Audio Session プログラミングガイド

Developer

目次

図、表、リスト

概要

iOSは音声まわりの動作を、（単独の）アプリケーション、複数のアプリケーション間、デバイスの各レベルで、*オーディオセッション*、すなわち`AVAudioSession`のAPIで処理します。

`AVAudioSession`のAPIで、次のような条件に応じた処理が可能です。

- 「着信／サイレント」スイッチに応じて消音するべきか。アプリケーションが機能を果たす上で音声が不可欠でなければ、答えは「はい」となります。たとえばメモ帳アプリケーションがこれに当たります。会議中は邪魔にならないよう、消音できるようになっていなければなりません。一方、発音辞典のように音声がなければ目的を果たせない場合は、「着信／サイレント」スイッチの状態に関係なく再生する必要があります。
- 音楽再生中、別の音声を出す必要が生じたとき、そのまま音楽を再生し続けるか。音楽ライブラリの曲に合わせてユーザが演奏する「仮想ピアノ」アプリケーションの場合、答えは「はい」となります。一方、インターネット経由でストリーミングラジオを再生するアプリケーションであれば、別の音声が出ている間は消すべきでしょう。

ユーザがヘッドセットのプラグを抜き差しする、電話がかかってくる、アラーム音が鳴る、といったことはいつでも起こりえます。このように、iOSデバイスの音声まわりの環境は、非常に複雑であると言えます。大部分の処理はiOSが受け持ちますが、開発者もオーディオセッションAPIを使って必要な設定を施し、システムの要求にきちんと対処できるよう実装しなければなりません。もっともこれは、ごくわずかのコードを記述するだけで可能です。

はじめに

`AVAudioSession`には、音声に関係するアプリケーションの動作を制御する働きがあり、次のような処理が可能です。

- アプリケーションに応じて、入出力の適切な信号経路を選択する
- 他のアプリケーションが再生する音声の取り扱いを判断する
- 他のアプリケーションからの割り込みを処理する
- アプリケーションの種類に応じて、自動的に適切な音声設定をする

オーディオセッションは音声まわりのさまざまな動作をカプセル化する

オーディオセッションは、アプリケーションとiOSの仲介役として、音声まわりの動作を設定するために使います。アプリケーションは起動後自動的に、シングルトンであるオーディオセッションを取得するようになっています。

> **関連する章:** “オーディオセッションの定義”（9 ページ）。

カテゴリは音声が果たす役割を表す

オーディオセッションには、音声まわりの動作を表現する基本的な仕組みとして、カテゴリというものがあります。アプリケーションにカテゴリを設定することにより、入出力の信号経路を使うか、音楽再生中に別の音声を出す際、そのまま音楽を再生し続けるか、などといった振る舞いを指定できるのです。ここで指定する振る舞いは、ユーザの予想を裏切らないものでなければなりません（『*iOS Human Interface Guidelines*』の「Sound」 in *iPhone Human Interface Guidelines* を参照）。

7種類のカテゴリに、オーバーライドスイッチや修飾スイッチを組み合わせることにより、アプリケーションの特性や役割に応じて振る舞いをカスタマイズできます。再生、録音、録音同時再生にそれぞれ対応したカテゴリがあります。システムは、アプリケーションにおける音声の役割を認識した上で、ハードウェア資源を効果的に活用します。また、デバイス上で再生される他の音声も、アプリケーションにとって具合のよい動作になるようシステムが制御します。たとえば「ミュージック(Music)」アプリケーションに割り込む必要が生じれば、そのとおりに動作するのです。

> **関連する章:** “カテゴリの取り扱い”（16 ページ）、“割り込みへの応答”（23 ページ）。

モードには各カテゴリの振る舞いをカスタマイズする働きがある

アプリケーションはそのカテゴリに応じた振る舞いをし、ユーザもそれを期待します。モードには、この振る舞いを特別扱いする働きがあります。たとえばVideoRecordingモードを指定すると、システムは、Defaultモードの場合とは別の組み込みマイクを選ぶ可能性があります。さらに、動画録音に適するよう、マイクの信号処理を調整するかも知れません。

> **関連する章:** “カテゴリの取り扱い”（16 ページ）.

通知機能には割り込みを処理する働きがある

音声割り込みとは、アプリケーションのオーディオセッションを停止し、したがって音声再生を止める動作のことです。割り込みが発生するのは、他のアプリケーションが競合するオーディオセッションを作動し、そのカテゴリが、音声の合成を認めるものでない場合です。すると、自アプリケーショ

ン側のセッションが停止した後、システムは割り込みが発生した旨のメッセージを配送します。これを受けて、状態を保存する、ユーザインターフェイスを更新する、などの処理をおこなうことになります。

割り込みに対処するためには、`AVAudioSession`の`AVAudioSessionInterruptionNotification`通知を受け取るよう登録する必要があります。ユーザが気づくような処理の中断を最小限に抑え、割り込み状態から円滑に復旧できるように、`beginInterruption`メソッドと`endInterruption`メソッドを記述してください。

関連する章: “割り込みへの応答” （23 ページ）.

通知機能を利用して音声信号の経路変更に対処する

ドック上でデバイスを着脱したり、ヘッドセットを抜き差ししたりすると、**音声信号の経路変化**が起こりますが、ユーザは当然、適切に対処されるものと期待するでしょう。『*iOS Human Interface Guidelines*』の「Sound」 in *iPhone Human Interface Guidelines* に、ユーザがどのような期待をし、どのように対応すればよいか、ガイドラインが載っています。経路変更に対処するためには、`AVAudioSessionRouteChangeNotification`通知が届くよう登録する必要があります。

関連する章: “信号経路の変更に対する応答” （34 ページ）。

各カテゴリの振る舞いをさらに洗練する

オーディオセッションのカテゴリは、さまざまな方法で振る舞いに手を入れることができます。できることはカテゴリによって異なりますが、たとえば次のようなことが可能です。

- 「ミュージック(Music)」アプリケーションなど、他の音声源からの信号を合成する（本来は許可していないカテゴリで特に許可）。
- 音声の出力経路をレシーバーからスピーカーに変更する。
- Bluetooth音声入力を許可する。
- 音声を再生する際、他の音声源からの信号は音量を下げるよう指定する。
- 実行時にデバイスハードウェアの処理を最適化する。アプリケーションが動作するデバイスの特性や、ユーザの操作（ヘッドセットのプラグの抜き差しなど）に合わせて、振る舞いを変える。

関連する章: “カテゴリの取り扱い”（16 ページ）、“デバイスハードウェアに合わせたアプリケーションの最適化”（29 ページ）、“プレーヤーを想定したオーディオセッションの微調整”（37 ページ）

必要事項

開発に当たっては、CocoaTouchにもとづくアプリケーション開発手順（『*iOS App Programming Guide*』を参照）や、Core Audioの基本事項（『*Core Audio Overview*』を参照）を把握しておく必要があります。オーディオセッションは、エンドユーザによる実際の使用状況に適合するように実装しなければならないので、iOSデバイスやiOSヒューマンインターフェイスガイドライン（特に『*iOS Human Interface Guidelines*』の「Sound」 in *iPhone Human Interface Guidelines* を参照）も頭に入れておかなければなりません。

関連項目

以下の資料も役に立つでしょう。

- 『*AVAudioSession Class Reference*』では、オーディオセッションを設定し、この技術を活用するための、Objective-Cインターフェイスについて説明しています。
- サンプルコードプロジェクト『*AddMusic*』には、音声再生アプリケーションにおけるオーディオセッションオブジェクトの使い方を示すコード例があります。「ミュージック(Music)」アプリケーションが再生する音声信号との協調のしかたを示す例にもなっています。

オーディオセッションの定義

オーディオセッションは、アプリケーションとiOSの仲介役として、音声まわりの動作を設定するために使います。アプリケーションは起動後自動的に、シングルトンであるオーディオセッションを取得するようになっています。これに設定を施すことにより、音声をどのように扱うか、開発者の意図を表現します。たとえば次のような事項です。

- 他のアプリケーション（「ミュージック(Music)」など）の音声信号を合成して出力するか、あるいは消音するか。
- 「時計(Clock)」のアラームのような音声割り込みに対してどのように対応するか。
- ユーザがヘッドセットを抜き差ししたとき、どのように対応するか。

オーディオセッションの設定は、アプリケーションの動作中、音声に関するあらゆる処理に影響します。ただし、「System Sounds Services」APIを使って再生される、ユーザインターフェイスまわりの効果音を除きます。オーディオセッションに対して、アプリケーションが動作するデバイスのハードウェア特性（チャネル数、サンプルレートなど）を問い合わせることができます。こういった特性はデバイスによって異なるだけでなく、動作中、ユーザの操作によって変わることもあります。

オーディオセッションは明示的に作動、停止できます。音声を再生したり、録音したりするためには、オーディオセッションを作動する必要があります。システムがオーディオセッションを停止することもあります。たとえば、電話が着信したりアラームが鳴ったりするとそうなります。このような動作を**割り込み**と呼びます。オーディオセッションのAPIを使って、割り込みに応じた処理をした後、通常状態に戻ることができます。

オーディオセッションの既定の動作

オーディオセッションにははじめからいくつか動作が組み込まれています。具体的には次のとおりです。

- 再生は有効、録音は無効になっている。
- 「サイレント」スイッチ（iPhoneの場合は「着信／サイレント」スイッチ）を「サイレント」側に切り替えると、音声は鳴らなくなる。
- 「スリープ／スリープ解除」ボタンを押して画面をロックするか、所定の時間が経過して自動ロックがかかると、音声は鳴らなくなる。

- 音声の再生を始めると、当該デバイスの他の音声（再生中の「ミュージック(Music)」アプリケーションなど）は鳴らなくなる。

上記はいずれも、既定のカテゴリ（`AVAudioSessionCategorySoloAmbient`）を指定したオーディオセッションの動作です。“カテゴリの取り扱い” （16 ページ）で、アプリケーションの特性に応じたカテゴリを指定する方法を解説します。

再生や録音を始めれば自動的にオーディオセッションも作動しますが、この挙動を当てにするのは危険です。たとえば、iPhoneの着信音が鳴ったとき、ユーザが電話には出ずにアプリケーションを実行し続けていても、採用している再生技術によっては音声が鳴らなくなります。このような問題への対処法をいくつか次の節で説明し、“割り込みへの応答” （23 ページ）でさらに詳しく解説します。

アプリケーション開発中は、既定の動作でもとりあえず作業を進めることができるでしょう。しかし、そのままで公開しても構わないのは、以下の条件を満たす場合に限ります。

- System Sound Services、またはUIKitの`playInputClick`メソッドだけで、音声を処理している場合。

  System Sound Servicesには、UI要素の効果音を再生したり、振動を起こしたりする役割があります。それ以外の目的には適していません（『*System Sound Services Reference*』と、サンプルコードプロジェクト『*Audio UI Sounds (SysSound)*』を参照）。

  UIKitの`playInputClick`メソッドを使えば、カスタム入力ビューやキーボードアクセサリビューで、標準的なキーボードクリック音を再生できます。そのオーディオセッションの動作は、システムが自動的に処理します。『*Text Programming Guide for iOS*』の「Playing Input Clicks」 in *Text, Web, and Editing Programming Guide for iOS* を参照してください。
- 音声をまったく扱わないアプリケーションの場合。

どちらにも該当しない場合、既定のオーディオセッションのままでアプリケーションを公開してはなりません。

既定のままのオーディオセッションでは具合が悪い理由

オーディオセッションを初期化し、必要な設定を施して明示的に使わないと、割り込みや音声信号の経路変化に対処できません。また、他のアプリケーションが出す音声信号との合成については、完全にOS任せになってしまいます。

いくつかのシナリオに沿って、オーディオセッションの既定の動作と、その変更方法を例示しましょう。

- シナリオ1。「オーディオブック」アプリケーション。ユーザが『*ヴェニスの商人*』を聴いていると、バサーニオが登場したとたん自動ロックがかかり、音声も止まってしまいました。

  画面がロックしても音声が止まらないようにするためには、再生可能なカテゴリをオーディオセッションに指定するとともに、フラグ`UIBackgroundModes`を立てる必要があります。

- シナリオ2。FPSシューティングゲーム（OpenALベースの効果音つき）。ゲームに合わせてサウンドトラックを再生してもよいのですが、これを切り、音楽ライブラリから自分で曲を選ぶこともできるようになっています。わくわくするような音楽が流れていますが、敵艦に向けて1発目の光子魚雷を射出したとたん、曲は止まってしまいました。

  音楽を止めないためには、音声合成を許可するようにオーディオセッションを設定する必要があります。具体的には、`AVAudioSessionCategoryAmbient`カテゴリを指定するか、または`AVAudioSessionCategoryPlayback`カテゴリを指定した上で、音声合成を許可するよう別途設定します。

- シナリオ3。Audio Queue Servicesを援用してストリーミングラジオを再生するアプリケーション。ラジオを聴いている最中に電話がかかってくると、ラジオの音は止まります（これは正しい反応です）。ユーザは、この電話には出ないことにして警告を閉じ、ストリーミングで音楽の続きを聴こうと「再生(Play)」をタップしますが、何も起きません。再生を再開するには、いったんアプリケーションを終了して立ち上げ直す必要があります。

  オーディオキューの割り込みにきちんと対処するため、カテゴリを適切に設定し、さらに`AVAudioSessionInterruptionNotification`通知が届くよう登録をした上で、この通知に対処するよう実装してください。

音声まわりの競合する要求に対するシステムの対処

iOSアプリケーションを起動したとき、バックグラウンドではすでに、組み込みアプリケーション（「SMS/MMS(Messages)」、「ミュージック(Music)」、「Safari」、「電話(Phone)」）が動作しているかも知れません。各アプリケーションは、テキストメッセージの着信音、10分前に開始したPodcastの再生など、音声を出力しようとするでしょう。

iOSデバイスを空港にたとえるなら、アプリケーションは離陸許可を待つ飛行機、システムは管制塔のようなものといえます。アプリケーションは音声まわりの要求を出したり、希望する優先度を表明したりできますが、実際に「滑走路」上で行われることを取りしきる最終決定権はシステムにあります。アプリケーションと「管制塔」との通信は、オーディオセッションを使って行います。図1-1に、

典型的なシナリオ、すなわち、「ミュージック(Music)」アプリケーションが曲を再生しているときに、音声まわりの処理をしたい新たなアプリケーションが現れた状況を示します。この場合、新たなアプリケーションは、「ミュージック(Music)」に割り込むことになります。

図 1-1　音声まわりの競合する要求をシステムが管理

図のステップ1で、アプリケーションはオーディオセッションを作動するよう要求します。これが起こるのは、アプリケーションの起動時、あるいは録音再生アプリケーションの「再生(Play)」ボタンをタップしたときなどです。ステップ2で、システムはこの要求の内容、特にオーディオセッションに指定されたカテゴリを検討します。図1-1でSpeakHereアプリケーションは、他の音声を消音するよう要求するカテゴリを指定しています。

ステップ3と4で、システムは「ミュージック(Music)」アプリケーションのオーディオセッションを停止し、その音声が再生されないようにします。最後のステップ5で、システムはSpeakHereアプリケーションのオーディオセッションを作動し、再生を始めます。

システムには、デバイス上のどのオーディオセッションでも作動/停止する、最終決定権があります。判断に当たってシステムは、「常に電話が優先する」という絶対的な規則に従います。どのようなアプリケーションがどれだけ高い優先度を求めても、電話の優先度を上回ることはできません。電話が着信すると、ユーザに通知し、アプリケーションに割り込みをかけます。何か音声まわりの操作をしている最中であっても、どのカテゴリを指定していても関係ありません。

AVCaptureSessionを組み込む

AV Foundationの取り込み処理API（`AVCaptureDevice`、`AVCaptureSession`）で、カメラやマイクの入力から、同期音声/画像を取り込むことができます。iOS 7以降、マイク入力を表す`AVCaptureDevice`オブジェクトは、アプリケーションの`AVAudioSession`を共有できるようになりました。通常は`AVCaptureSession`が、`AVCaptureSession`がマイクを使っているとき録音に適したように`AVAudioSession`を設定するようになっています。一方、`automaticallyConfiguresApplicationAudioSession`プロパティをNOにすると、`AVCaptureDevice`が、現在の`AVAudioSession`設定をそのまま使うようになります。詳しくは『*AVCaptureSession Class Reference*』および「メディアキャプチャ」を参照してください。

オーディオセッションを初期化する

アプリケーションを起動すると、システムがオーディオセッションのオブジェクトを用意します。しかし割り込みを処理するためには、セッションの初期化が必要です。

割り込み管理にAV Foundationフレームワークを使うことにすれば、次のように、`AVAudioSession`オブジェクトの参照を取得する際、自動的に初期化されます。

```
// implicitly initializes your audio session
AVAudioSession *session = [AVAudioSession sharedInstance];
```

`session`変数は既に初期化済みのオーディオセッションを表しており、すぐに使用できます。音声割り込みを処理する際には、`AVAudioSession`クラスの割り込み通知、または`AVAudioPlayer`クラスと`AVAudioRecorder`クラスのデリゲートプロトコルを使って、自動初期化の仕組みを活用するとよいでしょう。

音量や信号経路の制御機能を追加する

`MPVolumeView`クラスは音量や信号経路の制御機能を表します。音量ビューには、アプリケーション内から音量を制御するスライダと、出力信号経路を選択するボタンがあります。組み込みスピーカーに音声信号を流すためには、`AVAudioSessionPortOverride`を介して`MPVolumeView`の音声経路ピッカーを使うとよいでしょう。詳細については、『*MPVolumeView Class Reference*』を参照してください。

遠隔制御イベントに応答する

遠隔制御イベントを利用して、ユーザにマルチメディアを制御させることができます。アプリケーションに音声や画像の再生機能を組み込む場合、トランスポートコントロールや外付け機器から送られてくる、遠隔制御イベントに対処しなければなりませんiOSはコマンドを`UIEvent`オブジェクトに変換し、アプリケーションに配送します。アプリケーションはこれをファーストレスポンダに送信します。ファーストレスポンダが処理できなければ、レスポンダチェーンをたどって順次伝播していきます。

アプリケーションがイベントに応答するためには、「Now Playing」対応で、しかも音声再生中でなければなりません。詳しくは「Remote Control Events」および『*MPNowPlayingInfoCenter Class Reference*』を参照してください。

オーディオセッションを作動／停止する

システムはアプリケーションの起動時に、自動的にオーディオセッションを作動します。しかし実際には、明示的に作動する方がよいでしょう。通常はアプリケーションの`viewDidLoad`メソッド内で、望ましいハードウェア値の設定に先だっておこないます。ハードウェア値の設定については、コード

例が"最適なハードウェア値を設定する"（30 ページ）にあります。明示的に作動すれば、正常に作動できたか確認し、対処できるという利点があります。一方、再生/一時停止のUI要素がある場合は、ユーザがこのボタンを押して初めてセッションが作動するよう、コードを記述する必要があります。同様に、オーディオセッションの作動／停止状態を変更したときには、正常に変更できたかどうか確認しなければなりません。システムがセッションの作動を拒否した場合でも、きちんと対処できるように記述してください。

「時計(Clock)」や「カレンダー(Calendar)」のアラームが起動し、あるいは電話が着信すると、システムはオーディオセッションを停止します。ユーザがアラームを消したり、電話を無視したりした場合、アプリケーションはオーディオセッションを再び作動できるようになります。割り込み処理の最後にセッションを作動し直すか否かは、アプリケーションの種類によります（"アプリケーションの種類に応じた音声の取り扱いガイドライン"（41 ページ）を参照）。

リスト 1-1に、オーディオセッションを作動する方法を示します。

リスト 1-1　AV Foundationフレームワークを使ってオーディオセッションを作動するコード例

```
NSError *activationError = nil;

BOOL success = [[AVAudioSession sharedInstance] setActive: YES error:
&activationError];

if (!success) { /* activationErrorに示されているエラーを処理する */ }
```

`setActive`プロパティを`NO`とすれば、オーディオセッションは停止します。

音声の再生／録音に`AVAudioPlayer`オブジェクトや`AVAudioRecorder`オブジェクトを使っていれば、割り込み終了時にオーディオセッションを作動し直す処理はシステムがおこないます。しかし実際には、通知メッセージを配送するよう登録し、明示的にオーディオセッションを作動し直すようお勧めします。このようにすれば、正常に作動し直せたことを確認した上で、アプリケーションの状態やユーザインターフェイスを更新できるからです。詳しくは図3-1（25 ページ）を参照してください。

ほとんどのアプリケーションでは、オーディオセッションを明示的に停止する必要はありません。しかし重要な例外として、VoIP（Voice over Internet Protocol）アプリケーションや、道案内アプリケーション、さらに場合によっては、録音アプリケーションがあります。

- VoIPアプリケーションは通常、バックグラウンドで動作するので、オーディオセッションの作動は発着信の処理中に限定しなければなりません。バックグラウンドで着信を待機している状態では作動しないようにしてください。
- 「録音」カテゴリを指定したアプリケーションの場合、オーディオセッションを作動してよいのは録音中に限ります。録音の開始前や終了後はセッションを停止して、メッセージアラートの着信音など、他の音声を再生できる状態にしてください。

アプリケーション起動時に他の音声が再生中かどうか確認する

ユーザがアプリケーションを起動したとき、既に音声が再生中かも知れません。たとえばその時点で、「ミュージック(Music)」で音楽を再生していた、Safariで音声ストリーミングをしていた、などが考えられます。起動したのがゲームであれば、これは特に重要です。多くのゲームがサウンドトラックや効果音を活用しています。『*iOS Human Interface Guidelines*』の「Sound」で述べているように、ゲームの場合、その効果音は再生しつつ、起動前に再生していた音声もそのまま再生されるものとユーザは想定しています。

アプリケーションを起動する際、`otherAudioPlaying`プロパティで、音声が再生中であるかどうか調べてください。他の音声が再生中であれば、ゲーム側のサウンドトラックを無音にし、`AVAudioSessionCategorySoloAmbient`カテゴリを指定します。カテゴリについて詳しくは、“カテゴリの取り扱い” （16 ページ）を参照してください。

複数のアプリケーションにまたがって音声を処理する

複数のアプリケーションにまたがって音声を処理する基本的な形態では、あるひとつのアプリケーション（ノード）が、その音声出力を他のアプリケーション（ホスト）に送ります。しかし、ホストがその出力をノードに送信し、ノードが何らかの処理を施した上でホストに送り返す、ということも可能です。ホスト側ではオーディオセッションを作動する必要がありますが、ノード側でそれが必要なのは、ホスト、あるいはシステムから、入力を受け取る場合に限ります。アプリケーションをまたがる音声処理の設定に当たっては、以下のガイドラインに従ってください。

- ホスト、ノードの双方に、「inter-app-audio」エンタイトルメントを設定します。
- ホスト側アプリケーションは「UIBackgroundModes」 in *Information Property List Key Reference* 音声フラグを立てます。
- ノード側アプリケーションでは、音声入力／出力経路を用い、同時にホスト側アプリケーションにも接続する場合、「UIBackgroundModes」 in *Information Property List Key Reference* 音声フラグを立ててください。
- ホスト、ノードの双方に、`AVAudioSessionCategoryOptionMixWithOthers`を設定してください。
- ノード側アプリケーションでは、システムから音声入力を受信する（あるいは音声出力を送信する）と同時に、ホスト側アプリケーションにも接続する場合、オーディオセッションを作動する必要があります。

カテゴリの取り扱い

オーディオセッションの**カテゴリ**は、アプリケーションに対する音声まわりの振る舞いを表すキーとなります。カテゴリを指定することによって、「着信／サイレント」スイッチがフリップされたときに音声再生を続けるか、などといった音声まわりの振る舞いを、システムに伝えることができます。7種類のカテゴリに、オーバーライドスイッチや修飾スイッチを組み合わせることにより、振る舞いをカスタマイズできます。

各カテゴリは、次の振る舞いの可否（はい／いいえ）を表す組み合わせになっています（詳しくは表 B-1 （46 ページ）を参照）。

- **合成不可の音声に対する割り込み**：「はい」の場合、アプリケーションがオーディオセッションを作動すると、他のアプリケーションが処理する合成不可の音声に割り込みが発生します。
- **サイレントスイッチによる消音**：「はい」の場合、ユーザが「サイレント」スイッチをサイレント側に動かすと無音になります（iPhoneの場合は*「着信／サイレント」スイッチ*と呼びます）。
- **音声入力の可不可**：「はい」であれば、音声入力（録音）が可能になります。
- **音声出力の可不可**：「はい」であれば、音声出力（再生）が可能になります。

多くの場合、カテゴリの指定は、起動時に一度おこなうだけで充分です。しかし必要ならば、セッションの状態（作動/停止）によらず、いつでも指定し直すことができます。停止中であれば、実際には次に作動したときに、カテゴリ変更要求が送られることになります。既に作動中であれば直ちに送られます。

最適なカテゴリを選択する

各カテゴリに対応する実際の動作は、アプリケーションで制御することはできず、最終的にはオペレーティングシステムの判断によって決まります。また、将来のiOSバージョンでは、改良によってカテゴリの動作が変更される可能性もあります。アプリケーションの用途に応じ、最適な動作のカテゴリを選んでください。付録“オーディオセッションのカテゴリとモード” （46 ページ）に、各カテゴリの振る舞いを要約して示します。

カテゴリの選択に当たっては、次の事項を考慮してください。

- 再生と録音のどちらが必要か、両方とも必要か、あるいは単にオフラインで音声処理を行うだけでよいのか。

- 再生する音声は、アプリケーションの機能上、不可欠なものか否か。不可欠であれば、「着信／サイレント」スイッチがサイレント側になっても、再生を継続できるカテゴリが最適です。そうでなければ、サイレント側になったとき消音するカテゴリを選択します。
- アプリケーションを起動した時点で、他のアプリケーション（「ミュージック(Music)」など）が音声を再生していた場合にどうするか。起動時に状態を確認し、カテゴリを切り替えます。たとえばゲームの場合、再生中の音楽はそのまま再生するカテゴリも考えられますが、場合によっては、組み込みのアプリケーションサウンドトラックをサポートするカテゴリ設定が適しているかも知れません。

各カテゴリとその音声まわりの振る舞いを以下に示します。`AVAudioSessionCategoryAmbient`カテゴリは、他の音声の継続再生を許可し、したがって音声合成可能なアプリケーションになります。これ以外のカテゴリの場合、オーディオセッションを作動すれば他の音声は停止します。ただし、音声合成しない「再生」や「再生と録音」のカテゴリをカスタマイズして、合成できるようにすることも可能です（"カテゴリの振る舞いに手を入れる"（21 ページ）を参照）。

- `AVAudioSessionCategoryAmbient`：再生のみ。アプリケーションに磨きをかけ、雰囲気を盛り上げる目的で用いるだけで、その機能上、音声が不可欠ではない場合に指定します。「着信／サイレント」スイッチをサイレント側に動かし、あるいは画面がロックされたときには無音になります。
- `AVAudioSessionCategorySoloAmbient`：（既定値）再生のみ。「着信／サイレント」スイッチをサイレント側に動かし、あるいは画面がロックされたときには無音になります。`AVAudioSessionCategoryAmbient`カテゴリとの違いは、既に再生中の音声があれば割り込む、という点だけです。
- `AVAudioSessionCategoryPlayback`：再生のみ。「着信／サイレント」スイッチがサイレント側、あるいは画面がロックされた状態であっても、再生を続けます。アプリケーションの機能上、音声が不可欠である場合に指定してください。

> **注意:** 画面ロック時でも再生を継続するカテゴリを選んだ場合、アプリケーションのinfo.plistで、`UIBackgroundModes`の音声を設定しなければなりません。詳しくは「UIBackgroundModes」 in *Information Property List Key Reference*を参照してください。`idleTimerDisabled`プロパティでシステムのスリープタイマーを無効にすることができますが、通常は*避け*てください。スリープタイマーを無効にした場合、画面ロックを回避する必要がなくなり次第、このプロパティを`NO`にリセットしなければなりません。スリープタイマーとは、所定の時間が経過すると画面を暗くして、電池を節約する機能のことです。

- `AVAudioSessionCategoryRecord`：録音のみ。再生もおこなう場合は`AVAudioSessionCategoryPlayAndRecord`を指定します。

- `AVAudioSessionCategoryPlayAndRecord`：再生と録音。入力と出力は、必ずしも同時でなくても構いません。音声チャットアプリケーションに向いています。
- `AVAudioSessionCategoryAudioProcessing`：オフライン音声処理のみ。オフラインで音声を処理し、再生も録音もしません。
- `AVAudioSessionCategoryMultiRoute`：再生と録音。異なる音声ストリーム（USB出力とヘッドフォン出力など）に対する同時入出力を許可します。DJアプリケーションには、複数の信号経路を処理できる、このカテゴリが向いているでしょう。あるトラックの音声を聴きながら、別のトラックを再生する必要がしばしば生じます。踊りに合わせてトラックを再生しながら、ヘッドフォンでは少し先のトラックを聴いている、という使い方がしたいからです。

複数の信号経路を処理できるカテゴリで、機能を拡張する

複数の経路を処理するカテゴリには、他とは若干異なる振る舞いがあります。一般にどのカテゴリも、「最後が総取り」規則に従います。入出力の経路に最後に接続されたデバイスが、主たるデバイスになる、という規則です。しかし複数経路を処理するカテゴリの場合、アプリケーションは最後だけでなく、接続された出力ポートをすべて使えます。たとえば、HDMI出力経路を通して音声を再生しているときに、ヘッドフォンを接続すると、ここからも聞こえるようになります。HDMI出力経路を通して再生を継続したまま、ヘッドフォンでも再生できるのです。

さらに、出力経路ごとに、異なる音声ストリームを送信することも可能です。たとえば、ある音声ストリームをヘッドフォンの左、別の音声ストリームをヘッドフォンの右、そして第3の音声ストリームをHDMIの経路に送る、といった使い方ができるのです。図2-1に、音声経路ごとに異なる音声ファイルを送信する例を示します。

図 2-1　音声経路ごとにに異なる音声ファイルを送信する様子

デバイスや周辺機器によって異なりますが、次のような出力経路の組み合わせが可能です。

- USBとヘッドフォン
- HDMIとヘッドフォン
- ラインアウトとヘッドフォン

複数経路カテゴリでは、入力ポートも1系統使えます。

オーディオセッションのカテゴリを設定する

多くのiOSアプリケーションで、オーディオセッションのカテゴリを起動時に指定し、最後まで変更しないことにして良好に動作します。実行中、音声まわりの動作が一貫しているので、使っていて戸惑うことがないのです。

カテゴリの設定は`setCategory:error:`メソッドでおこないます（リスト 2-1を参照）。個々のカテゴリについては“最適なカテゴリを選択する” （16 ページ）を参照してください。

リスト 2-1　AV Foundationフレームワークを使ったオーディオセッションカテゴリの設定

```
NSError *setCategoryError = nil;
BOOL success = [[AVAudioSession sharedInstance]
                setCategory: AVAudioSessionCategoryAmbient
                      error: &setCategoryError];

if (!success) { /* setCategoryError示されているエラーを処理する */ }
```

モードを指定してカテゴリの動作を特殊化する

カテゴリによってアプリケーションの基本的な動作が決まりますが、モードにはそれを特殊化する働きがあります。カテゴリに追加する形でモードを指定することにより、音声まわりの動作をさらに細かく定義できるのです。次の7種類のモードがあります。

- `AVAudioSessionModeDefault`：どのカテゴリに対しても指定できる既定のモードで、デバイスの汎用的な使い方に向いています。
- `AVAudioSessionModeVoiceChat`：VoIP（Voice over IP）アプリケーション用。`AVAudioSessionCategoryPlayAndRecord`カテゴリの場合にのみ指定できます。システム組み込みの信号処理系で、人の声として最適な形に処理します。`AVAudioSessionCategoryOptionAllowBluetooth`も設定します。

  利用可能な一連の音声経路は、ボイスチャット用に最適化されます。組み込みマイクを使う場合、システムは自動的に、ボイスチャット用に最適な組み合わせを選択します。
- `AVAudioSessionModeVideoChat`：FaceTimeなどのビデオチャットアプリケーションに向いています。`AVAudioSessionCategoryPlayAndRecord`カテゴリの場合にのみ指定できます。システム組み込みの信号処理系で、人の声として最適な形に処理します。`AVAudioSessionCategoryOptionAllowBluetooth`および`AVAudioSessionCategoryOptionDefaultToSpeaker`も設定します。

利用可能な一連の音声経路は、ビデオチャット用に最適化されます。組み込みマイクを使う場合、システムは自動的に、ビデオチャット用に最適な組み合わせを選択します。

> **注意:** ボイスチャットやビデオチャットを利用するアプリケーションは、「Voice-Processing I/O」音声ユニットも使うようお勧めします。このユニットはVoIPアプリケーション向けに、自動利得補正、ボイス処理の補正、無音化など、いくつか機能を備えています。詳しくは「Voice-Processing I/O Unit」 in *Audio Unit Hosting Guide for iOS* を参照してください。

- `AVAudioSessionModeGameChat`：ゲームアプリケーション用。`GKVoiceChat`オブジェクトを使う`AVAudioSessionCategoryPlayAndRecord`カテゴリのアプリケーションは、このモードが自動的にオンになります。このモードはビデオチャットモードと同じ経路制御パラメータを使います。
- `AVAudioSessionModeVideoRecording`：カメラを使ってビデオを取り込むアプリケーション用。`AVAudioSessionCategoryPlayAndRecord`または`AVAudioSessionCategoryRecord`カテゴリの場合にのみ指定できます。信号の加工は、システム組み込みの信号処理系がおこないます。

  このモードでは、`AVCaptureSession` APIを使い、入出力の経路をさらに細かく制御できます。たとえば、`automaticallyConfiguresApplicationAudioSession`プロパティを設定すると、デバイスやカメラに応じて自動的に、最適な入力経路が選択されます。
- `AVAudioSessionModeMeasurement`：システム組み込みの信号処理系による加工を最小限に抑えたいアプリケーション向け。録音、再生と録音、再生の各カテゴリの場合にのみ指定できます。入力信号はデバイスの主たるマイクを通して伝送されます。
- `AVAudioSessionModeMoviePlayback`：動画再生アプリケーション用。`AVAudioSessionCategoryPlayback`カテゴリの場合にのみ指定できます。

AirPlayに合わせてカテゴリやモードを選択する

AirPlayに対応しているのは一部のカテゴリやモードに限ります。以下に示すカテゴリは、Airplayに対応しています（ミラー版、非ミラー版とも）。

- `AVAudioSessionCategorySoloAmbient`
- `AVAudioSessionCategoryAmbient`
- `AVAudioSessionCategoryPlayback`

`AVAudioSessionCategoryPlayAndRecord`カテゴリはミラー版にしか対応していません。

モードはいずれも、AirPlayに対応しているのは、「再生と録音」カテゴリの場合に限ります。以下のモードはAurPlayミラーリングにしか対応していません。

- `AVAudioSessionModeDefault`
- `AVAudioSessionModeVideoChat`
- `AVAudioSessionModeGameChat`

カテゴリの振る舞いに手を入れる

オーディオセッションのカテゴリは、さまざまな方法で振る舞いに手を入れることができます。できることはカテゴリによって異なりますが、たとえば次のようなことが可能です。

- 「ミュージック(Music)」アプリケーションなど、他の音声源からの信号を合成する（本来は許可していないカテゴリで特に許可）
- 音声信号の出力経路をレシーバーからスピーカーに変更する（他の経路が使えない場合）
- オーディオセッションが作動しているとき、他の音声の音量を下げる必要がある旨を指定する

`AVAudioSessionCategoryPlayback`、`AVAudioSessionCategoryPlayAndRecord`、`AVAudioSessionCategoryMultiRoute`カテゴリでは、割り込みの特性をオーバーライドすることにより、他の音声を合成できるようにすることも可能です。これは`AVAudioSessionCategoryOptionMixWithOthers`プロパティをオーディオセッションに適用することによりおこないます。合成可能であるよう設定すると、オーディオセッションを作動する際、他の合成不可のアプリケーションに対して割り込まないようになります。また、他のアプリケーション（「ミュージック(Music)」など）の、合成不可のオーディオセッションから割り込まれることもありません。

音声信号の出力経路をプログラムで変更することも可能です。`AVAudioSessionCategoryPlayAndRecord`カテゴリを指定した場合、音声信号は通常、レシーバー（電話で通話する際に耳に当てる小型スピーカー）に送られます。この信号をiPhoneの底部にあるスピーカーにリダイレクトするためには、`overrideOutputAudioPort:error:`メソッドを使います。

最後に、音声再生中は他の音声の音量を自動的に下げるようにして、カテゴリの振る舞いを改善できます。たとえばエクササイズアプリケーションなどに有用でしょう。たとえば、「ミュージック(Music)」で再生する曲に合わせて体操しているときに、「ボートこぎを始めてから10分経過しました」といったメッセージを再生する必要があるとします。メッセージが確実に伝わるようにしたければ、オーディオセッションに`AVAudioSessionCategoryOptionDuckOthers`プロパティを適用するとよいでしょう。他の音声はすべて、音量が下がるようになります（電話の音は例外）。このようなアプリケーションは、他のセッションの作動状態を管理しなければなりません。オーディオセッションを作動した上で音声を再生し、終了後にセッションを停止します。

録音の可否

iOS 7以降、音声を録音するためには、その可否を訊ね、許可を得なければならなくなりました。許可が得られなかった場合、録音しても無音状態が記録されるだけです。録音可能なカテゴリのアプリケーションが入力経路を使おうとすると、システムが自動的に、録音の許可を求める画面を表示します。

この仕組みを当てにせず、`requestRecordPermission:`メソッドで許可を求めることも可能です。この`requestRecordPermission:`の方法には、アプリケーションの自然な流れを妨げないタイミングで許可を求めることができる、という利点があり、使い勝手の改善に役立つでしょう。

割り込みへの応答

オーディオセッションに割り込み処理コードを実装すれば、電話が着信したとき、「時計(Clock)」や「カレンダー(Calendar)」のアラームが鳴ったとき、あるいは他のアプリケーションがオーディオセッションを作動したときでも、適切に動作を継続できるようになります。

音声割り込みは、アプリケーションのオーディオセッションを（外部から）停止することです。割り込み処理の方式により、即座に停止する場合と、一時停止する場合があります。割り込みが発生するのは、他のアプリケーションが競合するオーディオセッションを作動し、そのカテゴリが、音声の合成を認めるものでない場合です。すると、自アプリケーション側のセッションが停止した後、システムは割り込みが発生した旨のメッセージを配送します。これを受けて、状態を保存する、ユーザインターフェイスを更新する、などの処理をおこなうことになります。

割り込み発生後、アプリケーションが保留状態になることもあります。これは着信した電話に出た場合の動作です。電話に出なかったり、アラームを消したりした場合、システムは割り込み終了メッセージを発行し、アプリケーションは処理を続行します。再生を再開するには、オーディオセッションを作動し直す必要があります。

音声割り込み処理の技法

音声割り込みに対処するためには、適切な`NSNotification`（割り込み通知）を配送するよう登録する必要があります。割り込み処理コードでおこなう処理は、音声処理技術とその用途（再生、録音、音声ファイル形式変換、ストリーミングオーディオパケットの読み取りなど）に依存します。いずれにしても、ユーザから見て、処理が中断したことをできるだけ気づかれないようにしてください。

表 3-1に、オーディオセッションが割り込み処理としておこなうべき処理を要約して示します。`AVAudioPlayer`オブジェクトや`AVAudioRecorder`オブジェクトを使っていれば、いくつかの処理はシステムが自動的におこないます。

表 3-1　オーディオセッションの割り込み処理としておこなうべき事項

割り込み処理の開始後	• 状態とコンテキストを保存する • ユーザインターフェイスを更新する

割り込み処理の終了後	• 状態とコンテキストを復元する • オーディオセッションを作動し直す（これはアプリケーションの性質に依存） • ユーザインターフェイスを更新する

表 3-2に、音声処理技術に応じた音声割り込みの処理方法を要約して示し、以降の節で詳しく説明します。

表 3-2 音声処理技術に応じた音声割り込みの処理方法

音声処理技術	割り込み時の動作
AV Foundationフレームワーク	`AVAudioPlayer`クラスと`AVAudioRecorder`クラスは、割り込みの開始と終了に関するデリゲートメソッドを備えています。ユーザインターフェイスを更新し、割り込み終了後、必要であれば一時停止していた再生を再開するためにこれらのメソッドを実装します。割り込み時に、システムは自動的に再生や録音を一時停止し、その後再生や録音が再開されるとオーディオセッションを再作動します。 現在の再生位置を保存しておき、次に起動したとき同じところから再生する仕組みにしている場合、アプリケーション終了時だけでなく割り込み時にも、再生位置を保存しなければなりません。
Audio Queue Services、I/Oオーディオユニット	割り込み時の処理をアプリケーションが制御するためのサービスです。再生や録音の位置を保存し、割り込み終了後にオーディオセッションを作動し直す処理は、明示的に実装しなければなりません。
OpenAL	再生にOpenALを使う場合は、適切な`NSNotification`通知を配送するよう登録してください（Audio Queue Servicesを使う場合と同様）。デリゲートではさらに、OpenALのコンテキストも管理する必要があります。“OpenALと音声割り込み”（26 ページ）を参照してください。
System Sound Services	割り込み処理が始まると、System Sound Services再生される音声は鳴らなくなります。割り込み処理終了後は自動的に元の音量に戻ります。この動作をアプリケーションが変更することはできません。

Siriからの割り込みを処理する

再生中にSiriから割り込みが入った場合、オーディオセッションが割り込まれている間にSiriが発行した遠隔制御コマンドを保存しておく必要があります。割り込み状態の間、コマンドをすべて保持しておき、割り込み終了時にその内容に応じて対処するのです。たとえば、割り込み中、ユーザがSiriに

対し、音声再生を停止するよう指示したとします。この場合アプリケーションは、割り込み処理が終了した旨の通知を受けても再生を再開せず、一時停止状態であることを表す画面表示にしなければなりません。

割り込みのライフサイクル

音声再生アプリケーションについて、図 3-1に、オーディオセッションの割り込み発生前、発生中、終了後の、一連のイベントを示します。

図 3-1 オーディオセッションに割り込みが発生したときの動作

割り込みイベント（この例では電話の着信）は次のように進行します。番号付きのステップは、図中の番号に対応しています。

1. アプリケーションが動作し、音声を再生している状態です。
2. 電話が着信しました。システムは電話アプリケーションのオーディオセッションを作動します。
3. システムは他のアプリケーションのオーディオセッションを停止します。その結果、音声の再生も停止します。
4. システムは割り込み処理用のコールバック関数を呼び出し、オーディオセッションが停止した旨を通知します。
5. コールバック関数が適切な処理をします。状況が分かるよう画面を更新する、後で再生を再開するために必要な情報を保存する、などの処理が考えられます。
6. ユーザが割り込みを無視した（電話に出ないことにした）場合、システムはコールバック関数を呼び出し、割り込みが終了した旨を通知します。
7. これに応じて、コールバック関数が適切な処理をします。状況が分かるよう画面を更新する、オーディオセッションを作動し直す、再生を再開する、などの処理が考えられます。
8. （図には示されていませんが、）ステップ6でユーザが割り込みを無視せずに電話に出た場合、アプリケーションは保留状態になります。通話が終わると、割り込み終了の通知が配送されます。

割り込み開始に対応して必ず割り込み終了もあるとは限りません。フォアグラウンド実行状態に切り替わったり、ユーザが「再生」ボタンを押したりした場合、アプリケーションはそれを認識する必要があります。いずれの場合も、オーディオセッションを作動し直すべきかどうか、的確に判断しなければなりません。

OpenALと音声割り込み

音声の再生にOpenALを使う場合、Audio Queue Servicesの場合と同様に、割り込み処理コールバック関数を実装します。この関数ではさらに、OpenALのコンテキストも管理する必要があります。すなわち、割り込み時にOpenALコンテキストを`NULL`にし、割り込み終了後、元の状態に戻すのです。

AVAudioPlayerクラスによる音声割り込みの処理

`AVAudioPlayer`クラスには、独自の割り込みデリゲートメソッドがあります（『*AVAudioPlayerDelegate Protocol Reference*』を参照）。同様に`AVAudioRecorder`クラスにも、独自の割り込みデリゲートメソッドがあります（『*AVAudioRecorderDelegate Protocol Reference*』を参照）。どちらのクラスも考え方はよく似ています。

割り込み開始のデリゲートメソッドが呼び出された時点では、プレーヤーが一時停止状態、オーディオセッションも停止状態になっています。このメソッドの実装は、たとえばリスト 3-1のようになります。

リスト 3-1　割り込み開始時にプレーヤーを制御するデリゲートメソッドのコード例

```
- (void) audioPlayerBeginInterruption: (AVAudioPlayer *) player {
    if (playing) {
        playing = NO;
        interruptedOnPlayback = YES;
        [self updateUserInterface];
    }
}
```

割り込み発生時、実際にプレーヤーが音声に再生していたかどうか確認した上で、アプリケーションの状態や画面表示を更新します（プレーヤーの一時停止はシステムが自動的に実行）。

ユーザが電話に出なかった場合、システムは自動的にオーディオセッションを作動し直し、割り込み終了のデリゲートメソッドを呼び出します。その簡単な実装例をリスト 3-2に示します。

リスト 3-2　割り込み終了時にプレーヤーを制御するデリゲートメソッドのコード例

```
- (void) audioPlayerEndInterruption: (AVAudioPlayer *) player {
    if (interruptedOnPlayback) {
        [player prepareToPlay];
```

```
        [player play];
        playing = YES;
        interruptedOnPlayback = NO;
    }
}
```

メディアサーバのリセットに対処する

メディアサーバは、共有サーバプロセスの形で、音声その他のマルチメディア機能を提供します。ごく稀ですが、アプリケーション動作中にメディアサーバがリセットされることもありえます。そのようなイベントに対処するため、`AVAudioSessionMediaServicesWereResetNotification`通知を配送するよう登録してください。実際に通知が届いた場合、アプリケーションは次の処理をおこなう必要があります。

- 孤児状態になっている音声オブジェクトを破棄し、新たに音声オブジェクトを生成する
- 内部的に追跡している音声の状態をリセットする（`AVAudioSession`のプロパティもすべて対象）
- 妥当であれば、`setActive:error:`メソッドで`AVAudioSession`を作動し直す

Important: アプリケーションは、`AVAudioSession`の通知を配送するよう登録し直さなくても構いません。また、`AVAudioSession`プロパティのキー値監視をリセットする必要もありません。

メディアサーバが応答しなくなった時点ですぐに知る必要があれば、`AVAudioSessionMediaServicesWereLostNotification`通知も配送するよう登録しておくとよいでしょう。もっとも、多くのアプリケーションは、リセットの通知に応答するだけで充分です。メディアサーバ非応答の通知が必要なのは、その状態になった後に発生したユーザイベントに、リセット前であっても応答しなければならない場合だけです。

ユーザにガイドラインを提供する

競合する他のオーディオセッションに割り込まれると具合が悪い状況も考えられます。たとえば講演を録音中に割り込みが入ることは望ましくないでしょう。

しかし、オーディオセッションの割り込みをプログラムで確実に防止する方法はありません。これは、iOSでは常に電話が最優先の扱いを受けるからです。また、ある種のアラームやアラートもiOSでは高い優先度を持ちます。飛行機に乗り遅れないためわざわざアラームを設定したのに、鳴らないということは許されません。

したがって、録音中の割り込みを確実に防ぐには、ユーザが意識的に次の操作をおこない、他の音声まわりの処理を止める必要があります。

1. 機内モードであるべきデバイスについては、「設定(Settings)」アプリケーションで、実際にそうなっていることを確認します。
2. 「設定(Settings)」アプリケーションで、「Do Not Disturb」をオンにしておきます。
3. 「カレンダー(Calendar)」アプリケーションで、録音の終了予定時刻までのイベントアラームをすべて解除します。
4. 「時計(Clock)」アプリケーションで、録音の終了予定時刻までの時計アラームをすべて解除します。
5. サイレントスイッチ（iPhoneでは*「着信／サイレント」スイッチ*）付きのデバイスの場合、録音中はこのスイッチを操作しないように注意します。サイレントモードを変更すると、ユーザ設定によっては、たとえばバイブレーションが作動することがあります。
6. 録音中はヘッドセットを抜き差ししないように注意します。デバイスをドックに載せたりドックからはずしたりする操作も、同じく録音中は行わないようにします。
7. 録音中はデバイスの電源アダプタを接続しないように注意します。iOSデバイスに電源アダプタを接続すると、デバイスやユーザ設定によってはビープ音が鳴ったり、バイブレーションが作動したりする恐れがあります。

アプリケーションのユーザガイドなどに、これらの注意事項を明示しておくことをお勧めします。

デバイスハードウェアに合わせたアプリケーションの最適化

オーディオセッションのプロパティを使って、デバイスハードウェアに合わせて実行時に、音声まわりの振る舞いを最適化できます。デバイスの特性や、ユーザの操作（ヘッドセットの接続、デバイスのドッキングなど）に応じて、アプリケーションの動作を適合させることができるのです。

オーディオセッションのプロパティを設定することにより、以下のことが可能です。

- サンプルレートとI/Oバッファ時間に対して最適なハードウェア設定を指定する
- さまざまなハードウェア特性（入出力の遅延時間やチャネル数、ハードウェアのサンプルレートや音量設定、音声入力の可否など）を問い合わせる
- デバイスに特有の通知に応答する

よくあるプロパティ値変更イベントとして、信号経路の変更があります（"信号経路の変更に対する応答"（34 ページ）を参照）。また、ハードウェア出力音量の変化、音声入力の可否の変化などのイベントについても、これに対処するコールバックを実装することは少なくありません。

音声まわりの最適なハードウェア値を選択する

ハードウェアのサンプルレートやI/Oバッファ時間として最適な値を、オーディオセッションのAPIで設定できます。表 4-1に、設定する値によってどのような得失があるかを示します。

表 4-1　最適なハードウェア値の選択

設定項目	最適なサンプルレート	最適なI/Oバッファ時間
高い値	*例：44.1 kHz* + 音質が高い – ファイルサイズが大きい	*例：500 ミリ秒* + ディスクアクセスの頻度が低い – 遅延時間が長い
低い値	*例：8 kHz* + ファイルサイズが小さい – 音質が低い	*例：5 ミリ秒* + 遅延時間が短い – ディスクアクセスの頻度が高い

たとえば、音質を重視し、ファイルサイズが大きくなってもそれほど問題にならない場合は、高サンプルレートを指定します（表の中央上段）。

I/Oバッファ時間は多くの場合、既定値（サンプルレートが44.1 kHzの場合で約0.02秒）のままで充分です。楽器のライブ監視など、遅延が起こらないことを重視する場合は低めの値を指定することも考えられますが、通常は変更しなくても構いません。

最適なハードウェア値を設定する

最適なハードウェア値の設定は、オーディオセッションを作動する前におこないます。作動後に設定しても効果は現れません。オーディオセッションを作動し直したときには、選択した値を改めて確認してください。また、アプリケーション実行中に設定値を変更する場合は、いったんオーディオセッションを停止するようお勧めします。リスト 4-1に、最適なハードウェア値を設定し、実際に使う前にその値を確認する方法を示します。

リスト 4-1　ハードウェア値を設定し、問い合わせるコード例

```
NSError *audioSessionError = nil;
AVAudioSession *session = [AVAudioSession sharedInstance];
[session setCategory:AVAudioSessionCategoryPlayback error:&audioSessionError];
if (audioSessionError) {
     NSLog(@"Error %ld, %@, (long)audioSessionError.code,
audioSessionError.localizedDescription);
}

NSTimeInterval bufferDuration =.005;
[session setPreferredIOBufferDuration:bufferDuration error:&audioSessionError];
if (audioSessionError) {
     NSLog(@"Error %ld, %@, (long)audioSessionError.code,
audioSessionError.localizedDescription);
}

double sampleRate = 44100.0
[session setPreferredSampleRate:samplerate error:&audioSessionError];
if (audioSessionError) {
     NSLog(@"Error %ld, %@, (long)audioSessionError.code,
audioSessionError.localizedDescription);
```

```
}


[[NSNotificationCenter defaultCenter] addObserver:self
                                          selector:@selector(handleRouteChange:)

name:AVAudioSessionRouteChangeNotification
                                            object:session];


[session setActive:YES error:&audioSessionError];
if (audioSessionError) {
     NSLog(@"Error %ld, %@, (long)audioSessionError.code,
audioSessionError.localizedDescription);
}


sampeRate = session.sampleRate;
bufferDuration = session.IOBuffferDuration;
NSLog(@"Sampe Rate:%0.0fHZ I/O Buffer Duration:%f", sampleRate, bufferDuration);
```

ハードウェア特性を問い合わせる

iOSデバイスのハードウェア特性は、アプリケーションの動作中に変化することがあります。また、デバイスによっても異なります。たとえば、初代iPhoneの内蔵マイクを使う場合、録音サンプルレートは8 kHzに制限されますが、ヘッドセットを接続してそのマイクを使えば、もっと高いサンプルレートにすることができます。さらに、最新のiOSデバイスでは、内蔵マイクのサンプルレートをより高い値に設定できるようになっています。

オーディオセッションには、デバイスが備えるさまざまなハードウェア特性について、情報を取得する機能があります。ハードウェア特性はアプリケーション実行中にも変化する可能性があります。たとえばヘッドセットを接続したことにより、入力サンプルレートが変わるかも知れません。『*AVAudioSession Class Reference*』に、関係するプロパティの一覧が載っています。

最適なハードウェア特性を指定する際には、あらかじめオーディオセッションを停止しておかなければなりません。設定後、オーディオセッションを作動してから、改めて実際の特性を問い合わせます。指定どおりに設定されない場合もあるので、この最後の手順は重要です。

Important: ハードウェア特性について意味のある値を取得するため、オーディオセッションを確実に初期化、作動した上で問い合わせてください。

さまざまなハードウェア特性のうち特に有用なものとして、`sampleRate`と`outputLatency`の2つがあります。`sampleRate`プロパティはデバイスのサンプルレートを表します。`outputLatency`プロパティは再生遅延を表します。

最適なハードウェアI/Oバッファ時間を指定する

`AVAudioSession`クラスを使って、最適なサンプルレートやI/Oバッファ時間を指定できます（リスト4-2を参照）。サンプルレートを設定するコードも同じような構成です。

リスト 4-2 AVAudioSessionクラスを使って最適なI/Oバッファ時間を指定するコード例

```
NSError *setPreferenceError = nil;
NSTimeInterval preferredBufferDuration = 0.005;
[[AVAudiosession sharedInstance]
            setPreferredIOBufferDuration: preferredBufferDuration
                                  error: &setPreferenceError];
```

システムが要求に応じられない場合もあるため、設定後は必ず、実際の値をハードウェアに問い合わせてください。

ハードウェアサンプルレートを取得する

録音の際には、準備処理の一環としてハードウェアのサンプルレートを取得し、それに合った音声データ形式を選択してください。その方法をリスト4-3に示します。このコードは一般に、録音を実行するクラスの実装ファイルに置きます。入出力チャネル数など、他のハードウェア特性を取得するコードも同様の構成です。

現在のハードウェア特性を問い合わせる際には、あらかじめオーディオセッションを作動しておいてください。

リスト 4-3 AVAudioSessionクラスを使って現在のハードウェアサンプルレートを取得するコード例

```
double sampleRate;
sampleRate = [[AVAudioSession sharedInstance] currentHardwareSampleRate];
```

iPhoneシミュレータ上でアプリケーションを実行する

オーディオセッションの処理を組み込んだアプリケーションは、実デバイスだけでなくシミュレータ上でも実行できます。ただし、オーディオセッションの振る舞いをシミュレートし、あるいはデバイスのハードウェア機能にアクセスすることはできません。シミュレータ上では実行できないものとして、次のような処理があります。

- 割り込みを起こす
- サイレントスイッチの設定を変更する
- 画面のロックをシミュレートする
- ヘッドセットの抜き差しをシミュレートする
- 音声信号の経路情報を問い合わせる、あるいはオーディオセッションのカテゴリに応じた振る舞いを確認する
- 音声合成の動作をテストする - たとえば、「ミュージック(Music)」など他のアプリケーションの音声を合成して再生する

シミュレータの性質上、コードのある部分だけを実行するよう切り分けたい場合も起こりえます。

ひとつの方法として、APIの戻り値にもとづいて条件分岐する、というやり方があります。オーディオセッションに関係する関数は、戻り値を必ず検査し、適切に対処しなければなりません。実デバイスでは正常に動作しても、シミュレータではそうならない場合、結果コードを見ればその理由が分かる場合があります。

また、プリプロセッサの条件文を使い、シミュレータ上で実行するか否かによってコードを書き分ける、というやり方もあります。リスト 4-4にその方法を示します。

リスト 4-4　プリプロセッサの条件文の使い方

```
#if TARGET_IPHONE_SIMULATOR
#warning *** Simulator mode: audio session code works only on a device
    // Execute subset of code that works in the Simulator
#else
    // Execute device-only code as well as the other code
#endif
```

信号経路の変更に対する応答

アプリケーション実行中に、ユーザがヘッドセットを抜き差ししたり、音声接続つきのドッキングステーションを使ったりすることもありえます。『*iOS Human Interface Guidelines*』では、このようなイベントにiOSアプリケーションが対処する方法を解説しています。これに従い、オーディオセッションを使って、音声信号の経路変更に対処するためのコードを実装しましょう。実際には、ゲームのように、経路変化への対処が必要ないアプリケーションもあります。一方、たとえばメディアプレーヤーの場合、どのような経路変化にも的確に応じなければなりません。

音声信号のさまざまな経路変化

音声信号経路とは、音声信号を電気的に伝える有線の通り道です。ユーザがヘッドセットを抜き差しすると、システムは音声信号経路を自動的に変更します。アプリケーションはこのような状態変化を、`AVAudioSessionRouteChangeNotification`の仕組みを使って監視できます。

図 5-1に、録音／再生中に生じるさまざまな経路変化に伴って生じる、一連のイベントを示します。プロパティの変化を監視するコールバック関数の処理内容に応じて、図の一番下に示す4通りの結果が考えられます。

図 5-1　音声信号経路の変化に対する処理

図のように、アプリケーション起動後、システムはまず音声信号経路を判定します。それ以降もアプリケーション動作中は、アクティブな経路の監視を続けます。たとえば、ユーザが「録音(Record)」ボタンをタップした場合を考えてみましょう。図の左側、「録音を開始する(Recording starts)」の箱がこれに該当します。

録音中にユーザがヘッドセットを抜き差しするかも知れません（図の左下にある菱形の条件分岐）。システムはこれに応じて、経路変化の原因および以前の経路情報を収容した、`AVAudioSessionRouteChangeNotification`通知を配送します。アプリケーションはこれに応じて、録音を停止しなければなりません。

再生の場合も同様の流れですが、図の右側に示すように結果は異なります。再生中にヘッドセットが抜かれた場合、再生を一時停止します。一方、ユーザがヘッドセットを接続した場合は、再生をそのまま続行することになります。

サンプルコードプロジェクト『*AddMusic*』に、このフローチャートのうち再生に当たる部分の実装例があります。

音声信号の経路変化に対処する

音声信号の経路変化に対処するためには、次の2つを実装する必要があります。

1. 経路が変化したときに呼び出されるメソッドを実装する。
2. 経路変化時に`AVAudioSessionRouteChangeNotification`通知を配送するよう登録する。

たとえば、再生中にユーザがヘッドセットを抜くと、アプリケーションにこの通知が配送されます。Appleのガイドラインに従い、一時停止しなければなりません。次いで、再生を続けるかどうか、問い合わせを表示します。

システムは、経路変化の通知を配送する際、取るべきアクションの判断に必要な情報も提供します。`AVAudioSessionRouteChangeNotification`通知を配送するよう登録するコード例を以下に示します。

```
NSNotificationCenter *nc [NSNotificationCenter defaultCenter];
[nc addObserver:self
       selector:routeChanged:
           name:AVAudioSessionRouteChangeNotification
         object:nil];
```

通知が届くと、アプリケーションは適切なメソッドを呼び出して、通知に収容されている情報にもとづき挙動を変更します。`AVAudioSessionRouteChangeNotification`には`userInfo`辞書の形で、次の事項が収容されています。

- 経路変化の原因
- 以前の経路

辞書のキーはそれぞれ、`AVAudioSessionRouteChangeReasonKey`と`AVAudioSessionRouteChangePreviousRouteKey`です。経路変化の原因は`AVAudioSessionRouteChangeReason`列挙型の値で表され、`AVAudioSessionRouteChangeReasonKey`キーで取得できます。以前の経路は、音声経路を記述したオブジェクトの形で表され、`AVAudioSessionRouteChangePreviousRouteKey`キーで取得できます。

iOSでは、経路変化が起こる原因のひとつとして、`AVAudioSessionRouteChangeReasonCategoryChange`があります。言い替えれば、オーディオセッションのカテゴリを変更すると、システムは経路変化と看做し、その通知を配送するようになっているのです。したがって、通知に応答するメソッドが、ヘッドセットの抜き差しにのみ対処することにする場合、それ以外の原因による経路変化は明示的に無視しなければなりません。

実際的な例を示しましょう。録音／再生アプリケーションでは、録音や再生の開始直前にオーディオセッションのカテゴリを指定するのが正しい作法です。そのため、（それまで録音していた場合に）再生を開始し、あるいは（それまで再生していた場合に）録音を開始すると、経路変化通知が配送されます。しかし、録音ボタンや再生ボタンがタップされるたびに一時停止したり停止したりするのでは、使い勝手の面で問題があります。このような一時停止や停止を避けるため、通知に応じて呼び出されるメソッドは、経路変化の原因を調べ、カテゴリの変更であった場合は何もせずに制御を戻さなければなりません。この場合、カテゴリの変化はシステム側の視点に立ったものであって、アプリケーションにとってそれほど重要なものではないのです。

プレーヤーを想定したオーディオセッションの微調整

ムービープレーヤーは、ファイルやネットワーク経由で動画を再生できます。ミュージックプレーヤーを使うと、ユーザの音楽ライブラリにあるコンテンツを再生できます。こういった動画や音楽を、アプリケーションの音声と連動する形で再生するためには、各プレーヤーのオーディオセッションの特性を考慮する必要があります。

- ミュージックプレーヤー（`MPMusicPlayerController`クラスのインスタンス）は、常にシステムが提供するオーディオセッションを使う
- ムービープレーヤー（`MPMoviePlayerController`クラスのインスタンス）は、通常はアプリケーション側のオーディオセッションを使うが、システムが提供するオーディオセッションを使うよう設定することもできる

ミュージックプレーヤーと連動する

音楽ライブラリのコンテンツを再生しつつ、独自の音声も再生するためには（『*iPod Library Access Programming Guide*』を参照）、`AVAudioSessionCategoryAmbient`カテゴリ（*音声合成が可能*なカテゴリ）をオーディオセッションに指定するか、それ以外のカテゴリであれば`AVAudioSessionCategoryOptionMixWithOthers`オプションを設定する必要があります。このカテゴリであれば、アプリケーション側の音声がミュージックプレーヤーに割り込む（中断させる）ことも、その逆もありません。

> **Important:** アプリケーション側のオーディオセッションに対して合成可能なカテゴリを指定しないまま、ミュージックプレーヤーを使うことは避けてください。

システムは、ミュージックプレーヤーに対する経路変化や割り込みを、自動的に処理します。この組み込みの動作を変えることはできません。アプリケーション側のオーディオセッションを、これまでに説明したとおり適切に管理しているならば、ユーザがヘッドセットを接続した、アラームが鳴った、電話がかかってきた、などといったイベントへの対応は、ミュージックプレーヤーに任せても構いません。

アプリケーション側の音声再生時に、ミュージックプレーヤー側の音量を下げるよう、オーディオセッションを設定することができます。音量を下げる機能や、これを有効にする方法については、“カテゴリの振る舞いに手を入れる”（21 ページ）を参照してください。

ミュージックプレーヤークラスの詳細については、『*MPMusicPlayerController Class Reference*』を参照してください。

ムービープレーヤーと連動する

ムービープレーヤーは通常、アプリケーション側のオーディオセッションを共有します。その結果、ムービープレーヤーは、アプリケーション側の音声と合成するどころか、動画に付随する音声がアプリケーション側に属しているかのように振る舞います。カテゴリの指定やその具体的な設定方法にかかわらず、アプリケーション側の音声がムービープレーヤー側に割り込む（中断させる）ことも、その逆もありません。

オーディオセッションを共有することにより、「ミュージック(Music)」など他のアプリケーションの音声と、動画がやり取りする方法も、アプリケーション側で制御できるようになっています。たとえば、カテゴリを`AVAudioSessionCategoryAmbient`と指定してセッションを共有した場合、アプリケーション側で動画再生を開始しても、「ミュージック(Music)」側に割り込む（中断する）ことはありません。さらに、動画に付随する音声が「着信／サイレント」スイッチの状態に影響されるかどうかも、アプリケーション側で指定できます。

動画に付随する音声まわりの動作を設定するには、必要な動作を決め、それに応じてオーディオセッションの設定をおこないます（表 6-1を参照）。オーディオセッションの設定について詳しくは、“オーディオセッションの定義” （9 ページ）を参照してください。

表 6-1　ムービープレーヤー使用時のオーディオセッションの設定

必要な動作	オーディオセッションの設定
動画を再生する際、他の音声をすべて無音にする	• アプリケーション側で音声を再生しない場合、オーディオセッションは設定しない。 • アプリケーション側で音声を再生する場合、「ミュージック(Music)」など他の音声との合成が必要かどうかに応じて、合成可能なカテゴリ、またはそうでないカテゴリを、オーディオセッションに指定する。 • いずれの場合も、ムービープレーヤーに対し、独自のオーディオセッションを使うよう指定する： `myMoviePlayer.useappAudioSession = NO`

必要な動作	オーディオセッションの設定
動画に付随する音声と、アプリケーション側の音声を合成する（「ミュージック(Music)」など他の音声は無音にする）	• *合成不可*のカテゴリをオーディオセッションに指定する。 • ムービープレーヤーの`useApplicationAudioSession`の値として、`YES`（既定値）を指定する。
音声をすべて合成する	• *合成可能*なカテゴリをオーディオセッションに指定する。 • ムービープレーヤーの`useApplicationAudioSession`の値として、`YES`（既定値）を指定する。

アプリケーション側のオーディオセッションでは、通常どおり経路変化と割り込みの取り扱いを管理してください（"信号経路の変更に対する応答"（34 ページ）および"割り込みへの応答"（23 ページ）を参照）。必要ならば音量を下げる機能を有効にします（"カテゴリの振る舞いに手を入れる"（21 ページ）を参照）。

ムービープレーヤー側で独自のオーディオセッションを使う場合は、いくつか後処理が必要です。動画の再生終了後、あるいはユーザが動画を停止した後、音声を再生可能な状態に戻すには、次の処理が必要です。

1. ムービープレーヤーを破棄する（同じムービーを後で再び再生する予定がある場合でも）。
2. アプリケーション側のオーディオセッションを作動し直す。

ムービープレーヤークラスの解説については、『*MPMoviePlayerController Class Reference*』を参照してください。

Media Playerフレームワークのみを使用する

アプリケーションがムービープレーヤーのみ、またはミュージックプレーヤーのみを使い、なおかつアプリケーション独自の音声を再生することがない場合は、オーディオセッションを設定しないでください。

ムービープレーヤーのみを使用する場合は、次のコード例のように、ムービープレーヤー自身のオーディオセッションを使うように伝える必要があります。

```
myMoviePlayer.useappAudioSession = NO
```

ムービープレーヤーとミュージックプレーヤーの*両方*を使っている場合は、おそらくこの2つの相互作用の方法を設定するのが望ましいでしょう。 そのため、アプリケーション側では音声を再生しない場合でも、オーディオセッションを設定する必要があります。表 6-1 （38 ページ）に示したガイダンスを参考にしてください。

アプリケーションの種類に応じた音声の取り扱いガイドライン

最新のレーシングゲームは、音声まわりの要件が、実時間ビデオチャットとは違っています。以下の各節では、音声を扱うアプリケーションの種類ごとに、設計上のガイドラインを示します。

ゲームアプリケーションに関する音声まわりのガイドライン

多くのゲームでは、進行に応じて発生する事象ごとに、ユーザとのやり取りが発生します。ゲームの設計においては、カテゴリとして`AVAudioSessionCategoryAmbient`または`AVAudioSessionCategorySoloAmbient`を指定してください。ユーザは、他のアプリケーションが前面に来たり、画面がロックされたりしても、ゲーム音声の再生が続くとは期待しません。逆に他のアプリケーションの音声は、多くの場合、ゲーム中でも再生される方が望ましいでしょう。

Appleは次のガイドラインに従うよう推奨しています。

- ゲームの効果音を再生しつつ、他のアプリケーションの音声もそのまま再生する。
- ほかに再生されている音声がなければゲームのサウンドトラックを流すが、ある場合はそのまま再生し続ける。
- 割り込み終了イベントの後は、できるだけ割り込み前の箇所から再生を再開する。
- 経路変更は無視する（アプリケーション側に、特に必要がある場合を除く）。
- アプリケーション起動時、ビデオスプラッシュを表示する前に、音声カテゴリを設定する。

ユーザ制御の再生／録画アプリケーションに関する音声まわりのガイドライン

ビデオ録画アプリケーションと、PandoraやNetflixなどのアプリケーションには、同じガイドラインを適用します。こういったアプリケーションの場合、カテゴリとして`AVAudioSessionCategoryRecord`、`AVAudioSessionCategoryPlayAndRecord`、`AVAudioSessionCategoryPlayback`のいずれかを指定し、他のアプリケーションの音声と合成できるようにしてください。通常、他のアプリケーションが再生している音声の音量を下げることはありません。「再生／一時停止」または「録音／一時停止」のボタンを用意してください。

Appleは次のガイドラインに従うよう推奨しています。

- フォアグラウンド状態になっても、ユーザが再生ボタンや録音ボタンを押すまでは、オーディオセッションを作動しない。
- アプリケーション動作中は、割り込み時を除き、オーディオセッションを作動させたままにする。
- 割り込み処理中は、音声が一時停止状態であることが分かるよう画面を更新する。オーディオセッションを停止したり、プレーヤーオブジェクトを一時停止／停止したりしない。
- 割り込み終了時には、キー`AVAudioSessionInterruptionOptionKey`で表される定数値の有無を確認し、その値に従って処理する。音声再生を再開しない（割り込み前に再生していた場合を除く）。
- プラグが抜かれたために経路変更が生じた場合、オーディオセッションを一時停止するが、なお作動状態を維持する。
- 保留状態からフォアグラウンド状態に遷移したときは、オーディオセッションが停止しているものと想定して処理する。ユーザが「再生」ボタンを押したらオーディオセッションを作動し直す。
- `UIBackgroundModes`フラグを立てる。
- カテゴリとして、`AVAudioSessionCategoryRecord`ではなく`AVAudioSessionCategoryPlayAndRecord`を指定する。
- 無音をストリーミング再生する代わりに、バックグラウンドタスクを使うことにより、アプリケーションが保留状態になることを避ける。
- 音量調整スライダや経路ピッカーには`MPVolumeView`オブジェクトを使う。
- 録音の際には`requestRecordPermission:`メソッドでユーザに許可を求める。必要ならばiOSが自動的に問い合わせ画面を出すが、この機能に依存しない。
- 遠隔制御イベントを配送するよう登録する。

VoIP／チャットアプリケーションに関する音声まわりのガイドライン

VoIP／チャットアプリケーションには、入力経路、出力経路のどちらも必要です。カテゴリとして`AVAudioSessionCategoryPlayAndRecord`を指定し、他のアプリケーションの音声を合成しないでください。

Appleは次のガイドラインに従うよう推奨しています。

- ユーザが着信に応じたか、または発信したときにのみ、オーディオセッションを作動する。
- 割り込み通知が届いたら、通話音声に割り込まれたことが分かるよう画面を更新する。

- 割り込み後は、ユーザが着信に応じるかまたは発信するまで、オーディオセッションを作動しない。
- 通話終了後、オーディオセッションを停止する際には、オプションとして`AVAudioSessionSetActiveOptionNotifyOthersOnDeactivation`定数を指定する。
- 経路変更は無視する（アプリケーション側に、特に必要がある場合を除く）。
- `UIBackgroundModes`フラグを立てる。
- 音量調整スライダや経路ピッカーには`MPVolumeView`オブジェクトを使う。
- 録音の際には`requestRecordPermission:`メソッドでユーザに許可を求める。必要ならばiOSが自動的に問い合わせ画面を出すが、この機能に依存しない。

測定アプリケーションに関する音声まわりのガイドライン

測定アプリケーションは、入出力経路に適用するシステム組み込みの信号処理を、最小限にとどめなければなりません。そのため、カテゴリとして`AVAudioSessionCategoryPlayAndRecord`を指定し、さらに測定モードにしてください。また、他のアプリケーションの音声を合成しません。

Appleは次のガイドラインに従うよう推奨しています。

- 割り込み終了イベントの後は、できるだけ割り込み前の箇所から再生を再開する。
- 経路変更は無視する（アプリケーション側に、特に必要がある場合を除く）。
- アプリケーション起動時、ビデオスプラッシュを表示する前に、音声カテゴリを設定する。
- 録音の際には`requestRecordPermission:`メソッドでユーザに許可を求める。必要ならばiOSが自動的に問い合わせ画面を出すが、この機能に依存しない。

ブラウザ風のアプリケーション（音声の再生あり）に関する音声まわりのガイドライン

FacebookやInstagramなどのアプリケーションは、録音機能は不要で、音声や動画を再生するだけです。カテゴリとして`AVAudioSessionCategoryPlayback`を指定し、着信スイッチの切り替えには従わないでください。また、他のアプリケーションの音声を合成しません。

Appleは次のガイドラインに従うよう推奨しています。

- ユーザが明示的に操作しない限り再生しない。
- 動画再生が終わったらオーディオセッションを停止し、`AVAudioSessionSetActiveFlags_NotifyOthersOnDeactivation`キーを設定する。

- プラグが抜かれたために経路変更が生じた場合、オーディオセッションを一時停止するが、なお作動状態を維持する。
- 動画再生中は遠隔制御イベントを配送するよう登録し、再生終了後は解除する。
- 割り込み開始イベントが届いたらUIを更新する。
- 割り込み終了イベントが届いても、ユーザが明示的に操作しない限り再生を始めない。

ナビゲーション／トレーニングアプリケーションに関する音声まわりのガイドライン

ナビゲーション／トレーニングアプリケーションの場合、カテゴリとして`AVAudioSessionCategoryPlayback`または`AVAudioSessionCategoryPlayAndRecord`を指定します。こういったアプリケーションの音声は短い指示（プロンプト）が多いので、他のアプリケーションの音声と合成しても問題ありません。他のアプリケーションが再生中であっても、こういった指示の音声は聞き逃したくないはずなので、他のアプリケーション側の音量を下げてください。

Appleは次のガイドラインに従うよう推奨しています。

- 指示（プロンプト）を再生する必要が生じるまで、オーディオセッションを作動しない。
- 指示を再生した後はオーディオセッションを停止する。
- 割り込みや経路変更はすべて無視する。

協調型音楽制作アプリケーションに関する音声まわりのガイドライン

協調型音楽制作アプリケーションは、他のアプリケーションで再生しながら、同時に再生するよう設計されています。カテゴリとして`AVAudioSessionCategoryPlayback`または`AVAudioSessionCategoryPlayAndRecord`を指定し、他のアプリケーションの音声と合成して出力します。

Appleは次のガイドラインに従うよう推奨しています。

- 再生ボタンや停止ボタンがない場合は、ゲームアプリケーションのガイドラインにも従う。
- 再生ボタンや停止ボタンがある場合は、ユーザが再生ボタンを押したときにのみ、オーディオセッションを作動する。
- 遠隔制御イベントにサインアップしない。
- `UIBackgroundModes`フラグを立てる。

- 録音の際には`requestRecordPermission:`メソッドでユーザに許可を求める。必要ならばiOSが自動的に問い合わせ画面を出すが、この機能に依存しない。

オーディオセッションのカテゴリとモード

オーディオセッションのカテゴリを指定することによって、音声の取り扱い方法をiOSに指示することができます。表 B-1に、各カテゴリの詳細を示します。影付きの行は、カテゴリの既定値である`AVAudioSessionCategorySoloAmbient`を表します。カテゴリが機能するしくみについては"カテゴリの取り扱い"（16 ページ）を参照してください。音声合成に関する振る舞いを変更するスイッチについては、"最適なカテゴリを選択する"（16 ページ）を参照してください。

表 B-1　オーディオセッションに指定できる各カテゴリの動作

カテゴリ識別子	「着信／サイレント」スイッチや画面ロックに応じて消音 注を参照	合成不可の音声に割り込み	音声入力（録音）や音声出力（再生）を許可
`AVAudioSessionCategoryAmbient`	○	×	出力のみ可
`AVAudioSessionCategoryAudio-Processing`	–	○	入力、出力とも不可
`AVAudioSessionCategoryMultiRoute`	×	○	入力、出力とも可
`AVAudioSessionCategoryPlayAnd-Record`	×	既定値は○、スイッチで変更すれば×	入力、出力とも可
`AVAudioSessionCategoryPlayback`	×	既定値は○、スイッチで変更すれば×	出力のみ可
`AVAudioSessionCategoryRecord`	×（画面がロックした状態でも録音は継続）	○	入力のみ

カテゴリ識別子	「着信／サイレント」スイッチや画面ロックに応じて消音 注を参照	合成不可の音声に割り込み	音声入力（録音）や音声出力（再生）を許可
AVAudioSessionCategorySoloAmbient	○	○	出力のみ可

注意: 「着信／サイレント」スイッチが「サイレント」側になり、かつ画面がロックされても再生を続行するためには、UIBackgroundModesというaudioキーを、アプリケーションのInfo.plistファイルに追加する必要があります。これは、適切なカテゴリを指定した上で、さらに必要になる要件です。

表 B-2に、モードの一覧と、それぞれがどのカテゴリと組み合わせられるか、を示します。

表 B-2 モードと組み合わせ可能なカテゴリ

モードの識別子	組み合わせ可能なカテゴリ
AVAudioSessionModeDefault	すべてのカテゴリ
AVAudioSessionModeVoiceChat	AVAudioSessionCategoryPlayAndRecord
AVAudioSessionModeGameChat	AVAudioSessionCategoryPlayAndRecord
AVAudioSessionModeVideoRecording	AVAudioSessionCategoryPlayAndRecord AVAudioSessionCategoryRecord
AVAudioSessionModeMoviePlayback	AVAudioSessionCategoryPlayback
AVAudioSessionModeMeasurement	AVAudioSessionCategoryPlayAndRecord AVAudioSessionCategoryRecord AVAudioSessionCategoryPlayback
AVAudioSessionModeVideoChat	AVAudioSessionCategoryPlayAndRecord

書類の改訂履歴

この表は「*Audio Session プログラミングガイド*」の改訂履歴です。

日付	メモ
2014-09-17	AVAudioSessionCategoryMultiRouteがAVAudioSessionCategoryOptionMixWithOthersで修正できる旨の情報を追加しました。
2014-03-10	iOS 7に合わせて改訂しました。Cの参照をすべて削除しました。
2012-12-13	オーディオ処理グラフ使用中の割り込みの処理方法に関する情報を追加しました。
2010-11-15	playInputClickメソッドの説明に、オーディオセッションに関する検討を追加しました。
2010-09-01	iOS 4でアプリケーションのライフサイクルをサポートする方法の説明を改訂しました。
2010-07-09	細かな変更を行いました。
2010-04-12	iOS 3.2向けに、ムービープレーヤーのオーディオセッションに関する変更点を記述して更新しました。
2010-01-29	未定
2010-01-20	ハードウェア設定の使い方、画面ロックの処理方法に関する説明を追加しました。

日付	メモ
2009-10-19	未定
2009-09-09	iOS 3.1向けに更新して、新しいオーディオセッションプロパティの説明を追加しました。
2008-11-13	iOSアプリケーションのオーディオセッションの使いかたを説明する、iOS 2.2向け新規文書。

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
U.S.A.

Apple Japan
〒106-6140 東京都港区六本木
6丁目10番1号 六本木ヒルズ
http://www.apple.com/jp/

Offline copy. Trademarks go here.