Unity サウンドクラスライブラリ

USnd ver2.20.0

最終更新日: 2021/10/07

■目次

※デザイナー向けのチュートリアルは別文書です。

■更新履歴

2015/04/02 ver1.0.0	リリース
2015/06/12 ver1.1.0	USndPlugin.IsSpeaker()の Android 対応、USndPlugin.IsMannerMode()を追加
2015/06/18 ver1.2.0	Android のマナーモード対応を追加
2015/08/21 ver1.2.0	IsManngerMode の説明が間違っていた箇所を修正
2015/09/03 ver1.2.1	追加した GameObject が AudioManaegr の子に設定されるように修正
2015/09/16 ver1.2.2	AudioPlayer を使っていないときは非アクティブにするよう修正
2015/09/17 ver1.3.0	下記 API を追加、ドキュメントに API の説明と AudioSource 読み込みに関する説明を修正・追加。 GetAudioClipNameLoadId GetAudioClipNameAll GetAudioClipName SetAudioClipToLabelLoadId SetAudioClipToLabelAll SetAudioClipToLabelAll
2015/10/01 ver1.3.1	USndPlugin Android の IsMannerMode と IsSpeaker の実装を Java に変更
2015/10/04 ver1.4.0	AudioInstance と AudioSource のオブジェクトプーリング処理を追加、それに伴いプールをクリアする API を追加とドキュメントに説明を記載 ClearInstancePool ClearAudioSourcePool
2015/10/23 ver1.4.1	Android のボリューム変更ボタンを押して右側の歯車アイコンを押してマナーモード解除したとき、アプリを Suspend,Resume させないと変更が効かなかったのを修正。
2015/10/27 ver1.4.2	Android のマナーモード対応を有効がデフォルトから、無効がデフォルトの状態に変更。
2015/11/11 ver1.5.0	GetLabelVolume、GetInstanceCalcVolume を追加 サウンドデバッグツールの説明を追加
2015/11/12 ver1.5.1	実機用ビルドをした際のエラー、警告を修正。 Android でアプリを起動した際に、組み込みの音楽プレイヤーを強制停止させる処理を追加、実行するかのフラグはデフォルトで false。
2015/11/19 ver1.5.2	キャッシュ有効時、既に再生済みのインスタンスがあったときに XML 読み込みで更新すると、キャシュには更新が反映されないため想定と違う値で再生されることがあったのを修正。
2015/11/27 ver1.6.0	XML から読み込んだラベルを登録するときに AudioSource を生成せず、再生時にオブジェクトプールから AudioSource を取得するように変更。これに伴い AudioLabelSettings の IsCache は Prefab 作成時など AudioSource を直接登録するときのみ有効に挙動を変更。また、オブジェクトプールをクリアする API を追加。 ClearXMLAudioSourcePool ClearObjectPool
2015/12/03 ver1.6.1	AudioClip が読み込まれていない状態で XML を先に読み込むとエラーになっていたのを修正。
2015/12/07 ver1.7.0	ラベルに関連する AudioSource と AudioClip をまとめて削除するための API を追加 RemoveLabel

	RemoveLabelLoadId RemoveLabelAll 1.6.0 で追加したオブジェクトプールの処理の影響でランダム再生が正しく動作しなくなっていたのを修正。 ダッキング設定のあるラベル再生中に、同カテゴリをダッキングするラベルが再生されると音量が復帰しなかった不具合を修正。 ダッキング設定のあるラベルが複数再生中に、ダッキングから復帰するタイミングが正しくなかったのを修正。
2015/12/15 ver1.7.1	XML から読み込んだときにミキサーが設定されていなかったのを修正。 Android の OnApplicationFocus でフォーカス ON/OFF 時にポーズ設定/解除していたのを自動で行わないように変更。
2016/01/13 ver2.0.0	・DLL 版を追加 ・AudioManager の実装をシングルトンに変更 ・AudioSource の取り扱いの変更、それに伴い使用方法などを大幅に記載変更 ・バイナリ形式のサウンドテーブルのサポートを追加 ・ファイルの階層を変更 ・API の追加・名称変更・削除 AddMasterSettings → 廃止 AddCategorySettings → 廃止 AddAudioSource → 廃止 FindAudioMaster → FindMaster に変更 FindAudioCategory → FindCategory に変更 FindAudioSource → FindLabel に変更 RemoveAudioSource → 廃止、RemoveLabel を使用 RemoveAudioSourceLoadId → 廃止、RemoveLabelLoadId を使用 RemoveAudioSourceAll → 廃止、RemoveLabelAll を使用 ClearInstancePool → 廃止 ClearAudioSourcePool → 廃止 ClearXMLAudioSourcePool → 廃止 LoadBinaryTable を追加 GetAudioSourceNum → GetLabelNum に変更 GetAudioSourceNameList → GetLabelNameList に変更
2016/01/25 ver2.0.0b	・ゲームにデバッグとして組み込むための USndViewer パッケージの説明を追加
2016/01/29 ver2.1.0	・3D サウンド機能を実装、3D サウンド用の API を追加 Play3D SetTrackingObject ・ver2.0.0 で含まれていた不要なソース AudioSourcePool.cs を削除 ・ソースコード Audio3DSettings.cs を追加 ・3D サウンドについて説明を追加 ・トラブルシューティングを更新 ・PlayInstance 時にテーブル指定のディレイが実行されなかったのを修正

2016/02/01 ver2.2.0	・AudioMixerSettings を登録していたところを、UnityEngine.Audio.AudioMixer を登録するように変更 [削除]SetUnityMixerInfo → [new]SetAudioMixer [削除]UnsetUnityMixerInfo → [new]UnsetAudioMixer [変更]SetSnapshot → 第 1 引数が int のものを削除、string のみになります [廃止]GetSnapshotNum [廃止]GetSnapshotNameList ・Audio3DSettings が napespace USnd 内になっていなかったのを修正
2016/02/08 ver2.3.0	・USndViewer の説明に Snapshot 確認画面使用時の設定方法を追加 ・AudioMixer の ExposedParameters 機能を使用するための API を追加 SetAudioMixerExposedParam ・IsPlayingLabel の詳細が記載されていなかったので追加 ・Preload Audio Data が無効なとき再生終了後に自動で Unload するように処理を追加 ・LoadType が Streaming のときの注意点を追加
2016/02/17 ver2.4.0	・RemoveLabel が内部で AudioSource の終了処理が終わっていないうちに発生すると AudioSource を Active のままロックしてしまっていたのを修正 ・RemoveLabel の戻り値を bool に変更 ・RemoveLabel が成功するか調べる API を追加 CanRemoveLabel
2016/03/02 ver2.4.1	・フェードアウトが設定されているラベルを Delay 設定ありの Play を行ったとき Delay 完了前に Stop が実行されると Delay 秒後に再生を開始し同時にフェードアウト停止が実行されていたのを修正、Delay 秒経過前は即停止するように変更 ・Android のオーディオフォーカス有効時に onPause で再生を止めていたのを止めないように修正（キーボードなどの割り込みで再生がとまってしまうため）
2016/03/29 ver2.4.2	・ver2.4.1 で変更した Stop 処理が再生中のデータでも Delay 完了前と判定されることがあったので判定方法を変更 ・AudioPlayer.cpp の Update で行なっている UnloadAudioData の前に clip が null か判定を追加
2016/04/27 ver2.5.0	・ラベルをアンロードせずに AudioClip の参照をはずせるよう API を追加 UnsetAudioClipToLabel UnsetAudioClipToLabelLoadId UnsetAudioClipToLabelAll ・USndPlugin の各プラットフォーム依存部分を、各プラットフォーム環境以外ではビルド対象からはずすように変更 ・Android のオーディオフォーカスがポーズから復帰したときに再取得されていなかったのを修正
2016/06/15 ver2.5.1	・ContainsKey を使用していた箇所を一部 TryGetValue に修正
2016/10/13 ver2.6.0	・Terminate と Initialize を同じフレーム内で行なうと正しく動作しない症状の対応を追加 ・Android で SoundPool Java API を使用して再生する設定を追加

	・フェード制御の秒数で誤差が大きすぎるので、時間取得方法を Time.deltaTime から Time.time に変更
2016/10/25 ver2.6.1	・iOS の libsnd.a を Xcode8 でビルドしなおし（ソースコードは変更なし） ・[USndViewer]NameList 更新を実行したときに自動的に Reimport を行なうように処理を追加 ・[USndViewer]UI の表示不具合の修正 ・再生途中から再開するフラグが立っているラベルが混在していると seek error が出ることがあるので再生前に開始位置をリセットするように修正
2016/11/24 ver2.6.2	・3D サウンドのターゲットにするオブジェクトの参照を外す処理を一部追加 ・Audio3DSettings のアセット読み込みについてドキュメントへ追記
2016/11/29 ver2.6.3	・停止済みデータの PreloadAudioData パラメータが false のとき、自動的に UnloadAudioData を行なっていたが別ラベルで同一ファイルを参照しているときに再生中のデータが停止してしまうので自動では行わないように修正
2016/12/06 ver2.7.0	・Play3D、SetTrackingObject に引数が Transform のものを追加 ・3D 設定があるラベルを強制的に 2D 再生する Play2D を追加 ・フェード処理が Time.timeScale に影響して速度が変わっていたので unscaledTime を使用するように修正 ・moveTime を 0 にして SetVolume を行なうと値が反映されていなかったのを修正 ・ラベルに指定しているオーディオファイルの総再生秒数を取得する GetLabelLength, 総サンプル数を取得する GetLabelSamples を追加 ・再生を行ってから次に再生可能になるまでの間隔を設定する Interval パラメータを追加(テーブルで設定)
2017/01/26 ver2.7.1	・ランダム関連の min, max パラメータが同値だったとき止まらずに動くように修正（Debug 時は警告を表示） ・ピッチの計算を省略し、値をテーブルで持つように変更 ・処理の見直し ・ダッキングスタート秒数が 0 のときに音量が復帰しなくなるのを修正
2017/02/10 ver2.8.0	・SetTime, SetTimeSamples を追加
2017/04/04 ver2.9.0	・GetSpectrumData を追加 ・AudioDebugLog を USND_DEBUG_LOG が定義されているときのみ完全に有効になるように修正 ・usndplugin.jar を AndroidStudio で生成
2017/09/14 ver2.10.0	・JSON 形式で Master, Category, Label 情報と Audio3DSettings を読み込む機能を追加 ・LoadJson ・GetLoadJsonStatus ・SetAudio3DSettingsFromJson ・JSON 形式で初期化を行う方法の説明を追加 ・SetPosition で指定した値が反映されていない、反映タイミングが遅いのを修正 ・IsExistAudioClip を追加 ・SetAudioClip を行っていない状態で再生したとき Play 前に AudioClip を検索して自動的に紐付けを行うよう変更

2017/11/10 ver2.11.0	・Initialize の引数にデフォルトサンプルレートを指定できるように変更
2017/11/14 ver2.11.1	・ダッキング開始が 0 秒に指定されているとき、1 秒と同じ扱いになっていたのを正しく動作するように修正。
2017/12/27 ver2.12.0	・パラメータエディタで 3D サウンドパラメータの調整が行えるように機能を追加
2018/01/25 ver2.12.1	・SetCategoryVolume,SetMasterVolume の moveTime に 0 秒を指定してもパラメータが即時反映されなかったのを修正 ・SetCategoryVolume, SetMasterVolume の moveTime のデフォルト引数を-1 から 0 に変更
2018/02/22 ver2.13.0	・GetAudioClipNames を追加
2018/03/09 ver2.14.0	・GetRandomSourceNames を追加
2018/05/02 ver2.15.0	・ForceResetDucking, ForceResetDuckingAll を追加 ・ダッキング設定されているラベルが AudioManager の Update 実行前にラベル削除されていたときにダッキング復帰が正常に動作しなくなるのを修正
2018/12/25 ver2.15.1	・AudioClip の設定を一括変更する機能を追加 ・コール履歴を保存・出力する機能を追加 （両機能ともデザイナ向け機能なので詳細は USnd Designer Tutorial に記載） ・float.Parse の第 2 引数を System.Globalization.CultureInfo.InvariantCulture に未指定から変更。
2019/04/17 ver2.16.0	・ IsLoop 、 GetLabelMaxPlaybacksNum 、 GetCategoryMaxPlaybacksNum 、 GetCategoryMaxPlaybacksNumFromLabel を追加 ・加藤さんよりいただいたパフォーマンス改善のコードをマージ （重複して Update されていた箇所の修正と、3D サウンドの transform のキャッシュ方法を修正） ・USndEditor のパラメータコピーが正しく動かなくなっていたのを修正
2019/05/31 ver2.16.1	・USnd Designer Tutorial に AudioMixer に関する TIPS を追加 ・USnd ツールがアクティブな状態で再生失敗したとき、ラベル情報の設定でエラーになるのを修正 ・USndTools.xls のデバッグログに全角チェック、一部項目の出現チェックを追加 ・USndTool のレイアウト調整
2019/10/29 ver2.17.0	・OffPauseCategory に指定したインスタンスを除いて処理をする List<int>を引数にとるバージョンを追加 ・指定したラベルはインターバル中か調べる IsInterval を追加 ・targetTransform のリセットが抜けていたのを修正 ・USndTool の表示を調整、表示順の設定を追加
2020/03/17 ver2.18.0	・現在発音中の総数を取得する GetCurrentPlayNum を追加 ・OffPauseCategory の除外するインスタンスを指定するバージョンで、検査するインスタンスリストのインデックスが間違っていたのを修正 ・再生に使用する API によって、エラーログが正しく出力されていなかったのを修正 ・ USND_OUTPUT_CALL_LOG が有効なときに AssetBundle を出力すると AddCallLog が定義されていないというエラーが出てしまうのを修正 ・カテゴリが大量にあるとき、ダッキングの処理で Update の処理負荷が大きいのを軽減する対策を追加

2020/05/18 ver2.18.1	・パラメータエディタで Unity Mixer と 3D Group の変更ができるよう Apply ボタンを追加
2020/05/20 ver2.18.2	・ver2.18.0 で修正したダッキングの Update 処理負荷軽減によりダッキング復帰時間が設定されているとダッキング復帰しなくなっていたのを修正
2020/08/21 ver2.18.3	・Android プラグインを更新（Exception の Log.v を削除、deprecated になっていた箇所の対応） ・GetLabelVolume が返す値が正しくなかったのを修正
2021/03/12 ver2.19.0	・tvOS 用の libusnd_tvos.a を追加 ・USndPlugin を無効にする機能を追加 ・ドキュメントに USndPlugin 説明書を追加
2021/10/07 ver2.20.0	・Play2D に PlayOption と同等の引数を渡せるバージョンを追加

■概要

USnd は Unity5 用サウンドクラスライブラリです。
Unity5 の純正 API を使用してオーディオの再生を行い、Unity に足りない機能を USnd 側で吸収し再生管理を行うクラスです。
一部サポート機能はプラグインで実装しています。
下記のような機能が使用できます。

ver2.19.0 時点で Unity2020.2.4f1 まで動作確認済みです。

サポートしている機能

・ラベル単位で発音数制御を行う
・ラベルを任意のグループにわけ、発音制御を行う
・フェードイン、フェードアウト
・最後に停止した位置から再生を再開する
・ピッチをセント単位で扱う
・ピッチを指定した時間で変更する
・パンを指定した時間で変更する
・ランダムボリューム
・ランダムピッチ
・ランダムパン
・ランダムに指定した AudioSource を再生する
・グループ化した単位でのダッキング
・グループ化した単位での音量変更
・ミュート
・AudioClip のロード/アンロード(Preload Audio Data 無効時)
・AudioSnapshot の設定切り替え

プラグインでサポートしている機能

・他のアプリケーションがオーディオ再生しているか取得する(iOS のみ)
・端末スピーカーから再生されているか取得する
・マナーモードか取得する
・Android のオーディオフォーカス取得(有効時のみ、デフォルトは無効)
・Android の SoundPool API で再生を行う

UnityEditor のみの機能

・コンパイル設定（サウンド関連の define 設定切替）
・USndTool(ロード済み情報の参照、再生インスタンスの表示、ログ表示)
・パラメータエディタ（マスター、カテゴリ、ラベルの設定を実行中に変更する）

Unity でサポートされている機能

・サステインループの読み込み(Windows 版 SoundForge の smpl チャンクを反映)
・AudioMixer によるボリューム管理
・AudioMixer によるダッキング
・AudioMixer によるエフェクト設定
etc…

■用語説明

USnd の用語

Label	再生する 1 オーディオを表します。ラベルには AudioClip 名、ボリューム、再生制御方法などの情報が含まれています。ラベルにランダム再生が設定されている場合は、ひとつのラベルから複数のオーディオが再生される場合があります。ランダム再生で指定したラベルが再生されるとき、対象のラベルのランダム設定は無視されますが、それ以外のパラメータは各ラベルの設定に従います。 ラベルは文字列で管理します。ラベル名は重複できません。
Instance	ラベルの再生命令を実行し、実際に発音されたオーディオをインスタンスと呼びます。インスタンス ID は 0 以外の数値（マイナスを含む）で表現されます。
Category	複数のラベルをまとめてボリューム設定したり、同じカテゴリに所属するラベルの現在の発音数で発音制御を行ったりするための単位です。ダッキングもカテゴリ単位で行います。
Master	複数のカテゴリをまとめてボリューム設定するための単位です。 ゲームのコンフィグでボリューム設定を行うときなどに使用します。
XML	USnd でラベル、カテゴリ、マスターのパラメータ更新を、外部ファイルを用いて行う際に使用するファイル形式です。
XSD	XML が正しいかチェックするためのファイルです。XSD はなくても動きます。 Unity でそのまま読み込めないので拡張子を bytes にする必要があります。
LoadID	まとめてロード/アンロードするための ID です。ラベルに設定できます。 ロード ID は UnityEditor で編集するか、XML 読み込み時にプログラムで指定します。
Ducking	一時的に指定したカテゴリの音量を変更する機能です。

■パッケージ内容

Scripts

USnd
- ├ Plugins　※Plugins フォルダの内容は下記
- ├ Editor　※Editor フォルダの内容は下記
- ├ Audio3DSettings.cs
- ├ AudioCategorySettings.cs
- ├ AudioDebugLog.cs
- ├ AudioDefine.cs
- ├ AudioInstance.cs
- ├ AudioInstancePool.cs
- ├ AudioLabelSettings.cs
- ├ AudioMainPool.cs
- ├ AudioManager.cs
- ├ AudioManagerImpl.cs
- ├ AudioMasterSettings.cs
- ├ AudioMixerSettings.cs
- ├ AudioParamUpdater.cs
- ├ AudioPlayer.cs
- └ AudioXmlLoad.cs

Plugins
- ├ USndAndroidNativePlayer.cs
- ├ USndPlugin.cs
- ├ USndPlugin_abstract.cs
- ├ USndPlugin_ios.cs
- ├ USndPlugin_android.cs
- ├ iOS
- |　└ libusnd.a(bitcode_no)
- └ Android
- 　└ usndplugin.jar

Editor
- ├ USndAudioAssetLoader.cs
- ├ USndCompileSettings.cs
- ├ USndOutputCallLog.cs
- ├ USndParamEditor.cs
- ├ USndTableLog.cs
- └ USndToolEditor.cs

DLL

Debug/Release
└ USnd
　└ Editor
　|　└ USndEditor.dll
　└ USnd.dll

iOS_Plugin_bitcode

bitcode_no
└ libusnd.a

bitcode_yes
└ libusnd.a

Scripts または DLL どちらかを使用してください。
DLL を使用する場合、拡張メニューの USnd->コンパイル設定は効かないので、デバッグ機能を ON にするときは Debug フォルダの DLL を使用してください。
ver2.0.0 時点では ENABLE_BIT_CODE が No になっているものがデフォルトで入っています。Yes にする際は iOS_Plugin_bitcode/bitcode_yes/libusnd.a で上書きしてください。

■サウンド組み込みについて

ver2からはサウンドテーブル(XMLまたはバイナリファイル)から必要情報を読み込み、USnd内でAudioSourceを生成・使いまわして使用する方法のみになりました。

サウンドテーブルは ver1 と同様に XML が扱えます。ver2 からは XML をバイナリ形式に変換したファイルが読み込めるようになりました。XML はテキストエディタで直接書き換えが可能、バイナリ形式はロード速度が XML より早いですが実行しないと内容が把握できないので、開発環境は XML、本番はバイナリなど必要に応じて都合の良い方を使用してください。

■デザイナーとプログラマーの作業の切り分けについて

デザイナーの作業

・AudioClip の登録、パラメータの設定
・XML の編集
・AudioMixer の編集
上記の作業はデザイナー側で行います。

プログラマーの作業

・スクリプトからのラベルのコール
・AudioMixer の Snapshot 切り替え
・AudioMixer を AudioManager へ登録
・他のアプリケーションがオーディオ再生中だったときの処理
etc…
上記以外にもプログラマーから制御できる項目が多数あります。

AudioSource をスクリプトから直接再生する、ゲームオブジェクトに登録してゲームオブジェクトの発生時に USnd を通さずにオートで再生するといったことは絶対にしないでください。再生するときは必ず USnd を通して再生を行ってください。
USnd に登録していない AudioSource は管理外の再生オブジェクトになります。
共存は可能ですがサウンドデザイナーが把握できなくなるので、どうしても必要な場合は声を掛けてから行ってください。

AssetBundle の生成・管理についてはプログラマーとデザイナー間で相談して決めてください。

AudioMixer も AssetBundle で管理可能です。(Unity5.3.1 で確認)

■プログラマー向け　導入手順

本項で一通り説明を記載していますが、処理順次第ではうまく動作しないことがあります。下記説明は初期化順にはなっていないので、初期化サンプルを参照して実装を行なってください。
iOS/tvOS/Android がターゲットの場合は、USndPlugin 説明書もご参照ください。

1) スクリプトを配置する

USnd フォルダを Assets 以下の任意の場所に配置してください。

2) オーディオデータを Unity に登録する

アプリ内蔵にする場合は Resources に配置してください。AssetBundle をローカルで試す場合は StreamingAssets に配置します。

3) USnd.AudioManager の初期化

ver2 から USnd.AudioManager の実装がシングルトンになりました。
初期化すると GameObject を生成、DontDestroyOnLoad を行います。
USnd のクラスはすべて Usnd 名前空間に定義されているので、必要に応じて using USnd;を追加してください。

```
AudioManager.Initialize();
```

4) マスター、カテゴリ設定を読み込んで登録する

ラベルはカテゴリを参照、カテゴリはマスターを参照しているので、読み込む順番はマスター、カテゴリ、ラベルの順番に行います。カテゴリの登録時にマスターが存在しなかった場合、マスターが正しく参照されずボリュームコントロールが正しく動作しなくなるので、ロード順に注意してください。

XML を使って読み込む(XML を Resources に配置した場合)

```
TextAsset ta;
Stream xsd;
Stream xml;

ta = Resources.Load("Table/MasterSettings_xsd") as TextAsset;
xsd = new MemoryStream(ta.bytes);
ta = Resources.Load("Table/MasterSettings") as TextAsset;
xml = new MemoryStream(ta.bytes);
AudioManager.LoadMasterXml(xml, xsd);
while(AudioManager.GetLoadXmlStatus() != AudioDefine.LOAD_XML_STATUS.FINISH) {
    yield return null;
}

ta = Resources.Load("Table/CategorySettings_xsd") as TextAsset;
xsd = new MemoryStream(ta.bytes);
ta = Resources.Load("Table/CategorySettings") as TextAsset;
```

```
xml = new MemoryStream(ta.bytes);
AudioManager.LoadCategoryXml(xml, xsd);
while(AudioManager.GetLoadXmlStatus() != AudioDefine.LOAD_XML_STATUS.FINISH) {
    yield return null;
}

ta = Resources.Load("Table/LabelSettings_xsd") as TextAsset;
xsd = new MemoryStream(ta.bytes);
ta = Resources.Load("Table/LabelSettings") as TextAsset;
xml = new MemoryStream(ta.bytes);
AudioManager.LoadLabelXml(0, xml, xsd);
while(AudioManager.GetLoadXmlStatus() != AudioDefine.LOAD_XML_STATUS.FINISH) {
    yield return null;
}
```

上記の例では XSD ファイルを読み込んでいますが、XSD は必須ではありません。
XML の型チェックをしたいときのみ XSD が必要になります。XSD は Unity で認識されない拡張子なので、拡張子は bytes に変更して配置します。
XML のロード中は GetLoadXmlStatus()でステータスを確認して順番にロード処理を行ってください。同時に複数の XML のロード処理は行えません。

名前が重複していた場合は上書きになります。
LoadLabelXml の第 1 引数に指定する数字はアンロード時に使用する任意の値です。

バイナリファイルを使って読み込む(Resources に配置した場合)

```
TextAsset ta = Resources.Load("Table/MasterSettings) as TextAsset;
AudioManager.LoadBinaryTable(ta);

ta = Resources.Load("Table/CategorySettings) as TextAsset;
AudioManager.LoadBinaryTable(ta);

ta = Resources.Load("Table/LabelSettings) as TextAsset;
AudioManager.LoadBinaryTable(ta, loadId);
```

バイナリ形式の拡張子は bytes です。バイナリ形式の場合は LoadBinaryTable でマスター、カテゴリ、ラベルすべてのバイナリファイルが読み込み可能です。第 2 引数は LoadLabelXml の第 1 引数と同じ、アンロード時などで使用するための ID です。バイナリファイルの内容がラベルだったときのみ有効です。
バイナリ形式はブロックして処理するので、XML のような待機時間はありません。

JSON 形式ファイルを使って読み込む(Resources に配置した場合)

```
TextAsset ta;
ta = Resources.Load("Table/MasterSettings") as TextAsset;
```

```
AudioManager.LoadJson(ta.text);
while (AudioManager.GetLoadJsonStatus() == AudioDefine.LOAD_JSON_STATUS.LOADING) {
    yield return null;
}

ta = Resources.Load("Table/CategorySettings") as TextAsset;
AudioManager.LoadJson(ta.text);
while (AudioManager.GetLoadJsonStatus() == AudioDefine.LOAD_JSON_STATUS.LOADING) {
    yield return null;
}

ta = Resources.Load("Table/LabelSettings") as TextAsset;
AudioManager.LoadJson(ta.text);
while (AudioManager.GetLoadJsonStatus() == AudioDefine.LOAD_JSON_STATUS.LOADING) {
    yield return null;
}
```

JSON 形式の拡張子は json です。JSON 形式は内部で Master, Category, Label を判別するの共通で LoadJson を使用します。第 2 引数は LoadLabelXml の第 1 引数、LoadBinaryTable の第 2 引数と同じです。
JSON 形式は非同期で読み込むので読み込み完了を待つ必要があります。同時に複数ファイルのロード処理は行わないでください。

5) AudioClip を登録

USnd で使用するオーディオファイルをロードして登録します。
下記は Resources からまとめて読み込んでいますが、アセットバンドルを使う、LoadAsync を使うなどして読み込んだ方が良いです。

```
AudioClip[] clip = Resources.LoadAll<AudioClip>("");
AudioManager.AddAudioClip(clip);
```

既にテーブルファイルを読み込んでいる場合は、読み込んだ AudioClip をラベルに関連付けするための API を呼びます。
ver2.10.0 で紐付けを行わずに再生を呼び出した場合、自動的に登録済みの AudioClip を検索してセットするよう処理を追加しました。再生遅延を減らしたい場合は、従来どおり先に SetAudioClip を行ってください。

```
AudioManager.SetAudioClipToLabelAll();
```

先に AudioClip を読み込んでいる場合は、ラベルのロード時に割り当てを行うので SetAudioClipToLabel は不要です。

ラベルに指定されている AudioClip 名を取得することができます。
GetAudioClipNameLoadId ・・・ XML 読み込み時の第 1 引数ロード ID を指定してリストを取得します。
GetAudioClipNameAll ・・・ 登録されているすべてのラベルの AudioClip 名リストを取得します。
GetAudioClipName ・・・ 指定したラベル名に設定されている AudioClip 名を取得します。

Get で取得する名前リストと同じ単位で AudioClip の割り当ては実行可能です。
SetAudioClipToLabelLoadId ・・・ XML 読み込み時の第 1 引数ロード ID を指定して割り当てを行います。
SetAudioCLipToLabelAll ・・・ すべてのラベルに対して AuidoClip の割り当てを行います。
SetAudioClipToLabel ・・・ 指定したラベルの AudioClip の割り当てを行います。

6) テーブルからパラメータを読み込んで情報を上書きする

登録したあとにパラメータの変更を行いたいときは LoadMasterXml(), LoadCategoryXml(), LoadLabelXml()、または LoadBinaryTable()を使用します。初回ロード時と同じ処理内容です。
既に存在する名前のとき、項目を上書きします。存在しない名前だったら新しく登録されます。

7) 再生する

再生したいラベル名を指定して再生を行います。

```
int  instanceId = AudioManager.Play("bgm");
```

第２引数に秒数を指定すると、指定した秒数分だけ遅延して再生されます。

8) ボリュームを変更する

ボリュームの設定はマスター単位、カテゴリ単位、インスタンス単位で設定できます。
第２引数は変更後のボリュームです。
第３引数に指定した秒数で、現在のボリュームから指定のボリュームへ推移します。

```
AudioManager.SetVolume(instanceId, 0);
AudioManager.SetCategoryVolume("BGM", 0);
AudioManager.SetMasterVolume( “BGM” , 0);
```

ボリューム値は 0～1 の間で、テーブルで指定された値に対する割合になっています。例えばテーブルの音量が 0.8 だったら、プログラム 1 を指定したときは音量 0.8、プログラムで 0.5 を指定したときは音量 0.4 になります。

9) 停止する

停止はインスタンス、ラベル単位、カテゴリ単位、マスター単位、全部の５種類があります。
第２引数に指定した秒数でフェードアウトして停止します。指定無し（マイナス値）の場合はテーブルの設定に従ってフェードアウトします。

```
AudioManager.Stop(instancdId);
AudioManager.StopLabel( “bgm” );
AudioManager.StopCategory( “BGM” );
AudioManager.StopMaster( “BGM” );
AudioManager.StopAll();
```

10) ラベルの登録を解除する

使い終わったラベルを削除したいときは RemoveAudioSource()を使います。

```
AudioManager.RemoveLabel( “bgm” );   // ラベルを削除
```

```
AudioManager.RemovelabelLoadId(0);  // ラベルロード時に設定されているロードIDで削除
AudioManager.RemoveLabelAll();  // 全AudioSourceを削除
```

Stop 後、AudioManager の Update が行なわれた後に Remove 可能な状態になります。Stop 直後に Remove を行なうと失敗する場合があります。Remove が成功するか確認するには CanRemoveLabel を使用します。
USnd から参照を削除しただけなので、完全にリソースを消すときは Resources.UnloadUnusedAssets()を呼び出します。

11) 全設定をリセットする

AudioSource だけでなく、カテゴリ、マスター情報も含めたすべての情報を削除します。

```
AudioManager.RemoveAll();  // すべての登録を解除
```

USnd から参照を削除しただけなので、完全にリソースを消すときは Resources.UnloadUnusedAssets()を呼び出します。

12) 他のオーディオ再生状況を取得して処理を行う

必須の処理ではありません。実装が必要かはデザイナーと相談してください。
iOS の組み込みのミュージックアプリの再生などとゲーム BGM を排他にしたいときの実装方法です。USnd は判定方法のみ提供しています。
アプリケーション切り替え時にステータスをチェックします。下記の例では BGM のみ音量を 0 にしました。

```
void OnApplicationpause(bool pause)
{
    if ( pause == false )
    {
        if ( USndPlugin.IsMusicPlaying())
        {
            AudioManager.SetMasterVolume( “BGM”, 0, 0.5);
        }
        else
        {
            AudioManager.SetMasterVolume( “BGM”, 1, 0.5);
        }
    }
}
```

iOS のみ正しい値を返します。その他のプラットフォームでは常に false を返します。

判定方法は 2 種類あり、IsMusicPlaying()は iOS の組み込みのミュージックアプリのみ対象にとります。IsOtherAudioPlaying()は市販のアプリも対象に含みます。
IsOtherAudioPlaying()を使用する場合は、3 秒後くらいに再判定するように実装してください。市販のアプリから自分のアプリに切り替えるときに、自分のアプリに戻った時点で市販のアプリの音がフェードアウトを開始する場合があります。この場合、フェードアウト停止予定でもアプリを切り替えて処理を行った時点では「再生中」

と判定されるためです。

13) 手動でダッキングを行う

手動でダッキングを設定するときは、カテゴリごとに設定を行います。
解除するときはカテゴリ名を指定するか、一括で復帰させるか２種類の方法があります。

```
AudioManager.SetDucking("BGM", 0, 1);   // 1秒で音量0に変更
AudioManager.SetDucking("JINGLE", 0, 1);   // 1秒で音量0に変更
AudioManager.ResetDuckingAll(1);  // ダッキング中のインスタンスをすべて1秒で復帰
```

14) AudioMixer を設定する

AudioMixer を Resources または AssetBundle から取得して AudioManager へ設定してください。USnd 旧バージョンではアプリケーションに AudioMixer を組み込むよう記載していましたが、Unity5.3.1 で動作確認し AssetBundle から正しく AudioMixer が取得できることを確認しています。

```
mixer = Resources.Load<AudioMixer>("Mixer");
AudioManager.SetAudioMixer(mixer);
```

15) AudioMixer の Snapshot を切り替える

Snapshot は AudioMixer のパラメータプリセットのようなものです。現在の値から指定した Snapshot の値へ切り替えます。第１引数に Snapshot 名、第２引数に遷移にかかる秒数を指定します。

```
AudioManager.SetSnapshot("lowpass", 3);
```

16) AudioSource のポジションを変更する

3D サウンドを有効にするには、サウンドテーブルで Audio3DSettings アセット名を指定する、Audio3DSettings アセットをロードする、この 2 つの手順が必要です。
Audio3DSettings は ScriptableObject です。アセットをサウンドデザイナーが作成し、テーブルなどと一緒にお渡しするので、下記のようにアセットを登録してください。
受け渡し時にアセットのスクリプト参照が切れていることがあるので、Audio3DSettings.cs がアセットに登録されているか確認してください。

```
AudioManager. SetAudio3DSettings (asset);
```

Audio3DSettings も JSON 形式で読み込み可能です。テーブルファイル同様、文字列を指定してください。
Audio3DSettings はブロックして読み込むので終了待ちはありません。

```
AudioManager. SetAudio3DSettingsFromJson (json);
```

3D サウンド設定されているラベルは SetPosition で音源の位置を変更できます。
特になにも設定していなければ再生開始時の音源のポジションはデフォルトの(x,y,z)=(0,0,0)の位置に配置されています。

```
AudioManager.SetPosition(instanceId, new Vector3(x, y, z));
```

指定したゲームオブジェクトに追従して音源が移動するようにしたい場合は、SetTrackingObject で GameObject を指定します。指定したオブジェクトと同じ位置へ、音源の位置を自動で更新します。

```
AudioManager.SetTrackingObject(instanceId, objet);
```

上記 2 つはインスタンスを指定するので、再生前に設定を反映したい場合は Prepare で再生準備を行って PlayInstance 実行前に設定する必要があります。
再生開始前に音源の位置、追従するゲームオブジェクトが指定可能な場合は、Play3D を使用してください。

```
instanceId = AudioManager.Play3D(“PLAY_LABEL”, new Vector3(x, y, z));
```

```
instanceId = AudioManager.Play3D(“PLAY_LABEL”, object);
```

オーディオの再生が止まる前に GameObject が先に消滅する場合など、GameObject の登録を解除したいときは SetTrackingObject の引数に null を指定して設定を解除してください。

AudioListener については、特に変更していなければ AudioListener はシーンの作成をおこなったときに生成される MainCamera に設定されています。
リスナーの位置をどこに置くかは、サウンドデザイナーと相談して決めてください。

17) Pan(または Pitch)を指定した位置に指定秒数で遷移させる

ボリューム以外にも、パン、ピッチも現在の値から指定した値に、指定秒数で遷移させることができます。
第２引数が変更する目標値、第３引数が遷移にかかる秒数です。

```
AudioManager.SetPan(instanceId, -1, 1);  // １秒で左にパンをふる
AudioManager.SetPitch(instanceId, 200, 1);  // １秒で200セントピッチをあげる
```

18) 手動でオーディオデータの Load/Unload を管理する

AudioClip の設定で、Preload Audio Data のチェックをはずしておくと、プログラムから手動でデータのロード/アンロードが操作できます。チェックが入っている場合は Unity が自動でやってくれます。
Unity のデフォルトは Preload AudioData のチェックが ON なので、この処理は必須ではありません。必要な場合はデザイナーに声をかけてください。

読み込むときは下記のようにラベル単位か、LoadID を指定します。Preload Audio Data が無効でも、コールされたときにデータが読み込まれていないときは自動でロードするので、必ず呼ぶ必要はありません。

```
AudioManager.LoadAudioData(“bgm”);
AudioManager.LoadAudioDataLoadId(0);
```

削除するときは、ラベル単位、LoadID、すべてアンロードがあります。

```
AudioManager.UnloadAudioData(“bgm”);  // 指定したラベルのオーディオデータをアンロード
AudioManager.UnloadAudioDataLoadId(0);  // LoadIDが0に設定されているAudioSourceのオーディオデータをアンロード
AudioManager.UnloadAudioDataAll();   // すべてアンロード
```

Preload Audio Data にチェックが入っているときはこれらを呼ぶ必要はありません。Preload Audio Data が有効な状態で使い終わってラベルが不要になったときは、RemoveAudioSource()で登録を削除してください。
Preload Audio Data が有効なデータを AssetBundle から読み込んだとき、再生前に AssetBundle はアンロード

してしまって問題ありませんが、Preload Audio Data が無効なデータの場合は再生時に AssetBundle が開いたまま(AssetBundle.Unload(false)を行なっていない状態)でないと失敗するので注意してください。

19) ランダムソースの更新を行う

ラベルの中には複数の AudioSource からランダムで再生を行うような設定になっているラベルがあります。
例えば、ラベル A がラベル B,C をランダム候補として保持している場合、USnd に B,C を先に登録してから A を登録しないと参照が成功しません。
A,B,C の順にロードした場合は、UpdateRandomSourceInfo()または UpdateRandomSourceInfoAll()で正しい参照に更新してください。
SetAudioClipToLabel を呼ぶと実行されるので、通常の使い方であれば意図的に呼び出す必要はありません。

```
AudioManager.UpdateRandomSourceInfo(“se_random”);
AudioManager.UpdateRandomSourceInfoAll();
```

20) Android のマナーモード時の挙動を変更する

Android はゲームアプリの音量はメディア音量で制御されるため、マナーモードが反映されません。
マナーモードとゲームの音量を連動させたい場合は、UnityEditor の拡張メニューUSnd->コンパイル設定の USND_ANDROID_MANNER_MODE を ON にしてください。デフォルトは OFF です。

21) オブジェクトプールの削除

USnd で使用する AudioSource はカテゴリのロード時にカテゴリの発音数分確保され、空いている AudioSource を使いますようになっています。発音数が 0（無限）の場合は不足していたときに追加で生成します。
最大で使う数分の AudioSource が確保されるので、削除しなくても問題ないと思いますが定期的にクリアしたい場合は ClearObjectPool()を呼んでください。

22) アプリ起動時に強制的に音楽プレイヤーの再生を停止する

ゲームと同時に音楽プレイヤーが再生されると困る場合は下記対応を行ってください。iOS は組み込みのミュージックアプリのみが対象で、市販のアプリは処理対象外となります。Android は AudioFocus の制御が正しく実装されているアプリのみが対象となります。

iOS の場合

UnityEditor の File->Build Settings->PlayerSettings を開き、iOS のタブを選択します。Other Settings の Override iPod Music にチェックを入れると、アプリ起動時にミュージックアプリの再生が停止します。

Android の場合

UnityEditor の拡張メニューUSnd->コンパイル設定を開き、USND_ANDROID_AUDIO_FOCUS にチェックを入れてください。チェックを入れると起動時に AudioFocus を取得するようになるので、AudioFocus の処理を実装しているアプリの再生が停止します。プリインの音楽アプリや LISMO Player などのキャリア製アプリは停止するはずです。

23) デバッグモードの ON/OFF

拡張メニューの USnd->コンパイル設定でデバッグ機能の ON/OFF ができます。
USND_EDIT_MODE を ON にすると後述の USndTool が有効になります。USND_DEBUG_LOG を ON にす

ると Console に USnd のログが表示されます。

24) Android で SoundPool を使用する

Android は Unity を使うと、DSP Buffer Size を変更して低レイテンシな設定にしても音楽ゲームなどでは違和感を感じるくらい遅延します。Android の Java API、SoundPool を使用して再生するとほぼ遅延がなくなるので、使用用途によって SoundPool を使用してください。

サウンドテーブルで IsAndroidNative フラグが true になっているラベルは SoundPool API を使用して再生を行います。プログラム側で任意のラベルを SoundPool で再生することはできません。テーブル側の設定が必要です。

SoundPool で再生する場合は AssetBundle からデータを読み込むことはできないので、一度端末にオーディオファイルを書き出して別途ロードする処理が必要になります。

保存したらそのファイルを使用するラベル名と保存先のパスを指定して SetAndroidNativeToLabel()を呼び出します。

```
string toPath = Application.persistentDataPath + "/" + "se.ogg";
AudioManager.SetAndroidNativeToLabel("se_label_name", toPath, this.name, "onLoadFinishCallback");

void onLoadFinishCallback(string status)
{
    Debug.Log("!!! onLoadFinishCallback:" + status);
}
```

これで SoundPool を使用して再生する準備ができます。

第 3 引数にコールバックを実装しているオブジェクトの名前、第 4 引数にコールバック関数名を指定するとロード完了したときに通知が受け取れます。ロードに時間がかかるので、再生直前にロードはしないでください。

Play,Unload など後の操作は他のラベルと変わりません。

ただし SoundPool を使用する場合インスタンスの操作は行なえないので、ワンショットの短い SE を再生するときに使用してください。

■初期化サンプル

初期化

```
AudioManager.Initialize();

// AudioMixerはラベルのロードより前に設定、ミキサー未使用の場合は不要
AudioManager.SetAudioMixer(mixer);

// Audio3DSettingsは再生前までにセットしておく、3Dサウンド未使用の場合は不要
Audio3DSettings d3setting = Resources.Load<Audio3DSettings>("d3set");
AudioManager.SetAudio3DSettings(d3setting);
```

サウンドテーブルの読み込み（下記は XML の例）

loadStep が LOAD_EXIT になるまで毎ループ実行する。

```
TextAsset ta;
Stream xml;

switch(loadStep)
{
    case LOAD_MASTER:
        ta = Resources.Load("XML/MasterSettings" ) as TextAsset;
        if (ta != null)
        {
            xml = new MemoryStream(ta.bytes);
            AudioManager.LoadMasterXml(xml);
            loadStep = LOAD_CATEGORY;
        }
        break;
    case LOAD_CATEGORY:
        // LOAD_XML_STATUS.LOADINGの間はマスター読み込み中なので待機
        if (AudioManager.GetLoadXmlStatus() != AudioDefine.LOAD_XML_STATUS.LOADING)
        {
            ta = Resources.Load("XML/CategorySettings") as TextAsset;
            if (ta != null)
            {
                xml = new MemoryStream(ta.bytes);
                AudioManager.LoadCategoryXml(xml);
            }
            loadStep = LOAD_LABEL;
        }
        break;
```

```
    case LOAD_LABEL:
        if (AudioManager.GetLoadXmlStatus() != AudioDefine.LOAD_XML_STATUS.LOADING)
        {
            ta = Resources.Load("XML/" + CATEGORY_XML_NAME[subStep]) as TextAsset;
            if (ta != null)
            {
                xml = new MemoryStream(ta.bytes);
                AudioManager.LoadLabelXml(0, xml);
            }
            loadStep = LOAD_WAIT;
        }
        break;
    case LOAD_WAIT:
        // ラベルテーブルの読み込み終わりまで待つ
        if (AudioManager.GetLoadXmlStatus() != AudioDefine.LOAD_XML_STATUS.LOADING)
        {
            loadStep = LOAD_EXIT;  // すべて読み込み完了
        }
        break;
}
```

AudioClip の読み込み

```
IEnumerator loadClip()
{
    ResourceRequest req = null;
    // ラベルに設定されているAudioClip名を取得、ラベルを先に読み込んでいる場合はAudioManagerから一覧が取得可能
    string[] clipList = AudioManager.GetAudioClipNameAll();

    int currentLoad = 0;
    while (currentLoad < clipList.Length)
    {
        if (req == null)
        {
            req = Resources.LoadAsync<AudioClip>( "AudioClip/" + clipList[currentLoad]);
        }

        while (req.isDone == false)
        {
            yield return null;
        }
```

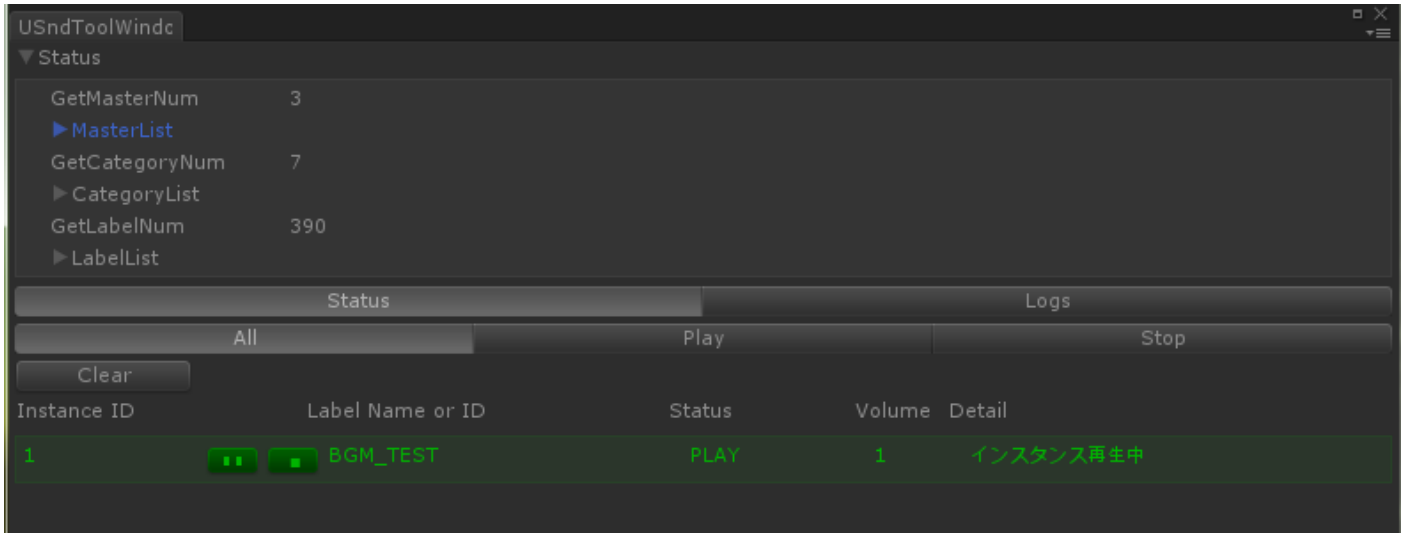
```
        if (req.isDone == true)
        {
            AudioClip clip = req.asset as AudioClip;

            if (clip != null)
            {
                AudioManager.AddAudioClip(clip);
                ++currentLoad;
                req = null;
            }
        }
    }
    // 読み込み終わったらラベルと関連付けを行なう、ラベルテーブルのロード前にAudioClipを読み込んでいる場合は不
要
    AudioManager.SetAudioClipToLabelAll();
}
```

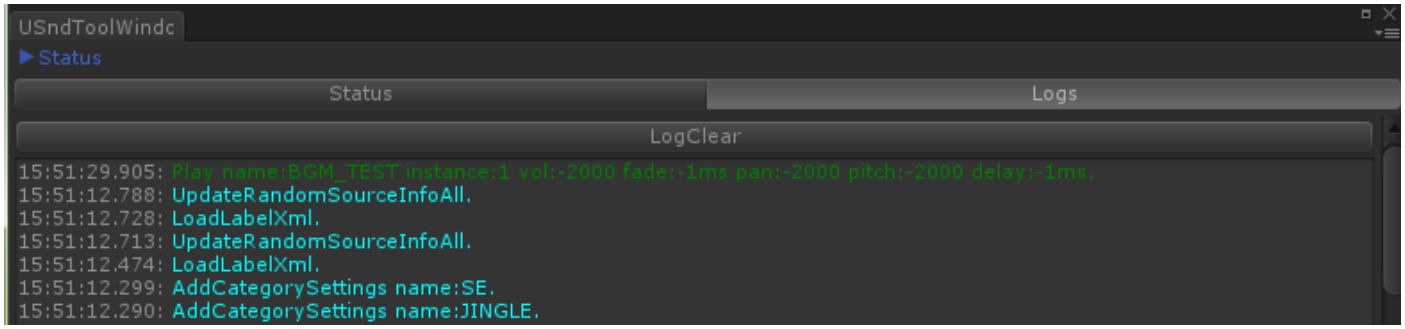
■サウンドデバッグツール

拡張メニューの USnd->コンパイル設定->USND_EDIT_MODE にチェックが入っているときのみ動作します。
拡張メニューUSnd->USndTool を選択すると、下記ツールウィンドウが開きます。

実行中に開くと、下図のように読み込んでいる Master、Category、Label の情報と、現在再生中のインスタンス情報が表示されます。LabelList は Play/Stop ボタンがあるので、再生のテストが可能です。



Logs タブに切り替えると、USnd のシステムログが表示されます。



ログは一部の API のみ出力を行っているので、ほしい情報がログに出ていない場合はご相談ください。

■サウンドテスト用シーン USndViewer について

USndViewer.unitypackage は USndViewer のプレイヤー部分のみを、ゲームのデバッグモードで使用するためにパッケージ化したものです。USnd 本体とは別の zip で配布していますが、サウンドデザイナー向けのデータの中にある USndViewer のプレイヤー部分と同じものです。
インポートすると USndViewer.unity と USndViewer.cs が配置されます。Assets 直下と Scripts 直下に配置されるので、プロジェクトごとの適当な場所に移動してください。
USndViewer ではサウンドテーブルの読み込みは行なっていません。表示に必要な情報の取得と再生・停止などの処理のみ実装しています。
USndViewer シーンの呼び出し前にロードしたテーブルから読み込んだラベルが再生できるラベルになるので、サウンドがロード済みの位置から USndViewer を呼び出せるように実装してください。
USndViewer から他のシーンに移動するには delegate を設定します。
USndViewer の左下の「<< Back」ボタンを押すと delegate に設定した処理が実行されます。
例えば下記のように指定すると、Back を押すと Boot シーンに戻ります。

```
USndViewer.backButtonMethod  = () =>
{
    SceneManager.LoadScene("Boot");
};
```

Snapshot の切替テスト画面を使用する場合、USndViewer にシーンを切り替える前に Snapshot 名のリストを登録してください。Snapshot がない場合は設定不要です。

```
string[] snapNameList = getSnapNameList();
USndViewer.SetSnapInfo(snapNameList);
```

■Streaming 設定時の注意点

Unity5.3.1 時点の仕様では AudioClip の LoadType に Streaming を設定しているときの挙動は以下のようになります。

~~AudioClip の Preload Audio Data 設定が有効なときの注意点~~

~~・1 ファイルあたりストリーミング用のメモリとして非再生時でも 158KB を消費する~~

~~・AudioClip 化した時点でファイルを開くため、大量に AudioClip を読み込むと OS の上限をオーバーすることがある~~

Unity5.3.5 から Streaming 設定は常に Preload Audio Data が無効な状態になりました。

AudioClip の Preload Audio Data 設定が無効なときの注意点

・AssetBundle から Load を行なうとき、AudioClip を取得した後 AssetBundle.Unload(false)を実行すると Load 時にファイルが開けなくなり再生失敗する

実装時は以下のことを注意してください。

・Preload を指定した場合、AudioClip 化するファイル数が多くなりすぎないようロードしておくデータを制限する

・長さの短いファイルは Compress In Memory を使用し、Streaming にしないようにする

・Compress In Memory では厳しい場合は、Streaming に指定した上で Preload の指定を外す。その際 AssetBundle から読み込むときは Unload(false)を行なわないようにする

通常 AssetBundle は読み込んだ後 Unload(false)を行なう実装ですが、現在の Unity の仕様では AudioClip を読み込んだときに Preload を使用しない設定の場合は解放してしまうと再生が行えません。
読み込み時点では Unload せずに、使用が終わった後に該当の AudioClip が設定されているラベルを AudioManager から削除し、その後 AssetBundle の破棄を行なうようにしてください。

Preload Audio Data が無効な AudioClip の再生・停止時は、ロードは Unity が自動で行います。アンロード処理は AudioManager が再生停止時に自動で行ないます。
明示的に操作したいときは AudioManager.LoadAudioData(), AudioManager.UnloadAudioData()でロード/アンロード処理が行えます。AudioClip 再生中に Unload が発生すると再生が止まるので、行なうときはタイミングに注意してください。

■トラブルシューティング

Q. 何もしていないのに勝手にループする or ループデータと指定されているがループしない
A. デザイナー側でループ設定は行います。プログラムからの制御はありませんので、ループ指定されていれば自動的にループする仕組みです。「ループする」と伝えられているのにループしない場合は設定が間違っているのでデザイナーに報告してください。

Q. ひとつのラベルを再生しているのに、再生毎にボリューム or ピッチ or パンが異なる音が再生される
A. デザイナー側で設定を行った結果の挙動なので問題ありません。デザイナーから不具合報告でこのような報告があがってきた場合は設定がされていないか確認するよう返答してください。

Q. ひとつのラベルを再生しているのに、再生毎に違う音が再生される
A. デザイナー側でランダムコール設定を行った結果の挙動なので問題ありません。デザイナーから不具合報告でこのような報告があがってきた場合は設定がされていないか確認するよう返答してください。

Q. 特定のラベルを再生すると勝手に BGM が消えて、再生が終わると BGM が鳴り出す
A. デザイナー側でダッキング設定を行った結果の挙動なので問題ありません。

Q. 特定のラベルを再生すると勝手に BGM が消えて、再生が終わったが BGM が復帰しない
A. ダッキング設定で自動で復帰しない設定があるため復帰しないことがあります。デザイナーの意図した挙動か確認し、意図通りであれば復帰タイミングを確認してプログラムから復帰処理を行ってください。意図した挙動でなければ自動で復帰するよう設定を変えてもらってください。

Q. Android で他のアプリケーションの再生が検知できない
A. Android, PC, Mac は非対応です。iOS のみ動作します。それ以外のプラットフォームは常に「再生していない」ステータスを返します。

Q. Play してから再生までに間があく
A. デザイナー側で設定を行った結果の挙動なので問題ありません。デザイナーから不具合報告でこのような報告があがってきた場合は、Play 時の引数でもディレイは設定できるので、プログラム側で設定が入っていないか確認してください。プログラム側で指定されていない場合は、設定がされていないか確認するよう返答してください。

Q. Error: Cannot create FMOD::Sound instance for resource archive:というエラーが出る
A. AudioClip の Preload Audio Data が無効な場合、AssetBundle を Unload してしまうと上記のエラーが出ます。Preload Audio Data を有効にするか、AssetBundle は使い終わってから Unload してください。

Q. iOS でアプリケーションを起動するとミュージックアプリの再生が自動的に止まってしまう
A. iOS の PlayerSettings にある Override I Pod Music にチェックを入れてください。チェックが入っていないとゲームの音声が優先されます。チェックを入れると、ゲームの音とミュージックアプリの音が同時に再生される状態になります。

ミュージックアプリ再生中はゲームの特定の音声の音量（BGM など）をさげるようにしたい場合は、USndPlugin.IsMusicPlaying()か USndPlugin.IsOtherAudioPlaying()で音声が再生中か判定を行い、マスターの音量を変更するなどしてボリュームの設定を行ってください。

Q. Android で音がスピーカーから鳴らない
A.コンパイル設定の USND_ANDROID_MANNER_MODE にチェックが入っていないか確認してください。チェックが入っているとマナーモードの状態にあわせてゲームの音量を自動的に制御します。

Q.Android で「Too many open file」で落ちる、または iOS で FMOD 内のファイルオープンエラーで落ちる
A.Unity に問い合わせたところ Unity 5.3.1f1 時点では、Streaming に設定した AudioClip はロードした時点でファイルを開いている扱いになり、OS のファイルディスクリプタ上限に達してしまうためということで仕様のようです。大量のストリーミング指定のファイルがある場合は、オンメモリを指定するように変更するか、こまめにロード/アンロードするようにしてください。設定の変更はデザイナーに依頼してください。

Q.Audio3DSettings アセットを登録しているが 3D サウンドにならない
A.Audio3DSetgins.cs がアセットに登録されているか確認してください。受け渡し時に参照が外れていることがあります。

Q.AudioMixer 設定が反映されない
A.ラベルのロード前に AudioMixer が読み込まれているか確認してください。

■関数リファレンス

システム関連一覧

Initialize	初期化
Terminate	終了
[廃止]AddMasterSettings	マスター情報を登録
[廃止]AddCategorySettings	カテゴリ情報を登録
[廃止]AddAudioSource	AudioSourceを登録
AddAudioClip	AudioClipを登録
IsExistAudioClip	登録済みのAudioClipか確認する
[廃止]FindAudioMaster	登録済みのマスターか調べる
[廃止]FindAudioCategory	登録済みのカテゴリが調べる
[廃止]FindAudioSource	登録済みのAudioSourceか調べる
[new]FindMaster	登録済みのマスターか調べる
[new]FindCategory	登録済みのカテゴリが調べる
[new]FindLabel	登録済みのラベルか調べる
[廃止]RemoveAudioSource	AudioSourceの登録を解除する
[廃止]RemoveAudioSourceLoadId	指定したLoadIDと一致するAudioSourceを解除する
[廃止]RemoveAudioSourceAll	すべてのAudioSourceを解除する
RemoveAll	すべての登録情報を解除する
RemoveAudioClip	AudioClipの登録を削除
RemoveAudioClipAll	AudioClipの登録をすべて削除
RemoveLabel	指定したラベル名に関連するAudioSourceとAudioClipを削除する
RemoveLabelLoadId	指定したLoadIDに含まれるラベルに関連するAudioSourceとAudioClipを削除する
RemoveLabelAll	すべてのラベルに関連するAudioSourceとAudioClipを削除する
CanRemoveLabel	指定したラベル名に関連するAudioSourceとAudioClipは削除可能か
UpdateRandomSourceInfo	ランダム再生の参照情報を更新する
UpdateRandomSourceInfoAll	すべてのAudioSourceのランダム再生の参照情報を更新する
LoadAudioData	PreloadAudioDataのチェックがはずれているときにオーディオデータをロードする
LoadAudioDataLoadId	PreloadAudioDataのチェックがはずれているときに指定したLoadIDと一致するAudioSourceのオーディオデータをロードする
UnloadAudioData	PreloadAudioDataのチェックがはずれているときにオーディオデータを解放する
UnloadAudioDataLoadId	PreloadAudioDataのチェックがはずれているときに指定したLoadIDと一致するAudioSourceのオーディオデータを解放する
UnloadAudioDataAll	PreloadAudioDataのチェックがはずれているときにすべてのオーディオデータを解放する
SetAudioClipToLabelLoadId	ロードIDを指定してラベルにAuidoClipの割り当てを行う
SetAudioClipToLabelAll	すべてのラベルにAuidoClipの割り当てを行う
SetAudioClipToLabel	指定したラベルにAuidoClipの割り当てを行う
SetAndroidNativeToLabel	指定したラベルにAndroidNative再生用のファイルパスをセットしてロード
UnsetAudioClipToLabelLoadId	ロードIDを指定してラベルに割り当てられたAudioClipの参照をはずす

UnsetAudioClipToLabelAll	すべてのラベルに割り当てられたAuidoClipの参照をはずす
UnsetAudioClipToLabel	指定したラベルに割り当てられたAuidoClipの参照をはずす
[廃止]ClearInstancePool	インスタンスのオブジェクトプールをクリア
[廃止]ClearAudioSourcePool	AudioSourceを直接割り当てたラベルのオブジェクトプールをクリア
[廃止]ClearXMLAudioSourcePool	XMLから読み込んだラベルのオブジェクトプールをクリア
ClearObjectPool	すべてのオブジェクトプールをクリア

ロード関連一覧

LoadMasterXml	マスター情報のXMLを読み込んでマスターを新規登録・上書き
LoadCategoryXml	カテゴリ情報のXMLを読み込んでカテゴリを新規登録・上書き
LoadLabelXml	ラベル情報のXMLを読み込んでラベルを新規登録・上書き
GetLoadXmlStatus	XMLの読み込み状況を取得する
LoadBinaryTable	バイナリ形式に変換したXMLテーブルを読み込んで新規登録・上書き
LoadJson	JSON形式のテーブル文字列を読み込んで新規登録・上書き
GetLoadJsonStatus	JSON形式の読み込み状況を取得する
SetAudio3DSettings	Audio3DSettingsのアセットを指定して3Dパラメータを登録
SetAudio3DSettingsFromJson	Audio3DSettingsパラメータをJSON形式で登録

再生制御一覧

Play	ラベルを再生する
PlayOption	ボリューム、ピッチ、パンを設定してラベルを再生する
Prepare	ラベルの再生準備を行う
PrepareOption	ボリューム、ピッチ、パンを設定してラベルの再生準備を行う
PlayInstance	再生準備で生成したインスタンスの再生を開始する
Play3D	3Dサウンドの位置情報を指定して再生する
Play2D	3Dサウンド設定されているラベルを強制的に2Dで再生する
SetTrackingObject	インスタンスのポジションを追従させるオブジェクトを指定
Stop	インスタンスを停止する
StopLabel	ラベル名を指定して再生中のインスタンスを停止する
StopCategory	指定したカテゴリに所属するラベルを停止する
StopMaster	指定したマスターが設定されているカテゴリに所属するラベルを停止する
StopAll	すべてのインスタンスを停止する
OnPause	インスタンスを一時停止する
OffPause	インスタンスの一時停止を解除する
OnPauseLabel	ラベル名を指定して再生中のインスタンスを一時停止する
OffPauseLabel	ラベル名を指定して再生中のインスタンスの一時停止を解除する
OnPauseCategory	指定したカテゴリに所属するラベルを一時停止する
OffPauseCategory	指定したカテゴリに所属するラベルの一時停止を解除する
OnPauseMaster	指定したマスターが設定されているカテゴリに所属するラベルを一時停止する
OffPauseMaster	指定したマスターが設定されているカテゴリに所属するラベルの一時停止を解除する
OnPauseAll	すべてを一時停止

OffPauseAll	すべての一時停止を解除
SetVolume	再生中のインスタンスIDまたはラベル名でボリュームを変更
GetInstanceVolume	インスタンスのボリュームを取得
GetInstanceCalcVolume	インスタンスの最終的なボリュームを取得
SetMasterVolume	マスターボリュームを変更
GetMasterVolume	マスターボリュームを取得（プログラム設定値）
SetCategoryVolume	カテゴリボリュームを変更
GetCategoryVolume	カテゴリボリュームを取得（プログラム設定値）
GetLabelVolume	ラベルに設定されているボリューム値を取得
SetPitch	再生中のインスタンスIDまたはラベル名でピッチを変更
SetPan	再生中のインスタンスIDまたはラベル名でパンを変更
SetPosition	再生中のインスタンスIDまたはラベル名が設定されているゲームオブジェクトのポジションを変更
ResetPlayPositionAll	再生開始位置をリセット
SetDucking	指定したカテゴリのダッキングを開始
ResetDucking	指定したカテゴリのダッキングを解除
ResetDuckingAll	すべてのダッキング設定を解除
ForceResetDucking	強制でダッキング解除
ForceResetDuckingAll	強制で全カテゴリのダッキング解除
SetMute	ミュートのON,OFFを設定
GetMuteStatus	ミュートがONかOFFかを取得

状態取得一覧

GetInstanceStatus	インスタンスの状態を取得
IsPlayingLabel	指定したラベルは再生中か取得
[廃止]GetAudioSourceNum	登録済みのAudioSourceの総数を取得
[廃止]GetAudioSourceNameList	登録済みのAudioSourceの名前リストを取得
GetCategoryNum	登録済みカテゴリの総数を取得
GetCategoryNameList	登録済みカテゴリの名前リストを取得
GetMasterNum	登録済みマスターの総数を取得
GetMasterNameList	登録済みマスターの名前リストを取得
GetCategoryNameSettingOfLabel	指定したラベルに設定されているカテゴリ名を取得
GetMasterNameSettingOfCategory	指定したカテゴリに設定されているマスター名を取得
GetPlayTime	インスタンスの現在の再生時間を取得
GetPlaySamples	インスタンスの現在の再生サンプル位置を取得
SetTime	インスタンスの再生位置を時間で設定
SetTimeSamples	インスタンスの再生位置をサンプル単位で設定
GetAudioClipNameLoadId	ロードIDに一致するラベルに指定されているAudioClip名のリストを取得
GetAudioClipNameAll	すべてのラベルに指定されているAudioClip名のリストを取得
GetAudioClipName	指定したラベルに設定されているAudioClip名を取得
GetAudioClipNames	指定したラベルに設定されているAudioClip名を取得（ランダム設定されているラ

	ベルのAudioClip名も含む)
GetRandomSourceNames	指定したラベルに設定されているRandomSource名（ラベル名）を取得する
GetLabelNum	登録済みラベルの総数を取得
GetLabelNameList	登録済みラベルの名前リストを取得
GetLabelLength	ラベルの総再生時間を秒単位で取得
GetLabelSamples	ラベルの総再生時間をサンプル単位で取得
GetSpectrumData	再生中インスタンスのスペクトラムを取得
IsLoop	ラベルのループ設定を取得
GetLabelMaxPlaybacksNum	ラベルの最大発音数を取得
GetCategoryMaxPlaybacksNum	カテゴリの最大発音数を取得
GetCategoryMaxPlaybacksNumFromLabel	ラベルに指定されているカテゴリの最大発音数を取得
IsInterval	指定したラベルがインターバル期間中か取得
GetCurrentPlayNum	停止予定も含む現在アクティブなプレイヤー数を取得

AudioMixer関連一覧

[削除]SetUnityMixerInfo	AudioMixer情報を登録
[削除]UnsetUnityMixerInfo	AudioMixer情報の登録を解除
SetAudioMixer	UnityEngine.Audio.AudioMixerを設定
UnsetAudioMixer	UnityEngine.Audio.AudioMixerを解除
[変更]SetSnapshot	Snapshotを切り替える
[廃止]GetSnapshotNum	Snapshotの総数を取得
[廃止]GetSnapshotNameList	Snapshotの名前リストを取得
SetAudioMixerExposedParam	Exposed ParametersでMixerの値を変更

USndPlugin関連一覧

USndPlugin.IsMusicPlaying	組み込みのミュージックプレイヤーが再生中か(iOSのみ)
USndPlugin.IsOtherAudioPlaying	他のアプリケーションが再生中か(iOSのみ)
USndPlugin.IsSpeaker	スピーカーから再生されているか
USndPlugin.IsMannerMode	マナーモードが設定されているか(Androidのみ)

void Initialize (int defaultSampleRate = 0)

AudioManager を初期化

引数

defaultSampleRate　[in]システムの出力サンプルレート

戻り値

なし

説明

AudioManager の初期化を行います。起動時または Terminate 後にもう一度システム起動する際に 1 回呼び出します。
defaultSampleRate は 0 で USnd の設定値 DEFAULT_SAMPLE_RATE = 44100 で初期化を行います。Unity のデフォルトに任せる場合は-1 を指定します。iPhone, 一部 Android の Unity のデフォルトは 24kHz なので Unity のデフォルト値を使用するのはお勧めしません。

void Terminate()

システムを終了

引数

戻り値

なし

説明

システムを終了します。終了後はサウンド再生を行えません。再度サウンド再生を行う場合は、Initialize から初期化処理を順番に行います。

void AddAudioClip(AudioClip clip)

AudioClip を登録

引数

clip	[in]登録する AudioClip

戻り値

なし

説明

XML ファイルから AudioSource の登録を行う際に使用する AudioClip を登録します。

void AddAudioClip(AudioClip[] clips)

AudioClip をまとめて登録

引数

clips	[in]登録する AudioClip の配列

戻り値

なし

説明

XML ファイルから AudioSource の登録を行う際に使用する AudioClip を登録します。

bool IsExistAudioClip(string clipName)

登録済みの AudioClip か

引数

clipName　　[in]確認する AudioClip の名前

戻り値

true:登録済み　flase:未登録

説明

指定した名前の AudioClip が登録済みか判定を行います。

bool FindMaster(string name)

登録済みのマスターか調べる

引数

name　　[in]調べるマスター名

戻り値

true: 存在する　false:存在しなし

説明

登録済みのマスター名か調べて結果を返します。

bool FindCategory(string name)

登録済みのカテゴリか調べる

引数

name　　[in]調べるカテゴリ名

戻り値

true: 存在する　false:存在しなし

説明

登録済みのカテゴリ名か調べて結果を返します。

bool FindLabel(string name)

登録済みのラベルか調べる

引数

name　　[in]調べる AudioSource 名(ラベル名)

戻り値

true: 存在する　false:存在しなし

説明

登録済みのラベルか調べて結果を返します。

void RemoveAll()

すべての登録情報を削除する

引数

戻り値

なし

説明

登録したすべての情報を削除します。

void RemoveAudioClip(string clipName)

AudioClipの登録を削除

引数

clipName　　[in]削除する AudioClip

戻り値

なし

説明

登録した AudioClip 情報を削除します。AudioSource に既に割り当てられている場合、該当の AudioSource が Remove されるまで解放されないので注意してください。

void RemoveAudioClipAll()

AudioClipの登録をすべて削除

引数

戻り値

なし

説明

登録した AudioClip 情報をすべて削除します。AudioSource に既に割り当てられている場合、該当の AudioSource が Remove されるまで解放されないので注意してください。

bool RemoveLabel(string labelName)

指定したラベル名に関連するAudioSourceとAudioClipを削除する

引数

labelName　　[in]削除するラベル名

戻り値

true:削除成功　false:削除失敗

説明

指定したラベル名に関連する AudioSource と AudioClip を削除します。
他のラベルからランダム再生設定されている場合、AudioClip は UpdateRandomSourceInfo を行うまで解放されません。
内部的にまだ再生中の状態だった場合、削除に失敗します。削除可能か調べるには CanRemoveLabel を使用します。

void RemoveLabelLoadId(int loadId)

指定したLoadIDに含まれるラベルに関連するAudioSourceとAudioClipを削除する

引数

loadId　　[in]削除するロード ID

戻り値

なし

説明

指定したロード ID に含まれるラベル名に関連する AudioSource と AudioClip を削除します。
他のラベルからランダム再生設定されている場合、AudioClip は UpdateRandomSourceInfo を行うまで解放されません。

void RemoveLabelAll()

すべてのラベルに関連するAudioSourceとAudioClipを削除する

引数

戻り値

なし

説明

すべてのラベルの関連する AudioSource と AudioClip を削除します。どのラベルにも関連付けられていない AudioClip は解放されないので、関連付けられていない AudioClip を削除する場合は RemoveAudioClip で削除します。

bool CanRemoveLabel(string labelName)

指定したラベル名に関連するAudioSourceとAudioClipは削除可能か

引数

labelName　　[in]削除可能か調べるラベル名

戻り値

true:削除可能　false:削除できない

説明

指定したラベル名に関連する AudioSource と AudioClip が削除可能なタイミングか調べます。

void UpdateRandomSourceInfo(string labelName)

ラベルのランダム再生情報を更新

引数

labelName　　[in]更新するラベル名

戻り値

なし

説明

ラベルにランダム再生が設定されているときに、ランダム再生の候補となる他のラベルへの参照情報を更新します。
ランダム候補を先に USnd に登録している場合は更新の必要がありませんが、ランダムの元となるラベルを先に登録している場合は更新処理を行ってください。

void UpdateRandomSourceInfoAll()

すべての AudioSource のランダム再生の参照情報を更新する

引数

戻り値

なし

説明

ラベルにランダム再生が設定されているときに、ランダム再生の候補となる他のラベルへの参照情報を更新します。
ランダム候補を先に USnd に登録している場合は更新の必要がありませんが、ランダムの元となるラベルを先に登録している場合は更新処理を行ってください。

void LoadAudioData(string labelName)

PreloadAudioData のチェックがはずれているときにオーディオデータをロードする

引数

labelName　　[in]ラベル名

戻り値

なし

説明

指定したラベルに設定されている AudioClip のオーディオデータをロードします。

AudioClip の設定で Preload Audio Data のチェックがはずれている場合のみ有効です。チェックが入っていない場合は UnityEditor で指定したタイミングに従って自動的にオーディオデータはロードされます。

void LoadAudioDataLoadId(int loadId)

PreloadAudioData のチェックがはずれているときに指定した LoadID と一致する AudioSource のオーディオデータをロードする

引数

loadId	[in]オーディオデータをロードするラベルの LoadId

戻り値

なし

説明

指定した loadId と AudioSource に設定されている loadId が一致する AudioSource に設定されている AudioClip のオーディオデータをロードします。

void UnloadAudioData(string labelName)

PreloadAudioData のチェックがはずれているときにオーディオデータを解放する

引数

labelName	[in]オーディオデータをアンロードするラベル名

戻り値

なし

説明

指定したラベルに設定されている AudioClip のオーディオデータをアンロードします。AudioClip の設定で Preload Audio Data のチェックがはずれている場合のみ有効です。チェックが入っている場合、AudioSource が Destroy されたタイミングでアンロードされます。

void UnloadAudioDataLoadId(int loadId)

PreloadAudioData のチェックがはずれているときに指定した LoadID と一致する AudioSource のオーディオデータを解放する

引数

loadId	[in]オーディオデータをアンロードするラベルの LoadId

戻り値

なし

説明

指定した loadId と AudioSource に設定されている loadId が一致する AudioSource に設定されている

AudioClip のオーディオデータをアンロードします。

void UnloadAudioDataAll()

PreloadAudioData のチェックがはずれているときにすべてのオーディオデータを解放する

引数

戻り値

なし

説明

すべての AudioClip のオーディオデータをアンロードします。

void SetAudioClipToLabelLoadId(int loadId)

ロード ID を指定してラベルに AuidoClip の割り当てを行う

引数

loadId	[in]AudioClip を割り当てるラベルのロード ID

戻り値

なし

説明

指定した loadId に一致するラベルに指定されている AudioClip 名から、AudioClip が AudioManager に登録されているか検索しラベルに AudioClip の割り当てを行います。
AudioClip が登録されていない状態で先に XML をロードした場合、この API を呼び出してラベルに AuidoClip の割り当てを行います。

void SetAudioClipToLabelAll()

すべてのラベルに AuidoClip の割り当てを行う

引数

戻り値

なし

説明

ラベルに指定されている AudioClip 名から、AudioClip が AudioManager に登録されているか検索しラベルに AudioClip の割り当てを行います。
AudioClip が登録されていない状態で先に XML をロードした場合、この API を呼び出してラベルに AuidoClip の割り当てを行います。

void SetAudioClipToLabel(string labelName)

指定したラベルに AuidoClip の割り当てを行う

引数

	labelName	[in]AudioClip を割り当てるラベル名

戻り値

なし

説明

指定したラベル名に設定されている AudioClip 名から、AudioClip が AudioManager に登録されているか検索しラベルに AudioClip の割り当てを行います。
AudioClip が登録されていない状態で先に XML をロードした場合、この API を呼び出してラベルに AuidoClip の割り当てを行います。

void SetAndroidNativeToLabel(string labelName, string filePath, string className, string funcName)

指定したラベルに AndroidNative 再生用のファイルパスをセットしてロード

引数

	labelName	[in]ファイルを割り当てるラベル名
	filePath	[in]ファイルのパス
	className	[in]コールバックを実装しているオブジェクト名
	funcName	[in]コールバック関数名

戻り値

なし

説明

指定した filePath のファイルを、指定したラベルのリソースとして割り当てます。サウンドテーブルで IsAndroidNative フラグが true になっているラベルのみ有効です。
コールバックは指定した関数がファイルのロード完了時に UnitySendMessage で呼び出されます。コールバックは使用しないときは null を指定してください。

void UnsetAudioClipToLabelLoadId(int loadId)

ロード ID を指定してラベルに割り当てられた AuidoClip の参照をはずす

引数

	loadId	[in]AudioClip を削除するラベルのロード ID

戻り値

なし

説明

指定した loadId に一致するラベルに指定されている AudioClip の参照を外します。ラベルを削除せずに AudioClip のみ削除したいときに使用してください。

void UnsetAudioClipToLabelAll()

すべてのラベルに割り当てられた AuidoClip の参照を外す

引数

戻り値

なし

説明

ラベルに指定されている AudioClip の参照を外します。ラベルを削除せずに AudioClip のみ削除したいときに使用してください。

void UnsetAudioClipToLabel(string labelName)

指定したラベルに割り当てられた AuidoClip の参照を外す

引数

labelName　　[in]AudioClip を削除するラベル名

戻り値

なし

説明

指定したラベル名に設定されている AudioClip の参照を外します。ラベルを削除せずに AudioClip のみ削除したいときに使用してください。

void ClearObjectPool ()

すべてのオブジェクトプールをクリア

引数

戻り値

なし

説明

ClearInstancePool, ClearAudioSourcePool, ClearXMLAudioSourcePool すべてを実行します。

bool LoadMasterXml(Stream xml, Stream xsd = null)

マスター情報の XML を読み込んでマスターを新規登録・上書き

引数

xml	[in]XML ファイルのストリーム
xsd	[in]XSD ファイルのストリーム

戻り値

true:読み込み開始　false:読み込み失敗

説明

マスター情報を XML ファイルから読み込み、名前が重複しない場合は新規登録、名前が重複している場合は XML ファイルに書かれている情報のみ上書きします。
XSD ファイルは XML ファイルの型チェックのためのファイルなので、型チェックが不要な場合は null を指定してください。XSD ファイルの拡張子.xsd は Unity で認識されないので、.bytes に変更して読み込んでください。

bool LoadCategoryXml(Stream xml, Stream xsd = null)

カテゴリ情報の **XML** を読み込んでカテゴリを新規登録・上書き

引数

xml	[in]XML ファイルのストリーム
xsd	[in]XSD ファイルのストリーム

戻り値

true:読み込み開始　false:読み込み失敗

説明

カテゴリ情報を XML ファイルから読み込み、名前が重複しない場合は新規登録、名前が重複している場合は XML ファイルに書かれている情報のみ上書きします。
XSD ファイルは XML ファイルの型チェックのためのファイルなので、型チェックが不要な場合は null を指定してください。XSD ファイルの拡張子.xsd は Unity で認識されないので、.bytes に変更して読み込んでください。

bool LoadLabelXml(int loadId, Stream xml, Stream xsd = null)

ラベル情報の **XML** を読み込んでラベルを新規登録・上書き

引数

loadId	[in]読み込んだラベルに割り振る LoadID
xml	[in]XML ファイルのストリーム
xsd	[in]XSD ファイルのストリーム

戻り値

true:読み込み開始　false:読み込み失敗

説明

ラベル情報を XML ファイルから読み込み、名前が重複しない場合は新規登録、名前が重複している場合は XML ファイルに書かれている情報のみ上書きします。
XSD ファイルは XML ファイルの型チェックのためのファイルなので、型チェックが不要な場合は null

を指定してください。XSD ファイルの拡張子.xsd は Unity で認識されないので、.bytes に変更して読み込んでください。

AudioDefine.LOAD_XML_STATUS GetLoadXmlStatus()

XML の読み込み状況を取得する

引数

戻り値

AudioDefine.LOAD_XML_STATUS.STANBY	待機中
AudioDefine.LOAD_XML_STATUS.LOADING	ロード中
AudioDefine.LOAD_XML_STATUS.FINISH	ロード完了
AudioDefine.LOAD_XML_STATUS.ERROR	エラー発生

説明

LoadMasterXml,LoadCategoryXml,LoadLabelXml の処理状況を返します。

bool LoadBinaryTable(byte[] tableData, int loadId = 0)

バイナリ形式のサウンドテーブルを読み込む

引数

tableData	[in]TextAsset として読み込んだバイナリ形式のテーブルファイル
loadId	[in] 読み込んだラベルに割り振る LoadID

戻り値

true:読み込み成功　false:読み込み失敗

説明

バイナリ形式のサウンドテーブルを読み込みます。マスター、カテゴリ、ラベル共通です。XML と異なりバイナリ形式はブロックして処理するので、この関数から戻ってきた時点で処理は終わっています。第 2 引数の loadId は LoadLabelXml で指定する第 1 引数の loadId と同じものです。テーブルの内容がマスター、カテゴリだった場合は無視されます。

bool LoadJson(string tableData, int loadId = 0)

JSON 形式のサウンドテーブルを読み込む

引数

tableData	[in]JSON 形式のテーブル情報（文字列）
loadId	[in] 読み込んだラベルに割り振る LoadID

戻り値

true:読み込み成功　false:読み込み失敗

説明

JSON 形式のサウンドテーブルを読み込みます。マスター、カテゴリ、ラベル共通です。XML 同様、非同期処理で読み込みを行うので GetLoadJsonStatus で読み込み完了を待ってください。複数ファイルを同時にロードはできません。第 2 引数の loadId は LoadLabelXml で指定する第 1 引数の loadId と同じものです。テーブルの内容がマスター、カテゴリだった場合は無視されます。

AudioDefine.LOAD_JSON_STATUS GetLoadXmlStatus()

JSON 形式の読み込み状況を取得する

引数

戻り値

AudioDefine.LOAD_JSON_STATUS.STANBY	待機中
AudioDefine.LOAD_JSON_STATUS.LOADING	ロード中
AudioDefine.LOAD_JSON_STATUS.FINISH	ロード完了
AudioDefine.LOAD_JSON_STATUS.ERROR	エラー発生

説明

LoadJson の処理状況を返します。

void SetAudio3DSettings(Audio3DSettings setting)

Audio3DSettings アセットを指定して 3D パラメータを登録

引数

settings	[in]Audio3DSettings アセット

戻り値

説明

Audio3DSettings のアセットを指定して 3D パラメータを登録します。3D パラメータを登録していないと 3D サウンドは正しく動作しないので、再生前にパラメータを登録してください。

void SetAudio3DSettingsFromJson(string jsonStr)

Audio3DSettings パラメータを JSON 形式で登録

引数

jsonStr	[in]JSON 形式の Audio3DSettings パラメータ

戻り値

説明

Audio3DSettings パラメータを JSON 形式で登録します。3D パラメータを登録していないと 3D サウンドは正しく動作しないので、再生前にパラメータを登録してください。

int Play(string labelName, float delay = -1)

ラベルを再生する

引数

labelName	[in]ラベル名
delay	[in]ディレイ秒数、-1 でラベル設定値に従う

戻り値

インスタンス ID(AudioDefine.INSTANCE_ID_ERROR のときはエラー)

説明

ラベルに設定したパラメータに従ってラベルの再生を行います。

int PlayOption(string labelName, float volume, float fadeTime, float pan, int pitch, float delay = -1)

ボリューム、ピッチ、パンを設定してラベルを再生する

引数

labelName	[in]ラベル名
volume	[in]ボリューム係数(0～1)
fadeTime	[in]フェードイン秒数
pan	[in]パン(-1(左)～0(中央)～1(右))
pitch	[in]ピッチ(-1200(１オクターブ下)～0(変化なし)～1200(1 オクターブ上))
delay	[in]ディレイ秒数、-1 でラベル設定値に従う

戻り値

インスタンス ID(AudioDefine.INSTANCE_ID_ERROR のときはエラー)

説明

ラベルに設定されているパラメータ以外の数値で再生を開始するときに使用します。
ラベルに設定されている値（デフォルト値）のままで良い項目はそれぞれ不正値を指定するとデフォルト値になります。AudioDefine にデフォルト値設定用の定義があるので、下記を使用してください。
AudioDefine.DEFAULT_VOLUME
AudioDefine.DEFAULT_PAN
AudioDefine.DEFAULT_PITCH
AudioDefine.DEFAULT_FADE

int Prepare(string labelName)

ラベルの再生準備を行う

引数	
	labelName　　[in]ラベル名
戻り値	
	インスタンス ID(AudioDefine.INSTANCE_ID_ERROR のときはエラー)
説明	
	ラベルに設定したパラメータに従ってラベルの再生準備を行います。Prepare()は PlayInstance()を呼ぶまで再生を開始しませんが、発音数は消費します。

int PrepareOption(string labelName, float volume, float fadeTime, float pan, int pitch)

ボリューム、ピッチ、パンを設定してラベルの再生準備を行う	
引数	
	labelName　　[in]ラベル名
	volume　　[in]ボリューム係数(0～1)
	fadeTime　　[in]フェードイン秒数
	pan　　[in]パン(-1(左)～0(中央)～1(右))
	pitch　　[in]ピッチ(-1200(１オクターブ下)～0(変化なし)～1200(1 オクターブ上))
戻り値	
	インスタンス ID(AudioDefine.INSTANCE_ID_ERROR のときはエラー)
説明	
	ラベルに設定されているパラメータ以外の数値で再生準備を行うときに使用します。PrepareOption()は PlayInstance()を呼ぶまで再生を開始しませんが、発音数は消費します。 ラベルに設定されている値（デフォルト値）のままで良い項目はそれぞれ不正値を指定するとデフォルト値になります。AudioDefine にデフォルト値設定用の定義があるので、下記を使用してください。 AudioDefine.DEFAULT_VOLUME AudioDefine.DEFAULT_PAN AudioDefine.DEFAULT_PITCH AudioDefine.DEFAULT_FADE

void PlayInstance(int instanceId, float delay = -1)

再生準備で生成したインスタンスの再生を開始する	
引数	
	instanceId　　[in]インスタンス ID
	delay　　[in]ディレイ秒数
戻り値	
	インスタンス ID(AudioDefine.INSTANCE_ID_ERROR のときはエラー)

説明

Prepare()または PrepareOption()で準備したインスタンスの再生を開始します。Prepare 後は再生中同様発音数を消費するので、状態によっては再生前に発音制御でインスタンスが無効になることがあります。Prepare したインスタンスは速やかに再生開始してください。

int Play3D (string labelName, GameObject object, float delay = -1)

追従させるオブジェクトを指定して再生(3D サウンド指定ラベルのみ有効)

引数

labelName	[in]ラベル名
object	[in]追従する対象ゲームオブジェクト
delay	[in]ディレイ秒数

戻り値

インスタンス ID(AudioDefine.INSTANCE_ID_ERROR のときはエラー)

説明

AudioSource のポジションを指定したゲームオブジェクトに追従するよう設定し、再生を行います。Prepare で取得したインスタンスに SetTrackingObject でゲームオブジェクトを指定し､PlayInstance で再生したときと同じ結果になります。

オーディオの再生より先にゲームオブジェクトが消える場合は SetTrackingObject に null を指定して参照を解除してください。

int Play3D(string labelName, Vector3 position, float delay = -1)

位置を指定して再生開始(3D サウンド指定ラベルのみ有効)

引数

labelName	[in]ラベル名
position	[in]音源に設定するポジション
delay	[in]ディレイ秒数

戻り値

インスタンス ID(AudioDefine.INSTANCE_ID_ERROR のときはエラー)

説明

AudioSource のポジションを指定して再生を行います。GameObject を引数に取る API との違いは、指定した定位置から自動では移動しないことです。

Prepare で取得したインスタンスに SetPosition で位置を指定し再生したときと同じ結果になります。再生後に音源の位置を変更したいときは SetPosition を指定してください。

int Play2D(string labelName, float delay = -1)

3D サウンド設定されているラベルを強制的に 2D で再生する	
引数	
labelName	[in]ラベル名
delay	[in]ディレイ秒数
戻り値	
インスタンス ID(AudioDefine.INSTANCE_ID_ERROR のときはエラー)	
説明	
3D サウンド設定のあるラベルを強制的に 2D で再生を行います。2.16.0 時点では、Play で 3D サウンド設定のあるラベルを再生すると原点で再生されます。2D で再生したいときは Play2D を使用してください。	

int Play2D(string labelName, float volume, float fadeTime, float pan, int pitch)

3D サウンド設定されているラベルを強制的に 2D で再生する。PlayOption と同等の引数があるバージョン。	
引数	
labelName	[in]ラベル名
volume	[in]ボリューム係数(0～1)
fadeTime	[in]フェードイン秒数
pan	[in]パン(-1(左)～0(中央)～1(右))
pitch	[in]ピッチ(-1200(１オクターブ下)～0(変化なし)～1200(1 オクターブ上))
戻り値	
インスタンス ID(AudioDefine.INSTANCE_ID_ERROR のときはエラー)	
説明	
3D サウンド設定のあるラベルを強制的に 2D で再生を行います。2.16.0 時点では、Play で 3D サウンド設定のあるラベルを再生すると原点で再生されます。2D で再生したいときは Play2D を使用してください。 こちらは PlayOption と同じ引数指定が可能なバージョンです。	

void SetTrackingObject(int instanceId, GameObject target)

インスタンスのポジションを追従させるオブジェクトを指定(3D サウンド指定ラベルのみ有効)	
引数	
instanceId	[in]インスタンス ID
target	[in]追従するゲームオブジェクト
戻り値	
なし	

説明

指定したインスタンス ID の音源位置を、指定したゲームオブジェクトのポジションに自動で更新されるようにゲームオブジェクトを登録します。ゲームオブジェクトが音源の再生終了よりも先に消滅する場合は、target に null を指定して登録解除してください。

void Stop(int instanceId, float fadeTime = -1)

インスタンスを停止する

引数

instanceId	[in]インスタンス ID
fadeTime	[in]フェードアウト秒数

戻り値

なし

説明

指定したインスタンスを停止します。フェードアウト秒数にマイナス値を指定するとラベルの設定に従ってフェードアウトを実行します。

IsPlayingLael()で再生中か調べたとき、フェードアウト中は停止中扱いになります。

void StopLabel(string labelName, float fadeTime = -1)

ラベル名を指定して再生中のインスタンスを停止する

引数

labelName	[in]停止するラベル名
fadeTime	[in]フェードアウト秒数

戻り値

なし

説明

指定したラベルの再生中インスタンスを停止します。

void StopCategory(string categoryName, float fadeTime = -1)

指定したカテゴリに所属するラベルを停止する

引数

categoryName	[in]停止するカテゴリ名
fadeTime	[in]フェードアウト秒数

戻り値

なし

説明

指定したカテゴリに所属するラベルの再生中インスタンスを停止します。

void StopMaster(string masterName, float fadeTime = -1)

指定したマスターが設定されているカテゴリに所属するラベルを停止する

引数

masterName	[in]停止するマスター名
fadeTime	[in]フェードアウト秒数

戻り値

なし

説明

指定したマスターが設定されているカテゴリに所属するラベルの再生中インスタンスを停止します。

void StopAll(float fadeTime = -1)

すべてのインスタンスを停止する

引数

fadeTime	[in]フェードアウト秒数

戻り値

なし

説明

すべての再生中インスタンスを停止する。

void OnPause(int instanceId, float fadeTime = -1)

インスタンスを一時停止する

引数

instanceId	[in]一時停止するインスタンス
fadeTime	[in]フェードアウト秒数

戻り値

なし

説明

指定したインスタンスを一時停止します。

void OffPause(int instanceId, float fadeTime = -1)

インスタンスの一時停止を解除する

引数	
instanceId	[in]一時停止解除するインスタンス
fadeTime	[in]フェードイン秒数
戻り値	
なし	
説明	
一時停止中インスタンスの再生を再開します。	

void OnPauseLabel(string labelName, float fadeTime = -1)

ラベル名を指定して再生中のインスタンスを一時停止する	
引数	
labelName	[in]一時停止するラベル名
fadeTime	[in]フェードアウト秒数
戻り値	
なし	
説明	
指定したラベルの再生中インスタンスを一時停止します。	

void OffPauseLabel(string labelName, float fadeTime = -1)

ラベル名を指定して再生中のインスタンスの一時停止を解除する	
引数	
labelName	[in]一時停止解除するラベル名
fadeTime	[in]フェードイン秒数
戻り値	
なし	
説明	
指定したラベルの一時停止中インスタンスの再生を再開します。	

void OnPauseCategory(string categoryName, float fadeTime = -1)

指定したカテゴリに所属するラベルを一時停止する	
引数	
categoryName	[in]一時停止するカテゴリ名
fadeTime	[in]フェードイン秒数
戻り値	
なし	

説明

指定したカテゴリに所属するラベルの再生中インスタンスを一時停止します。

void OffPauseCategory(string categoryName, float fadeTime = -1)

指定したカテゴリに所属するラベルの一時停止を解除する

引数

categoryName	[in]一時停止解除するカテゴリ名
fadeTime	[in]フェードイン秒数

戻り値

なし

説明

指定したカテゴリに所属するラベルの一時停止中インスタンスの再生を再開します。

void OnPauseMaster(string masterName, float fadeTime = -1)

指定したマスターが設定されているカテゴリに所属するラベルを一時停止する

引数

masterName	[in]一時停止するマスター名
fadeTime	[in]フェードイン秒数

戻り値

なし

説明

指定したマスターが設定されているカテゴリに所属するラベルの再生中インスタンスを一時停止します。

void OffPauseMaster(string masterName, float fadeTime = -1)

指定したマスターが設定されているカテゴリに所属するラベルの一時停止を解除する

引数

masterName	[in]一時停止解除するカテゴリ名
fadeTime	[in]フェードイン秒数

戻り値

なし

説明

指定したマスターが設定されているカテゴリに所属するラベルの一時停止中インスタンスの再生を再開します。

void OnPauseAll(float fadeTime = -1)

すべてを一時停止

引数

fadeTime	[in]フェードイン秒数

戻り値

なし

説明

再生中インスタンスをすべて一時停止します。

void OffPauseAll(float fadeTime = -1)

すべての一時停止を解除

引数

fadeTime	[in]フェードイン秒数

戻り値

なし

説明

すべての一時停止中インスタンスを再開させる。

void SetVolume(int instanceId, float newVolume, float moveTime)

再生中のインスタンスのボリュームを変更

引数

instanceId	[in]変更するインスタンス ID
newVolume	[in]ボリューム
moveTime	[in]移動時間(秒)

戻り値

なし

説明

インスタンスを指定してボリューム係数を変更します。

void SetVolume(string labelName, float newVolume, float moveTime)

指定したラベルの再生中インスタンスのボリュームを変更

引数

labelName	[in]変更するラベル名

	newVolume	[in]ボリューム
	moveTime	[in]移動時間(秒)
戻り値		
	なし	
説明		
	指定したラベルの再生中インスタンスのボリューム係数を変更します。	

float GetInstanceVolume(int instanceId)

インスタンスのボリュームを取得		
引数		
	instanceId	[in]取得するインスタンス ID
戻り値		
	なし	
説明		
	再生中インスタンスのボリューム係数を取得します。	

float GetInstanceCalcVolume(int instanceId)

インスタンスの最終的なボリュームを取得		
引数		
	instanceId	[in]取得するインスタンス ID
戻り値		
	なし	
説明		
	再生中インスタンスに最終的に設定されたボリュームを取得します。	

void SetMasterVolume(string masterName, float volume, float moveTime = 0)

マスターボリュームを変更		
引数		
	masterName	[in]変更するマスター名
	volume	[in]変更後のボリューム（0～1）
	moveTime	[in]移動時間(秒)
戻り値		
	なし	

説明

指定したマスターのボリューム係数を変更します。

float GetMasterVolume(string masterName)

マスターボリュームを取得（プログラム設定値）

引数

masterName	[in]取得するマスター名

戻り値

なし

説明

マスターのボリューム値を取得します。

void SetCategoryVolume(string categoryName, float volume, float moveTime = 0)

カテゴリボリュームを変更

引数

categoryName	[in]変更するカテゴリ名
volume	[in]変更後のボリューム（0～1）
moveTime	[in]移動時間(秒)

戻り値

なし

説明

指定したカテゴリのボリューム係数を変更します。

float GetCategoryVolume(string categoryName)

カテゴリボリュームを取得（プログラム設定値）

引数

categoryName	[in]取得するカテゴリ名

戻り値

なし

説明

カテゴリのボリューム値を取得します。

float GetLabelVolume(string labelName)

ラベルに設定されているボリュームを取得

引数

labelName	[in]取得するラベル名

戻り値

なし

説明

ラベルのボリューム値を取得します。プログラムの設定値ではありません。

void SetPitch(int instanceId, int newPitch, float moveTime)

再生中のインスタンスのピッチを変更

引数

instanceId	[in]インスタンス ID
newPitch	[in]変更後のピッチ(-1200～1200)
moveTime	[in]移動時間(秒)

戻り値

なし

説明

指定したインスタンスのピッチを変更します。moveTime を指定すると、現在の値から指定した値まで moveTime 秒をかけて線形に推移します。

void SetPitch(string labelName, int newPitch, float moveTime)

指定したラベルの再生中インスタンスのピッチを変更

引数

labelName	[in]変更するラベル名
newPitch	[in]変更後のピッチ
moveTime	[in]移動時間(秒)

戻り値

なし

説明

指定したラベルの再生中インスタンスのピッチを変更します。

void SetPan(int instanceId, int newPan, float moveTime)

再生中のインスタンスのパンを変更

引数		
	instanceId	[in]インスタンス ID
	newPan	[in]変更後のパン(-1～0～1)
	moveTime	[in]移動時間(秒)
戻り値		
	なし	
説明		
	指定したインスタンスのパンを変更します。moveTime を指定すると、現在の値から指定した値まで moveTime 秒をかけて線形に推移します。	

void SetPan(string labelName, int newPan, float moveTime)

指定したラベルの再生中インスタンスのパンを変更		
引数		
	labelName	[in]変更するラベル名
	newPan	[in] 変更後のパン(-1～0～1)
	moveTime	[in]移動時間(秒)
戻り値		
	なし	
説明		
	指定したラベルの再生中インスタンスのパンを変更します。	

void SetPosition(int instanceId, Vector3 position)

再生中のインスタンスが設定されているゲームオブジェクトのポジションを変更		
引数		
	instanceId	[in]インスタンス ID
	position	[in]移動後の座標
戻り値		
	なし	
説明		
	指定したインスタンスの座標を変更します。AudioSource の SpetialBlend が 0 以上の値のときのみ有効です。	

void SetPosition(string labelName, Vector3 position)

指定したラベルの再生中のインスタンスが設定されているゲームオブジェクトのポジションを変更
引数

	labelName	[in]変更するラベル名
	position	[in]移動後の座標
戻り値		
	なし	
説明		
	指定したラベルの再生中インスタンスの座標を変更します。	

void ResetPlayPositionAll()

再生開始位置をリセット
引数
戻り値
なし
説明
停止した位置から再生をはじめる設定になっているラベルの再生開始位置をすべてリセットする。

void SetDucking(string categoryName, float targetVolumeFactor, float fadeTime)

指定したカテゴリのダッキングを開始		
引数		
	categoryName	[in]ダッキングするカテゴリ名
	targetVolumeFactor	[in]ダッキング中の目標ボリューム係数
	fadeTime	[in]ボリューム変更にかかる秒数
戻り値		
	なし	
説明		
	指定したカテゴリのダッキングを開始します。 categoryName を fadeTime 秒かけてボリューム値 targetVolumeFactor まで変更します。解除するときは ResetDucking()または ResetDuckingAll()を使用します。	

void ResetDucking(string categoryName, float fadeTime)

指定したカテゴリのダッキングを解除		
引数		
	categoryName	[in]ダッキング解除するカテゴリ名
	fadeTime	[in]ボリューム変更にかかる秒数

戻り値

なし

説明

指定したカテゴリのダッキングを解除します。
SetDucking()でダッキングしたときのほか、ラベル再生時に自動でダッキングしたカテゴリをプログラム側で復帰させたい場合に使用します。

void ResetDuckingAll(float fadeTime)

すべてのダッキング設定を解除

引数

fadeTime	[in]ボリューム変更にかかる秒数

戻り値

なし

説明

すべてのカテゴリのダッキングを解除します。

void ForceResetDucking(string categoryName, float fadeTime)

強制でダッキング解除

引数

categoryName	[in]ダッキング解除するカテゴリ名
fadeTime	[in]ボリューム変更にかかる秒数

戻り値

なし

説明

指定したカテゴリのダッキングを解除します。
通常は ResetDucking を使用してください。ダッキングが正常に動作しなくなった場合にForceResetDucknng を試してください。自動でダッキングを行う設定になっていて、ForceResetDucking を呼ばないと解除されない・音量が戻っても次に再生を行うとダッキングが解除されないといった場合は不具合なのでご連絡ください。

void ForceResetDuckingAll(float fadeTime)

強制でダッキング解除

引数

fadeTime	[in]ボリューム変更にかかる秒数

戻り値

なし

説明

指定したカテゴリのダッキングを解除します。
通常は ResetDuckingAll を使用してください。ダッキングが正常に動作しなくなった場合に ForceResetDuckngAll を試してください。自動でダッキングを行う設定になっていて、ForceResetDuckingAll を呼ばないと解除されない・音量が戻っても次に再生を行うとダッキングが解除されないといった場合は不具合なのでご連絡ください。

void SetMute(bool onMute)

ミュートのON,OFFを設定

引数

onMute	[in]true:ミュート ON　false:ミュート OFF

戻り値

なし

説明

true を指定するとミュート設定が ON になり、すべてのサウンドの音量が 0 になります。false を指定するとミュート設定が OFF になり、すべてのサウンドの音量が復帰します。

bool GetMuteStatus()

ミュートがONかOFFかを取得

引数

戻り値

true:ミュート ON　false:ミュート OFF

説明

現在のミュート設定状況を取得します。

AudioDefine.INSTANCE_STATUS GetInstanceStatus(int instanceId)

インスタンスの状態を取得

引数

instanceId	[in]取得するインスタンス ID

戻り値

なし

説明

現在のインスタンスの状態を取得します。

AudioDefine.INSTANCE_STATUS.STOP	停止

AudioDefine.INSTANCE_STATUS.STOP_SOON	停止予定(フェードアウト中)
AudioDefine.INSTANCE_STATUS.PREPARE	準備完了
AudioDefine.INSTANCE_STATUS.PLAY	再生中
AudioDefine.INSTANCE_STATUS.PAUSE	ポーズ中
AudioDefine.INSTANCE_STATUS.PAUSE_SOON	ポーズ予定(フェードアウト中)

bool IsPlayingLabel(string labelName)

ラベルが再生中か取得する

引数

labelName	[in]ラベル名

戻り値

true:再生中　false:停止済み

説明

ラベルが再生中か調べます。指定したラベルが複数再生中だった場合、再生本数が 0 になったときに false になります。

int GetLabelNum()

登録済みのラベルの総数を取得

引数

戻り値

ラベル総数

説明

登録済みのラベルの総数を取得します。

string[] GetLabelNameList()

登録済みのラベルの名前リストを取得

引数

戻り値

ラベル名リスト

説明

登録済みのラベルの名前リストを取得します。

int GetCategoryNum()

登録済みのカテゴリの総数を取得

引数

戻り値

カテゴリ総数

説明

登録済みのカテゴリの総数を取得します。

string[] GetCategoryNameList()

登録済みのカテゴリの名前リストを取得

引数

戻り値

カテゴリ名リスト

説明

登録済みのカテゴリの名前リストを取得します。

int GetMasterNum()

登録済みのマスターの総数を取得

引数

戻り値

マスター総数

説明

登録済みのマスターの総数を取得します。

string[] GetMasterNameList()

登録済みのマスターの名前リストを取得

引数

戻り値

マスター名リスト

説明

登録済みのマスターの名前リストを取得します。

string GetCategoryNameSettingOfLabel(string labelName)

指定したラベルに設定されているカテゴリ名を取得

引数	
	labelName　　[in]調べるラベル名
戻り値	
	なし
説明	
	指定したラベルに設定されているカテゴリ名を取得します。

string GetMasterNameSettingOfCategory(string categoryName)

指定したカテゴリに設定されているマスター名を取得	
引数	
	categoryName　　[in]調べるカテゴリ名
戻り値	
	なし
説明	
	指定したカテゴリに設定されているマスター名を取得します。

float GetPlayTime(int instanceId)

インスタンスの現在の再生時間を取得	
引数	
	instanceId　　[in]取得するインスタンス ID
戻り値	
	現在の再生時間(秒)
説明	
	指定したインスタンス ID の現在の再生時間を秒単位で取得します。波形上何秒の位置かを取得するので、再生時にディレイを設定したときのディレイ部分は含みません。またループ設定のデータの場合も、例えば 30 秒でループするデータの２回目の 15 秒目の位置になったら総計の 45 秒ではなく 15 秒を返します。

int GetPlaySamples(int instanceId)

インスタンスの現在の再生サンプル位置を取得	
引数	
	instanceId　　[in]取得するインスタンス ID
戻り値	
	現在の再生サンプル位置
説明	

指定したインスタンス ID の現在の再生サンプル位置を取得します。

void SetTime(int instanceId, float time)

インスタンスの再生位置を時間で設定

引数

instanceId	[in]再生位置を変更するインスタンス ID
time	[in]再生位置(秒)

戻り値

なし

説明

指定したインスタンス ID の再生位置を時間で指定します。最大再生時間を越える値が指定された場合は何も行ないません。

void SetTimeSamples(int instanceId, int samples)

インスタンスの再生位置をサンプル単位で設定

引数

instanceId	[in] 再生位置を変更するインスタンス ID
samples	[in]再生位置

戻り値

なし

説明

指定したインスタンス ID の再生位置をサンプル単位で指定します。最大再生時間を越える値が指定された場合は何も行ないません。

string[] GetAudioClipNameLoadId(int loadId)

ロードIDに一致するラベルに指定されているAudioClip名のリストを取得

引数

loadId	[in]AudioClip 名リストを取得したいラベルに設定されているロード ID

戻り値

AudioClip 名のリスト

説明

ラベルに設定されている AudioClip 名のリストを取得します。
ロード ID を指定した場合、ロード ID に一致するラベルのみ対象にします。

string GetAudioClipName(string labelName)

指定したラベルに設定されているAudioClip名を取得

引数

	labelName	[in]AudioClip 名を取得したいラベル

戻り値

AudioClip 名

説明

ラベルに設定されている AudioClip 名を取得します。

string[] GetAudioClipNames(string labelName)

指定したラベルに設定されているAudioClip名を取得（ランダム設定されているラベルのAudioClip名も含む）

引数

	labelName	[in]AudioClip 名を取得したいラベル

戻り値

AudioClip 名

説明

ラベルに設定されている AudioClip 名を取得します。

string[]GetRandomSourceNames (string labelName)

指定したラベルに設定されているRandomSource名（ラベル名）を取得する

引数

	labelName	[in]AudioClip 名を取得したいラベル

戻り値

RandomSource 名（ラベル名）

説明

ラベルに設定されているランダム候補のラベル名リストを返します。

ランダム候補に含まれるラベルは、指定したラベル名自身も含まれます。

string[] GetAudioClipNameAll()

すべてのラベルに指定されているAudioClip名のリストを取得

引数

戻り値

AudioClip 名のリスト

説明

すべてのラベルに設定されている AudioClip 名のリストを取得します。

void SetAudioMixer(AudioMixer mixer)

UnityEngine.Audio.AudioMixerを登録

引数

mixer	[in]設定する AudioMixer

戻り値

なし

説明

AudioMixer をセットします。
ラベルを設定する前に Mixer のセットを行なってください。

float GetLabelLength(string labelName)

ラベルの総再生時間を秒単位で取得

引数

labelName	[in]ラベル名

戻り値

指定したラベルに設定されているオーディオファイルの総再生秒数

int GetLabelSamples(string labelName)

ラベルの総再生時間をサンプル単位で取得

引数

labelName	[in]ラベル名

戻り値

指定したラベルに設定されているオーディオファイルの総サンプル数

bool GetSpectrumData(int instanceId, float[] sample, int channel, FFTWindow window)

Snapshotを切り替える

引数

instanceId	[in]取得するインスタンス ID
sample	[out]オーディオサンプルの出力配列。２の累乗を指定。
channel	[in]サンプルのチャンネル

window	[in]窓関数のタイプ
戻り値	
true:成功　false:失敗	
説明	
現在の Snapshot から、AudioMixer に登録されている Snapshot へ切り替えます。	

bool IsLoop(string labelName)

ラベルのループ設定を取得	
引数	
labelName	[in]ラベル名
戻り値	
true:ループする　false:ループしない	

int GetLabelMaxPlaybacksNum(string labelName)

ラベルの最大発音数を取得	
引数	
labelName	[in]ラベル名
戻り値	
ラベルに設定されている最大発音数、0 は無限、-1 は失敗	

int GetCategoryMaxPlaybacksNum(string categoryName)

カテゴリの最大発音数を取得	
引数	
labelName	[in]ラベル名
戻り値	
カテゴリに設定されている最大発音数、0 は無限、-1 は失敗	

int GetCategoryMaxPlaybacksNumFromLabel(string labelName)

ラベルに指定されているカテゴリの最大発音数を取得	
引数	
labelName	[in]ラベル名
戻り値	
ラベルに設定されているカテゴリの最大発音数、0 は無限、-1 は失敗	

bool IsInterval(string labelName)

指定したラベルがインターバル期間中か取得

引数

	labelName	[in]ラベル名

戻り値

true:インターバル期間中　false:インターバル期間中ではない

int GetCurrentPlayNum()

停止予定も含む現在アクティブなプレイヤー数を取得

引数

戻り値

停止予定も含む、アクティブなプレイヤー数を返す

void UnsetUnityMixerInfo()

UnityEngine.Audio.AudioMixer情報を削除

引数

戻り値

なし

説明

AudioMixer の登録を解除します。

void SetSnapshot(string snapName, float time)

Snapshotを切り替える

引数

	snapName	[in]スナップショットの名前
	time	[in]パラメータの変更にかかる時間(秒)

戻り値

なし

説明

現在の Snapshot から、AudioMixer に登録されている Snapshot へ切り替えます。

void SetAudioMixerExposedParam(string paramName, float value)

Exposed ParametersでMixerの値を変更

引数

paramName	[in]Unity 上で設定された名前
value	[in]paramName と関連付けられたパラメータに設定する数値

戻り値

なし

説明

AudioMixer 上のパラメータをスクリプトから変更するための機能です。この機能を使うには AudioMixer の ExposedParameters に変更したいパラメータの関連付けを追加する必要があります。
数値範囲は Unity 上のパラメータの数値範囲に従うので、ボリュームは dB 値で扱います。数値範囲は割り当てたパラメータにより異なるので、使用する際は注意してください。

USndPlugin.IsMusicPlaying()

組み込みのミュージックアプリケーションが再生中か取得する。(iOSのみ)

引数

戻り値

true:再生中　false:再生していない

説明

iOS の組み込みのミュージックアプリケーションが再生中か判定します。

USndPlugin.IsOtherAudioPlaying()

起動中のその他のアプリケーションが音声を再生中か取得する。(iOSのみ)

引数

戻り値

true:再生中　false:再生していない

説明

組み込みのミュージックアプリのほか、起動中のアプリケーションが音声を再生中か調べます。他のメーカーのアプリケーション以外にも、システム側の発音である電話着信、アラーム、Siri の音声再生も判定に含みます。
アプリ切り替わり時に OS が自動的にフェードアウトを掛けている場合があり、そのときのフェードアウトも再生中と判定されるので、IsOtherAudioPlaying()で判定を行うときはアプリケーションの Suspend,Resume 以外のタイミングでも判定を行うようにしてください。

USndPlugin.IsSpeaker ()

端末スピーカーから再生されているか取得する。(iOS, Android)

引数

戻り値

true:端末スピーカーから再生中　false:ヘッドホンから再生中

説明

端末スピーカーから再生中のときは true, ヘッドホンから再生中のときは false を返します。iOS と Android 以外の OS では false を返します。

USndPlugin.IsMannerMode ()

マナーモードが設定されているか取得。(Android)

引数

戻り値

true:マナーモード設定中　false:マナーモードではない

説明

マナーモード設定中のときは true, マナーモードが設定されていないときは false を返します。Android 以外の OS では false を返します。

iOS のサイレントスイッチは OS 側で処理されるので、何もしなくても音量は連動して自動で変更されます。Android も USnd 側でマナーモードに必要な処理は行っているので自動的に状態が反映されます。UI などマナーモード設定中なのか判定して何か行いたいときにこちらを使用してください。