AquesTalk iOS マニュアル

（株）アクエスト

http://www.a-quest.com/

1. 概要

本文書は、規則音声合成ライブラリ AquesTalk iOS をアプリケーションに組み込んで使用するためのプログラミングに関しての方法および注意点を示したものです。

AquesTalk はかな表記のよみ記号にアクセント等の情報を付与した音声記号列から音声波形データを生成する日本語の音声合成ライブラリです。

AquesTalk iOS は、非常にコンパクトな音声合成エンジンで、これを用いれば、iPhone、iPod Touch、iPad などのアプリケーションに簡単に音声合成機能を実現することができます。

AquesTalk iOS は、メモリ上に音声波形を生成する基本的な機能をもっています。生成する波形データのフォーマットは WAV 形式で同じす。また、同期、非同期に音声を内蔵スピーカから直接出力する DA(サウンド出力)機能も持っています。

なお、漢字かな混じり文からの音声合成には、別途、言語処理ライブラリ AqKanji2Koe を利用してください。

AquesTalk の入力である音声記号列の文字コードは Shift JIS または UTF-8 で、それぞれ関数名が異なります。AquesTalk に与える文字コードは必ずプログラム上で指定の文字コードに変換して用いてください。

特長

・耳なじみの良い声
　聴きやすさや明瞭性を重視し、人の声にこだわらずに開発しました。

・超小型軽量
　他に類を見ない少ないリソース量で音声合成が実現できます。

・簡単に実装
　簡単なインターフェース関数。
　ファイルアクセスも不要なため、ライブラリをリンクするだけで簡単に使用できます。

2. 仕様

入力	かな表記音声記号列　SJIS または UTF-8
出力	WAV フォーマット(8KHz サンプリング, 16bitPCM, モノラル)バイナリデータ
	またはサウンドデバイスへ出力
声種	基本1種
ライブラリ形式	スタティックライブラリ
	armv7/armv7s/arm64/i386/x86_64 Universal Binary, 各々bitcode を含む
関数 I/F	C関数呼び出し （Objective-C や swift から呼び出し可能）
OS	iOS 5.1.1 以上
デバイス	iPhone / iPod Touch / iPad
マルチスレッド	対応
ライブラリサイズ	約 120KB（リンク後）
外部依存 framework	AudioToolbox.Framework、libc++
動作確認環境	Xcode 7.1

3. アーキテクチャと Universal Binary

libAquesTalk.a は、Simulator(シミュレータ)用と Device(実機)用の2つの異なるアーキテクチャのライブラリがひとつにパックされている Universal Binary となっています。Xcode がビルド時にターゲットにあわせて適切な方を選択しますので、開発者がターゲットに応じて切り替える必要は有りません。

なお、本ライブラリには、Device 用として、armv7(iPhone4S 以前)、armv7s(iPhone5)、armv64(iPhone5s 以降)の3つ、Simulator 用として i386 と x86_64 の2つの計5つのアーキテクチャのライブラリが含まれています。

さらに、bitcode にも対応しましたので、iOS9 からのアプリ最適化の仕組み AppThinning 用の bitcode に対応したアプリを作成できます。

4. 関数 API

[メモリ上に音声波形を生成する]

AquesTalk_iPhone.h

AquesTalk_Synthe

説明	音声記号列(SJIS)から音声波形を生成します 生成した音声データは、使用後、呼び出し側で free()してください。
構文	unsigned char * **AquesTalk_Synthe**(const char **koe*, int *iSpeed*, int **size*)
引数	
koe	音声記号列 (SJIS、 C 言語文字列)
iSpeed	発話速度[%] 50-300 の間で指定 デフォルト:100 値を大きく設定するほど、速くなる
size	生成した音声データのサイズが返る[byte](エラーの場合はエラーコードが返る)
戻り値	WAV フォーマットのデータ(内部で領域確保、解放は呼び出し側で AquesTalk_FreeWave()で行う)の先頭アドレスを返す。エラー時は、nil を返す。このとき size にエラーコードが設定される。

AquesTalk_iPhone.h

AquesTalk_Synthe_UTF8

説明	音声記号列から音声波形を生成します 生成した音声データは、使用後、呼び出し側で free()してください。
構文	unsigned char * **AquesTalk_Synthe_Utf8**(const char **koeUtf8*, int *iSpeed*, int **size*)
引数	
koeUtf8	音声記号列(UTF-8、 C 言語文字列)
iSpeed	発話速度[%] 50-300 の間で指定 デフォルト:100 値を大きく設定するほど、速くなる
size	生成した音声データのサイズが返る[byte](エラーの場合はエラーコードが返る)
戻り値	WAV フォーマットのデータ(内部で領域確保、解放は呼び出し側で AquesTalk_FreeWave()で行う)の先頭アドレスを返す。エラー時は、nil を返す。このとき size にエラーコードが設定される。

AquesTalk_iPhone.h

AquesTalk_FreeWave

説明	音声データの領域を開放。 AquesTalk_Synthe()と対で使用します。
構文	void **AquesTalk_FreeWave** (unsigned char *wav)
引数	なし
wav	WAV フォーマットのデータ(AquesTalk_Synthe()で生成した音声データ)
戻り値	なし

[デバイスに音声出力する(同期タイプ)]

AquesTalk_iPhone.h

AquesTalkDa_PlaySync

説明	同期タイプの音声合成。デバイスに音声出力する。発声が終了するまで戻らない。
構文	int **AquesTalkDa_PlaySync**(const char **koe*, int *iSpeed*)
引数	
koe	音声記号列 (SJIS、 C 言語文字列)
iSpeed	発話速度[%] 50-300 の間で指定 デフォルト:100 値を大きく設定するほど、速くなる
戻り値	0:正常終了 それ以外:エラー

AquesTalk_iPhone.h

AquesTalkDa_PlaySync_Utf8

説明	同期タイプの音声合成。デバイスに音声出力する。発声が終了するまで戻らない。
構文	int **AquesTalkDa_PlaySync_Utf8**(const char **koeUtf8*, int *iSpeed*)
引数	
koeUtf8	音声記号列 (UTF-8、 C 言語文字列)
iSpeed	発話速度[%] 50-300 の間で指定 デフォルト:100 値を大きく設定するほど、速くなる
戻り値	0:正常終了 それ以外:エラー

[デバイスに音声出力する(非同期タイプ)]

AquesTalk_iPhone.h

AquesTalkDa_Create

説明	音声合成エンジンのインスタンスを生成(非同期タイプ)
構文	H_AQTKDA **AquesTalkDa_Create**()
引数	
戻り値	音声合成エンジンのハンドル

AquesTalk_iPhone.h

AquesTalkDa_Release

説明	音声合成エンジンのインスタンスを解放(非同期タイプ)
	発声終了前にこの関数でインスタンス解放すると、その時点で発声が終了してしまうので注意
構文	void **AquesTalkDa_Release**(H_AQTKDA *hMe*)

引数

hMe	音声合成エンジンのハンドル
戻り値	なし

AquesTalk_iPhone.h

AquesTalkDa_Play

説明	非同期タイプの音声合成。発声終了を待たずに戻る。デバイスに音声出力する。 notification を指定すると、再生終了後に指定したイベントが通知される。 ※phontDat および notification を指定した場合は、発声が完了するまで、これらを開放してはいけません。
構文	int **AquesTalkDa_Play**(H_AQTKDA *hMe,* const char *koe, int *iSpeed*, NSNotification* notification)

引数

hMe	音声合成エンジンのハンドル
koe	音声記号列（SJIS、 C 言語文字列）
iSpeed	発話速度[%] 50-300 の間で指定 デフォルト:100 値を大きく設定するほど、速くなる
notification	再生終了イベントの指定。イベント通知不要のときは nil を指定します。
戻り値	0:正常終了 それ以外:エラー

AquesTalk_iPhone.h

AquesTalkDa_Play_Utf8

説明	非同期タイプの音声合成。発声終了を待たずに戻る。デバイスに音声出力する。 notification を指定すると、再生終了後に指定したイベントが通知される。 ※phontDat および notification を指定した場合は、発声が完了するまで、これらを開放してはいけません。
構文	int **AquesTalkDa_Play_Utf8**(H_AQTKDA *hMe,* const char *koeUtf8, int *iSpeed*, NSNotification* notification)

引数

hMe	音声合成エンジンのハンドル
koeUtf8	音声記号列（UTF-8、 C 言語文字列）
iSpeed	発話速度[%] 50-300 の間で指定 デフォルト:100 値を大きく設定するほど、速くなる
notification	再生終了イベントの指定。イベント通知不要のときは nil を指定します。
戻り値	0:正常終了 それ以外:エラー

AquesTalk_iPhone.h

AquesTalkDa_Stop

説明	発声の停止。Play()で発声中に使用する。 再生中に Stop()によって発声が終了した場合、、notification が指定されていたなら、再生終了イベントが通知される。 再生終了後に呼び出すと、何もしない。

構文 void **AquesTalkDa_Stop**(H_AQTKDA *hMe*)

引数

hMe 音声合成エンジンのハンドル

戻り値 なし

AquesTalk_iPhone.h

AquesTalkDa_IsPlay

説明 再生中か否か

構文 int **AquesTalkDa_IsPlay**(H_AQTKDA *hMe*)

引数

hMe 音声合成エンジンのハンドル

戻り値 1:再生中 0:再生中でない

5. エラーコード表

関数が返すエラーコードの内容は、次の通りです。

値	内容
100	その他のエラー
101	メモリ不足
102	音声記号列に未定義の読み記号が指定された
103	韻律データの時間長がマイナスなっている
104	内部エラー(未定義の区切りコード検出)
105	音声記号列に未定義の読み記号が指定された
106	音声記号列のタグの指定が正しくない
107	タグの長さが制限を越えている(または[>]がみつからない)
108	タグ内の値の指定が正しくない
109	WAVE 再生ができない(サウンドドライバ関連の問題)
110	WAVE 再生ができない(サウンドドライバ関連の問題 非同期再生)
111	発声すべきデータがない
200	音声記号列が長すぎる
201	1つのフレーズ中の読み記号が多すぎる
202	音声記号列が長い(内部バッファオーバー1)
203	ヒープメモリ不足
204	音声記号列が長い(内部バッファオーバー1)

6. 音声記号列

本ライブラリの入力である音声記号列の書き方、仕様については、付属の「音声記号列仕様」を参照ください。

7. 音声データ形式

本ライブラリで生成する音声データは、次の形式となります。
AquesTalk_Synthe()等で生成する音声データには、先頭部に WAV ヘッダが付与されています。
ストレート PCM データが必要な場合は、別途ヘッダを除いて使用してください(先頭の44バイト)。
ほかのフォーマットをご希望の場合はカスタマイズを承りますので、お問い合わせください。

サンプリング周波数	8KHz
量子化 bit 数	16bit
チャンネル数	モノラル
エンコード	リニア PCM
フォーマット	WAV 形式

8. アプリケーションへの実装方法

8.1. ヘッダ、ライブラリ

AquesTalk ライブラリをアプリケーションに組み込むときは、ヘッダファイル(AquesTalk_iPhone.h)をインクルードし、ビルド時に libAquesTalk.a をリンクするようにします。

具体的な方法として、ヘッダファイルは AquesTalk_iPhone.h ひとつだけですので、アプリケーションのソースファイルと同じディレクトリにコピーして、ソース内で次のようにインポートすると良いでしょう。

```
#import "AquesTalk_iPhone.h"
```

libAquesTalk.a(評価版の場合は libAquesTalkEva.a)も、ソースファイルと同じディレクトリにコピーしてから、Xcode 上のメニュ>File>Add Files to"..."にてプロジェクトに追加します。この操作により、Project Navigator(通常画面左側のプロジェクトで使用するファイルリスト)に、libAquesTalk.a が追加されていることがわかります。

8.2. 外部 Framework

本ライブラリは、AudioToolbox.Framework と libc++.tbd の2つ外部フレームワーク(ライブラリ)を使用します。前者はサウンドデバイスへの音声出力機能のため、後者は C++の標準ライブラリです。
これらを指定し忘れるとアプリのビルド時に関数が見つからないというリンクエラーが発生します。

アプリのプロジェクトにこれらを追加する方法は、Target の設定画面＞General＞Linked Frameworks and Libraries で＋部分を押下して追加できます。この操作により、Project Navigator(通常画面左側のプロジェクトで使用するファイルリスト)に、AudioToolbox.Framework と libc++.tbd が追加されていることがわかります。

9. サンプルプログラム

本パッケージにはサンプルプログラムのプロジェクト一式(xcode 7)が含まれています。

HelloAqTk は、任意の音声記号列を指定して Play ボタンを押下すると、合成音声を発声するアプリケーションです。

9.1. 実行方法

1. アプリケーションプロジェクトを開く

HelloAqTk.xcodeproj をダブルクリックして xcode でプロジェクトを開きます。

2. ビルド・実行

メニュー>Product>Run でビルド・実行してください。
エラーが無ければ、右上のようなアプリ画面が立ち上がり、
テキストボックスに任意の音声記号列を入力して[Play]ボタンのクリックで音声が聞こえれば OK です。
なお、再生中は、図の[Done]の部分が[Playing...]になります。

9.2. コード説明

HelloAqTk サンプルアプリでは、音声合成を非同期に行い、サウンドデバイスへ出力する一連の動作を行っています。

ViewController.h では、UI オブジェクトの他に、AquesTalk のオブジェクト（インスタンス）を保存する変数 m_pAqTk を追加しています。

ViewController.h

```
#import <UIKit/UIKit.h>
#import "AquesTalk_iPhone.h"

@interface ViewController : UIViewContoller
{
    IBOutlet UITextField* textfield;
    IBOutlet UILabel* onPlayLabel;
    H_AQTKDA m_pAqTk;     //AquesTalk エンジンのインスタンス
}
- (IBAction)play:(id)sender;
- (IBAction)stop:(id)sender;
@end
```

次は、ViewController.m のコードの抜粋です。
本サンプルでは viewDidLoad()内で AquesTalk のインスタンスを生成し、dealloc()にてインスタンスを解放しています。このインスタンスは発声終了時まで維持する必要があります。どのタイミングで AquesTalk のインスタンスを生成し解放するかはアプリでの音声合成の利用方法により適宜変更してください。

また、発声終了のタイミングでイベントを受け取れるように Notification の登録をします。ここでは、イベントのメソッド名は「daDone」としましたが、任意でかまいません。また、イベントの名前も「AquesTalkDaDoneNotify」としましたが、これも任意でかまいません。この処理を追加することによって、音声合成が終了したタイミングで daDone メソッドが呼ばれるようになります。

メソッド play は、Play ボタンをタップされたときに呼び出されるメソッドです。
まず、終了通知イベントを作成します。name は、は applicationDidFinishLaunching で指定したものと同じ文字列を指定します。
次に、テキストボックスから NSString の文字列を取得し、これを Shift JIS へ文字コードを変換します。
その後、AquesTalkDa_Play()で音声合成を開始します。3 番目の引数は話速です。4 番目の引数は、終了通知イベントを指定します。この関数は同期型なので、発声の終了を待たずに返ります。
戻り値が 0 の場合は正常終了、それ以外はエラーとなります。本サンプルでは、エラーの場合はアラート表示を出すようにしています。

ViewController.m

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // AquesTalk インスタンス生成
    m_pAqTk = AquesTalkDa_Create();

    // Notification の登録　発声終了を検知しなければ不要　メソッド名や name は任意で
    [[NSNotificationCenter defaultCenter]
     addObserver:self
     selector:@selector(daDone:)
     name: @"AquesTalkDaDoneNotify"
     object:nil];
}

- (void)dealloc {
    if(m_pAqTk) AquesTalkDa_Release(m_pAqTk);
    [[NSNotificationCenter defaultCenter] removeObserver:self];
}

-(IBAction)play:(id)sender
{
    //終了通知イベントの作成　name 部分は NSNotificationCenter に設定した値に合わせる
    NSNotification *notification = [NSNotification
```

```
                notificationWithName:@"AquesTalkDaDoneNotify" object:self userInfo:nil];

    // テキストボックスから文字列取得し、文字コードを ShiftJIS に変換し、音声合成出力（非同期）
    char *sjis = (char*)[[textfield text] cStringUsingEncoding:NSShiftJISStringEncoding];
    int iret = AquesTalkDa_Play(m_pAqTk, sjis, 100, notification);

    if(iret!=0){ // エラーの場合はアラート表示
        UIAlertController *alertController = [UIAlertController
          alertControllerWithTitle:@"Error" message:@"音声記号列の指定が正しくありません"
          preferredStyle:UIAlertControllerStyleAlert];
        [alertController  addAction:[UIAlertAction actionWithTitle:@"はい"
          style:UIAlertActionStyleDefault handler:^(UIAlertAction *action) { }]];
        [self presentViewController:alertController animated:YES completion:nil];
        return;
    }
    onPlayLabel.text = @"Playing...";
}

-(IBAction)stop:(id)sender
{
    AquesTalkDa_Stop(m_pAqTk);
}
- (void)daDone:(NSNotification*)notification
{
    onPlayLabel.text = @"Done";
}
・
・
・
```

このサンプルは、非同期の音声合成でデバイスに直接合成音声を出力するものですが、この他に音声データをメモリ上(配列データ)に取得する関数もあり、生成した音声を任意の方法で転送したり出力することも可能です。
また、本サンプルは Objective-C で記述していますが、swift でも使用可能です。具体的なコーディングについてはネットで検索してください。

10. ライセンス

本ライブラリを含んだアプリケーションを使用する場合は「使用ライセンス」の購入が必要となります。また、本ライブラリを含んだアプリの配布には事前に当社と『配布ライセンス契約』が必要となります。ライセンスの詳細は、本パッケージ内のライセンスドキュメントを参照ください。

11. サポート

バグや不具合に関しては、infoaq@a-quest.com にご連絡いただければ幸いです。
使用方法についてのご質問は、BBS http://www.a-quest.com/aquestalk/patio/patio.cgi に投げていただければと思います。

12. 履歴

日付	版	変更箇所	更新内容	更新者
2011/02/03	1.0	新規作成	AquesTalk2 iOS から加筆修正	N.Y
2012/10/01	1.1	Ver.1.4 に対応	iOS6 に対応 iOS4.2.1 以前は非対応に	N.Y
2013/07/08	1.2	Ver.1.5 に対応	UTF8 関数の追加	N.Y
2014/12/22	1.3	Ver.1.6 に対応	iOS 64bit(arm64)に対応	N.Y
2016/03/19	1.4	Ver.1.61 に対応	Xcode7 用に、x86_64,bitcode 記述追加	N.Y