

# AquesTalk2 Android Manual

## 1. 概要

本文書は、規則音声合成ライブラリ AquesTalk2 Android 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。

AquesTalk2 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。

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 を利用します。

### 1.1. 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  
声種の切り替えは簡単で、システムに複数の声種を入れることもできます
- ・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

### AquesTalk2

ライブラリ	so 形式ダイナミックライブラリ (JNI 実装)
形式	各種 ABI に対応 armeabi-v7a, arm64-v8a, x86, x86_64, riscv64
対応 OS	Android 5 (API レベル 21) 以降
入力データ	音声記号列 (UTF-8)
形式	
出力データ	WAV フォーマットデータ (8KHz サンプリング, 16bitPCM)
形式	
開発環境	Android NDK r29
プログラム	約 58KByte
サイズ	
Phont デー	約 16KB/声種
タサイズ	

### 3. 関数 API

#### 3. 1. Android 版 Java ラップクラス

JNI をラップした Java クラスを用いてアクセスします

syntheWav		package aquestalk2
説明	音声記号列から音声データを生成します	
構文	byte[] syntheWav(String kanaText, int speed, byte[] phontDat)	
引数		
kanaText	音声記号列 (UTF-8)	
speed	発話速度 (%) 50 - 300	
phontDat	Phont データ デフォルトを用いるときは null を指定	
戻り値	wav フォーマットのデータ エラー時には、長さ1で、先頭にエラーコードが返される	

### 4. ファイル

ビルド及び実行時に下記のファイルを使用します。

libAquesTalk2.so : AquesTalk2 ライブラリの実体 (ネイティブコード)  
AquesTalk2.java : libAquesTalk2.so を Java からアクセスするためのラッパークラス  
phont/ : Phont ファイル (複数)

任意のアプリケーションを開発するときは、libAquesTalk2.so と AquesTalk2.java ファイルを、アプリケーションのプロジェクトに追加します。Android Studio の環境であれば、プロジェクトディレクトリに、上のファイルを下記のようなディレクトリ構造上に (ディレクトリがなければ作成する) コピーします。

「AquesTalk2 Android」の実体はネイティブコードで書いてあり、JNI 経由で呼び出しています。そのため、使用するシステム環境 (ABI) に応じたライブラリを使用する必要があります。たとえば、使用端末が ARM CPU の場合は arm64-v8a か armeabi-v7a を、Intel CPU の場合は x86, x86\_64 などのライブラリを使用します。

Andorid では FAT-Binary (シングル APK) によって、複数の異なる ABI のライブラリを1つの apk に含めることができ、動作マシン側で自動的に選んで実行します。そのため、apk のサイズさえ気にしなければ、SDK パッケージの jniLibs 以下すべてをアプリのプロジェクト側にコピーしても良いでしょう。

```
/ <app/src/main>
  / java
    / <アプリのソース>
      / aquestalk2
        / AquesTalk2.java
  / jniLibs
    / arm64-v8a
      / libAquesTalk2.so
    / armeabi-v7a
      / libAquesTalk2.so
    / x86
      / libAquesTalk2.so
    / x86_64
      / libAquesTalk2.so
  / res
```

```
.
/raw
  /aq_f1c.phont
  /aq_f3a.phont
.
```

なお、呼び出しの関係は、次のようになります。

\* アプリ -> AquesTalk2.java -> libAquesTalk2.so

#### 4.1. Phont ファイル

本 SDK パッケージには、phont ディレクトリに、声種を規定する phont ファイルがいくつか含まれています。なお、この中の aq\_rm.phont は、DLL に内臓のデフォルト Phont と同じものです。今後、公開される新しい Phont ファイルは、別途ダウンロードしてお使いください。

### 5. アプリへの実装方法

任意のアプリに AquesTalk2 を組み込む方法を SDK に含まれているサンプルアプリのコードを参照しながら以下に示します。

(samples¥AqTk2App¥app¥src¥main¥java¥com¥a\_quest¥aquestalk2¥AqTkApp2.java)

AquesTalk2 のクラスをインポートします。

```
import aquestalk2.AquesTalk2;
```

AquesTalk2.java の中を見ればわかりますが、静的メソッドと、通常のメソッドが用意されています。いまのところ、関数は1つだけなので、どちらを使っても違いはありません。AqTkApp2.java では、アプリ起動時の onCreate()のなかで、インスタンスを生成しています。

```
aquestalk2 = new AquesTalk2();
```

そして、発声ボタンが押されたときに、音声記号列 と発話速度を取得して、次のように音声データをメモリ上に生成します。なお、音声記号列の文字コードは UTF-8 で指定します (Android のデフォルトのままでも ok)。最期の引数には Phont データを指定します。デフォルトの Phont(声種)を使用するときは、null で構いません。Phont データの指定方法は、後述します。

```
onPlayBtn():
    byte[] wav = aquestalk2.syntheWav(koe, speed, null);
```

ここで得られる音声データは wav フォーマット (8KHz, Mono, Straight PCM) です。音声記号列が不正な場合など生成エラーの時は、byte[] のサイズが 1 になり、配列の先頭にエラーコードが返されます。エラーコードは、samples¥AqTk2App¥src¥aquestalk2¥AquesTalk2.java の後ろに記述されています。

現時点では、「AquesTalk Android」に直接音声を出力するメソッドは用意されていないので、音声再生はアプリ側で記述する必要があります。AudioTrack クラスを用いて音声出力するコードの例が AqTk2App.java にありますので、参考にしてください。

注意すべき点は、AquesTalk2 の出力は wav フォーマットであるのに対し、AudioTrack で再生するデータは、ヘッダを含まない生のデータ列を指定する必要があります。そこで、生成した音声データの先頭にある WAV ヘッダ (44 バイト) を除いて、それ以降を AudioTrack に渡します。また、データの長さ (サンプル

数)は、1つのサンプルが2バイトですので、次のようになります。

```
サンプル数=(wav.length-44)/2
```

#### Phont の指定方法

`aquestalk2.syntheWav()` の最後の引数に Phont データを指定することが出来ます。これにより異なる声種で合成が可能になります。

各種の Phont ファイル(拡張子が.phont)が、AquesTalk2 ライブラリのパッケージ中(/phont フォルダ内)に含まれています。なお、Phont ファイルは、Windows 版その他のプラットフォーム間で互換がありますので、今後公開される Phont ファイルをダウンロードして利用することもできます。

Android アプリプログラムで Phont を指定する方法ですが、Phont ファイルを Android 規定の場所に配置してアプリからファイルを読み込んでも良いのですが、ここではアプリのリソースに Phont データをいれて、それを実行時に読み出す方法を示します。

まず、Phont ファイルをリソースに含まれるようにします。

エクスプローラなどでアプリパッケージのフォルダを見ると、res フォルダがあるはずです。その下に raw フォルダを作成し、さらにその下に使用する各 Phont ファイルをコピーします。Android Studio 上の Project で確認すると、次のようになります。これでビルド後にアプリパッケージの中に自動的に Phont データが含まれるようになります。

`AqTkApp2.java` 中の `AqTk2App.LoadPhont()` は、Phont 名(ファイル名の拡張子を除いたもの)を指定して、Phont データをリソースから byte 配列に読み込むコードです。

当該 Phont のリソース ID は、`android.content.res.Resources` クラスの `getIdentifier()` で、リソース名から取得しています。その後、`InputStream` を使って、Phont データを `byte[]` に読み込んでいます。この関数の戻り値を、`aquestalk2.syntheWav()` の最後の引数に指定すれば、その声種で音声合成ができるようになります。

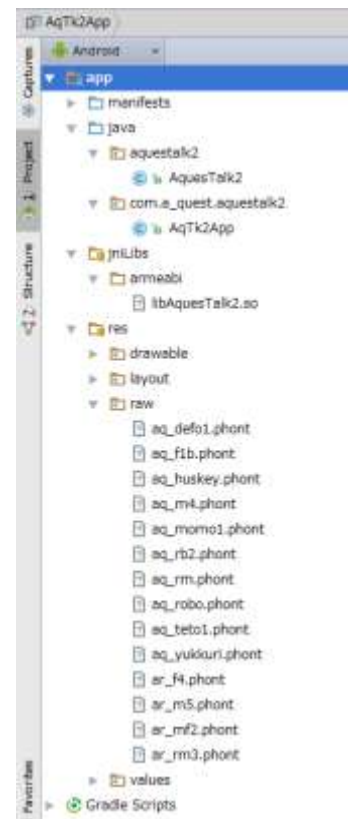
## 6. サンプルアプリ

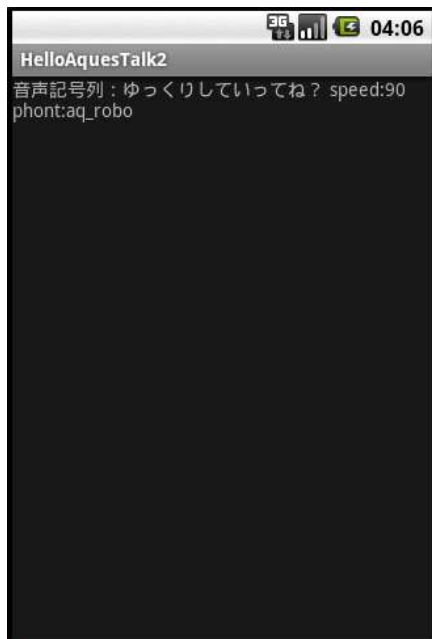
SDK パッケージには、ライブラリと一緒に、サンプルアプリが2つ

(`HelloAquesTalk2`, `AqTk2App`)入っています。

サンプルアプリのソースコード(ライブラリやライブラリのヘッダファイルは含まれません)は、ご自由に改変してご使用いただけます。

- \* `HelloAquesTalk2`:  
最も単純なサンプルで、起動時に規定のメッセージを合成出力するだけのものです。
- \* `AqTkApp2`:  
GUI を持たせ、任意の音声記号列と話速、声種を指定して音声合成するものです。





#### サンプルアプリの動かし方

Android Studio の開発環境が構築されていることが前提です。以下に HelloAquesTalk2 アプリを動かす手順を示します。もうひとつのサンプル AqTk2App や、独自のアプリの場合も同様の手順で使用できます。

1. SDK パッケージ内の samples/HelloAquesTalk2 フォルダを任意の場所にコピー
2. SDK パッケージ内の java/aquestalk2 フォルダを HelloAquesTalk2/app/src/main/java フォルダにコピー
3. SDK パッケージ内の phont フォルダの各 xxx.phont ファイルを、HelloAquesTalk2/app/src/main/res/raw フォルダ内にコピー
4. SDK パッケージ内の jniLibs フォルダ以下を HelloAquesTalk2/app/src/main フォルダにコピー
5. Android Studio を起動
6. menu>File>Open でコピーした HelloAquesTalk2 フォルダを Open
7. 各種設定ファイルや build が自動的に行われます
8. あとは、menu>Run>Run で実行

## 7. 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になっている
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[>]が見つからない)
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)
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)

## 8. 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、SDK 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。

## 9. 履歴

日付	版	変更箇所	更新内容
2011/01/25	1.0	新規作成	
2016/03/11	2.4		SDK Ver.2.4 、 IDE を AndroidStudio に変更
2019/05/29	2.4.1		Android Studio 3.4 用に修正
2023/01/31	2.4.2		「サンプルアプリの動かし方」に加筆
2025/09/06	2.4.3		ライブラリの VerUp に応じて仕様を更新