



FlashAir™ Tutorial

FlashAir のチュートリアル

FlashAir+Azure で「しゃべるデジカメ」

1	用意するもの	4
2	サブスクリプションキーの取得(無料)	5
3	本体機能の作成(Lua スクリプト)	11
4	ビューアーの作成(JavaScript)	19
5	FlashAir の準備	30
6	実行	31
7	参考	32



FlashAir Developers

FlashAir Tutorial -FlashAir のチュートリアル
FlashAir+Azure で「しゃべるデジカメ」

2016 年 8 月 11 日 第 1 版第 2 刷発行

著者	綾瀬ヒロ
表紙イラスト	じむ
編集	綾瀬ヒロ
発行	FlashAir Developers
連絡先	著者
	Twitter: @ayasehiro
	メール: ayase@big.or.jp
	または
	support@flashair-developers.com

FlashAir+Azure で「しゃべるデジカメ」

FlashAir 上の Lua スクリプトで Azure Cognitive Services の Computer Vision API を使って画像にキャプション情報を付加し、キャプション情報を JavaScript で Bing Speech API の Bing Text To Speech API を使ってしゃべらせるチュートリアルです。

FlashAir をインターネットに接続し、Lua 機能及び Web サーバ機能を用いて Azure Cognitive Services の環境に接続することで Microsoft 社の API を活用したシステムの構築が可能となります

本書では、デジタルカメラで撮影した写真に、Computer Vision API を使って写真の状況を分析したキャプションを付け、これをスマートフォン等のブラウザで表示しつつ、Bing Text To Speech API を使って音声データに変換し、再生することで、「しゃべるデジカメ」を作成してみます。

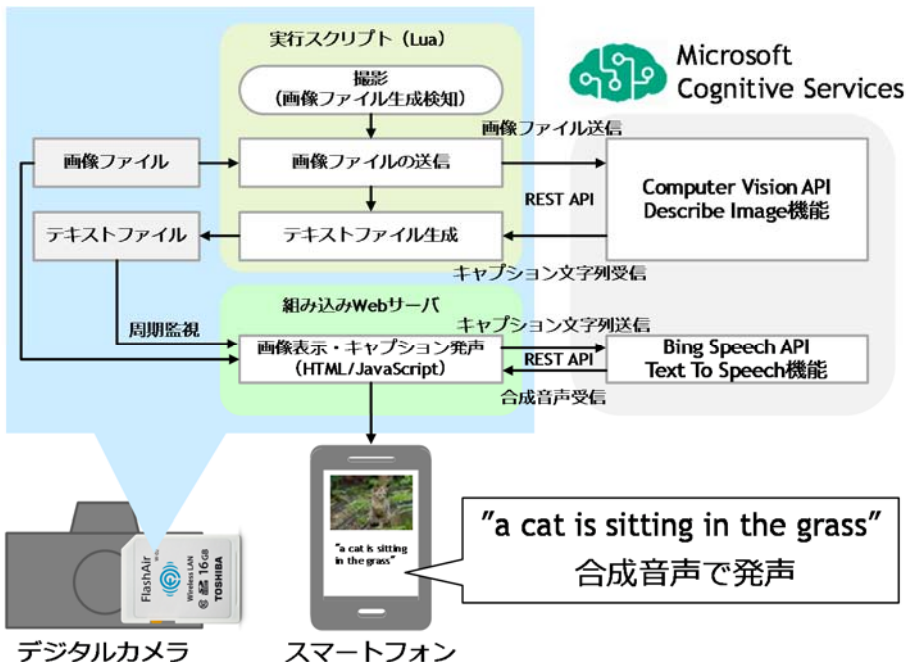


図 1: 全体の構成

1 用意するもの

下記表 1 に必要な部品を示します。

表 1: 用意するもの

部品	個数	備考
FlashAir	1 個	W-03 (ファームウェア V3.00.01) 本書の説明内で、以下の設定をします。 APPMODE = 6 インターネット同時接続モード APPNAME = flashair_azure APPNETWORKKEY = flashair_nets BRGSSID = flashair_internet_ap BRGNETWORKKEY = flashair_nets LUA_SD_EVENT = /main.lua
デジタルカメラ	1 個	FlashAir が使えれば、なんでも構いません。 本書では RICOH WG-5 GPS で試しました。
スマートフォン	1 個	ブラウザで JavaScript が実行できるもの。 本書では iPhone6s(iOS9.3.3)の Safari で説明を行います。
PC	1 個	テキストエディタと FlashAir が USB 経由で読み書きできればなんでも構いません。 本書では Windows10 で説明を行います。
インターネット接続環境	1 本	FlashAir からインターネット接続できる環境を準備してください。 本書では、以下の設定を想定します。 SSID: flashair_internet_ap Password: flashair_nets

2 サブスクリプションキーの取得(無料)

はじめに、Microsoft Cognitive Services の Computer Vision API と Bing Speech API を利用するために、サブスクリプションキーを取得します。

現在、Microsoft アカウントをお持ちの方はトライアル版(無料)のキーを取得することができます。

Microsoft Cognitive Services の web サイト(英語)にアクセスします。

<https://www.microsoft.com/cognitive-services>

「Get started for free」をクリックします。

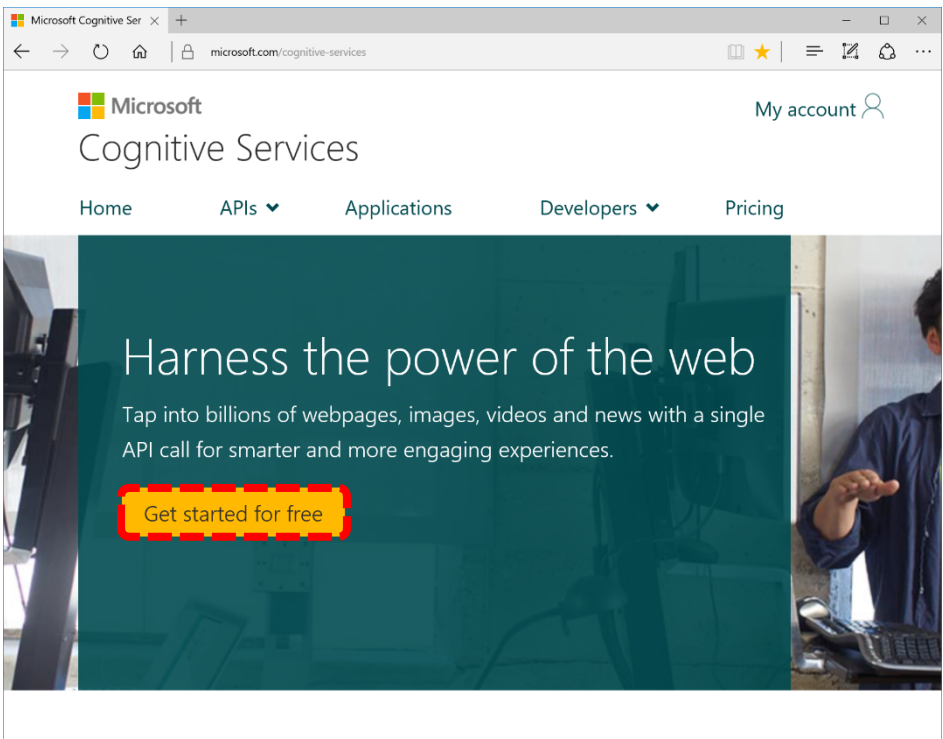
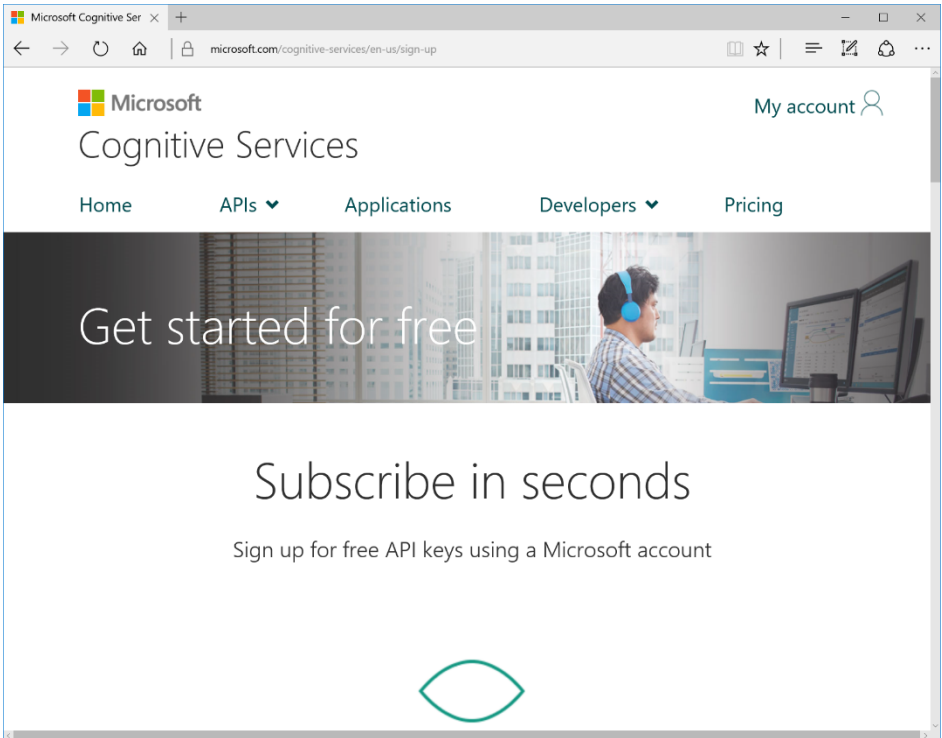


図 2: Microsoft Cognitive Service のサイト(英語)

表示された画面を下方にスクロールして、「Let's go」をクリックします。

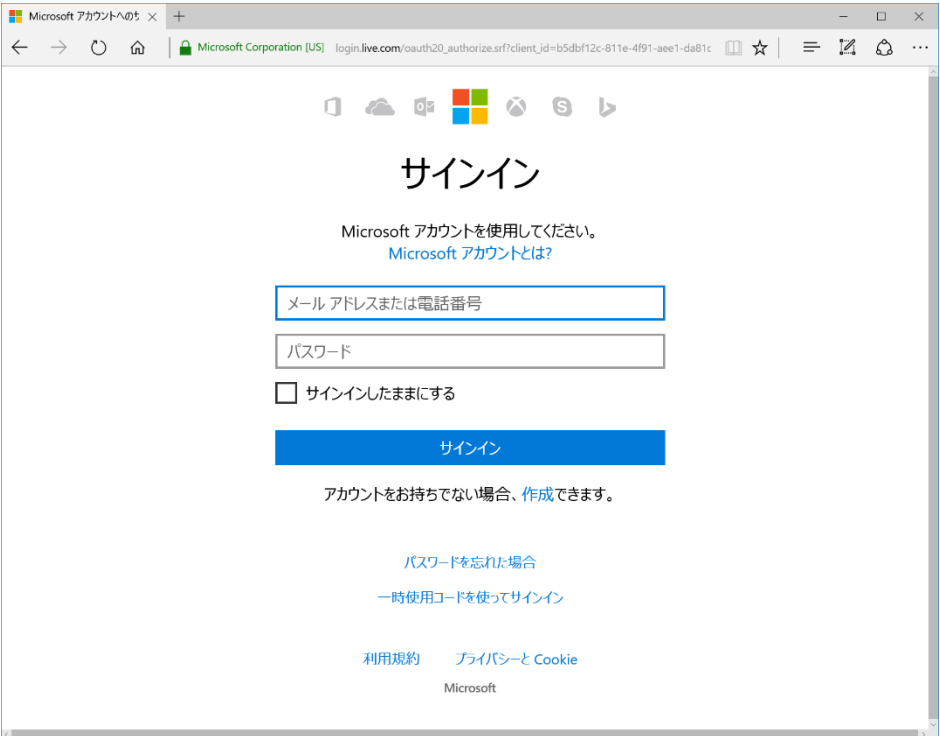


Make your apps more intelligent with the power of Bing APIs. A single call accesses data from billions of web pages, images, videos, and news.

Let's go

図 3: Microsoft Cognitive Service のスタートページ

Microsoft アカウントでサインインを求められますので、お持ちの Microsoft アカウント（もし、お持ちでなければ無料で作成可能です）でサインインしてください。



The screenshot shows a web browser window with the Microsoft account sign-in page. The browser's address bar shows the URL `login.live.com/oauth20_authorize.srf?client_id=b5dbf12c-811e-4f91-aae1-da81c`. The page features the Microsoft logo at the top, followed by the large text "サインイン" (Sign In). Below this, it says "Microsoft アカウントを使用してください。" (Use your Microsoft account.) and provides a link "Microsoft アカウントとは?" (What is a Microsoft account?). There are two input fields: "メール アドレスまたは電話番号" (Email address or phone number) and "パスワード" (Password). A checkbox labeled "サインインしたままにする" (Keep me signed in) is present. A large blue "サインイン" (Sign In) button is centered. Below the button, it states "アカウントをお持ちでない場合、作成できます。" (If you don't have an account, you can create one.). At the bottom, there are links for "パスワードを忘れた場合" (Forgot your password?), "一時使用コードを使ってサインイン" (Sign in with a one-time use code), "利用規約" (Terms of use), and "プライバシーと Cookie" (Privacy and Cookies). The Microsoft logo is at the very bottom.

図 4: Microsoft アカウントのサインインページ

以下のサブスクリプションキー取得ページが表示されます。下方にスクロールしてください。

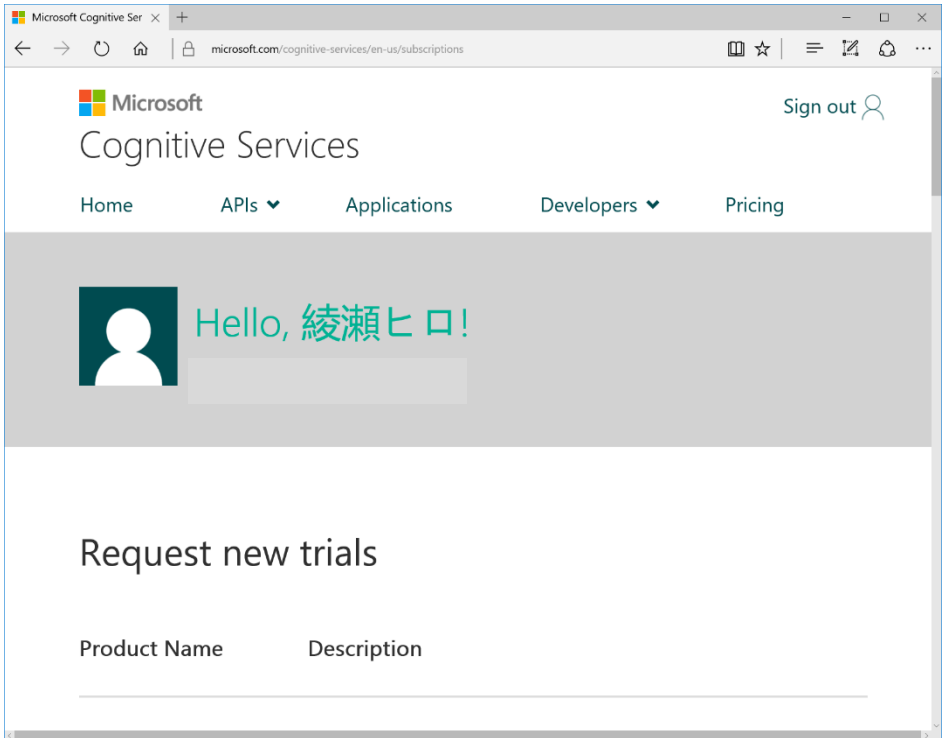


図 5: Microsoft Cognitive Service のサブスクリプションキー取得ページ(1)

サブスクリプションキー取得ページのなかで、「Computer Vision - Preview」と「Speech - Preview」のチェックボックスをクリックし、「Microsoft Cognitive Services Terms and Microsoft Privacy Statement.」をお読みいただいた上で同意するチェックボックスをクリックし、「Contact me with promotional offers and updates about Microsoft Cognitive Services.」については任意にチェックボックスをクリックして、最後に「Subscribe」をクリックしてください。

Microsoft Cognitive Ser x +

← → ↺ 🏠 | microsoft.com/cognitive-services/en-us/subscriptions

Home APIs ▼ Applications Developers ▼ Pricing

per second for evaluate, 6 per minute for calcHistogram. [Sign out](#) 👤

☒ Computer Vision - Preview 5,000 transactions per month, 20 per minute.

☒ Speech - Preview 5,000 transactions per month, 20 per minute for each feature for a total of 60 per minute.

☒ I agree to the [Microsoft Cognitive Services Terms and Microsoft Privacy Statement.](#)

☐ Contact me with promotional offers and updates about Microsoft Cognitive Services.

Cancel **Subscribe**

図 6: Microsoft Cognitive Service のサブスクリプションキー取得ページ(2)

以下のように、サブスクリプションキーが生成されます。

のちほど、それぞれのキーを使用しますので、Computer Vision は Key1 を、Speech は Key1 と Key2 の両方をテキストファイルにコピー&ペーストして保管しておいてください。

Description に書かれているとおり、トライアル(無料)のサブスクリプションキーでは、Speech は機能ごとに月間 5,000 回または 1 分あたり 20 回まで、また全機能あわせて 1 分あたり 60 回まで利用できます。また Computer Vision は月間 5,000 回または 1 分あたり 20 回まで利用できます。

Microsoft Cognitive Services

Home APIs Applications Developers Pricing Sign out

My free subscriptions (2)

Request new trials

Product	Description	Keys	State	Created	Quota
Speech - Preview	5,000 transactions per month, 20 per minute for each feature for a total of 60 per minute.	Key 1: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate Show Copy Key 2: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate Show Copy	active	2016年7月31日 11:27:33	Show Quota <input type="button" value="Cancel"/>
Computer Vision - Preview	5,000 transactions per month, 20 per minute.	Key 1: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate Show Copy Key 2: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate Show Copy	active	2016年7月31日 11:27:34	Show Quota <input type="button" value="Cancel"/>

図 7: Microsoft Cognitive Service のサブスクリプションキー取得ページ(3)

3 本体機能の作成(Lua スクリプト)

FlashAir の上で動作させる Lua スクリプトを作成します。ファイル名を `main.lua` としてテキストエディタなどでテキストファイルを作成してください。

まず、以下にスクリプトすべてを掲載します。その後に、Lua スクリプトの部分部分について少し解説します。

手っ取り早く、すべてのスクリプトをコピー＆ペーストしていただいて構いませんが、「[ここに Computer Vision のサブスクリプションキーを記述する](#)」とある部分だけは、前章で取得した自分のサブスクリプションキーに置き換えてください。

```
local cJSON = require "cjson"
print("HTTP/1.1 200 Internal OK¥n¥n")
last_fname = ""
last_fpath = ""
last_modif = 0
last_moddir = 0
last_dirname = "/DCIM"

fpath = "/DCIM"
for dirname in lfs.dir(fpath) do
    dirpath = fpath .. "/" .. dirname
    mod_dir = lfs.attributes( dirpath, "mode" )
    if mod_dir == "directory" then
        dir_modifficate = lfs.attributes( dirpath, "modification" )
        if dir_modifficate > last_moddir then
            last_moddir = dir_modifficate
            last_dirname = dirpath
        end
    end
end

for filename in lfs.dir(last_dirname) do
    if(string.sub(filename, 1, 1) ~= ".") then
        filepath = last_dirname .. "/" .. filename
        mod = lfs.attributes( filepath, "modification" )
        if mod > last_modif then
            last_modif = mod
            last_fname = filename
            last_fpath = filepath
        end
    end
end
```

```

    end
end

boundary = "1234567890"
contenttype = "multipart/form-data; boundary=" .. boundary
mes = "--" .. boundary .. "¥r¥n"
    .. "Content-Disposition: form-data; name=¥"file¥"; filename=¥"" .. last_fname .. "¥"¥r¥n"
    .. "Content-Type: image/jpg¥r¥n¥r¥n"
    .. "<!--WLANSDFILE-->¥r¥n"
    .. "--" .. boundary .. "--¥r¥n"

blen = lfs.attributes(last_fpath, "size") + string.len(mes) - 17
b, c, h = fa.request{url =
"https://api.projectoxford.ai/vision/v1.0/analyze?visualFeatures=Description,Tags",
    method = "POST",
    headers = {[ "Content-Length" ] = tostring(blen),
[ "Content-Type" ] = contenttype,
[ "Ocp-Apim-Subscription-Key" ] = "ここに Computer Vision のサブスクリプションキーを記述する",
    [ "maxCandidates" ] = "1"},
    file = last_fpath,
    body = mes
}

if(c == 200) then
    res = cJSON.decode(b)
    tagstr = ""
    for idx=1,#res.description.tags do
        tagstr = tagstr .. res.description.tags[idx] .. " "
    end

    local fh = io.open("result.txt", "w")
    fh:write("filepath  [" .. filepath .. "]¥n")
    fh:write("Tags      [" .. tagstr .. "]¥n")
    fh:write("Captions  [" .. res.description.captions[1].text .. "]¥n")
    fh:write("Confidence[" .. res.description.captions[1].confidence .. "]¥n")
    fh:flush()
    fh:close()

    fh = io.open("caption.txt", "w")
    fh:write(filepath .. ";" .. res.description.captions[1].text .. "¥n")
    fh:flush()

```

```
    fh:close()
else
    local fh = io.open("result_error.txt", "w")
    fh:write("filepath [" .. filepath .. "]¥n")
    fh:write("h [" .. h .. "]¥n")
    fh:write("b [" .. b .. "]¥n")
    fh:write("c [" .. c .. "]¥n")
    fh:flush()
    fh:close()
end
```

3.1 最新の画像ファイルの検索と取得

デジタルカメラで撮影された写真は画像ファイルとして FlashAir の DCIM フォルダ内に作成されます。このとき撮影日付別にデジタルカメラのメーカーによって異なる名称のフォルダが作成され、そのなかに画像ファイルが格納されます。

今回の「しゃべるデジカメ」では、FlashAir の「LUA_SD_EVENT」(SD カード内にファイル生成されたイベント)が発生したときに、最新の画像ファイル(デジタルカメラが直近で撮影した写真の画像ファイル)を取得したいため、まず Lua スクリプトの最初のところで、最新の画像ファイルを検索します。

まず変数を宣言します。

```
last_fname = ""      --最新の画像ファイルのファイル名が格納されます。
last_fpath = ""      --最新の画像ファイルのあるフォルダのパスが格納されます。
last_modif = 0       --画像ファイルの生成日時を一時的に格納しておく変数
last_moddir = 0      --フォルダの生成日時を一時的に格納しておく変数
last_dirname = "/DCIM" --最新の画像ファイルのあるフォルダ名が格納されます。
                    --初期値として/DCIMを指定しておきます。
fpath = "/DCIM"      --初期フォルダの位置を指定します。
```

続いて、まず最新のフォルダを検索します。

今回は最新の画像ファイルを検索する際に、すべてのフォルダ内のファイルを検索していると時間がかかるため、まずフォルダの生成日時だけで最新のフォルダを検索し、そのフォルダ内に最新の画像ファイルがある前提としています。仮に最新の生成日時でないフォルダ内に最新の画像ファイルが生成されてしまった場合は、今回の検索処理では検知できません。ここは処理速度と検索の正確性のバスターになりますので、お使いの環境やデジタルカメラの仕様をご確認の上で、必要に応じて各自調整をお願いします。

最新のフォルダは、変数 `last_dirname` に格納されます。

```
for dirname in lfs.dir(fpath) do
    dirpath = fpath .. "/" .. dirname
    mod_dir = lfs.attributes( dirpath, "mode" )
    if mod_dir == "directory" then
        dir_modifficate = lfs.attributes( dirpath, "modification" )
        if dir_modifficate > last_moddir then
            last_moddir = dir_modifficate
            last_dirname = dirpath
        end
    end
end
```

次に、最新のフォルダ(last_dirname)内の最新の生成日時の画像ファイルを検索します。このときファイル名の先頭にピリオド"."のあるファイルは除外しています。MacOS でFlashAir 内のフォルダにファイルコピー等した場合に自動生成される隠しファイルを除外するためです。

最新の画像ファイルのファイル名が変数last_fnameに、ファイルのあるフォルダの絶対パス+ファイル名が変数last_fpathに格納されます。

```
for filename in lfs.dir(last_dirname) do
  if(string.sub(filename, 1, 1) ~= ".") then
    filepath = last_dirname .. "/" .. filename
    mod = lfs.attributes( filepath, "modification" )
    if mod > last_modif then
      last_modif = mod
      last_fname = filename
      last_fpath = filepath
    end
  end
end
```

3.2 Computer Vision の REST API の呼び出し

続いて、得られた最新の画像ファイルを Computer Vision の REST API に送信しキャプション情報を取得します。今回は、汎用的な Analyze API を使用します。この API ではキャプション情報以外にもさまざまな情報を得ることができます。

まず、REST API に送信するメッセージ本文を生成します。

Content-Disposition の filename に、さきほど取得した最新の画像ファイルのファイル名を格納した last_fname を指定します。

```
boundary = "1234567890"
contenttype = "multipart/form-data; boundary=" .. boundary
mes = "--" .. boundary .. "\r\n"
.. "Content-Disposition: form-data; name="file"; filename=" .. last_fname .. "\r\n"
.. "Content-Type: image/jpeg\r\n\r\n"
.. "<!--WLANSDFILE-->\r\n"
.. "--" .. boundary .. "\r\n"
blen = lfs.attributes(last_fpath, "size") + string.len(mes) - 17
```

次に FlashAir の Lua 機能で提供されている HTTP リクエスト機能 `fa.request` で Computer Vision の REST API を呼び出します。Ocp-Apim-Subscription-Key に指定する文字列は、前述のとおり、Computer Vision のサブスクリプションキーに置き換えてください。

呼び出し結果は、変数 `b`、`c`、`h` に格納されます。

変数 `b` には、呼び出した結果の HTTP response body の文字列が格納されます。Computer Vision では JSON 形式で結果が返ります。

変数 `c` には、呼び出した結果の HTTP ステータスコードが格納されます。呼び出した結果が成功だった場合は 200 が返ります。

変数 `h` には、呼び出した結果の HTTP response header の文字列が格納されます。

```
b, c, h = fa.request{url =
"https://api.projectoxford.ai/vision/v1.0/analyze?visualFeatures=Description,Tags",
method = "POST",
headers = {[ "Content-Length" ] = tostring(blen),
[ "Content-Type" ] = contenttype,
[ "Ocp-Apim-Subscription-Key" ] = ここに Computer Vision のサブスクリプションキーを記述する ,
[ "maxCandidates" ] = "1"},
file = last_fpath,
body = mes
}
```


3.3 キャプション情報のテキストファイル出力

Computer Vision の REST API を呼び出した結果、成功した場合(変数 `c` の HTTP ステータスコードが 200 の場合)は、まず JSON 形式で格納されている変数 `b` の内容を、`cjson` ライブラリの `decode` 関数で Lua スクリプトの Table 形式に変換して扱いやすくします。変換した内容は変数 `res` に格納しています。

変数 `res` は、Table 形式になっており、`description` テーブル内の `tags` テーブル内に画像ファイルを分析してつけられたタグ情報が格納されています。「しゃべるデジカメ」でタグ情報は扱いませんが、デバッグ用に取得しています。タグ情報を使って、写真の分類などしても面白いかもしれません。以下のコードでは、文字列 `tagstr` に空白区切りですべて列挙しています。

```
res = cjson.decode(b)
tagstr = ""
for idx=1,#res.description.tags do
    tagstr = tagstr .. res.description.tags[idx] .. " "
end
```

まずデバッグ用の結果を出力したテキストファイル"result.txt"を生成します。このテキストファイルには、Computer Vision API で画像ファイル(`filepath` で示されたファイル)を分析した結果得られたタグ情報、キャプション情報(`res.description.captions[1].text`)、キャプション情報の確からしさ(`res.description.captions[1].confidence`)を出力しています。このとき `captions` 情報が複数得られる場合もあるようですが、今回は決めうちで1つめのもののみ取り出しています。

```
local fh = io.open("result.txt", "w")
fh:write("filepath [" .. filepath .. "]%n")
fh:write("Tags      [" .. tagstr .. "]%n")
fh:write("Captions  [" .. res.description.captions[1].text .. "]%n")
fh:write("Confidence[" .. res.description.captions[1].confidence .. "]%n")
fh:flush()
fh:close()
```

続いて、後ほど JavaScript から読み込むためのテキストファイル"caption.txt"を生成します。JavaScript で読み込んだ際に余計な処理が必要ないように、区分文字を";"として、画像ファイルの絶対パス+ファイル名とキャプション情報を出力するだけにします。

```
fh = io.open("caption.txt", "w")
fh:write(filepath .. ";" .. res.description.captions[1].text .. "%n")
fh:flush()
fh:close()
```

最後は、Computer Vision の REST API を呼び出した結果、成功しなかった場合(変

数 `c` の HTTP ステータスコードが 200 以外の場合)に、`fa.request` の返り値を直接テキストファイル `result_error.txt` に出力しておき、デバッグしやすいようにしておきます。

```
local fh = io.open("result_error.txt", "w")
fh:write("filepath [" .. filepath .. "]¥n")
fh:write("h [" .. h .. "]¥n")
fh:write("b [" .. b .. "]¥n")
fh:write("c [" .. c .. "]¥n")
fh:flush()
fh:close()
```

4 ビューアーの作成 (JavaScript)

残念ながらデジタルカメラ単体で HTTP リクエストを受けて音声再生できるものが見当りませんので、今回はスマートフォンで FlashAir の Web サーバにアクセスして、HTML ファイル (ビューアー) を表示させ、JavaScript で先に Lua スクリプトで生成したテキストファイル "caption.txt" を定期的に監視して、文字列に変化があれば、表示している画像ファイルを自動更新し、さらに Bing Text To Speech API を使って得られた音声を実再生します。

まず、FlashAir の Web サーバ上で動作させる JavaScript を含む HTML ファイルを作成します。ファイル名を viewer.htm としてテキストエディタなどで HTML ファイルを作成してください。

以下に HTML すべてを掲載します。その後に、JavaScript の部分部分について少し解説します。

手っ取り早く、すべてのスクリプトをコピー＆ペーストしていただいて構いませんが、「ここに Speech のサブスクリプションキー (Key1) を記述する」と「ここに Speech のサブスクリプションキー (Key2) を記述する」の2つの部分だけは、前章で取得した自分のサブスクリプションキーに置き換えてください。また、アプリケーションを区別するために「任意の 32 桁の 16 進数の文字列①」と「任意の 32 桁の 16 進数の文字列②」という項目がありますので、こちらはハイフンなしの GUID を指定しますが、任意の文字列に置き換えてください。

今回の JavaScript による音声再生における注意点として、ビューアーを iOS デバイスで表示させる場合、JavaScript 実行環境の制限により、人間のアクションによる場合のみ音声再生されることから、表示時に「開始」ボタンを押すアクションが必要になります。

```
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta charset="UTF-8">
<title>FlashAir + Azure でしゃべるデジカメ</title>
<script language="javascript" type="text/javascript">
<!--
var AudioContext = window.AudioContext || window.webkitAudioContext;
var context = new AudioContext();
var speech_lang = "en-US";
var speech_gend = "Female";
var speech_name = "Microsoft Server Speech Text to Speech Voice (en-US, ZiraRUS)";

var last_str = "";
function getCaptionText() {
    var request = new XMLHttpRequest();
```

```

request.open("GET", "http://flashair/caption.txt", false);
request.onload = function(res) {
    var statestr = request.responseText;
    if(last_str != statestr) {
        result = statestr.split( ";" );
        str_caption = result[1];
        str_image = result[0];
        document.getElementById('AUTOTXT').value = str_caption;
        document.getElementById("AUTOImage").src = "http://flashair" + str_image;
        last_str = statestr
        Synthesize()
    }
}
request.send(null);
}

function getWaveFile(OxfordAccessToken, talkstring) {
    var ttsServiceUri = "https://speech.platform.bing.com/synthesize";
    var post_data = "<speak version='1.0' xml:lang='en-us'><voice xml:lang=' " + speech_lang
+ "' xml:gender=' " + speech_gend + "' name=' " + speech_name + "'>" + talkstring +
"</voice></speak>";
    var xhr = new XMLHttpRequest();
    xhr.open("POST", ttsServiceUri);
    xhr.setRequestHeader("Content-Type", "application/ssml+xml");
    xhr.setRequestHeader("X-Microsoft-OutputFormat", "riff-16khz-16bit-mono-pcm");
    xhr.setRequestHeader("Authorization", OxfordAccessToken);
    xhr.setRequestHeader("X-Search-AppId", "任意の 32 桁の 16 進数の文字列①");
    xhr.setRequestHeader("X-Search-ClientID", "任意の 32 桁の 16 進数の文字列②");
    xhr.responseType = 'arraybuffer';
    xhr.onload = function(res) {
        var req_data = xhr.response;
        if (req_data instanceof ArrayBuffer) {
            var successCallback = function(audioBuffer) {
                var source = context.createBufferSource();
                source.buffer = audioBuffer;
                source.loop = false;
                source.loopStart = 0;
                source.loopEnd = audioBuffer.duration;
                source.playbackRate.value = 1.0;
                source.connect(context.destination);
                source.start = source.start || source.noteOn;
                source.start(0);
            }
        }
    }
}

```

```

    };
    var errorCallback = function(error) {
        if (error instanceof Error) {
            window.alert(error.message);
        } else {
            window.alert('Error : "decodeAudioData" method. ');
        }
    };
}

context.decodeAudioData(req_data, successCallback, errorCallback);
};
xhr.send(post_data);
}

function Synthesize() {
    var clientId    = "ここに Speech のサブスクリプションキー (Key1) を記述する";
    var clientSecret = "ここに Speech のサブスクリプションキー (Key2) を記述する";
    var speechHost  = "https://speech.platform.bing.com";
    var post_token_data      = "grant_type=client_credentials&client_id=" +
    encodeURIComponent(clientId) + "&client_secret=" + encodeURIComponent(clientSecret) +
    "&scope=" + encodeURIComponent(speechHost);
    var AccessTokenUri      = "https://oxford-speech.cloudapp.net/token/issueToken";
    var xhr_token = new XMLHttpRequest();
    xhr_token.open("POST", AccessTokenUri);
    xhr_token.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    xhr_token.responseType = "json";
    xhr_token.onload = function(res) {
        var _data = xhr_token.response;
        var OxfordAccessToken = "Bearer " + _data.access_token;
        talkstring = document.getElementById('AUTOTXT').value;
        getWaveFile(OxfordAccessToken, talkstring);
    };
    xhr_token.send(post_token_data);
}

function StartPlay() {
    if (document.getElementById('AUTOSTART').value != "動作中") {
        context.createBufferSource().start(0);
        document.getElementById('AUTOSTART').value = "動作中";
        setInterval("getCaptionText()", 1000);
    }
}
}

```

```
</script>
</head><body>
FlashAir+Microsoft Cognitive Services<br>
<br>
<form name="selectButtonFuncion">
<div id="start">キャプション</div><br>
<input type="text" id="AUTOTXT" style="WIDTH: 640px;"><br>
<br>
<input type="button" id="AUTOSTART" value="開始" onclick="StartPlay()" width="80px"
height="80px" />
</form>
</body></html>
```

4.1 全体の流れ

ビューアーは以下の流れで動作します。

- ① 表示するデバイスを、デジタルカメラの FlashAir に接続します
- ② デバイスのブラウザで、以下の URL にアクセスして FlashAir 上の HTML ファイルを表示することで、「4.2 ビューアーの初期表示」が行われます
<http://flashair/viewer.htm>
- ③ ビューアーの初期表示の「開始」ボタンを押下すると、「4.3 キャプション情報テキストファイルの周期監視」が開始されます。デジタルカメラで写真を撮影して FlashAir 上に画像ファイルが生成され Lua スクリプトが"caption.txt"を生成したことを検知します
- ④ Bing Text To Speech API の呼び出すため、「4.4 アクセストークンの取得」を行います
- ⑤ 取得したアクセストークンを使って、「4.5 Bing Text To Speech API の呼び出しと音声再生」を行います

4.2 createBufferSource ビュアーの初期表示

まず、ビューアーの初期表示は、図 8 に示すような状態で表示されます。左下の「開始」ボタンを押下することで、ビューアーの処理が開始されます。

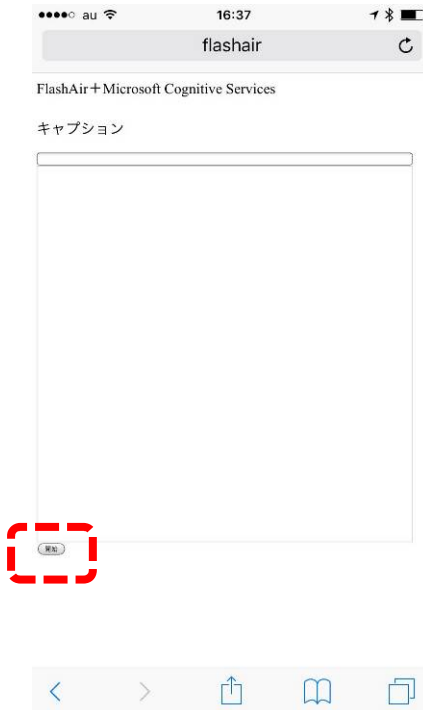


図 8: ビュアー(viewer.htm)の初期表示

「開始」ボタンの onclick イベントで StartPlay 関数が呼び出されます。「開始」ボタンは 2 回以上押されないようにします。StartPlay 関数では、以下の処理を行います。

- ① 音声再生のための AudioContext の createBufferSource をダミーでスタート start(0)させます。まだ createBufferSource のバッファには音声データがありませんので、当然、音声は再生されません。しかし、このボタン押下時の処理で、一度、AudioContext を再生させておかないと、後ほどのキャプション取得後の音声再生で音が出ません。これは人間のアクションによる処理でないと音が出ない iOS の仕様を回避するためのものです。
- ② 「開始」ボタンのキャプションを「動作中」に切り替えます。
- ③ setInterval で、1000[ms] (1秒) 周期に、getCaptionText 関数を呼び出すように設定します。これ以降、周期的に getCaptionText 関数が実行されます。


```
var AudioContext = window.AudioContext || window.webkitAudioContext;  
var context = new AudioContext();
```

```
function StartPlay() {  
    if (document.getElementById('AUTOSTART').value != "動作中") {  
        context.createBufferSource().start(0);  
        document.getElementById('AUTOSTART').value = "動作中";  
        setInterval("getCaptionText()", 1000);  
    }  
}  
</script>  
</head><body>  
FlashAir+Microsoft Cognitive Services<br>  
<br>  
<form name="selectButtonFuncion">  
<div id="start">キャプション</div><br>  
<input type="text" id="AUTOTXT" style="WIDTH: 640px;"><br>  
<br>  
<input type="button" id="AUTOSTART" value="開始" onclick="StartPlay()" width="80px"  
height="80px" />  
</form>  
</body></html>
```

4.3 キャプション情報テキストファイルの周期監視

FlashAir 上にある"caption.txt"を取得するため、"http://flashair/caption.txt"に HTTP リクエストでアクセスします。取得した文字列は区分文字";"でキャプション情報と画像ファイルの絶対パス+ファイル名情報に分けて、それぞれビューアーのテキストボックス 'AUTOTXT'とイメージタグ"AUTOImage"の src に指定します。

変数 last_str には、最後に処理した"caption.txt"の内容を保存しておきます。この文字列と取得した文字列を比較して、保存したものと異なる場合は、"caption.txt"が新しくなったとみなして処理を行います。同じ場合は"caption.txt"は前回処理した時のままであるとみなして、処理をスキップします。

最後に、音声再生するための Synthesize 関数を呼び出します。

```
var last_str = "";
function getCaptionText() {
    var request = new XMLHttpRequest();
    request.open("GET", "http://flashair/caption.txt", false);
    request.onload = function(res) {
        var statestr = request.responseText;
        if(last_str != statestr) {
            result = statestr.split(";");
            str_caption = result[1];
            str_image = result[0];
            document.getElementById('AUTOTXT').value = str_caption;
            document.getElementById("AUTOImage").src = "http://flashair" + str_image;
            last_str = statestr
            Synthesize()
        }
    }
    request.send(null);
}
```

4.4 アクセストークンの取得

Bing Text To Speech API は、Computer Vision API と異なり、一度、サブスクリプションキーで認証を行ってアクセストークンを取得する必要があります。アクセストークンの取得も REST API となっているため、XMLHttpRequest を使って POST してデータ(_data)を得ます。このデータは JSON 形式となっており、このなかに access_token という項目が含まれているので、これに "Bearer " (Bearer の後に半角空白が必要) を付加して、アクセストークン (OxfordAccessToken) とします。

この得られたアクセストークンと、さきほど取得してテキストボックス 'AUTOTXT' に書き込んだキャプション情報を引数に指定して getWaveFile 関数を呼び出します。

```
function Synthesize() {
    var clientId      = "ここに Speech のサブスクリプションキー (Key1) を記述する";
    var clientSecret  = "ここに Speech のサブスクリプションキー (Key2) を記述する";
    var speechHost    = "https://speech.platform.bing.com";
    var post_token_data = "grant_type=client_credentials&client_id=" +
        encodeURIComponent(clientId) + "&client_secret=" + encodeURIComponent(clientSecret) +
        "&scope=" + encodeURIComponent(speechHost);
    var AccessTokenUri = "https://oxford-speech.cloudapp.net/token/issueToken";
    var xhr_token = new XMLHttpRequest();
    xhr_token.open("POST", AccessTokenUri);
    xhr_token.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    xhr_token.responseType = "json";
    xhr_token.onload = function(res) {
        var _data = xhr_token.response;
        var OxfordAccessToken = "Bearer " + _data.access_token;
        talkstring = document.getElementById('AUTOTXT').value;
        getWaveFile(OxfordAccessToken, talkstring);
    };
    xhr_token.send(post_token_data);
}
```

4.5 Bing Text To Speech API の呼び出しと音声再生

Bing Text To Speech API では、application/ssml+xml 形式という文字列でデータを渡すと、音声データを得られます。application/ssml+xml 形式は HTML に似た記述内容となっており、speak タグの中で、voice タグに JavaScript の先頭で定義していた変数 speech_lang (扱う言語)、変数 speech_gend (話す音声の性別)、変数 speech_name (話す音声の種類) を指定し、タグ内に音声データにしたい文字列 (引数の talkstring を指定している) を指定します。(現在、18 言語 29 種類の音声種類があります。今回は英語音声を使用していますが、日本語にも対応しています。詳細は 7 章参考の⑦Bing Text To Speech API の合成音声の種類を参照ください)

```
var speech_lang = "en-US";
var speech_gend = "Female";
var speech_name = "Microsoft Server Speech Text to Speech Voice (en-US, ZiraRUS)";
```

```
var post_data = "<speak version='1.0' xml:lang='en-us'><voice xml:lang='" + speech_lang + '" xml:gender='" + speech_gend + '" name='" + speech_name + "'>" + talkstring + "</voice></speak>";
```

XMLHttpRequest を使って Bing Speech API に POST しますが、このときにヘッダ情報の X-Microsoft-OutputFormat には音声データの種類を指定します。(現在、4 種類あります。詳細は 7 章参考の⑥Bing Text To Speech API の出力音声形式の種類を参照ください)

また、Authorization には、さきほど取得したアクセストークンを指定します。前述したとおり X-Search-AppId と X-Search-ClientID には任意の 32 桁の 16 進数の文字列を記述してください。一応、世界中で一意となる文字列である必要があります。

```
var xhr = new XMLHttpRequest();
xhr.open("POST", ttsServiceUri);
xhr.setRequestHeader("Content-Type", "application/ssml+xml");
xhr.setRequestHeader("X-Microsoft-OutputFormat", "riff-16khz-16bit-mono-pcm");
xhr.setRequestHeader("Authorization", OxfordAccessToken);
xhr.setRequestHeader("X-Search-AppId", "任意の 32 桁の 16 進数の文字列①");
xhr.setRequestHeader("X-Search-ClientID", "任意の 32 桁の 16 進数の文字列②");
xhr.responseType = 'arraybuffer';
```

正常にリクエストが返ると、レスポンスデータに音声データが格納されてきますので、これを `AudioContext` の `source = context.createBufferSource()` のバッファ `buffer` にセットして、再生開始 `source.start(0)` します。

```
var successCallback = function(audioBuffer) {  
  var source = context.createBufferSource();  
  source.buffer = audioBuffer;  
  source.loop = false;  
  source.loopStart = 0;  
  source.loopEnd = audioBuffer.duration;  
  source.playbackRate.value = 1.0;  
  source.connect(context.destination);  
  source.start = source.start || source.noteOn;  
  source.start(0);  
};
```

5 FlashAir の準備

5.1 CONFIG ファイルの設定

FlashAir の設定ファイル CONFIG を以下の通りに設定します。

CONFIG ファイルは、FlashAir の SD_WLAN フォルダの下にあります。SD_WLAN フォルダは隠し属性となっているため、必要に応じて属性変更を行ってください。

MacOS の場合は、以下のコマンドをターミナルで実行します。

```
macos:~ username$ chflags nohidden /Volumes/{SD カード名}/SD_WLAN
macos:~ username$ chflags nohidden /Volumes/{SD カード名}/SD_WLAN/CONFIG
```

CONFIG ファイルの記述例です。既存の CONFIG の必要な部分を書き換え、既存以内行は追加してください。BRGSSIDとBRGNETWORKKEYはご自分のインターネット接続環境のアクセスポイントの設定に合わせて設定してください。

```
APPSSID=flashair_azure
APPNETWORKKEY=flashair_nets
APPMODE=6
BRGSSID=flashair_internet_ap
BRGNETWORKKEY=flashair_nets
LUA_SD_EVENT=/main.lua
```

5.2 必要なファイルのコピー

図 9 のように、先に作成した Lua スクリプトファイル(main.lua)と、ビューアHTML ファイル(viewer.htm)を FlashAir の最上位階層のフォルダにコピーします。

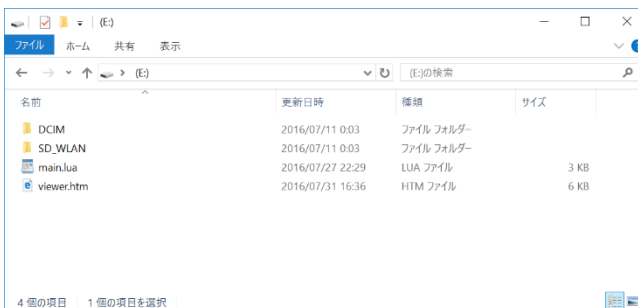


図 9: FlashAir のファイル配置

6 実行

それでは「しゃべるデジカメ」を実際に稼働させてみましょう。

FlashAir をデジタルカメラに装着して、デジタルカメラの電源を入れます。デジタルカメラのスリープ設定は解除しておいたほうが良いです。デジタルカメラの機種によっては撮影していない場合に SD カードへの電源供給が停止（スリープ）してしまう場合があります、そうすると FlashAir の無線 LAN が稼働しなくなって、スマートフォンとの接続が切れてしまいます。なるべくスリープしないように設定しておきましょう。

また、デジタルカメラの写真の画質設定で、画像ファイルのサイズが 4MB（メガバイト）以内になるように設定してください。これは Computer Vision API で送信できる画像ファイルの上限が 4MB に制限されているためです。また、あまり大きなサイズの画像ファイルの場合、Computer Vision API に送信する時間がかかるようになり、処理速度が低下しますので、適切な画質で撮影されることをお勧めします。

デジタルカメラで窓の外の空を撮影してみた例を図 10 に示します。キャプション情報には「a blue cloudy sky」と付加され、合成音声でキャプションをしゃべってくれました。

FlashAir+Microsoft Cognitive Services

キャプション



図 10: 実行例

いろいろ撮影して、試してみてください。

7 参考

本書に記述した内容は、2016年7月31日現在に入手できる情報をもとにしています。将来、内容が変更される可能性がありますこと、ご容赦ください。

以下に、参考文献を紹介します。

- ① Microsoft Cognitive Services の公式サイト
<https://www.microsoft.com/cognitive-services>
- ② Computer Vision API の公式サイト
<https://www.microsoft.com/cognitive-services/en-us/computer-vision-api>
- ③ Computer Vision API の Analyze Image API の API リファレンス
<https://dev.projectoxford.ai/docs/services/56f91f2d778daf23d8ec6739/operations/56f91f2e778daf14a499e1fa>
- ④ Bing Speech API の公式サイト
<https://www.microsoft.com/cognitive-services/en-us/speech-api>
- ⑤ Bing Text To Speech API の API リファレンス
<https://www.microsoft.com/cognitive-services/en-us/speech-api/documentation/api-reference-rest/bingvoiceoutput>
- ⑥ Bing Text To Speech API の出力音声形式の種類
API リファレンスより抜粋。

Header	Value	Comments
X-Microsoft-OutputFormat	1) raw-8khz-8bit-mono-mulaw, 2) raw-16khz-16bit-mono-pcm, 3) riff-8khz-8bit-mono-mulaw, 4) riff-16khz-16bit-mono-pcm	The output audio format

- ⑦ Bing Text To Speech API の合成音声の種類
API リファレンスより抜粋。

Locale	Gender	Service name mapping
ar-EG*	Female	"Microsoft Server Speech Text to Speech Voice (ar-EG, Hoda)"
de-DE	Female	"Microsoft Server Speech Text to Speech Voice (de-DE, Hedda)"
de-DE	Male	"Microsoft Server Speech Text to Speech Voice (de-DE, Stefan, Apollo)"
en-AU	Female	"Microsoft Server Speech Text to Speech Voice (en-AU, Catherine)"
en-CA	Female	"Microsoft Server Speech Text to Speech Voice (en-CA, Linda)"
en-GB	Female	"Microsoft Server Speech Text to Speech Voice (en-GB, Susan, Apollo)"
en-GB	Male	"Microsoft Server Speech Text to Speech Voice (en-GB, George, Apollo)"
en-IN	Male	"Microsoft Server Speech Text to Speech Voice (en-IN, Ravi, Apollo)"
en-US	Female	"Microsoft Server Speech Text to Speech Voice (en-US, ZiraRUS)"
en-US	Male	"Microsoft Server Speech Text to Speech Voice (en-US, BenjaminRUS)"
es-ES	Female	"Microsoft Server Speech Text to Speech Voice (es-ES, Laura, Apollo)"
es-ES	Male	"Microsoft Server Speech Text to Speech Voice (es-ES, Pablo, Apollo)"
es-MX	Male	"Microsoft Server Speech Text to Speech Voice (es-MX, Raul, Apollo)"
fr-CA	Female	"Microsoft Server Speech Text to Speech Voice (fr-CA, Caroline)"
fr-FR	Female	"Microsoft Server Speech Text to Speech Voice (fr-FR, Julie, Apollo)"
fr-FR	Male	"Microsoft Server Speech Text to Speech Voice (fr-FR, Paul, Apollo)"
it-IT	Male	"Microsoft Server Speech Text to Speech Voice (it-IT, Cosimo, Apollo)"
ja-JP	Female	"Microsoft Server Speech Text to Speech Voice (ja-JP, Ayumi, Apollo)"
ja-JP	Male	"Microsoft Server Speech Text to Speech Voice (ja-JP, Ichiro, Apollo)"
pt-BR	Male	"Microsoft Server Speech Text to Speech Voice (pt-BR, Daniel, Apollo)"
ru-RU	Female	"Microsoft Server Speech Text to Speech Voice (pt-BR, Daniel, Apollo)"
ru-RU	Male	"Microsoft Server Speech Text to Speech Voice (ru-RU, Pavel, Apollo)"
zh-CN	Female	"Microsoft Server Speech Text to Speech Voice (zh-CN, HuihuiRUS)"
zh-CN	Female	"Microsoft Server Speech Text to Speech Voice (zh-CN, Yaoyao, Apollo)"

zh-CN	Male	"Microsoft Server Speech Text to Speech Voice (zh-CN, Kangkang, Apollo)"
zh-HK	Female	"Microsoft Server Speech Text to Speech Voice (zh-HK, Tracy, Apollo)"
zh-HK	Male	"Microsoft Server Speech Text to Speech Voice (zh-HK, Danny, Apollo)"
zh-TW	Female	"Microsoft Server Speech Text to Speech Voice (zh-TW, Yating, Apollo)"
zh-TW	Male	"Microsoft Server Speech Text to Speech Voice (zh-TW, Zhiwei, Apollo)"

*ar-EG supports Modern Standard Arabic (MSA)

⑧ FlashAir Developers サイト

<https://www.flashair-developers.com/ja/>

⑨ FlashAir Developers サイト

ユーザーチュートリアル「Bluemix に繋げてみよう」

<https://www.flashair-developers.com/ja/documents/tutorials/users/4/>



綾瀬ヒロ

某 IT 企業の運輸系インダストリーマネージャです。

自称 FlashAir エバンジェリストとして、FlashAir を使った電子工作や応用例を Twitter や Web で公開しています。鉄道模型とのコラボが多めですね。

Twitter: @ayasehiro

Qiita: <http://qiita.com/ayasehiro>

Web: <http://www14.big.or.jp/~ayase/>