

---

# VDS-API リファレンス

## Voice Delivery System API References

株式会社 ナレッジクリエーション  
横浜市西区高島2-6-38 岩井本社ビル3階

---



**We Do Voice API Right.**

## ご注意

このドキュメントに記載されている情報（URLなどのインターネットWebサイトに関する情報を含む）は、将来予告なく変更することがあります。このドキュメントに記載されている会社名、製品名には、各社の商標のものがあります。

このドキュメントに記載されている内容に関し、特許、特許申請、著作権、または無体財産権を有する場合があります。このドキュメントはこれらの特許、著作権、またはその他の無体財産権に関する権利を利用者に許諾するものではありません。

© 2007 Knowledge Creation Inc., All rights reserved.

落丁、乱丁はお取り替えいたします。

# 目次

<b>VDSの概要</b>	<b>1</b>
VDSのしくみ	1
クライアントにおけるVDSの動作条件	2
VDS利用上の制限事項	2
まとめ	2
<b>はじめのいっぽ</b>	<b>3</b>
Step 1 : 準備	3
Step 2 : 読み上げ範囲の指定と文字列の取り出し	5
Step 3 : 文字列を読み上げる	6
まとめ	8
<b>VDS API詳説</b>	<b>9</b>
VoiceDeliveryオブジェクトとVoiceDeliveryPlayerオブジェクト	9
VDSのエラー処理	10
VDSの認証とセッション	10
API一覧	10
■ VoiceDeliveryオブジェクトのメソッド	12
<i>VoiceDelivery</i>	12
<i>speak</i>	13
<i>setCast</i>	15
<i>setRate</i>	16
<i>setString</i>	17
<i>setFinishCallback</i>	18
<i>getCast</i>	19

<i>getCastList</i>	20
<i>getRate</i>	21
<i>getString</i>	22
<i>getStringLimit</i>	23
<i>getServerError</i>	24
■ <i>VoiceDeliveryPlayer</i> オブジェクトのメソッド	25
<i>VoiceDeliveryPlayer</i>	25
<i>setPan</i>	26
<i>setVolume</i>	27
<i>getPan</i>	28
<i>getVolume</i>	29
<i>speakPause</i>	30
<i>speakStop</i>	31
<i>isPlaying</i>	32

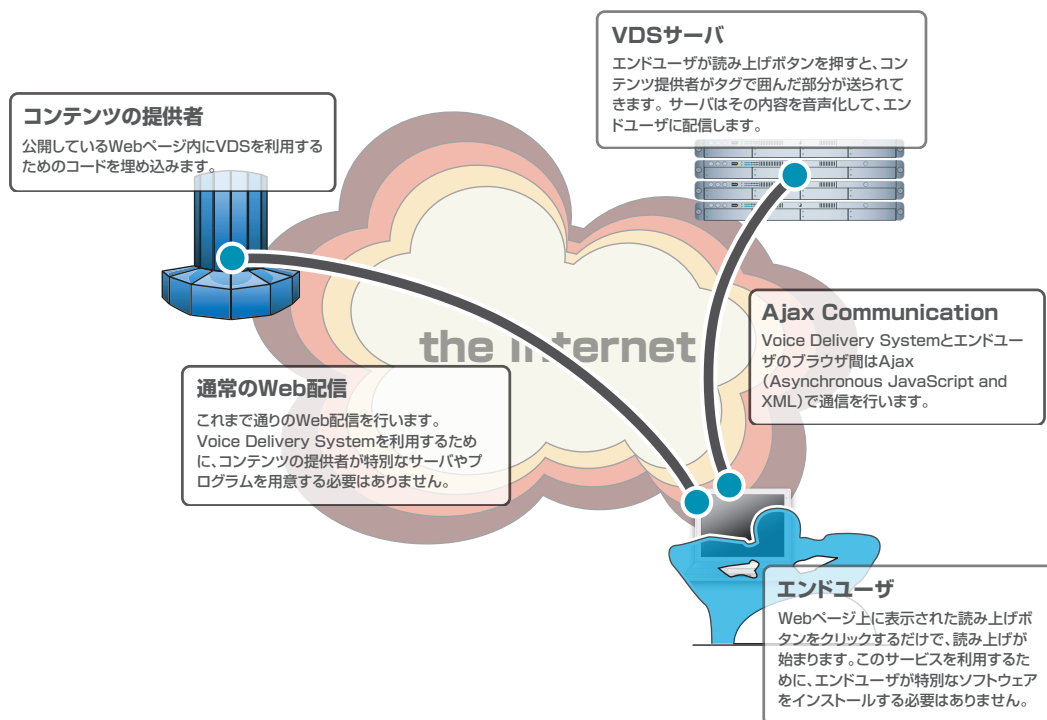
# VDSの概要

## VDSの動作概略と、APIの概要説明

この章では、Voice Delivery System(以下、VDS)の動作の概要を解説します。  
動作の概要を把握しておくことは、APIの使い方を理解する助けとなることでしょう。

### VDSのしくみ

VDSの概略図を以下に示します。



Voice Delivery System概略図

図の左上には、コンテンツの提供者がいます。そして、右下にはエンドユーザ、つまりWebページを閲覧している人がいます。この2者間では通常のWeb配信が行われます。ただし、VDSを使った音声配信を行う場合には、コンテンツの一部に、VDSサーバを利用するための認証キーなどを埋め込んでおく必要があります。

ユーザが、VDSによる音声配信を利用する場合には、コンテンツ提供者があらかじめコンテンツの一部として作成したボタンなどを操作します。すると、読み上げ対象の文字列がVDSサーバに送信され、音声合成された後、エンドユーザの所にストリーミング配信されます。エンドユーザ側でストリーミングを受けるために必要なプログラムは、FlashのオブジェクトとしてVDSサーバから自動的にロードされ起動します。エンドユーザの元から文字列が送られる部分と、サーバから音声ストリーミング配信される部分は、Ajaxによる非同期の通信により実現されています。

## クライアントにおけるVDSの動作条件

VDSは、OSやブラウザなどクライアントの環境に影響されることなく、できるだけ多くの環境で動作するように設計されています。以下の環境で動作確認を行っています。

動作保障の対象ではありませんが、JavaScriptやFlashPlayerに関する一定の条件を満たせば、他の環境でも動作するでしょう。

OS	Microsoft Windows XP SP2、Microsoft Windows Vista、Mac OS X 10.4
ブラウザ	InternetExplorer 6以上、Firefox 2.0以上、Safari 2.0以上 (ただし、いずれのブラウザでもJavaScriptが実行可能であること)
Flash Player	Adobe FlashPlayer 9.0以上
出力装置	サウンドデバイスおよび、1つ以上のスピーカー

## VDS利用上の制限事項

VDS利用するには、以下のような制限があります。この制限は無償版を対象にしたものです。また、この制限は将来予告なく変更されることがあります。

Page View	10,000 PV/day以下
文字数	50 文字/request
リクエスト数	1,000 request/day
音声	日本語男性音 (TakashiJPm)、日本語女性音 (KeikoJPf)

## まとめ

このセクションでは、VDSの概要と動作環境についてまとめました。導入に際してさらに詳しい解説が必要な場合は、VDS技術サポート (tech@vdsapi.ne.jp) まで、ご相談ください。

この次のセクションでは、VDSのAPIの具体的な使い方について解説していきます。

---

# はじめのいっぽ

## VDSを使った音声ボタンの設置 — “HELLO, VDS WORLD”

---

このセクションでは、実際にVDSを使ってWebページに音声読み上げボタンを設置する例を示します。この簡単な例を通して、VDSの使い方をつかんでください。このセクションで一度ステップを把握してしまえば、あとはこの次のセクション「VDS API詳説」が、あなたの作品作りの助けになってくれることでしょう。

このセクションでは、以下のようなページを作ります。



読み上げ機能付きのページ

ページの左上にあるボタンを押すと、そのボタンのすぐ下にある文字列を読み上げます。

### Step 1 : 準備

VDSを利用した開発を行うためには、まず認証キーを取得する必要があります<sup>1</sup>。認証キーは、VDSのWebサイト (<http://www.vdsapi.ne.jp/>) で取得できます。

さて、認証キーを手に入れたら、以下のようなコードをWebページのhead要素の中に記述します。

---

<sup>1</sup> 無償版を使う場合は、Webから簡単な操作で取得できます。有償版では、契約書類でお知らせします。

```
1. <script
2.     src="http://api.vdsapi.ne.jp/cgi-bin/vds.cgi?key=[認証ID]"
3.     type="text/javascript">
4. </script>
```

### 認証キーの入力とオブジェクトの生成

この部分は、VDSのAPIサーバからJavaScriptのプログラムを入手している部分です。認証キーは間違いがないように、コピーして確実に入力してください。type属性やcharset属性もこの通りに入力します。

前述のとおり、VDSは音声の再生にFlashを利用しています。このFlashのオブジェクトは画面表示を一切持ちませんが、ページ内のどこかに埋め込まれている必要があります。body要素の先頭に以下のように埋め込み場所を用意しておきます。

```
1. <body>
2.   <div id="vdsp"></div>
```

### Flashオブジェクトの埋め込み場所

このように、div要素で指定します。この時、id要素には任意の値を持たせることが可能です。

```
1. <script type="text/javascript">
2.   <!--
3.     var vdsp;
4.     var vds;
5.
6.     window.onload=function(){
7.       try{
8.         vdsp = new VoiceDeliveryPlayer("vdsp");
9.         vds = new VoiceDelivery(vdsp, "vds");
10.      } catch(e) {
11.        alert("Cannot Create Object.");
12.      }
13.    }
14.   -->
15. </script>
```

### 2つのオブジェクト：VoiceDeliveryPlayerとVoiceDelivery

VDSを使用するためには、2つのオブジェクトVoiceDeliveryPlayerオブジェクトとVoiceDeliveryオブジェクトが必要となります。ここでは、その準備を行います。このコードは、head要素の中に入れておくのが適当です。

2つのオブジェクトは、ページの読み込みが完了した後に生成する必要があります。そこで、6行目のようにwindow.onloadの無名関数にオブジェクトの生成部分を記述します。

VoiceDeliveryPlayerオブジェクトは、音声のボリュームやパンを制御する役割を持っています。引数に先ほど準備したFlashオブジェクトの埋め込み場所のid属性"vdsPlayer"を渡すことで、オブジェクトの生成と同時に、Flashオブジェクトが埋め込まれます。(8行目)

VoiceDeliveryオブジェクトは、音声の生成を担当するオブジェクトです。引数にVoiceDeliveryPlayerオブジェクトとVoiceDeliveryオブジェクトの変数名を渡します。(9行目)

上のリストのように、オブジェクトの生成部分にtry-catch文を使っておくと、万が一、オブジェクトの生成に失敗した場合の処理をcatch節に記述することが出来ます。

これで、準備は完了です。Step 2に進みましょう。

## Step 2 : 読み上げ範囲の指定と文字列の取り出し

ページ中の要素をJavaScriptから取り出すには、div要素やspan要素で文字列を囲んで、適当なid属性をつけておくと便利です。

1. `<div id="readOutHere">`
2.     こんにちは、世界
3. `</div>`

**読み上げ文字列をdiv要素のid属性で指定する**

上の例のように指定された範囲は、以下のような方法で取り出すことができます。

1. `var buf = (document.getElementById("readOutHere")).innerHTML;`

**文字列を取り出して、speakメソッドに渡す**

id属性で指定された要素は、getElementByIdで取り出します。引数には先ほどdiv要素で指定したid属性"readOutHere"を渡します。さらに、ここでは要素の中に含まれる文字列を取得したいので、プロパティinnerHTMLを参照して、変数bufに代入しています。

### Step 3 : 文字列を読み上げる

さて、いよいよ音声を実際に出力する部分です。ここではボタンが押されたタイミングで、文字列が読み上げられるようにするので、先ほどの文字列の取り出し部分と、読み上げ部分を1つの関数にまとめておきます。

```
1. function readOut(){
2.     var buf = (document.getElementById("readOutHere")).innerHTML;
3.     vds.speak(buf);
4. }
```

#### 文字列を取り出して、読み上げる関数

このコードは、先ほどオブジェクトを生成したscript要素の中に記述します。2行目は、Step2でとりあげた文字列の取り出し部分です。3行目が読み上げを行っている部分です。VoiceDeliveryオブジェクトのspeakメソッドを呼び出します。このとき、読み上げさせたい文字列を引数として渡します。

最後に、読み上げボタンを文字列の上に設置して、それがクリックされたら関数readOutを呼ぶようにします。

```
1. <input type="button" value="読み上げ" onClick="readOut()" />
```

#### 読み上げボタンの設置

以上で、VDSを使った音声読み上げの基本的な利用方法を一通り見てきたことになります。次のページに、ここで紹介したサンプルの全ソースを載せておきます。

```
1. <html>
2. <head>
3.   <script src="http://api.vdsapi.ne.jp/cgi-bin/vds.cgi?key=[認証キー]"
4.     type="text/javascript">
5. </script>
6. <script type="text/javascript">
7.   <!--
8.     var vdsp;
9.     var vds;
10.
11.     window.onload=function(){
12.       try{
13.         vdsp = new VoiceDeliveryPlayer("vdsp");
14.         vds = new VoiceDelivery(vdsp, "vds");
15.       } catch(e) {
16.         alert("Cannot Create Object");
17.       }
18.     }
19.
20.     function readOut(){
21.       buf = (document.getElementById("readOutHere")).innerHTML;
22.       vds.speak(buf);
23.     }
24.     -->
25. </script>
26. </head>
27.
28. <body>
29.   <div id="vdsp"></div>
30.
31.   <input type="button" value="Read Out" onClick="readOut()" />
32.   <div id="readOutHere">
33.     こんにちは、世界
34.   </div>
35. </body>
36. </html>
```

読み上げボタンを使った読み上げ機能付きページの例

## まとめ

このセクションでは、VDSの基本的な使い方を紹介しました。ここで紹介した流れは、複雑なアプリケーションの開発においても同様です。全体の流れを右の図に示します。

まず、VoiceDeliveryPlayerオブジェクトを生成します。これは、音声の制御を担当するオブジェクトです。このオブジェクトが生成されると、同時に音声の再生を行うFlashのオブジェクトも挿入されます。

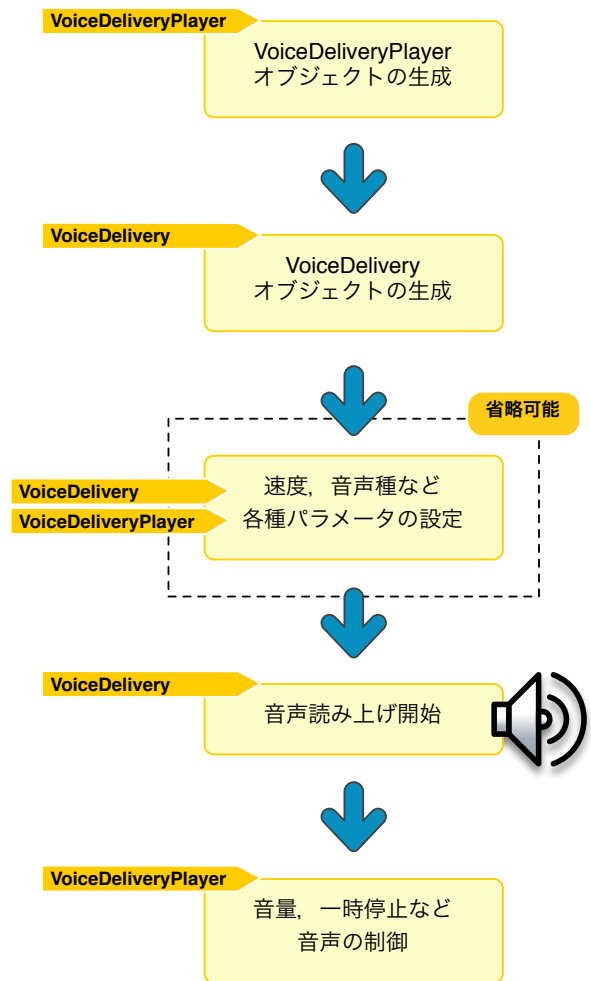
生成されたVoiceDeliveryPlayerオブジェクトを引数に渡し、VoiceDeliveryオブジェクトを生成します。このオブジェクトが、音声の合成を担当します。

このセクションでは省略しましたが、VoiceDeliveryオブジェクトのメソッドでは、音声の種類や音声の速度など、音声合成の際に使用されるパラメータが設定できます。また、再生する際の音量や左右のパンに関する値もこの時点でVoiceDeliveryPlayerオブジェクトのメソッドで設定することも可能です。これらのパラメータにはデフォルト値がそれぞれ指定されているので、読み上げようとする文字列以外は設定しなくても動作します。

VoiceDeliveryオブジェクトのspeakメソッドを呼ぶと音声合成され、再生が始まります。

音声の再生が始まった後の制御は、VoiceDeliveryPlayerオブジェクトで行います。音量の変更や一時停止、停止、等を行うことができます。

これ以降のセクションでは、この一連の流れを踏まえた上で、VDSの機能をフル活用するため、用意されているメソッド全てについて詳細に解説をしていきます。



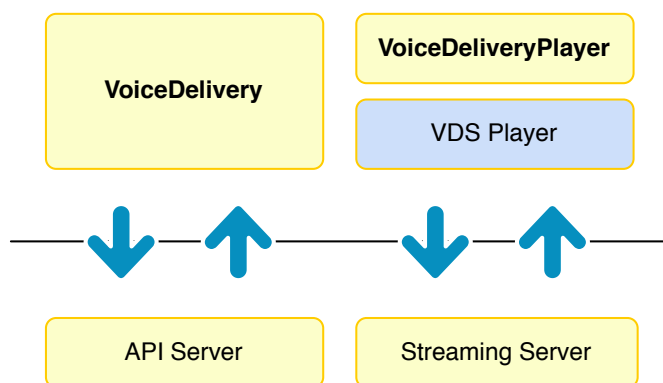
# VDS API詳説

## VDS APIの詳しい説明とサンプルコード

VDSの基本的な使い方を、前のセクションで解説しました。VDSには、より豊かなユーザインタフェースを実現するため多くのメソッドが用意されています。このセクションでは、VDSの動作について掘り下げた解説をした後、VDSが提供するAPIの詳細な説明とそのサンプルコードを示します。

### VoiceDeliveryオブジェクトとVoiceDeliveryPlayerオブジェクト

前のセクションでも触れたように、VDSには音声を生成するためのVoiceDeliveryオブジェクトと、音声読み上げを制御するVoiceDeliveryPlayerオブジェクトの2種類が用意されています。



2つのオブジェクト：「VoiceDelivery」と「VoiceDeliveryPlayer」

2つのオブジェクトの役割について、上の概念図を基に説明していきましょう。VoiceDeliveryオブジェクトは、APIサーバと通信して音声合成を行うためのオブジェクトです。文字列の他に、音声合成に必要な各種パラメータ（音声種別、音声速度など）も同時に送信されます。一方、VoiceDeliveryPlayerオブジェクトは、音声の再生を行うFlashのプログラム VDS Playerを制御するためのオブジェクトです。音声の再生に伴って必要となる様々な制御（停止、一時停止、音量など）を行うことができます。

この2つのオブジェクトを使いこなすことで、音声の合成、再生、一時停止、などを自由自在に行うことができます。

では、ここでクライアントで音声合成をしてから音声再生されるまでの流れを見ていくことにしましょう。VoiceDeliveryオブジェクトのspeakメソッド（詳しくは後述）を呼ぶと、サーバに音声合成のリクエストが送信されます。リクエストが受理され、サーバで音声の合成が終わり再生可能な状態になると、それがクライアントに通知されます。この情報を基にVoiceDeliveryオブジェクトは、VoiceDeliveryPlayerオブジェクトを介してVDS Playerを操作します。こうして、音声の再生が再生されるわけです。

## VDSのエラー処理

VDS APIを利用してエラーが発生した場合、いくつかの方法でエラーに対処することができます。

### 1. 各メソッドの戻り値

各メソッドには、メソッドが正常に動作しているかどうかを確認するため、戻り値が用意されています。正常に処理された場合は0が、何らかのエラーが発生した場合は負の値が戻ります。

### 2. try-catch文

認証キーが誤っているなどの理由で、APIが記述されているJavaScriptのファイルを入手できなかった場合、オブジェクトの生成に失敗します。この場合のエラーは、try-catch文を使用して取得します。

### 3. getServerErrorメソッド

サーバでの処理中に発生したエラーを取得する場合には、getServerErrorメソッドの戻り値を利用します。メソッドによっては、戻り値が正常終了を示していても、正しく音声再生されない場合がありますので、getServerErrorメソッドを使った確認も併せて利用します。サーバ側での処理が一通り完了していなければ、getServerErrorメソッドで正確なエラーコードを取得することはできませんので、setFinishCallbackで指定した関数の中で、使用するのが効果的です。（詳しくは、VoiceDeliveryオブジェクトのgetServerErrorの項を参照してください）

ここで示した、3通りのエラー取得方法を使用することで、VDSの使用において発生する問題に対処することができます。どのメソッドに対して、どのエラー取得方法が有効であるかは、この後のAPI一覧で解説します。

## VDSの認証とセッション

VDSでは、script要素でJavaScriptのプログラムをダウンロードする際に認証を行っています。この認証に失敗するとJavaScriptのプログラムがクライアントにダウンロードされないため、VDSの利用はできません。

この1回の認証によるセッションの有効期限を1時間としています。セッションが有効であるうちは、VDSの様々な機能を使用することができます。一度このセッションが失われてしまった場合は、リロードすれば、自動的に再度認証されセッションが有効になります。

## API一覧

これ以降では、VDS APIのオブジェクトごとにメソッドを紹介していきます。各メソッドは以下のような形式で解説されています。

**メソッド名** .....

**機能：**

メソッドが持つ機能の概要

**引数：**

メソッドが取る引数の説明

**戻り値：**

メソッドが返す値の説明

**[ getServerErrorによる戻り値 ]：**

「戻り値」でエラーが把握できない場合のエラーの取得方法（一部のメソッドのみ）

**[ 例外 ]：**

try-catch文で取得できる例外（一部のメソッドのみ）

**使用例：**

実際にメソッドを使ったプログラムの例

**[ 関連メソッド ]：**

関連のあるメソッドの一覧

## ■ VoiceDeliveryオブジェクトのメソッド

# VoiceDelivery

### 機能：

音声合成を行うためのVoiceDeliveryオブジェクトを生成するメソッド。

### 引数：

- \* **VoiceDeliveryPlayerオブジェクト**  
合成された音声の再生に使用するVoiceDeliveryPlayerオブジェクトを渡す。
- \* **VoiceDeliveryオブジェクトの変数名**  
生成されたVoiceDeliveryオブジェクトを代入する変数の名前を文字列で渡す。

### 戻り値：

- \* **VoiceDeliveryオブジェクト**

### 例外：

- \* **ReferenceError - Can't find variable**  
第1引数で指定したVoiceDeliveryPlayerオブジェクトが存在しない場合

### 使用例：

```
try{
  vdsp = new VoiceDeliveryPlayer("vdsp");
  vds = new VoiceDelivery(vdsp, "vds");
} catch(e) {
  alert("Cannot Create Object\n" + e);
}
```

- VoiceDeliveryオブジェクトの生成は、VoiceDeliveryPlayerオブジェクトの生成をした後に行います。
- オブジェクトの生成に失敗する場合に備えて、try-catch文でエラー処理を行います。
- VoiceDeliveryメソッドの2番目の引数は、変数名を文字列として与えます。

### 関連メソッド：

- ➔ **VoiceDeliveryPlayer**：音声を出力、調整するオブジェクトVoiceDeliveryPlayerを生成する。

# speak

## 機能：

音声で読み上げを行うメソッド。

## 引数：

### \* 読み上げ文字列 (省略可能)

省略する場合は、setStringメソッドで事前に読み上げ文字列を設定しておきます。

## 戻り値：

### \* 0

正常に読み上げのリクエストが受理された場合。

### \* -103

読み上げ文字列が許可されている長さを超えた場合。

### \* -105

読み上げ文字列が設定されていない場合。

文字列が、スペースや改行コードなどのみの場合も、このエラーを返します。

## getServerErrorによる戻り値：

### \* 0

正常に読み上げのリクエストが受理された場合。

### \* -301

1日あたりの再生回数を超えたリクエストを受け付けた場合

## 使用例：

```
vds.speak("VDSは、音声配信するためのWebAPIです。");
```

```
vds.setString("VDSは、音声配信するためのWebAPIです。");  
vds.speak();
```

文字コードはVDSが自動認識して処理します。

音声は、サーバで生成され配信されるため、speakメソッドを呼んでから実際に音声の再生が始まるまでには、若干のタイムラグが生じます。

☑ 連続してspeakメソッドを呼ぶ場合

- ☑ speakメソッドが呼ばれると、読み上げ音声はVDS Playerのキューに追加されます。キューの内容は自動的に再生処理にまわされます。複数回speakメソッドを呼んだ場合、先に呼ばれたspeakメソッドの音声から順に読み上げが行われます。
- ☑ 先に呼んだspeakメソッドの音声読み上げが始まってから、次のspeakメソッドを呼ぶようにします。でないと、最初のspeakメソッドが無視されることがあります。
- ☑ 音声読み上げが実際に始まっているかどうかは、VoiceDeliveryPlayerのisSpeakingメソッドで確認することができます。例えば、以下のようなコードを使うことで、タイミングをとることができます。

```
var waitId  
vds.setString("VDSは、音声配信するためのWebAPIです。");  
vds.speak();  
waitId = setInterval("if(vdsp.isSpeaking()==1){  
    clearInterval(waitId);  
    vds.speak("この声は、2番目に読み上げられます。");}" , 100);
```

**関連メソッド：**

- ➔ **setString**：読み上げる文字列を設定するメソッド。
- ➔ **isSpeaking**：音声の出力中であるか否かを調べるメソッド。

# setCast

## 機能：

読み上げに使用する音声の種類を指定します。  
デフォルトでは、「TakashiJPm」が設定されています。

## 引数：

- \* **音声種別**  
音声合成に使用する音声名を文字列で渡します。

## 戻り値：

- \* **0**  
正しい値が設定された場合。
- \* **-101**  
無効な音声指定された場合。

## 使用例：

```
vds.setCast("TakashiJPm");  
vds.speak("この声は、Takashiの声です");
```

- 現在使用できる音声は2種類（"TakashiJPm"、"KeikoJPf"）です。
- 大文字と小文字を正確に入力する必要があります。

## 関連メソッド：

- ➔ **getCast**：現在設定されている音声の種類を取得する。
- ➔ **getCastList**：設定可能な音声のリストを取得する。

# setRate

## 機能：

合成される音声の速度を設定します。音声の速度を速く設定すると、早口言葉のようになります。デフォルト値は0に設定されています。

## 引数：

### \* 音声の速度

-10(遅い)から10(速い)の21段階の値を整数で渡します。デフォルト値は0です。

## 戻り値：

### \* 0

正しい値が設定された場合。

### \* -102

無効な値が指定された場合。

有効範囲外の値が指定されてもこのエラーは戻されません。-11以下の値が渡された場合は、-10として処理をします。同様に、11以上の値が渡された場合は、10として処理をします。

## 使用例：

```
vds.setRate(-5);  
vds.speak("ゆっくりとしゃべることができます。");
```

```
vds.setRate(5);  
vds.speak("早口言葉も得意です。生麦、生米、生卵");
```

- 設定できる値は、整数です。
- しゃべっている途中で、音声の速度を変えることはできません。
- 音声の速度は、speakメソッドを呼ぶ前に指定しておく必要があります。

## 関連メソッド：

- ➔ **getRate**：現在設定されている音声の種類を得る。

# setString

## 機能：

読み上げる文字列を設定します。

## 引数：

### \* 読み上げ文字列

読み上げの内容を引数に取ります。読み上げに使用できる言語は音声種別によって異なります。現在は日本語のみの対応です（一部の英単語はカタカナ読みで読み上げます）。

## 戻り値：

### \* 0

正しい値が設定された場合。

### \* -103

引数で渡された文字列が許可されている長さを超えている場合。

### \* -105

引数で渡された文字列の長さが0の場合。

## 使用例：

```
vds.setString("事前に文字列を設定できます");  
vds.speak();
```

- 文字コードはVDSが自動認識して処理します。
- speakメソッドの前にsetStringで文字列を指定しておくことで、speakメソッドの引数を省略することができます。
- setStringで文字列を設定した後、speakメソッドを引数付きで呼び出すと、setStringで設定された値は破棄されます。

## 関連メソッド：

- ➔ **getString**：現在設定されている読み上げ文字列を返します。
- ➔ **getStringLimit**：一度に読み上げ可能な文字列の長さを返します。
- ➔ **speak**：speakメソッドの引数として読み上げ文字列を渡すことも可能です。

# setFinishCallback

## 機能：

サーバ側で音声の生成が終了した際に呼び出される関数名を指定します。  
音声の準備中であることをユーザに伝えるローダアイコンなどの制御に使用できます。

## 引数：

- \* 関数名呼び出し文  
当該関数を呼び出す際の文を文字列として渡す。

## 戻り値：

- \* なし  
引数で指定された関数が実際に存在しているか否かは、一切関知しません。

## 使用例：

```
function serverProcessFinished(){
    alert("Server Process is Finished.");
}
vds.setFinishCallback("serverProcessFinished()");
vds.speak("この文字列が読み上げられます。");
```

- setFinishCallbackの引数には、関数の呼び出し形式を記述します（カッコのつけ忘れに注意）。
- 引数を取る関数を呼び出す場合には、その引数も併せて記述します。
- 「読み上げが終了したタイミング」ではありません。

## 関連メソッド：

- **speak**：speakメソッドの引数として読み上げ文字列を渡すことも可能です。

# getCast

## 機能：

現在設定されている読み上げに使用する音声の種類を返す。

## 引数：

\* なし

## 戻り値：

\* 音声の種類 (文字列)  
音声の種類が文字列で得られる。

## 使用例：

```
voice = vds.getCast();
switch(voice){
    case "TakashiJPm":
        vds.speak("私の名前は、たかしです。");
        break;
    case "KeikoJPF":
        vds.speak("私の名前は、けいこです。");
        break;
}
```

上の例では、設定されている音声名によって、しゃべる内容を切り替えています。

## 関連メソッド：

➔ **setCast**：読み上げに使用する音声の種類を指定します。

# getCastList

## 機能：

現在使用可能な音声の種類の一覧を配列で返す

## 引数：

\* なし

## 戻り値：

\* 文字列の配列  
現在使用可能な音声の種類一覧

## 使用例：

```
var myCast;
var castList = new Array();

castList = vds.getCastList();

for (var i = 0; i < castList.length; i++) {
    var value = castList[i];
    if( value.match(/^Keiko/) ){
        myCast = value;
    }
}

vds.setCast(myCast);
```

- 上の例では、音声名の一部から音声を検索して、setCastメソッドに検索結果を渡しています。
- 利用可能な音声の中からユーザーに1つを選ばせるようなインタフェースを動的に生成する時などに、使用できるでしょう。
- 音声の種類や数、それぞれの名称は、今後変更される可能性があります。

## 関連メソッド：

- ➔ **setCast**：読み上げに使用する音声の種類を指定します。
- ➔ **getCast**：現在設定されている音声の種類を得る

# getRate

## 機能：

現在設定されている音声の読み上げ速度を返す。

## 引数：

\* なし

## 戻り値：

\* **音声の読み上げ速度**  
設定されている読み上げ速度の値（数値）

## 使用例：

```
var currentRate;  
currentRate = getRate();  
alert("現在の読み上げ速度は：" + currentRate + "です。");
```

## 関連メソッド：

→ **setRate**：音声の読み上げ速度を設定する

# getString

## 機能：

読み上げ対象として設定されている文字列を返す。

## 引数：

- \* なし

## 戻り値：

- \* **読み上げ文字列**  
setStringメソッドで設定された文字列、またはspeakメソッドの引数で指定された文字列。

## 使用例：

```
var currentString;  
currentString = getString();  
alert("現在の読み上げ文字列は:[" + currentString + "]です。");
```

## 関連メソッド：

- **setString**：読み上げ文字列設定する。
- **getStringLimit**：一度に読み上げ可能な文字列の長さを調べる

# getStringLimit

## 機能：

speakメソッドに渡すことができる文字列の長さを返す。

## 引数：

- \* なし

## 戻り値：

- \* **文字列の長さ（数値）**  
サーバ側で許可されている一回のspeakメソッド呼び出しで読み上げられる最大の文字数。

## 使用例：

```
var currentString;
currentString = getStringe();
if( currentString.lenght > vds.getStringLimit() ){
    alert("読み上げ文字列が長過ぎます。");
}
```

- ユーザに入力させた文字列を読み上げさせる場合には、上のような方法で、文字列の長さをチェックすることができます。
- 1回のspeakメソッド呼び出しで渡すことができる文字列の長さは、予告なく変更されることがあります。よって、スタティックな値をプログラム内で持つのではなく、getStringLimitを使って取得することをお勧めします。

## 関連メソッド：

- ➔ **setString**：読み上げ文字列設定する。

# getServerError

## 機能：

サーバでの音声合成時に発生したエラーを確認するためのメソッドです。  
このセクションの最初にある「VDSのエラー処理」を併せて参照してください。

## 引数：

- \* なし

## 戻り値：

- \* -301  
1日あたりの再生回数が規定値を超えた場合。
- \* -501  
セッションが失われた場合。

## 使用例：

```
vds.speak("この文字列を読み上げます。");  
switch( vds.getServerError() ){  
    case -301:  
        alert("残念ながら、1日あたりの再生回数が規定数を超えました。たくさん再生してくれて、ありがとう。");  
        break;  
    case -501:  
        alert("ページをリロードして、もう一度再生ボタンを押してみてください。");  
        break;  
}
```

- 上の例では、speakメソッドを呼んだ後、正しく再生出来たかどうか確認して、必要に応じてエラーを出力しています。
- このセクションの最初のエラー処理でも述べましたが、「メソッドの戻り値チェック」、「try-catch文による例外処理」、そしてこの「getServerError」がVDSで発生するエラーに対処する3つの方法です。

## ■ VoiceDeliveryPlayerオブジェクトのメソッド

# VoiceDeliveryPlayer

### 機能：

音声を出力、調整するためのオブジェクトVoiceDeliveryPlayerを生成するメソッド。

### 引数：

**\*** Flashオブジェクトの埋め込み場所のid

音声の再生は、Flashのオブジェクトを利用しています。そのFlashオブジェクトを埋め込むdiv要素のid属性の値を引数に取ります。

### 戻り値：

**\*** VoiceDeliveryPlayerオブジェクト

このオブジェクトを介して、音量や音声の左右バランスを調節します。

### 例外：

**\*** TypeError - Null value

引数で指定したidを持つdiv要素が見つからない場合は、この例外を投げます。

### 使用例：

```
try{
  vdsp = new VoiceDeliveryPlayer("vdsp");
  vds = new VoiceDelivery(vdsp, "vds");
} catch(e) {
  alert("Cannot Create Object\n" + e);
}
. . . . 中略 . . . .
<body>
<div id="vdsp"></div>
```

VoiceDeliveryオブジェクトを作る前にVoiceDeliverPlayerオブジェクトを生成します。

Flashオブジェクトの挿入先であるdiv要素は、bodyタグの直後におきます。

# setPan

## 機能：

左右のスピーカから出力される音声のバランスを設定します。

## 引数：

### \* バランスの数値

-10(左)から10(右)まで21段階の設定が可能です。

-10より小さい値や、10より大きな値を設定しようとした場合、自動的に最小値

-10、最大値10に設定されます。

## 戻り値：

### \* なし

## 使用例：

```
var waitId;
vds.setCast("KeikoJPf");
vdsp.setPan(-10);
vds.speak("私は左から読み上げます。");
waitId = setInterval( "if(vdsp.isSpeaking()==1){
    clearInterval(waitId);
    vds.setCast('TakashiJPm');
    vdsp.setPan(10);
    vds.speak("私は右から読み上げます。");
}" , 100);
```

- 上の例では、「KeikoJPf」の音声を右から出力し、その後「TakashiJPm」の音声で右から出力します。
- タイミングの調整には、VoiceDeliveryPlayerオブジェクトのisSpeakメソッドで紹介している方法を使っています。

## 関連メソッド：

- ➔ **getPan**：現在設定されているパンの値を返します。
- ➔ **speak**：speakメソッドの引数として読み上げ文字列を渡すことも可能です。

# setVolume

## 機能：

出力音声の音量を調節するメソッド。

## 引数：

### \* 音量の数値

0(ミュート)から100までの設定が可能です。デフォルト値は1です。

## 戻り値：

### \* なし

## 使用例：

```
vdsp.setVolume(5);  
vds.speak("大きな声で読み上げます。");
```

上の例では、ボリューム5で読み上げを行います。

10より大きな数値は、かなり大きな音に感じられますので注意が必要です。

## 関連メソッド：

→ **getVolume**：現在設定されているボリュームの値を返します。

→ **speak**：speakメソッドの引数として読み上げ文字列を渡すことも可能です。

# getPan

## 機能：

現在設定されている音声の左右バランスの値を取得します。

## 引数：

\* なし

## 戻り値：

\* バランスの数値  
-10(左)から10(右)まで21段階

## 使用例：

```
var panValue = vdsp.getPan();  
if(panValue < 0){  
    alert("左寄り");  
} else if(panValue > 0){  
    alert("右寄り");  
} else {  
    alert("まんなか");  
}
```

getPanメソッドの戻り値を基に、左右の音量バランス、つまり音声の定位が分かります。

## 関連メソッド：

➔ **setPan**：左右のスピーカから出力される音声のバランスを設定します。

# getVolume

## 機能：

現在設定されているボリュームの値を取得します。

## 引数：

- \* なし

## 戻り値：

- \* 音量の数値  
0(ミュート)から100までの設定が可能です。デフォルト値は1です。

## 使用例：

```
var volValue = vdsp.getVolume();  
  
if(volValue == 0){  
    alert("注意：音量設定が0です。ミュートされています。")  
}
```

- getValueメソッドを呼ぶことで、現在の音量が分かります。
- 上の例では、音量の値を基にユーザに警告を与えています。

## 関連メソッド：

- ➔ **setVolume**：出力音声の音量を調節するメソッド。

# speakPause

## 機能：

音声出力の一時停止・再開するメソッド。

トグルになっています。つまり、再生中には一時停止し、一時停止中には音声読み上げを再開します。

## 引数：

\* なし

## 戻り値：

\* なし

## 使用例：

```
<input type="button" value="一時停止" onClick="vdsp.speakPause()" />
```

上のようなコードで、一時停止ボタンを設置することができます。

## 関連メソッド：

➔ **speak** : speakメソッドの引数として読み上げ文字列を渡すことも可能です。

# speakStop

## 機能：

音声出力を停止するメソッド。

## 引数：

\* なし

## 戻り値：

\* なし

## 使用例：

```
<input type="button" value="停止" onClick="vdsp.speakStop()" />
```

上のようなコードで、停止ボタンを設置することができます。

## 関連メソッド：

→ **speak**：speakメソッドの引数として読み上げ文字列を渡すことも可能です。

# isPlaying

## 機能：

音声の出力中であるか否かを調べるメソッド。

## 引数：

\* なし

## 戻り値：

\* 出力の状態を表す数値  
音声出力中（1）もしくは、停止中（0）

## 使用例：

```
var waitId;
vds.speak("最初に読み上げます");
waitId = setInterval("if(vdsp.isSpeaking()==1){
    clearInterval(waitId);
    vds.speak("2番目に読み上げます");
}" , 100);
```

- ☑ speakメソッドを2つ続けて呼び出す場合、2番目のspeakメソッドの呼び出しは、音声読み上げが始まった後に行う必要があります。上の例では、isSpeakingメソッドをsetIntervalの中で使用してタイミングを計っています。

## 関連メソッド：

➔ **speak**：speakメソッドの引数として読み上げ文字列を渡すことも可能です。

## **VDS API リファレンス**

---

2007年8月21日 初版発行

株式会社ナレッジクリエーション  
〒220-001 神奈川県横浜市西区高島2-6-38  
岩井本社ビル3階