PC 用フロントエンド 仕様 Ver1.0

最終更新日： 2003/4/7
作成者 ： 中島将宏

1　はじめに

本ドキュメントは MMI システム PC 用フロントエンドが備えるモダリティを規定する。
本章で PC 用フロントエンドが要求するマシンスペック、ソフトウェア構成を示し、2 章で入力モダリティの記述、3 章で出力モダリティの記述について述べる。

1.1　フロントエンドの概要

端末として使用する PC 用フロントエンドは以下の性能を要求する。

CPU　　－ Intel® Pentium®III 800MHz
Memory　－ 256MB

PC 用フロントエンドは次のユーティリティ・アプリケーションを利用する。

1．Microsoft® Windows®2000
2．MSXML 4.0
3．Microsoft® InternetExplorer6.0
4．Microsoft® SAPI ver.5.1
5．Microsoft® Agent2.0
6．東芝音声システム V6.1

以上の要求に加え、TCP/IP 通信が可能な環境が必須である。

1.2　PC 用フロントエンドで利用可能なモダリティ

PC 用フロントエンドは以下に示すモダリティが利用可能である。

－ 入力モダリティ
1. 音声
2. マウス
3. キーボード入力
4. タイマー

－ 出力モダリティ
1. ブラウザ
2. 擬人化エージェント
3. TextToSpeech
4. タイマー

次章以降で、各モダリティの詳細について述べる。

2　入力モダリティ

2.1　タッチ

2.2　クリック

▼ 機能

- 表示中の HTML ドキュメントに対する、マウスの左ボタンクリックを入力として受け付ける。

▼ 仕様

- HTML ドキュメントを表示していなければならない。
- type 属性には”touch”を記述する。
- event 属性には”click”を記述する。
- match 属性には HTML 要素の ID 属性値を記述する。
- return 属性の変数には二つの変数を記述する。
- return 属性の変数には、左から順に、クリックされた位置の X 座標値、Y 座標値が格納される。
- 座標値はブラウザの表示領域左上を原点（0.0）とする値である。

▼ エラー

なし

▼ XISL 記述

type 属性	event 属性	match 属性	return 属性
“touch”	“click”	HTML 要素の ID 属性値	変数１ X 座標値 変数２ Y 座標値

▼ 記述例

例１）ブラウザで表示されているボタンのクリックを受け付ける

```
<input type=”touch” event=”click” match=”button1”/>
```

- 動作
  ユーザがボタン１をクリックすると入力として受け付けられる。

例１で入力の対象となる HTML の一部

```
<p>サンプル</p>
<img src=”./image.jpg” id=”image01”><br/>
<form>
<input type=”button” value=”ボタン１” id=”button1”>
<input type=”button” value=”ボタン２” id=”button2”>
</form>
```

2.3　ダブルクリック

▼ 機能

- 表示中の HTML ドキュメントに対する、マウスの左ボタンダブルクリックを入力として受け付ける。

▼ 仕様

- HTML ドキュメントを表示していなければならない。
- type 属性には”touch”を記述する。
- event 属性には”dclick”を記述する。
- match 属性には HTML 要素の ID 属性値を記述する。
- return 属性の変数には二つの変数を記述する。
- return 属性の変数には、左から順に、ダブルクリックされた位置の X 座標値、Y 座標値が格納される。
- 座標値はブラウザの左上を原点（0.0）とする値である。

▼ エラー

なし

▼ XISL 記述

type 属性	event 属性	match 属性	return 属性
“touch”	“dclick”	HTML 要素の ID 属性値	変数１ X 座標値 変数２ Y 座標値

▼ 記述例

例１）ブラウザで表示されているボタンのダブルクリックを受け付ける

```
<input type=”touch” event=”dclick” match=”button1”/>
```

- 動作
  ユーザがボタン１をダブルクリックすると入力として受け付けられる。

例１で入力の対象となる HTML の一部

```
<p>サンプルです</p>
<img src=”./image.jpg” id=”image01”><br/>
<form>
<input type=”button” value=”ボタン１” id=”button1”>
<input type=”button” value=”ボタン２” id=”button2”>
</form>
```

2.4　キー

2.4.1　プレス

▼ 機能

- “０”から“９”までの数字キーと“Ａ”から“Ｚ”までのアルファベットキーのタイプを、入力として受け付ける。

▼ 仕様

- type 属性には”key”を記述する。
- event 属性には”press”を記述する。
- match 属性には 0~9、A~Z のうち一文字を記述できる。
- match 属性の記述は半角で、アルファベットは大文字を用いる
- return 属性の変数には押されたキーの文字を返す。
- 複数キーの同時入力には対応しない。

▼ エラー

なし

▼ XISL 記述

type 属性	event 属性	match 属性	return 属性
“key”	“press”	0~9,A~Z の一文字	押された文字

▼ 記述例

例 1）”Y”キーが押された時に入力を受け付ける

```
<input type=”key” event=”press” match=”Y”/>
```

- 動作
  ユーザが”Y”キーを押すと入力として受け付けられる。

2.5　音声

2.5.1　認識

▼ 機能

- 指定された音声認識語彙文法に沿った発話の認識を、入力として受け付ける。

▼ 仕様

- type 属性には”speech”を記述する。
- event 属性には”recognize”を記述する。
- match 属性には文法ファイル名とその中で適用するルール名を記述する。（文法ファイルの詳細は付録１を参照のこと）。
- 文法ファイル名とルール名の区切りには”#”を使用する。
- return 属性に記述された変数には認識時の戻り値が格納される。

▼ エラー

- 音声認識語彙文法が正しく記述されていないとき。

▼ XISL 記述

type 属性	event 属性	match 属性	return 属性
“speech”	“recognize”	“文法ファイル名#ルール名”	文法規則と同名の変数の集合

▼ 記述例

例１）ユーザの「ハンバーガーをください」という発話を受け付ける

```
<input type=”speech” event=”recognize” match=”./grammar.txt#buy” return=”buy”/>
```

- 動作
  ユーザが「ハンバーガーをください」と発話するとそれが入力として受け付けられる。
  変数 buy には”はんばーがー”が格納される。

例１で用いられている文法ファイルの記述

```
$buy = ( $foods | $drinks ) をください;
$foods = はんばーがー | ぽてと;
$drinks = こーひー | こーら;
```

2.6　タイマー

2.6.1　タイムアウト

▼ 機能

- タイマーの”timeout”イベントを入力として受け付ける

▼ 仕様

- type 属性には”timer”を記述する。
- event 属性には”timeout”を記述する。
- match 属性にはタイマーの ID を記述する。
- タイマーは先に生成し、そのタイマーがカウント中である必要がある。

▼ エラー

- match 属性に、存在しないタイマーの ID が記述された時

▼ XISL 記述

type 属性	event 属性	match 属性	return 属性
“timer”	“timeout”	タイマーの ID	なし

▼ 記述例

例１）タイマー”timer-1”のタイムアウトイベントを受け付ける

```
<input type=”timer” event=”timeout” match=”timer-1”/>
```

- 動作
  タイマー”timer-1”のタイムアウトイベントが発火すると入力となる。
  タイマー”timer-1”は事前に作成されているものとする。詳細は出力モダリティのタイマーの項を参照。

3　出力モダリティ

3.1　ブラウザ

▼ 機能

- 通常のウィンドウで XML/HTML ドキュメントを表示する。
- モーダルウィンドウで XML/HTML ドキュメントを表示する。
- タッチ入力を受け付ける。

▼ 仕様

- type 属性には”browser”を記述する。event 属性の記述法は以下の通りである。
  - close ： ブラウザを閉じる

- <param>の name 属性
  - id（必須）：操作対象のブラウザの ID
- navigate ： XML/HTML ドキュメント・Web ページを表示する
  - <param>の name 属性
    - id（必須）：操作対象のブラウザの ID
    - uri（必須）：表示するコンテンツの URI
    - modal：モーダル表示の指定
    - width：横幅
    - height：高さ
    - position：位置（値は”move”の移動先の指定と同様）
- back ： 一つ前に表示していたページを再表示する
  - <param>の name 属性
    - id（必須）：操作対象のブラウザの ID
- reload ： ブラウザの表示を更新する
  - <param>の name 属性
    - id（必須）：操作対象のブラウザの ID
- max ： ブラウザの最大化
  - <param>の name 属性
    - id（必須）：操作対象のブラウザの ID
- min ： ブラウザの最小化
  - <param>の name 属性
    - id（必須）：操作対象のブラウザの ID
- resize ： サイズ変更
  - <param>の name 属性
    - id（必須）：操作対象のブラウザの ID
    - width（必須）：横幅
    - height（必須）：高さ
- move ：ブラウザの移動
  - <param>の name 属性
    - id（必須）：操作対象のブラウザの ID
    - position（必須）：移動先
      “center”→中央、”left-top”→左上、”left-bottom”→左下、”right-top”→右上、”right-bottom”→右下
- get ： フォームの値を読み込む
  - <param>の name 属性
    - id（必須）：操作対象のブラウザの ID
    - target（必須）：値を取得する対象の HTML 要素の ID 属性値
- set ： フォームに値を書き込む
  - name 属性のみの<param>の記述
    - id（必須）：操作対象のブラウザの ID
  - name 属性と target 属性を持つ<param>の記述
    - name 属性値（必須）：”value”
    - target 属性値（必須）：値を書き込む先の HTML 要素の ID 属性値
    - <param>要素の内容：書き込む値

- ブラウザは複数作成できる。
- ブラウザに与える ID はユニークな値にする。

- ブラウザで表示するドキュメントやページには HTML の<frame>要素は使用できない。
- event 属性が”navigate”の時に、’ <param name=”modal”/>’を指定することでモーダルブラウザとして表示できる。
- ブラウザが最大化中の event 属性が”max”の<output>、最小化中の”min”の<output>は無視される。
- event 属性が“set”の時に、指定する HTML 要素の数は制限しない。
- event 属性が“get”の時に、指定する HTML 要素の数は制限しない。
- event 属性が“get”の時、フォームの値を格納する変数の名前は操作対象の HTML 要素の ID 属性値と等しくなければならない。
- 必須項目以外で、記述に誤りがある、または記述されていない場合はすべてデフォルトが指定される。

▼ 動作

- **ブラウザのモーダル表示**
  - モーダル化されているブラウザがある時は、他のブラウザは<output>で操作できない。
  - 同時に二つ以上のモーダルブラウザを作成、または表示できない。
- **タッチ入力について**
  - HTML ドキュメントを表示している時にはタッチ入力が可能になる。

▼ エラー

- 既に存在するブラウザの ID を使ってブラウザを作成しようとしたとき。
- 存在しないブラウザを操作しようとしたとき。
- <param>の name 属性の必須項目の記述に誤りがある、または記述されていないとき。

▼ XISL 記述

<table>
<tr><th rowspan="2">type 属性</th><th rowspan="2">event 属性</th><th rowspan="2">return 属性</th><th colspan="2">param 要素</th></tr>
<tr><th>name 属性</th><th>値</th></tr>
<tr><td rowspan="10">“browser”</td><td>“close”</td><td rowspan="10">なし</td><td>id</td><td>ブラウザの ID</td></tr>
<tr><td rowspan="7">“navigate”</td><td>id</td><td>ブラウザの ID</td></tr>
<tr><td>uri</td><td>表示するコンテンツの URI</td></tr>
<tr><td>modal</td><td></td></tr>
<tr><td>width</td><td>ブラウザの幅</td></tr>
<tr><td>height</td><td>ブラウザの高さ</td></tr>
<tr><td>position*1</td><td>ブラウザの移動先
center － 中央
left-top － 左上
left-bottom － 左下
right-top － 右上
right-bottom － 右下</td></tr>
<tr><td rowspan="2">“back”</td><td rowspan="2">id</td><td rowspan="2">ブラウザの ID</td></tr>
<tr><td>“reload”</td></tr>
</table>

	“max”			
	“min”			
	“resize”		id	ブラウザの ID
			width	ブラウザの幅
			height	ブラウザの高さ
	“move”		id	ブラウザの ID
			position	*1 と同様
	“get”	フォーム中の値を格納する変数	id	ブラウザの ID
			target	値を取得する HTML 要素の ID 属性値

type 属性	event 属性	return 属性	param 要素		
			name 属性	target 属性	値
“browser”	“set”	なし	id		ブラウザの ID
			value	書き込む対象の HTML 要素の ID 属性値	書き込む値

▼ 記述例

例１）HTML ページを表示する

```
<output type=”browser” event=”navigate”>
<![CDATA[
<param name=”id”>browser1</param>
<param name=”uri”>./main.xml</param>
]]>
</output>
```

- 動作

“browser1”というブラウザが既に存在していればそのブラウザで main.xml が表示される。

存在していなければブラウザが新規に作成される。

例２）フォームに値を書き込む

```
<output type=”browser” event=”set”>
<![CDATA[
<param name=”id”>browser1</param>
<param name=”value” target=”goods_name”>鉛筆</param>
<param name=”value” target=”number”>10</param>
]]>
</output>
```

- 動作
  “browser1”に表示されているフォーム中の、”goods_name”と”number”を ID 属性値に持つ要素に対し”鉛筆”と”10”を書き込む。

例３）フォームの値を読み込む

```
<output type=”browser” event=”get” return=”goods_name, number”>
<![CDATA[
<param name=”id”>browser1</param> ← フォームを表示中のブラウザを指定
<param name=”target”>goods_name</param>
<param name=”target”>number</param>
]]>
</output>
```

- 動作
  “browser1”で表示中のフォーム中の、”goods_name”と”number”を ID 属性値に持つ要素の値を同名の変数に格納する。

例２と例３で操作の対象となる HTML ページ

```
<html>
        <head>
        <title>サンプル</title>
        </head>
        <body>
                <form> 商品名： <input type="text" id="goods_name">
                        個数： <input type="text" id="number">
                </form>
        </body>
</html>
```

3.2 エージェント

▼ 機能

- エージェントのアクション、音声出力、テキストの吹き出し表示を行う。

▼ 仕様

- type 属性には”agent”を記述する。event 属性の記述法は以下の通りである。
  - create ： 生成
    - <param>の name 属性
      - name（必須）：生成するエージェントの名前
  - destroy ： 削除
    - <param>の name 属性
      - name（必須）：削除するエージェントの名前
  - show ： 開始
    - <param>の name 属性
      - name（必須）：操作するエージェントの名前
  - hide ： 停止
    - <param>の name 属性
      - name（必須）：操作するエージェントの名前
  - move ： 一時停止
    - <param>の name 属性
      - name（必須）：操作するエージェントの名前
      - x（y とセットで記述）：移動先の X 座標
      - y（x とセットで記述）：移動先の Y 座標
      - object（pos とセットで記述）：移動先の HTML 要素
      - pos（object とセットで記述）：移動先の HTML 要素上での位置
  - character ： 再開
    - <param>の name 属性
      - name（必須）：操作するエージェントの名前
      - voice（必須）：TTS キャラクタ
  - action ： 発火間隔の設定
    - <param>の name 属性
      - name（必須）：操作するエージェントの名前
      - action（必須）：アクションの名前
  - speech ： 出力テキストを TTS 出力と吹き出し表示する
      - name（必須）：操作するエージェントの名前
      - text（必須）：出力テキスト
  - balloon ： 出力テキストを吹き出し表示する
      - name（必須）：操作するエージェントの名前
      - text（必須）：出力テキスト

- エージェントは同じキャラクタを 2 体以上生成することはできない。
- “move”で移動先の指定では座標指定と HTML 要素指定は同時に使用できない。
- “move”の移動先で HTML 要素を指定した時、ポジションは全て対象要素の上部での位置になる。
- “move”で HTML 要素を指定する時、対象要素は最前面のブラウザに存在しなければならない。

- 必須項目以外で、記述に誤りがある、または記述されていない場合はすべてデフォルトが指定される。

▼ 動作

- **エージェントの操作について**
  - エージェントの生成には以下の名前を用いる。ただし、同時に 2 体以上の同じキャラクタの生成はできない。
    - Genie
    - Merlin
    - Robby
    - Peedy
  - 表示中のエージェントに対する show、非表示中のエージェントに対する hide は無視される。

▼ エラー

- 既に存在するエージェントを生成しようとしたとき。
- 存在しないエージェントを操作しようとしたとき。
- event 属性が move であり、指定された移動先が存在しなかった時。
- <param>の name 属性の必須項目の記述に誤りがある、または記述されていないとき。

▼ XISL 記述

<table>
<tr><th rowspan="2">type 属性</th><th rowspan="2">event 属性</th><th colspan="3">param 要素</th></tr>
<tr><th colspan="2">name 属性</th><th>値</th></tr>
<tr><td rowspan="13">“agent”</td><td>“create”</td><td colspan="2" rowspan="4">name</td><td rowspan="4">エージェントの名前
”Merlin”、”Genie”、”Robby”、”Peedy”</td></tr>
<tr><td>“destroy”</td></tr>
<tr><td>“show”</td></tr>
<tr><td>“hide”</td></tr>
<tr><td rowspan="5">“move”</td><td colspan="2">name</td><td>エージェントの名前</td></tr>
<tr><td rowspan="2">I</td><td>x</td><td>X 座標（スクリーン座標）</td></tr>
<tr><td>y</td><td>Y 座標（スクリーン座標）</td></tr>
<tr><td rowspan="2">II</td><td>object</td><td>移動先のオブジェクト ID</td></tr>
<tr><td>pos</td><td>オブジェクト上での位置
center：中央
left：左
right：右</td></tr>
<tr><td rowspan="2">“character”</td><td colspan="2">name</td><td>エージェントの名前</td></tr>
<tr><td colspan="2">voice</td><td>TTS で指定できるキャラクタ</td></tr>
<tr><td rowspan="2">“action”</td><td colspan="2">name</td><td>エージェントの名前</td></tr>
<tr><td colspan="2">action</td><td>アクション名</td></tr>
<tr><td>“speech”</td><td colspan="2">name</td><td>エージェントの名前</td></tr>
</table>

		text	読み上げテキスト
	“balloon”	name	エージェントの名前
		text	表示テキスト

▼ **記述例**

例１）エージェント Merlin を表示する

```
<output type=”agent” event=”show”>
<![CDATA[
<param name=”name”>Merlin</param>
]]>
</output>
```

例２）エージェント Merlin を画面上の（100,150）の位置へ移動させる

```
<output type=”agent” event=”move”>
<![CDATA[
<param name=”name”>Merlin</param>
<param name=”x”>100</param>
<param name=”y”>150</param>
]]>
</output>
```

例３）エージェント Merlin をブラウザで表示中の画像 image01 の中央に移動させる

```
<output type=”agent” event=”move”>
<![CDATA[
<param name=”name”>Merlin</param>
<param name=”object”>image01</param>
<param name=”pos”>center</param>
]]>
</output>
```

例４）エージェント Merlin にアクションをさせる

```
<output type=”agent” event=”action”>
<![CDATA[
<param name=”name”>Merlin</param>
<param name=”action”>DoMagic1</param>
]]>
</output>
```

3.3 プレイヤー（Not Implemented）

▼ 機能

- 音声・動画ファイルの再生、停止、一時停止ができる。

▼ 仕様

- type 属性には”player”を記述する。event 属性の記述法は以下の通りである。
  - open ： プレイヤーを開く
    - <param>の name 属性
      - id（必須）：プレイヤーにつける ID
      - app（必須）：再生に使用するアプリケーション
  - close ： プレイヤーを閉じる
    - <param>の name 属性
      - id（必須）：閉じるプレイヤーの ID
  - play ： 再生の開始
    - <param>の name 属性
      - id（必須）：操作対象のプレイヤーの ID
  - stop ： 再生の停止
    - <param>の name 属性
      - id（必須）：操作対象のプレイヤーの ID
  - pause ： 再生の一時停止
    - <param>の name 属性
      - id（必須）：操作対象のプレイヤーの ID
  - open_file ： ファイルオープン
    - <param>の name 属性
      - id（必須）：ファイルを開くプレイヤーの ID
      - app：再生に使用するアプリケーション
      - uri（必須）：開くファイルの URI
      - start_flag：ファイルオープンと同時に再生を開始するか
        “true”→開始する（デフォルト）
        “false”→開始しない
      - window_flag：再生が終わった時にプレイヤーを閉じるか
        “true”→閉じる（デフォルト）
        “false”→閉じない
  - max ： プレイヤーの最大化

        - <param>の name 属性
            - id（必須）：操作対象のプレイヤーの ID
    - min ： プレイヤーの最小化
        - <param>の name 属性
            - id（必須）：操作対象のプレイヤーの ID
    - resize ： サイズ変更
        - <param>の name 属性
            - id（必須）：操作対象のプレイヤーの ID
            - width（必須）：横幅
            - height（必須）：高さ
    - move ：プレイヤーの移動
        - <param>の name 属性
            - id（必須）：操作対象のプレイヤーの ID
            - pos（必須）：移動先
                “center”→中央、”left-top”→左上、”left-bottom”→左下、”right-top”→右上、”right-bottom”→右下

- プレイヤーを作成する時に、使用するアプリケーションを指定する。
- ファイルを開く時、プレイヤーがまだ開いていなければそこで開かれる。
- “open_file”で既にプレイヤーが開かれているときにアプリケーションが指定されていても無視される。
- 必須項目以外で、記述に誤りがある、または記述されていない場合はすべてデフォルトが指定される。
- アプリケーションの指定をする時、アプリケーション名のアルファベットの大文字小文字は区別しない。

▼ 動作

- **使用できるアプリケーション**
    - 現在”open”や”open_file”で指定できるアプリケーションは以下のものに限る
        - MediaPlayer

▼ エラー

- 既に存在するプレイヤーの ID を使ってプレイヤーを作成しようとしたとき。
- 存在しないプレイヤーを操作しようとしたとき。
- 使用するアプリケーションにサポート外の記述があった時。
- <param>の name 属性の必須項目の記述に誤りがある、または記述されていないとき。

▼ XISL 記述

type 属性	event 属性	param 要素	
		name 属性	値
“player”	“open”	id	プレイヤーにつける ID
		app	使用するアプリケーション
	“close”	id	プレイヤーの ID

	“play”		
	“stop”		
	“pause”		
	“open_file”	id	プレイヤーの ID
		app	使用するアプリケーション
		uri	ファイルの URI
		start_flag	ファイルオープンと同時に再生を開始するか true：再生（default） false：再生しない
		window_flag	再生終了後プレイヤーを閉じるか true：閉じる（default） false：閉じない
	“max”	id	プレイヤーの ID
	“min”		
	“resize”	id	プレイヤーの ID
		width	プレイヤーの幅
		height	プレイヤーの高さ
	“move”	id	プレイヤーの ID

▼ **記述例**

例１）プレイヤーを作成する

```
<output type=”player” event=”open”>
<![CDATA[
<param name=”id”>player1</param>
<param name=”app”>MediaPlayer</param>
]]>
</output>
```

- 動作
  プレイヤーのウィンドウが作成される。再生には MediaPlayer を利用する。

例２）ファイルを開く

```
<output type=”player” event=”open_file”>
<![CDATA[
<param name=”id”>player1</param>
<param name=”app”>MediaPlayer</param>
<param name=”uri”>./welcome.mpg</param>
<param name=”window_flag”>false</param>
]]>
</output>
```

- 動作
  “welcome.mpg”がロードされ、再生が開始される。

3.4 TTS

▼ 機能

- テキストを合成音声で読み上げる。
- 合成音声の声質を変更できる。

▼ 仕様

- type 属性には”tts”を記述する。event 属性の記述法は以下の通りである。
  - character ： 出力音声の選択
    - <param>の name 属性
      - character（必須）：使用するキャラクタ名
        “brother”、“sister”、“boy”、“girl”、“g-father”、“g-mother”、“robot-a”、“robot-b”、“robot-c”
  - speech ： TTS 出力する
    - <param>の name 属性
      - text（必須）：出力テキスト
      - english：英語の読み上げ方法
        “true”→単語読み（デフォルト）
        “false”→スペル読み
      - number：数字の読み上げ方法
        “true”→桁読み（デフォルト）
        “false”→数字読み
      - speed：読み上げ速度
        最大値 23（早い）
        最小値 0（遅い）
        デフォルト 13
      - volume：ボリューム
        最大値 65535（大きい）
        最小値 0（小さい）
        デフォルト 56540

- 必須項目以外で、記述に誤りがある、または記述されていない場合はすべてデフォルトが指定される。

▼ 動作

- **キャラクタの変更**
  - 一度変更したキャラクタはそれ以降の<output>でも有効である。
- **TTS 出力のパラメータ**
  - TTS のパラメータ変更は継続して有効になる。

▼ エラー

- <param>要素の name 属性の必須項目の記述に誤りがある、または記述されていないとき。

▼ XISL 記述

type 属性	event 属性	param 要素	
		name 属性	値
“tts”	“character”	character	キャラクタ名
	“speech”	text	出力テキスト
		english	“true”か”false”
		number	“true”か”false”
		speed	0 ～ 23 までの数字
		volume	0～65535 までの数字

▼ 記述例

例１）TTS キャラクタを変更する

```
<output type=”tts” event=”character”>
<![CDATA[
<param name=”character”>robot-a</param>
]]>
</output>
```

- 動作
  TTS の声質が robot-a になる。

例２）TTS 出力する

```
<output type=”tts” event=”speech”>
<![CDATA[
<param name=”text”>窓は日本語、window は英語です。</param>
]]>
</output>
```

- 動作
  「まどはにほんご、ウィンドウはえいごです」と TTS 出力される。

3.5 タイマー

▼ 機能

- タイマーの生成、開始、停止、一時停止、削除、再設定を行う。
- タイムアウト時、そのイベントを入力として受け取ることができる。

▼ 仕様

- type 属性には”timer”を記述する。event 属性の記述法は以下の通りである。
  - create ： 生成
    - <param>の name 属性
      - id（必須）：生成するタイマーの ID
      - interval（必須）：タイマーの間隔(秒)
      - start_flag：タイマー生成時にカウントを始めるかどうか
        ”true”→生成時にカウント状態にする(デフォルト)
        “false”→生成時に停止状態にする
  - delete ： 削除
    - <param>の name 属性
      - id（必須）：削除対象のタイマーID
  - start ： カウントを 0 から開始
    - <param>の name 属性
      - id（必須）：操作対象のタイマーID
  - stop ： 停止
    - <param>の name 属性
      - id（必須）：操作対象のタイマーID
  - suspend ： 一時停止
    - <param>の name 属性
      - id（必須）：操作対象のタイマーID
  - resume ： 一時停止中のカウントを再開
    - <param>の name 属性
      - id（必須）：削除対象のタイマーID
  - set ： 発火間隔の設定
    - <param>の name 属性
      - id（必須）：操作対象のタイマーID

・ interval ： 間隔（秒）

- 各操作について、操作対象のタイマーは複数記述できるが、interval 等のパラメータは 1 つしか記述できない。
- 生成したタイマーの ID は一つのセッション中で常に有効である。タイマー削除後は無効となる。
- タイマーの ID には任意の文字列を指定可能である。タイマーID は必須項目であるので必ず記述する。
- タイマー生成時、start_flag が true のときはすぐにカウント状態に、false のときは停止状態にする。start_flag を指定しないときはすぐにカウント状態にする。
- interval の値は秒間隔で指定し、少数の記述は無効。値は数字のみで記述する。記述例：”10”，”150”
- interval の設定はカウント状態ではできない。
- タイマーはタイムアウト後、再び interval 分の時間の経過を待つ。
- カウント状態のタイマーは明示的に stop、suspend、delete しない限り、セッション中は動きつづける。
- 生成するタイマーの数は制限しない。
- タイマーは厳密なものでなく、システムの状態によっては実時間より若干のタイムラグが起こる可能性がある。
- 必須項目以外で、記述に誤りがある、または記述されていない場合はすべてデフォルトが指定される。

▼ 動作

- **タイマーの状態**
  - カウント状態 ： タイマーがカウントをしている状態。カウント状態になってから interval(タイマーの間隔)時間経過するとタイムアウトを発生する。
  - 一時停止状態 ： タイマーが一時停止している状態。タイマー開始から一時停止したときまでの時間を経過時間として保持している。
  - 停止状態 ： タイマーが停止している状態。interval 時間経過してもタイムアウトを発生しない。
- **各状態における動作の可否**

<table>
<tr><th></th><th>start</th><th>suspend</th><th>stop</th><th>resume</th><th>set</th><th>delete</th></tr>
<tr><td>カウント状態</td><td rowspan="2">無視</td><td>実行</td><td rowspan="2">実行</td><td>無視</td><td>無視</td><td rowspan="3">実行</td></tr>
<tr><td>一時停止状態</td><td rowspan="2">無視
無視</td><td>実行</td><td rowspan="2">実行</td></tr>
<tr><td>停止状態</td><td>実行</td><td>無視</td><td>無視</td></tr>
</table>

▼ エラー

- すでに宣言した ID を指定してタイマーを生成しようとしたとき。
- 存在しない ID でタイマーを開始、停止、一時停止、削除、再設定しようとしたとき。
- interval の値が正数以外のとき。
- <param>の name 属性の必須項目の記述に誤りがある、または記述されていないとき。

▼ **XISL 記述**

type 属性	event 属性	param 要素	
		name 属性	値
“timer”	“create”	id	タイマーの ID
		interval	timeout を発生させる間隔(sec)
		start_flag	同時にカウントを開始するかどうか true：開始（デフォルト） false：停止
	“delete”	id	タイマーの ID
	“start”		
	“stop”		
	“suspend”		
	“resume”		
	“set”	id	タイマーの ID
		interval	インターバル値

▼ **記述例**

例１）タイマーを生成する

```
<output type=”timer” event=”create”>
<![CDATA[
<param name=”id”>timer-1</param>
<param name=”interval”>10</param>
<param name=”start_flag”>false</param>
]]>
</output>
```

例２）既存のタイマーの interval を変更する

```
<output type=”timer” event=”set”>
<![CDATA[
<param name=”id”>timer-1</param>
<param name=”interval”>15</param>
]]>
</output>
```

付録 1：音声（DTMF）文法記述の仕様

1. はじめに

この文書は PC 用フロントエンドと電話用フロントエンドの音声入力で使用される認識文法を規定する。

文法の記述には SRGS(Speech Recognition Grammar Specification)を使用する。

2. 記法

2.1 トークン（Tokens）

空白、あるいは特別な構文機能を持つ記号（ ; = | () [] {} ）によって区切られた単位の事をいう。

```
hello
bon voyage
this is a test  // ４つのトークンからなる順列
“San Francisco”
2
```

なお、引用符で囲まれたトークン内には空白があってもよい。以下は全て同等なトークンとして扱われる。

```
“San Francisco”
“San
Francisco”
“    San      Francisco“
```

2.2 ルール参照（Rule References）

今回はローカル参照のみ使用する。ローカルルール参照では、参照先が同じグラマー内に存在しており、常にローカルなルール名だけから成る単純なルール名参照を用いる。
ルール参照は、ルール名の頭に$をつけることで行う。

```
$city
$digit
```

2.3 順列（Sequences）

トークン、ルール参照が一続きとなった順番列。

```
this is a test            // トークンの順列
$action $object       // ルール参照の順列
the $object is $color// トークンとルール参照の順列
```

集合は”()”によって区切られても良い。

```
Adachi | Kobayashi | Nakamura | $otherNames
( 1 | 2 | 3 )
```

2.4 タグ（Tags）

タグ”{}”内の記述は、前に記述されているトークンやルール参照に関連付けられる。発話内容に替わって、タグ内の値が<input>に渡されるようになる。

```
hamburger  |  burger {hamburger}  |  ( chicken [sandwich] ) {chicken}
```

2.5 その他

(...)	グルーピング
[...]	省略可能

“()”で囲われたトークンやルール参照はグループ化される。
ルール中に“[]”で囲われたトークンやルール参照があれば、それがない発話でもそのルールに適合させることができる。

3. ルール定義

ルール定義は以下のように記述される。

```
// ABNF 形式
$ruleName = ruleExpansion;
```

```
// 例
$city = とよはし | さやま | かなざわ
$exp = $city [の] へるぷ;
```

4. XISL の記述

<input>の match 属性にパスを含むファイル名と参照するルール名を指定する。ファイル名とルール名は”#”で区切ること。return 属性には複数の変数を記述できる。変数には認識した語彙か、もしあるれば 2.5 のタグに記述した値が格納される。
以下に<input>および、グラマーファイルの記述例を示す。

<input type=”speech” event=”recognize” match=”grammar.txt#purchase” return =”goods” />

grammar.txt

```
$goods = てぃーしゃつ | ぱんつ | そっくす | くつした{そっくす};
$num = ( いっこ| ひとつ){1} | (にこ | ふたつ ){2} | ( さんこ | みっつ ){3};
$bug = こうにゅう | かう | ください;
$purchase = $goods [を] $num [$buy];
```