PDA 端末用フロントエンドの仕様　Ver1.0

仕様作成者　：　中島将宏、中嶋真里子

最終更新日：2003/04/7

1. はじめに

本文書は PDA 端末用フロントエンドの入出力モダリティを規定する。

XISL 記述者は本文書の仕様に従って PDA 端末用の XISL アプリケーションを記述することが求められる。

1.1　フロントエンドの概要

端末として使用する PDA は以下の OS 及びデバイスを搭載しているものとする。

OS　　－　Microsoft (R) Pocket PC 2002 Software(Windows CE 3.0)

入出力インターフェース　－　マイク、スピーカー、液晶ディスプレイ、カーソルキー

LAN またはダイアルアップによるインターネットへの接続が可能なこと。

また、以下の機能が利用可能であることを前提としている。

入力機能

- ペンタッチの受け付け
- カーソルキーの受け付け
- ソフトキーボードの受け付け
- 音声認識（単語認識が可能であること）

出力機能

- Web ページの表示
- ダイアログボックスの表示
- 音声出力（Wave ファイルの再生が可能であること）

1.2　フロントエンドで利用可能な入出力モダリティ

PDA 端末用フロントエンドで利用する入力モダリティは

- ペンタッチ
- 音声入力
- カーソルキー入力
- タイマー

である。

出力モダリティは

- ブラウザ
- ダイアログボックス
- サウンド出力
- タイマー

である。

音声入力に用いる音声認識エンジンには、単語認識が可能なものを使用する。認識単語は音声認識文法記述規則に従って記述しファイルに保存する。また、認識単語が記述されたファイルを文法ファイルと呼ぶ。音声認識文法規則は付録 1 に示す。

2. 入力モダリティ

各入力モダリティにおける共通の仕様を以下に挙げる。

- 仕様以外の type 属性、event 属性が記述された場合エラーになる

2.1 ペンタッチ

ペンタッチはタップとタップ&ホールドの二つの操作が可能である。タップは画面を軽く押す操作を、タップ&ホールドは画面を数秒間押す操作を表す。

HTML ドキュメントが表示されてる場合は<a>要素の内容や画像に対して、ダイアログボックスが表示されている場合はボタンに対して、タップやタップ&ホールドを行うことが可能である。

表示されている<a>要素の内容にタップまたはタップ&ホールドを行う場合、その<a>要素の href 属性の属性値には先頭に”#”が記述されている必要がある。（例：サンプル ）

2.1.1 タップイベント

◆ **機能**

- 表示されている<a>要素の内容をタップした場合に発生するタップイベントを待ち受ける。

◆ **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”、event 属性には”tap”を指定する。
- match 属性にはタップされた<a>要素の href 属性の属性値を記述する。ただし、属性値の先頭の”#”は記述しない。
- return 属性には何も指定しない。
- 同じ HTML ドキュメントに二つ以上の同じ<a>要素の href 属性の属性値が記述されていた場合、それらへのタップは全て同じイベントとなる。

◆ **エラー**

- 共通のエラーのみ。

◆ **XISL 記述**

入力モダリティ	type属性	event属性	match属性	return属性
ペンタッチ	″touch″	″tap″	<a>要素の″#″を除いた href属性の属性値	なし

◆ **記述例&動作例**

- ブラウザで表示されているボタンに対するタップを受け付ける。

```
<input type=”touch” event=”tap” match=”sample” />
```

- 表示する HTML コンテンツの記述

```
<html>
    <head>
        <title>sample</title>
    </head>
    <body>
        <a href="#sample">サンプル</a>
    </body>
</html>
```

- 動作例
  ① ユーザが表示されている<a>要素の内容「サンプル」をタップ。
  ② 待ち受け中の<input>イベントが発生。

2.1.2 タップ&ホールドイベント

◆ **機能**

- 表示されている<a>要素の内容または画像をタップ&ホールドした場合に発生するタップ&ホールドイベントを待ち受ける。

◆ **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”、event 属性には”hold”を指定する。
- 表示されている<a>要素の内容をタップ&ホールドした時に発生するイベントを待ち受ける場合、match 属性には<a>要素の href 属性の属性値を記述する。ただし、属性値の先頭の”#”は記述しない。
- 表示されている画像をタップ&ホールドした時に発生するイベントを待ち受ける場合、match 属性には<img>要素の src 属性の属性値（画像ファイルへの URI）を記述する。
- 二つ以上の同じ画像ファイル名が存在する場合、それら画像へのタップ&ホールドは全て同じイベントとなる。
- return 属性には何も記述しない。

◆ **例外処理**

- <a>要素の内容に<img>要素が記述された画像が表示されていた場合、画像へのタップ&ホールドイベントは発生せず、<a>要素の内容へのタップイベントのみが発生する。詳しくは『2.1.1 タップイベント』を参照のこと。（例：）

◆ **エラー**

- 共通のエラーのみ。

◆ **XISL 記述**

- 表示されている<a>要素の内容へのタップ&ホールドイベント

入力モダリティ	type属性	event属性	match属性	return属性
ペンタッチ	″touch″	″hold″	<a>要素の″#″を除いたhref属性の属性値	なし

- 画像へのタップ&ホールドイベント（<img>が<a>要素内部に無い場合）

入力モダリティ	type属性	event属性	match属性	return属性
ペンタッチ	″touch″	″hold″	<img>要素のsrc属性の属性値	なし

◆ **記述例&動作例**

- XISL 中の<input>リスト

```
<input type=”touch” event=”hold” match=”../picture/sample.jpg” />
<input type=”touch” event=”hold” match=”sample” />
```

- 表示する HTML ドキュメントの記述

```
<html>
    <head>
        <title>sample</title>
    </head>
    <body>
        <img src="../picture/sample.jpg" />
        <a href=”#sample”>サンプル</a>
    </body>
</html>
```

- 動作例
  ① ユーザが画像「sample.jpg」をタップ&ホールドする。
  ② 待ち受け中の<input>イベント（画像へのタップ&ホールドイベント）が発生。

2.1.3 ダイアログボックスイベント

◆ **機能**

- ダイアログボックスの「OK」ボタン、または「cancel」ボタンがタップまたはタップ&ホールドされた場合に発生するダイアログボックスイベントを待ち受ける。

◆ **仕様**

- type 属性には”touch”、event 属性には”dialogbox”を指定する。
- 「OK」ボタンがタップまたはタップ&ホールドされた場合に発生するダイアログボックスイベントを待ち受ける場合、match 属性には”OK”を指定し、return 属性にはダイアログボックスのテキストフィールドへ入力された値を格納するための変数を記述する。
- 「cancel」ボタンがタップまたはタップ&ホールドされた場合に発生するダイアログボックスイベントを待ち受ける場合、match 属性には”cancel”を指定し、return 属性には何も記述しない。
- 予め、ダイアログボックス表示イベントによってダイアログボックスが表示されているときのみ、ダイアログボックスイベントが発生する。詳しくは『3.1.2　ダイアログボック

ス表示』を参照のこと。

- 「OK」ボタン、「cancel」ボタンが押されてもダイアログボックスは閉じられない。ダイアログボックスを閉じるためにはダイアログボックス終了イベントを出力する必要がある。詳しくは『3.1.3.　ダイアログボックス終了』を参照のこと。

◆ エラー

共通のエラーのみ。

◆ XISL 記述

- 「OK」ボタンへのダイアログボックスイベント

入力モダリティ	type属性	event属性	match属性	return属性
ペンタッチ	″touch″	″dialogbox″	″OK″	ダイアログボックスのユーザフィールドに入力された値を格納するための変数

- 「cancel」ボタンへのダイアログボックスイベント

入力モダリティ	type属性	event属性	match属性	return属性
ペンタッチ	″touch″	″dialogbox″	″cancel″	なし

◆ 記述例&動作例

- XISL 中の<input>の記述

```
<input type=”touch” event=”dialogbox” match=”OK” return=”name” />
<input type=”touch” event=”dialogbox” match=”cancel” />
```

- ダイアログボックスを表示するための<output>記述

```
<output type=”dialogbox” event=”open”>
        <![CDATA[
        <param name=”system_field”>名前を入力してください</param>
        ]]>
</output>
```

- 動作例
  ① 予め、ダイアログボックス表示の<output>を出力しておく。
  ② ユーザがダイアログボックスにデータを入力。（例：中嶋真里子）
  ③ ユーザが「OK」ボタンを押す。

④ 待ち受けていた<input>イベント（「OK」ボタンへのダイアログボックスイベント）が発生。（return 属性 name には中嶋真里子が格納される）

2.2 音声入力

2.2.1 音声認識完了イベント

◆ **機能**

- 指定された音声認識文法とマッチする発話が入力された場合に発生する、認識完了イベントを待ち受ける。

◆ **仕様**

- type 属性には”speech”、event 属性には”recognize”を指定する。
- match 属性には文法ファイル名と認識単語のルール名を記述する。
- 文法ファイル名とルール名は、文法ファイル名の後ろにルール名を指定し、両者を”#”で区切る。(例：match=”grm.txt#$goods”)
- 文法ファイルの記述にミスがあった場合、音声入力を不可にする。
- 認識単語の記述は、付録 1 の音声認識文法記述に従う。
- return 属性には認識単語のタグを格納するための変数を記述する。（タグについては付録１を参照のこと）
- 同時に待ち受けることができるルール名は 30 個までとする。
- 一つの認識単語は 31 文字まで。

◆ **エラー**

- 文法ファイルの認識文法が正しく記述されていない。
- 音声認識完了の<input>が 31 個以上指定された。

◆ **XISL 記述**

入力モダリティ	type属性	event属性	match属性	return属性
音声入力	″speech″	″recognize″	文法ファイル名とルール名	認識単語のタグを格納するための変数

◆ **記述例&動作例**

- XISL 中の<input>記述

```
<input type=”speech” event=”recognize” match=”grm.txt#$goods” return=”clothes”/>
```

- 文法ファイル(grm.txt)の記述

```
$goods = すかーと{0001} | てぃーしゃつ{0002} | せーたー{0003};
$help = へるぷ | せつめい;
                ・・・
```

- 動作例
  ① ユーザが「スカート」と発話。
  ② 待ち受け中の<input>イベントが発生。（return 属性 clothes には「0001」が格納される）

2.2.2 リジェクト（認識失敗）イベント

◆ **機能**

- 指定された音声認識文法とマッチしない発話が入力された場合に発生する、リジェクトイベントを待ち受ける。

◆ **仕様**

- type 属性には”speech”、event 属性には”reject”を指定する。
- match 属性には何も指定しない。（match=””）
- return 属性は指定しない。

◆ **エラー**

- 共通のエラーのみ。

◆ **XISL 記述**

入力モダリティ	type属性	event属性	match属性	return属性
音声入力	″speech″	″reject″	なし	なし

◆ **記述例&動作例**

- XISL 中の現在待ち受け中の<input>リスト記述

```
<input type=”speech” event=”recognize” match=”grm.txt#$goods” return=”clothes” />
<input type=”speech” event=”recognize” match=”grm.txt#$help” return=”help” />
<input type=”speech” event=”reject” match=”” />
```

- 文法ファイル（grm.txt）の記述

```
$goods = すかーと | てぃーしゃつ | せーたー;
$help = へるぷ | せつめい;
        ・・・
```

- 動作例
  ① ユーザが待ち受けていない語彙を発話（例：「おはよう」と発話）
  ② 待ち受け中のリジェクト<input>イベントが発生。（<input>リストにリジェクトの待ち受けがない場合、いずれの<input>も満たされない）

2.3 カーソルキー入力

◆ **機能**

- カーソルキーを操作された場合に発生する、カーソルキーイベントを待ち受ける。ソフトウェアキーボードのカーソルキーを押しても同様のイベントが発生する。ただし、ダイアログボックスが表示されていると、カーソルキーイベントは発生しない。

◆ **仕様**

- type 属性には”key”を指定する。
- 操作に対する各属性値を以下に示す。
  - “key_push”・・・カーソルキーが押された。
  - “key_up”・・・カーソルキーが上方向に押された。
  - “key_down”・・・カーソルキーが下方向に押された。
  - “key_right”・・・カーソルキーが右方向に押された、またはソフトウェアキーボードの右矢印が押された。
  - “key_left”・・・カーソルキーが左方向に押された、またはソフトウェアキーボードの左矢印が押された。
- match 属性には何も指定しない。（match=””）
- return 属性は記述しない。

◆ **エラー**

- 共通のエラーのみ。

◆ **XISL 記述**

入力モダリティ	type属性	event属性	match属性	return属性
カーソルキー入力	″key″	″key_push″	なし	なし
		″key_up″		
		″key_down″		
		″key_right″		
		″key_left″		

◆ **記述例&動作例**

- XISL 中の<input>記述

```
<input type=”key” event=”key_push” match=””/>
```

- 動作例
  ① ユーザがカーソルキーを押した
  ② 待ち受け中の<input>イベントが発生

2.4 タイマー入力

2.4.1 タイムアウトイベント

◆ **機能**

- カウント中のタイマーのタイムアウト（指定時間の経過）した場合に発生する、タイムアウトイベントを待ち受ける。

◆ **仕様**

- type 属性には”timer”、event 属性には”timeout”を指定する。
- match 属性には予めタイマー出力で指定したタイマーID を記述する。詳しくは『3.3　タイマー出力』を参照のこと。
- return 属性は何も記述しない。

◆ **エラー**

- 共通のエラーのみ。

◆ **XISL 記述**

入力モダリティ	type属性	event属性	match属性	return属性
タイマー	″timer″	″timeout″	タイマーID	なし

◆ **記述例&動作例**

- XISL 中の<input>記述

```
<input type=”timer” event=”timeout” match=”timer-no1”/>
```

- タイマー（タイマーID は”timer-no1”）を生成するための<output>記述

```
<output type=”timer” event=”create”>
        <![CDATA[
        <param name=”id”>timer-no1</param>
        <param name=”duration”>10</param>
        <param name=”start_flag”>true</param>
        ]]
</output>
```

- 動作例
  ① 予め、タイマー生成の<output>を出力しておく。
  ② タイマー（タイマーID は”timer-no1”）を生成。
  ③ 10 秒が経過する。（name 属性”interval”の条件が満たされる）
  ④ 待ち受け中の<input>イベントが発生。

3. 出力モダリティ

各出力モダリティにおける共通の仕様を以下に挙げる。

- 仕様以外の type 属性、event 属性が記述された場合はエラーとする。
- <output>に省略可以外の<param>の記述がない場合はエラーとする。
- <output>に name 属性の値が同じ<param>要素が複数記述された場合は、上に記述されている方を優先して出力する。

3.1 ブラウザ

3.1.1 ブラウザナビゲート(コンテンツ表示)

◆ **機能**

- 指定されたコンテンツを表示する。

◆ **仕様**

- type 属性には”browser”、event 属性には”navigate”を指定する。
- CDATA セクション内で記述する<param>要素の name 属性には”uri”を指定し、表示するコンテンツの URI を記述する。
- コンテンツの URI が正しいかどうかは評価しない。
- 正しくページが表示されなかったときは、PocketIE の動作に従う。
- 表示する HTML ドキュメントの記述は基本的に自由である。ただし、<a>要素の href 属性の属性値の先頭に”#”を記述すること。
- 画像を表示する場合、表示される画像のファイル形式は PDA の PocketIE で表示できる形式とする。

◆ **エラー**

- name 属性の記述に誤りがある、または記述されていない。

◆ **XISL 記述**

出力モダリティ	type属性	event属性	param要素	
			name属性	値
ブラウザ	″browser″	″navigate″	″uri″	表示するコンテンツのURI

◆ **記述例&動作例**

- XISL 中の<output>記述

```
<output type=”browser” event=”navigate”>
        <![CDATA[
        <param name=”uri”>http://www.vox.tutkie.tut.ac.jp/</param>
        ]]>
</output>
```

- 動作例
  ① ブラウザナビゲートの<output>を出力。
  ② http:// www.vox.tutkie.tut.ac.jp/を表示する。

3.2 ダイアログボックス

3.2.1 ダイアログボックスの表示

◆ **機能**

- ユーザからデータを取得するために、ダイアログボックスを表示する。
- 表示するダイアログボックスはシステムが指定したテキストを表示するシステム用フィールドと、ユーザが値を入力するユーザ用フィールド、「OK」ボタン、「cancel」ボタンから構成されている。

◆ **仕様**

- type 属性には”dialogbox”、event 属性には”open”を指定する。
- CDATA セクション内で記述する<param>要素の name 属性には以下を指定する。
  - ”system_field”：システム用フィールドに表示するテキストを指定する。
  - “user_field”：ユーザ用フィールドに表示するテキストを指定する。（省略可）
- 一度に表示できるダイアログボックスは一つとする。

◆ **エラー**

- name 属性の記述に誤りがある、または記述されていない。
- すでにダイアログボックスが表示されている。

◆ **XISL 記述**

出力モダリティ	type属性	event属性	param要素	
			name属性	値
ブラウザ	″dialogbox″	″open″	″system_field″	システム用フィールドに表示するテキスト
			″user_field″ [省略可]	ユーザ用フィールドに表示されるテキスト

◆ **記述例&動作例**

- XISL 中の<output>記述

```
<output type=”dialogbox” event=”open”>
        <![CDATA[
        <param name=”system_field”>お名前を入力してください</param>
        ]]>
</output>
```

- 動作例
  ① ダイアログボックス表示の<output>を出力。
  ② テキストフィールドに「お名前を入力してください」と書かれたダイアログボックスを表示する。

  ダイアログボックスへの入力に関しては『2.1.3　ダイアログボックスイベント』の記述例&動作例を参照のこと。

3.2.2　フィールド値の変更

◆ **機能**

- すでに表示してあるダイアログボックスの各フィールドの値を変更する。

◆ **仕様**

- type 属性には”dialogbox”、event 属性には”set”を指定する。
- CDATA セクション内で記述する<param>要素の name 属性には以下を指定する。
  - ”system_field”：システム用フィールドに表示するテキストを記述する。（省略可）
  - ”user_field”：ユーザ用フィールドに表示するテキストを記述する。（省略可）
- <param>要素の値を指定しない場合、それぞれのフィールドの値は空になる。

◆ **エラー**

- name 属性の記述に誤りがある、または記述されていない。
- ダイアログボックスが表示されていない。

◆ **XISL 記述**

出力モダリティ	type属性	event属性	param要素	
			name属性	値
ブラウザ	″dialogbox″	″set″	″system_field″ [省略可]	システム用フィールドに 表示するテキスト
			″user_field″ [省略可]	ユーザ用フィールドに 表示するテキスト

◆ **記述例&動作例**

- XISL 中の<output>記述

```
<output type=”dialogbox” event=”set”>
<![CDATA[
<param name=”system_field”>お名前は以下でよろしいですか</param>
<param name=”user_field”>中嶋真里子</param>
]]>
</output>
```

- 動作例
  ① ダイアログボックス表示変更の<output>を出力。
  ② システム用フィールドに「お名前は以下でよろしいですか」が、ユーザ用フィールドには「中嶋真里子」が表示される。

  ダイアログボックスへの入力に関しては『2.1.3　ダイアログボックスイベント』の記述例&動作例を参照のこと。

3.2.3　表示内容を取得

◆ **機能**

- ダイアログボックスのユーザ用フィールドに入力された値を取得する。

◆ **仕様**

- type 属性には”dialogbox”、event 属性には”get”を指定する。
- CDATA セクション内で記述する<param>要素はない。

- return 属性を持ち、ユーザ用フィールドに入力された値を格納するための変数を記述する。

◆ **エラー**

- name 属性の記述に誤りがある、または記述されていない。
- ダイアログボックスが表示されていない。

◆ **XISL 記述**

出力モダリティ	type属性	event属性	return属性	param要素	
				name属性	値
ブラウザ	″dialogbox″	″get″	ユーザ用フィールドに入力された値を格納するための変数	なし	なし

◆ **記述例&動作例**

- XISL 中の<output>記述

```
<output type=”dialogbox” event=”get” return=”input_text”/>
```

- 動作例
  ① ユーザがダイアログボックスのユーザ用フィールドに値を入力。（例：中嶋真里子）
  ② ダイアログボックスデータ取得の<output>を出力。
  ③ <output>実行イベントを送信。（return 属性で指定した変数 input_text には「中嶋真里子」が格納される）

3.2.4 ダイアログボックス終了

◆ **機能**

- 表示されているダイアログボックスを閉じる。

◆ **仕様**

- type 属性には”dialogbox”、event 属性には”close”を指定する。
- CDATA セクション内で記述する<param>要素はない。
- ダイアログボックスが表示されていない場合は無視する。

◆ **エラー**

- 共通のエラーのみ。

◆ **XISL 記述**

出力モダリティ	type属性	event属性	param要素	
			name属性	値
ブラウザ	″dialogbox″	″close″	なし	なし

◆ **記述例&動作例**

- XISL 中の<ouput>記述

```
<output type=”dialogbox” event=”close”/>
```

- 動作例
  ① ダイアログボックス終了の<output>を出力。
  ② ダイアログボックスを閉じる。（ダイアログボックスが表示されていない場合は無視する）

ダイアログボックスの表示は『3.1.2　ダイアログボックス表示』を、ダイアログボックスへの入力に関しては『2.1.3 ダイアログボックスイベント』を参照のこと。

3.3　サウンド出力

◆ **機能**

- 指定された音声ファイルの再生、停止、一時停止を行う。

◆ **仕様**

- type 属性には”sound”を指定する。
- event 属性に指定する各属性値を以下に示す。
  - “open_file”･･･ファイルを開き再生する。
    - CDATA セクション内に記述する<param>要素の name 属性には以下を指定する。
      - “file_name”：再生するファイル名を記述する
      - “start_flag”：ファイルを開くと同時に再生するかどうか（省略可）
        - “true”･･･再生する（デフォルト）
        - “false”･･･再生しない
    - name 属性の”start_flag”が指定されていなかった場合は、同時に再生される。
    - 別のファイルが再生中または一時停止中の場合、停止状態にして新たなファイルを開き再生する。

- “stop”･･･再生を止める。
  - CDATA セクション内に記述する<param>要素の name 属性は指定しない。
  - 停止状態の場合は、無視する。
  - 一時停止中の場合、一時停止を解除して停止状態にする。
- “play”･･･現在開いているファイルを再生する。
  - CDATA セクション内に記述する<param>要素の name 属性は指定しない。
  - 再生中の場合、無視する。
  - 一時停止中の場合、一時停止を解除し停止状態にして再生する。
- “pause”･･･再生の一時停止と再開。
  - CDATA セクション内に記述する<param>要素の name 属性は指定しない。
  - 再生中の場合は一時停止、一時停止中の場合は続きから再生する。
  - 停止状態の場合は、無視する。

➢ 出力するファイルは Wave 形式に限定する。

➢ Wave ファイルはドキュメントサーバーからダウンロードして使用する。

◆ **エラー**

➢ 指定されたファイルが存在しない。

➢ name 属性の記述に誤りがある、または記述されていない。

➢ 開いているファイルがない状態で、”play”,”stp”,”pause”が指定された。

◆ **XISL 記述**

出力モダリティ	type属性	event属性	param要素	
			name属性	値
サウンド出力	″sound″	″open_file″	″file_name″	再生するファイルのURI
			″start_flag″ [省略可]	同時に再生するかどうか true:再生（デフォルト） false:停止
		″stop″	なし	なし
		″play″		
		″pause″		

◆ **記述例&動作例**

➢ XISL 中の<output>記述

```
<output type=”sound” event=”open_file”>
        <![CDATA[
        <param name=”file_naem”>sample.wav</param>
        <param name=”start_flag”>true</param>
        ]]>
</output>
```

➢ 動作例

① 音声出力の<output>を出力。

② sample.wav を再生する。

3.4 タイマー出力

◆ **機能**

➢ タイマーの生成、削除、開始、終了、一時停止、再開、再設定を行う。

◆ **仕様**

➢ type 属性には”timer”を指定する。

➢ event 属性に指定する各属性値を以下に示す。

- “create”･･･生成する
  - CDATA セクション内に記述する<param>要素の name 属性には以下を指定する。
    - “id”：生成するタイマーの ID
    - “duration”：生成するタイマーの timeout を発生させる間隔（sec）
    - “start_flag”：生成と同時にカウントをはじめるかどうか（省略可）
      - true･･･開始（デフォルト）
      - false･･･停止
  - name 属性”start_flag”が記述されていない場合、生成と同時にカウントが開始される。
  - 同じ ID のタイマーを二つ以上生成することはできない。
- “destroy”･･･削除する
  - CDATA セクション内に記述する<param>要素の name 属性には以下を指定する。
    - “id”：削除するタイマーの ID
  - カウントの状態にかかわらずタイマーを削除する。
- “start”･･･カウントを開始する

- CDATA セクション内に記述する<param>要素の name 属性には以下を指定する。
  - “id”：カウントを開始するタイマーの ID
- 停止状態ならばカウントを開始、一時停止状態ならばカウントを再開する。

● “stop”･･･カウントを停止する
- CDATA セクション内に記述する<param>要素の name 属性には以下を指定する。
  - “id”：カウントを停止するタイマーの ID
- カウント状態、一時停止状態のときカウントを停止する。

● “suspend”･･･カウントを一時停止する
- CDATA セクション内に記述する<param>要素の name 属性には以下を指定する。
  - “id”：カウントを一時停止するタイマーの ID
- カウント状態ならば一時停止状態に、停止状態ならば無視する。

● “duration”･･･インターバル値を再設定する
- CDATA セクション内に記述する<param>要素の name 属性には以下を指定する。
  - “id”：再設定するタイマーの ID
  - “duration”：再設定する timeout 発生間隔（sec）
- カウントの状態にかかわらず再設定する。
- 再設定後、タイマーの状態はそのままでカウント値を 0 に戻す。

➢ システムが可能なかぎり、タイマーを生成することができる。
➢ タイマーがカウントしている状態をカウント状態、カウントをしていない状態を停止状態、カウントを一時停止している状態を一時停止状態と呼ぶ。
➢ タイマーの状態が出力実行前と変わらない場合は無視する。(例えば、タイマー停止中に”stop”が指定されるなど)
➢ タイマーはタイムアウト後、再びカウント値 0 からカウントを開始する。
➢ duration の値は秒単位で指定し、0~9 の半角数字のみ使用できる。

◆ **エラー**

➢ すでに同じ ID のタイマーが生成されている。
➢ 存在しない ID のタイマーを削除、開始、終了、一時停止、再設定を行う。
➢ name 属性の記述に誤りがある、または記述されていない。

◆ **XISL 記述**

出力モダリティ	type属性	event属性	param要素	
			name属性	値
タイマー	"timer"	"create"	"id"	生成するタイマーのID
			"duration"	timeoutを発生させる間隔（sec）
			"start_flag" [省略可]	同時にカウントを開始する かどうか "ture"→開始（デフォルト） "false"→停止したまま
		"destroy"	"id"	操作タイマーのID
		"start"		
		"stop"		
		"suspend"		
		"duration"	"id"	設定するタイマーのID
			"duration"	timeoutを発生させる間隔(sec)

◆ **記述例&動作例**

- XISL 中の<output>記述

```
<output type=”timer” event=”create”>
        <![CDATA[
        <param name=”id”>timer-no1</param>
        <param naem=”duration”>10</param>
        <param name=”start_flag”>true</param>
        ]]
</output>
```

- 動作例

① タイマー出力の<output>を出力。

② タイマーID が timer-no1、タイムアウト間隔 10 秒のタイマーを生成し、すぐにカウントを開始する。

タイムアウトに関しては、『2.4.1 タイムアウトイベント』の記述例&動作例を参照のこと。

付録 1：音声認識文法記述規則について

- **音声認識文法記述規則**

1. RULE := RULENAME '=' WORD {'|' WORD}* ';'
2. RULENAME := '$' ALPHA*
3. WORD := CHAR '{' TAG '}' *:

※RULE をルール、RULENAME をルール名、WORD を認識単語、
TAG をタグと呼ぶ

- 各ルールは一行で記述する。
- ALPHA は"a"～"z"、0～9 の英数字からなり、大文字小文字の区別をする。
- WORD の"{TAG}"は記述しなくても良い。
- CHAR は平仮名からなる。
- TAG は"a"～"z",0～9 の英数字からなり、大文字小文字の区別をする。

- **認識文法記述例**

```
$help = へるぷ | せつめい;
$buy = かう | こうにゅう | ください;
$about = しょうさい;
$goods = せーたー{0001} | ぱんつ{0002} | すうぇっと{0003};
```

- 上の例において、ルール名「$help」の認識単語は「へるぷ」と「せつめい」である。
- 「せーたー」が認識された場合、タグの値は「0001」である。