Tellme Voice Studio

Tellme Voice Studio is a voice application design and development tool based on Microsoft's Visual Studio Domain Specific Language (DSL) toolkit.

Version 0.07
August 4, 2008

Mike Trinkala
trink@tellme.com

Contents

Overview

Voice Studio is a graphical voice application design and development tool that stores a representation of a voice application as an XML model. The application model is used to generate various artifacts: documentation, recording scripts, VXML, and JavaScript to name a few. The generation process is open and extensible; meaning the developer can change the way the current artifacts are generated or even create new artifacts.

Terms

The following terms appear throughout this document, and refer to specific terminology used in the Tellme Voice Studio application.

Application – A collection of modules comprising an entire voice application.

Call flow – The visual representation of the application flow.

Canvas – The visual work area where the call flow design takes place.

Model – The XML representation of a module (in this context it is synonym for Module).

Module – A collection of states and transitions comprising a functional piece of a voice application (one call flow).

State – A single shape on the canvas.

Transition – A line connecting two shapes on the canvas that indicates how the call passes from one state to another.

Voice Studio Toolbox: States and Transitions

This section describes all of the state tools in the *Voice Studio* toolbox, how they behave on the design canvas, and what their various properties do. All states have the following properties: **Name**, **Encoded Name**, and **Notes**. Some states have additional properties.

- **Name:** Displays in the center of the state shape on the diagram (with the exception of the Start and Exit states). The state names MUST be unique per module with the exception of Module state names and can be edited directly by double-clicking on the label.
- **Encoded Name** is a read only value derived from the **Name** property. It is used for filenames and identifiers in generated code.
- **Notes** property is used in the generation process as supplemental documentation.

Start

Icon	White circle ○
Occurrences (min,max)	1,1
Inbound Transitions (min,max)	0,0
Outbound Transitions (min,max)	1,1

The Start state serves as the entry point into the module. The **Name** property can be modified but it is not displayed within the tool. It is recommended to leave the default name provided (*e.g.,* Start1).

Exit

Icon	Black, Red, or Blue circle ●
Occurrences (min,max)	1,Many
Inbound Transitions (min,max)	1,Many
Outbound Transitions (min,max)	0,0
Double-click	Black (Return) or Blue (Reprompt) will return to the module that opened it (if this module was opened directly nothing will happen)
Right-click	Since double-click is used for navigation you will need to use the right-click menu to access the state properties

The Exit state serves as the exit point from the module. The **Name** property can be modified but it is not displayed within the tool. It is recommended to leave the default name provided (*e.g.,* Exit1).

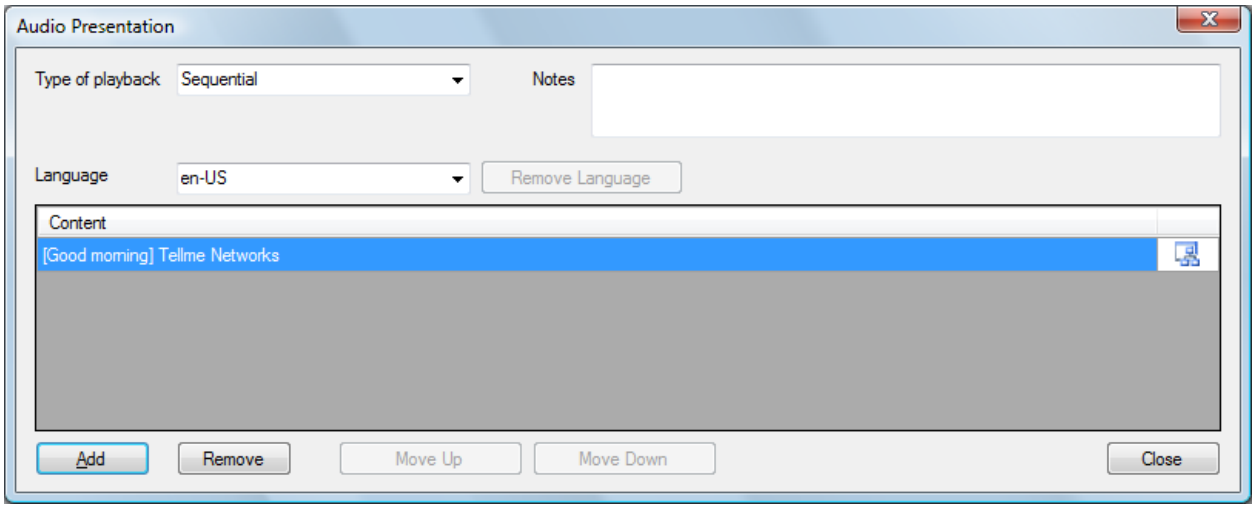
Properties

- **Type –** enumeration
  - **Return** (Black) – Exits, returning to the calling module (if there is no calling module an error will be thrown).
  - **Disconnect** (Red) – Ends the call at this point.
  - **Reprompt** (Blue) – Only used in a Confidence Base Confirmation module. This should be configured as the exit path of a negative confirmation as it will add the item to the skip list and reprompt the user for input.

Presentation

Icon	Antique White rounded rectangle
Occurrences (min,max)	0,Many
Inbound Transitions (min,max)	1,Many
Outbound Transitions (min,max)	1,1
Double-click	Opens the Audio Presentation dialog

The presentation state is designed to support multi-modal and multi-lingual output. However, in the current release the only modality exposed is audio. Also, to date testing has focused exclusively on English but will be expanded as soon as possible.



Properties

- **Type –** enumeration (applies to all languages)
  - **Sequential** (default) – The audio files will be played in the order they are listed.
  - **Random** – One of the audio files in the list will be randomly played.
  - **Custom** – A JavaScript stub function will be created and a developer will need to enter the logic to control which audio file will playback.
- **Language** – Displays the language of the current presentation. If a new language is added, all presentation states in the module must contain that language.

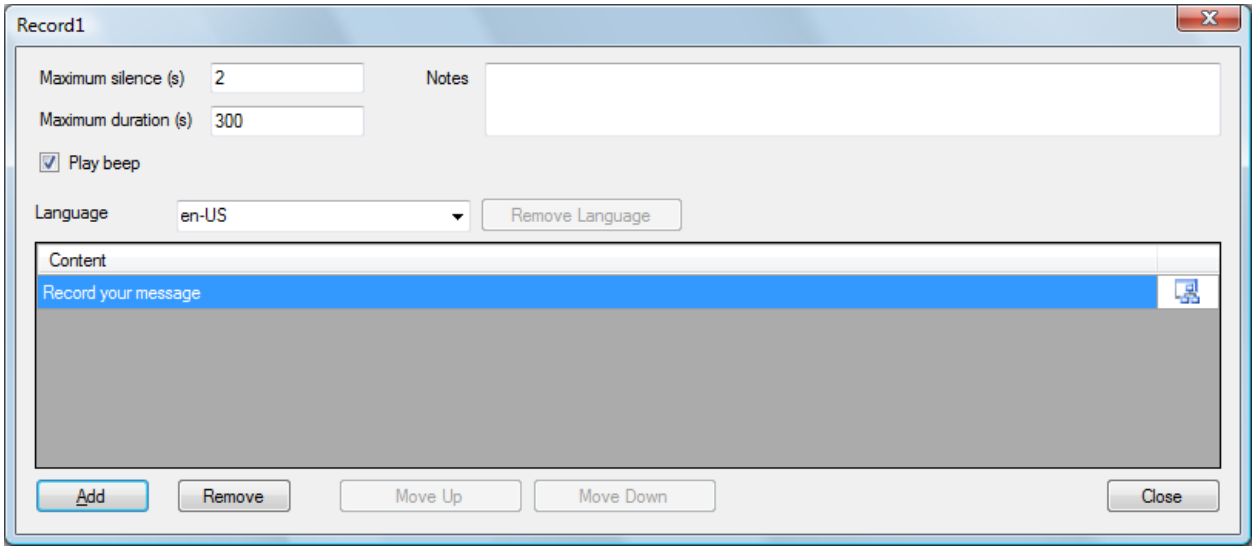
Audio Columns

- **Content** – See Audio Content Format.
- **Icon –** Click to load the Audio Builder (advanced audio configuration tool)**.**

Record

Icon	Green rectangle
Occurrences (min,max)	0,Many
Inbound Transitions (min,max)	1,Many
Outbound Transitions (min,max)	1,1
Double-click	Opens the Record dialog

The record state is designed to support multi-modal and multi-lingual output. However, in the current release the only modality exposed is audio. Also, to date testing has focused exclusively on English but will be expanded as soon as possible. The resulting recording is stored in a JavaScript variable named **appsRecordedAudio.**



Properties

- **Maximum silence –** The interval of silence that indicates the end of speech in seconds.
- **Maximum duration –** The maximum recording time in seconds. The maximum limit is 1800 seconds (30 minutes).
- **Play beep –** When checked; a beep is played before recording begins.
- **Language** – Displays the language of the current presentation. If a new language is added, all presentation states in the module must contain that language.

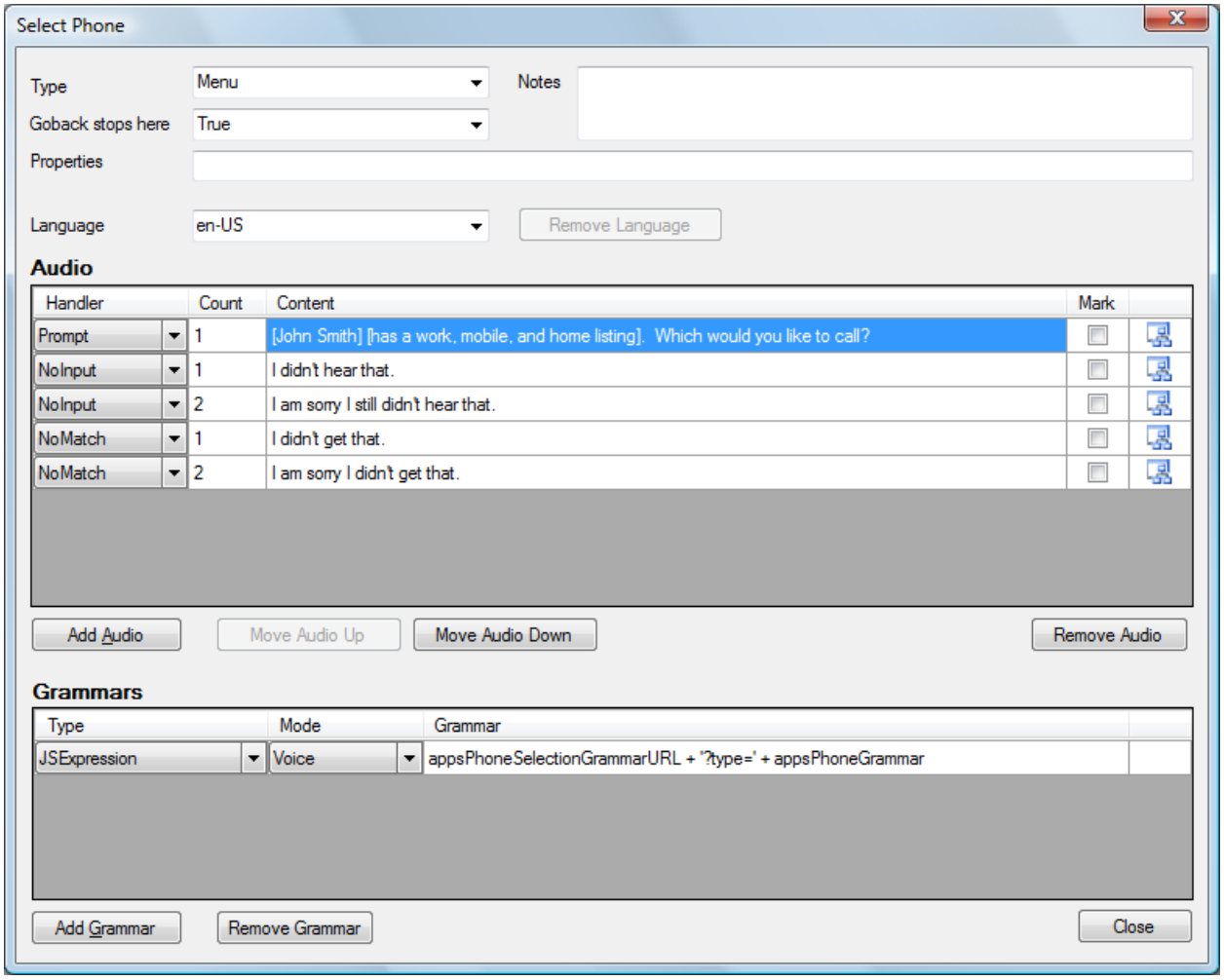
Audio Columns

- **Content** – See Audio Content Format.
- **Icon –** Click to load the Audio Builder (advanced audio configuration tool)**.**

UserInput

Icon	Light blue rectangle
Occurrences (min,max)	0,Many
Inbound Transitions (min,max)	1,Many
Outbound Transitions (min,max)	1,Many
Double-click	Opens a User Input dialog

The user input state is designed to support multi-modal and multi-lingual input and output. However, in the current release the only modality exposed is audio. Also, to date testing has focused exclusively on English but will be expanded as soon as possible.



Properties

- **Type** – enumeration
  - **Menu** (default) – Standard user input.
  - **ConfidenceBasedConfirmation** – Adds infrastructure to allow for confirmation of user input.

- **Goback stops here –** enumeration
    - **True** (default) – The state is the target of a goback command (the state will have a green border).
    - **False –** The state is not the target of a goback command (the state will have the standard black border).
    - **End –** The state is the target of a goback command and the user can goback no further (the state will have a red border).
- **Properties –** A semi-colon delimited list of name=value pairs (*e.g.,* 'bargein=true; timeout=7s').

Audio Columns

- **Handler –** enumeration
    - **Prompt –** Plays when the input state is entered.
    - **NoInput –** Plays when the user does not say anything at the input prompt.
    - **NoMatch** – Plays when the user's input is not recognized as part of the active grammar(s).
    - **Help –** Plays when the user says 'help'.
- **Count** – The invocation of the handler when the audio will play. If multiple pieces of audio have the same handler and count, they will be grouped together and will play in the order they appear in the list.
- **Content –** See Audio Content Format.
- **Mark –** When checked; a bookmark is set allowing you to determine when the user barged in on the prompt. The bookmark name is the name of the handler with the count value appended e.g., **prompt1**. If there is more than one mark per handler and count the subsequent names will be numbered e.g., **prompt1**, **prompt1_1**, **prompt1_2**
- **Icon –** Click to load the Audio Builder (advanced audio configuration tool)**.**

Grammar Columns

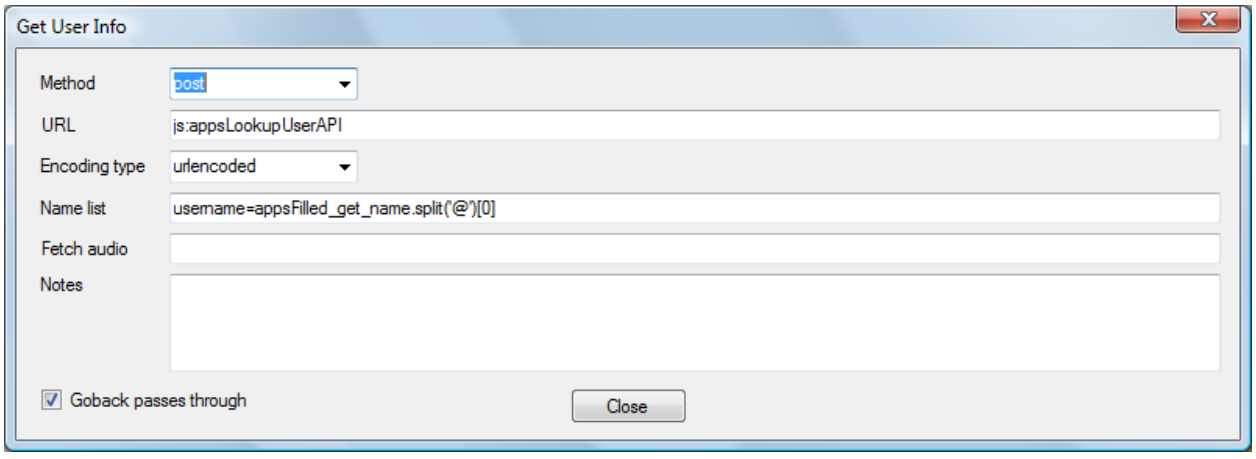
- **Type –** enumeration
    - **Custom** (default) **–** The **Grammar** column should contain the name of a GRXML file in the Grammar directory (*e.g.*, 'firstnames.grxml').
    - **AlphaNumericRegex –** The **Grammar** column should contain an alpha-numeric regular expression (*e.g.,* '\d{3,5}').
    - **RangePositiveIntCardinal –** The **Grammar** column should contain Number1 hyphen Number2 with no spaces and in ascending order (*e.g.,* '1-20').
        - For grammars like 'one, two, three'.
    - **RangePositiveIntOrdinal –** The **Grammar** column should contain Number1 hyphen Number2 with no spaces and in ascending order (*e.g.,* '1-20').
        - For grammars like 'first, second, third'.
    - **SetPositiveIntCardinal –** The **Grammar** column should contain a comma delimited list of values with no spaces (*e.g.,* '1,3,5,7,9').
        - For grammars like 'one, three,five…'.

    - **SetPositiveIntOrdinal –** The **Grammar** column should contain a comma delimited list of values with no spaces (*e.g.,* '1,3,5,7,9').
        - For grammars like 'first, third, fifth…'.
    - **ItemList –** The **Grammar** column should contain the name of a text file in the Grammar directory (*e.g.*, 'airports.txt').
    - **ItemListBySpelling –** Same as ItemList but the grammar would require the user to spell the item (*e.g.*, S-A-N J-O-S-E).
- **Mode –** enumeration
    - **Voice –** voice grammar
    - **DTMF –** DTMF grammar
- **Grammar –** See the **Grammar Type** property above for the correct syntax for each type.
- **Icon –** Click to open a simple editor window for the following grammar types:
    - **Custom –** Opens the GRXML file specified in the grammar column. If the file has to be created it will be populated with a simple yes/no grammar.
    - **ItemList, ItemListBySpelling –** Opens the text file list. If the file has to be created it will contain a short sample list (one item per line). Each line can contain a tab delimited semantic value if necessary (*e.g.*, San Jose<tab>SJC<newline>).

DataFetch

Icon	Blue parallelogram 
Occurrences (min,max)	0,Many
Inbound Transitions (min,max)	1,Many
Outbound Transitions (min,max)	1,Many (if there is more than one a default must be specified)
Double-click	Opens the Data Fetch dialog

The data fetch state provides programmatic access to data in XML format from an HTTP server.



Properties

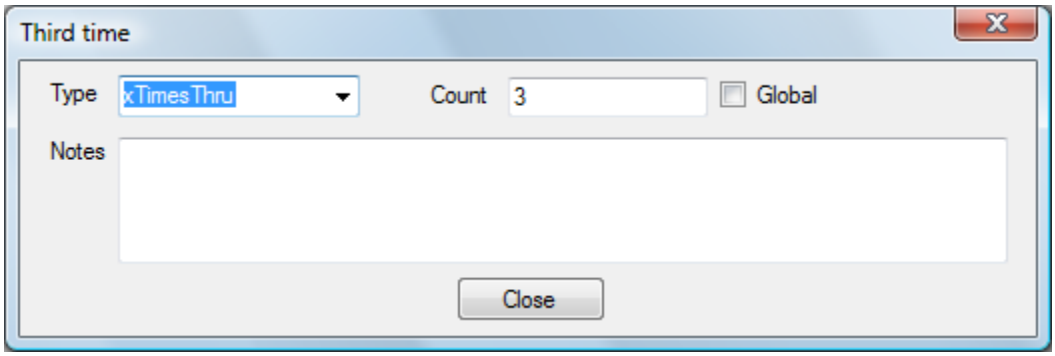
- **Method** – enumeration – HTTP Request Type
    - **get** (default)
    - **post**

- **URL** – URL used to fetch the application data.
- **Encoding Type** – enumeration – HTTP data encoding type
  - **urlencoded** (default)
  - **multipart**
- **Name List** – Parameters to pass to the URL (semi-colon delimited list of name=JavaScript expression pairs; *e.g.*, 'ani=session.telephone.ani').
- **Fetch Audio** – URL of the audio clip to play while the interpreter fetches the resource.
- **Goback passes through** – Boolean
  - **Unchecked** (default) – a goback command cannot traverse beyond this point in the call flow (the state will have a red border).
  - **Checked** – a goback command can traverse further back in the call flow (the state will have a standard black border).

The returned DOM is parsed into a JavaScript object and stored in an application root variable named **appoAPI_*EncodedName*** .

Decision

Icon	Light yellow diamond 
Occurrences (min,max)	0,Many
Inbound Transitions (min,max)	1,Many
Outbound Transitions (min,max)	2, 2 (Many for Custom) in either case a default else condition must be specified
Double-click	Opens the Decision Dialog



Properties

- **Type** – enumeration
  - **Custom** (default) – A JavaScript stub function per transition will be created and a developer will need to enter the decision logic.
  - **xTimesThru –** Tests if the application has been through this path x times (where x is specified in the **Count** property). The test can be local (module scope) or global (application scope) and is specified in the **Global** property.

- **LowConfidence –** Tests if the last recognition was below the **Confidence Threshold** property.
- **HighConfidence** – Tests if the last recognition was above the **Confidence Threshold** property.
- **UsedDTMF –** Tests if the last recognition used DTMF input.
- **UsedVoice –** Tests if the last recognition used voice input.

- **Confidence Threshold –** Used only for **Low/High Confidence** types (defaults to 0.60).
- **Count –** Used only for the **xTimesThru** type (defaults to 3).

Global – Used only for the xTimesThru type (defaults to False).Module

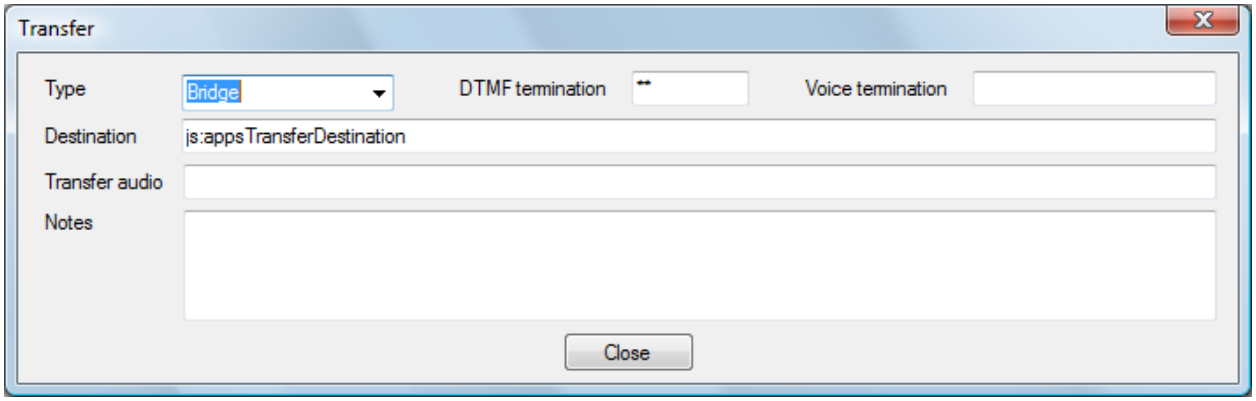
Icon	Light gray rectangle
Occurrences (min,max)	0,Many
Inbound Transitions (min,max)	1,Many
Outbound Transitions (min,max)	0,1 (an outbound transition is only allowed if the module returns)
Double-click	Opens/Creates the referenced module
Right-click	Since double-click is used for navigation you will need to use the right-click menu to access the state properties

The module state is associated with an actual module using the **Encoded Name** property. The encoded name is treated as the filename of the target module. If the module state is double-clicked the target module is created/opened. If multiple module states target the same module they should have the same **Name** property.

Transfer

Icon	White rectangle with dashed border
Occurrences (min,max)	0,Many
Inbound Transitions (min,max)	1,Many
Outbound Transitions (min,max)	0,1 (an outbound transition is only allowed on bridge transfers)

The transfer state transfers the caller to the specified destination.



Properties

- **Type** – enumeration – Determines whether or not the caller's session with the VoiceXML interpreter resumes after the call initiated by the transfer ends. Unless Tellme Networks has explicitly provisioned your voice application to support blind transfers, the transfer will fail.
    - **Blind**
    - **Bridge** (default)**–** resumes after the transfer
- **Destination –** The destination must be a SIP URL, Telephone Number (with country code), or JavaScript expression. The destination must be prefixed with **sip:**, **tel:+,** or **js:** respectively**.**
- **Transfer Audio** – The URL of an audio file played to the caller during a bridged transfer. If the callee picks up, the interpreter terminates playback of the recorded audio immediately. If the end of the audio file is reached and the callee has not yet picked up, the interpreter plays the audio tones from the far end of the call (ringing, busy).
- **DTMF Termination** – DTMF termination string for bridge transfers (default '**'**).
- **Voice Termination** – Voice termination utterance. Caution: the utterance is active for the duration of the bridged call.

The bridge transfer results are stored in an application root variable **`appoBridgeTransfer`**.

SubDialog

Icon	Light gray rounded rectangle 
Occurrences (min,max)	0,Many
Inbound Transitions (min,max)	1,Many
Outbound Transitions (min,max)	1,1

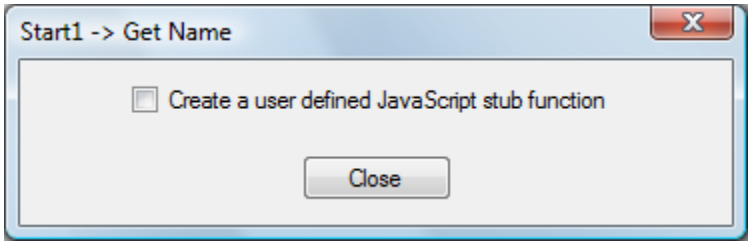
The subdialog state allows the application to call a self-contained VoiceXML dialog; values can be passed into and returned from the called VoiceXML dialog.

Properties

The properties are identical to the Data Fetch state.

The returned object is stored in an application root variable **appoSub*_EncodedName*** .

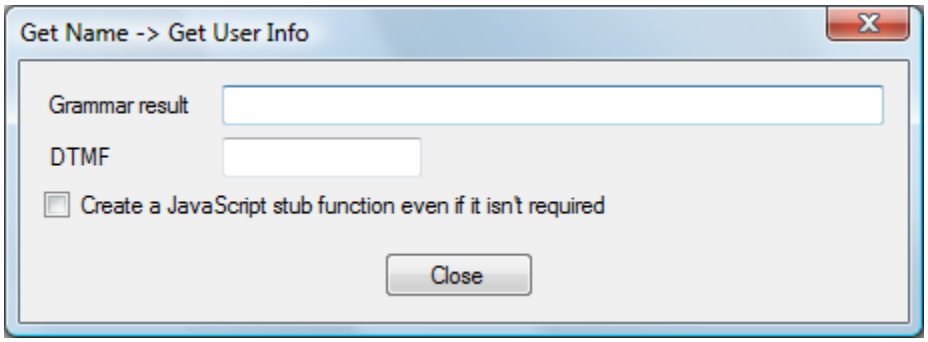
Transition



Properties

- **Create a user defined JavaScript function** – Boolean
  - **Unchecked** (default) – no JavaScript stub function is created.
  - **Checked** – Creates a JavaScript stub function for the developer to insert custom code. This should RARELY be used on this type of transition but it was included to provide additional flexibility in a design.

RecoTransition



The RecoTransition is the outbound transition for all UserInput states.

Properties

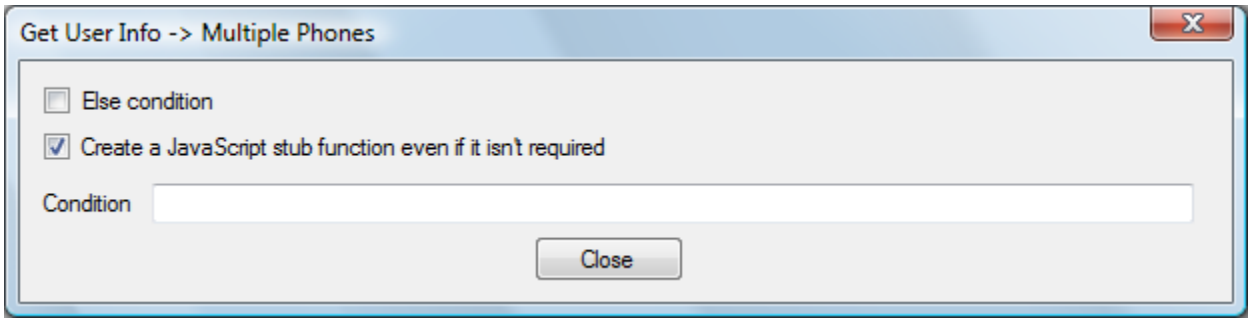
- **Grammar Result –** Grammar return value that would trigger the transition.
- **DTMF –** DTMF input that would trigger this transition (the string can contain multiple DTMF characters; any one of them will trigger the transition) *e.g.,* '1*'. In the alpha release the generators do not use this value. For now you would still have to create your own DTMF grammars and return the semantic specified in the Grammar result property.
- **Create a JavaScript stub function even if it isn't required** – Boolean
  - **Unchecked** (default) – a JavaScript stub function is not created if it isn't required.
  - **Checked** – a JavaScript stub function is created even if it isn't required.

Properties exposed only through the Properties window

- **Name** – Short sample voice input that would trigger the transition; *e.g.,* 'Retirement' (optional).
- **Label** – (read only) Label used on the transition line. It is a combination of the **Name** and **DTMF** property and cannot be set directly**.**

- **Label Position** – enumeration **–** This was added to compensate for the label overwrite problem in the DSL. Currently only the **hidden** value is functional.
    - **SourceTop**
    - **SourceBottom**
    - **TargetTop**
    - **TargetBottom**
    - **Hidden –** disable the label

DataTransition



The DataTransition is the outbound transition for all **Decision** and **DataFetch** states.

Properties

- **Else condition (default) –** The transition followed if no other transitions are triggered (no condition is necessary since it will be the else case). The default transition is colored Green.
- **Condition –** A more detailed textual description of the condition; *e.g., '*Daytime is between 6am and 6pm'. The condition is used as documentation on the generated JavaScript function.
- **Create a JavaScript stub function even if it isn't required** – Boolean
    - **Unchecked** (default) – a JavaScript stub function is not created if it isn't required.
    - **Checked** – a JavaScript stub function is created even if it isn't required.

Properties exposed only through the properties window

- **Name** – Short conditional title displayed on the transition (optional); *e.g.,* 'Daytime'.
- **Label Position –** enumeration **–** This was added to compensate for the label overwrite problem in the DSL.
    - **SourceTop**
    - **SourceBottom**
    - **TargetTop**
    - **TargetBottom**
    - **Hidden –** disable the label

Embedded Tools (Ports)

A port is a small colored square associated with a state. The port represents functionality that is coupled to a state but is not actually part of the state.

Error Handler

Icon	Red Square
Occurrences (min,max)	1,1 (An error handler is attached to UserInput, DataFetch, Subdialog, and Transfer states)
Inbound Transitions (min,max)	0,0
Outbound Transitions (min,max)	0,1 (An outbound transition is only allowed if the error handler module returns)
Double-click	Opens/Creates the referenced module
Right-click	You will need to use the right-click menu to access the error handler properties

The error handler port works much like a Module state. The error handler is associated with a module using the **Encoded Name** property. The encoded name is treated as the filename of the target module. If the error handler port is double-clicked, the target module is created/opened. The error handler is part of the state it is attached to; attempting to delete it will delete the entire state.

Unlike the module state, the error handler can be unnamed. In this case an outbound transition is required and allows the error to be trapped and handled in the current module.

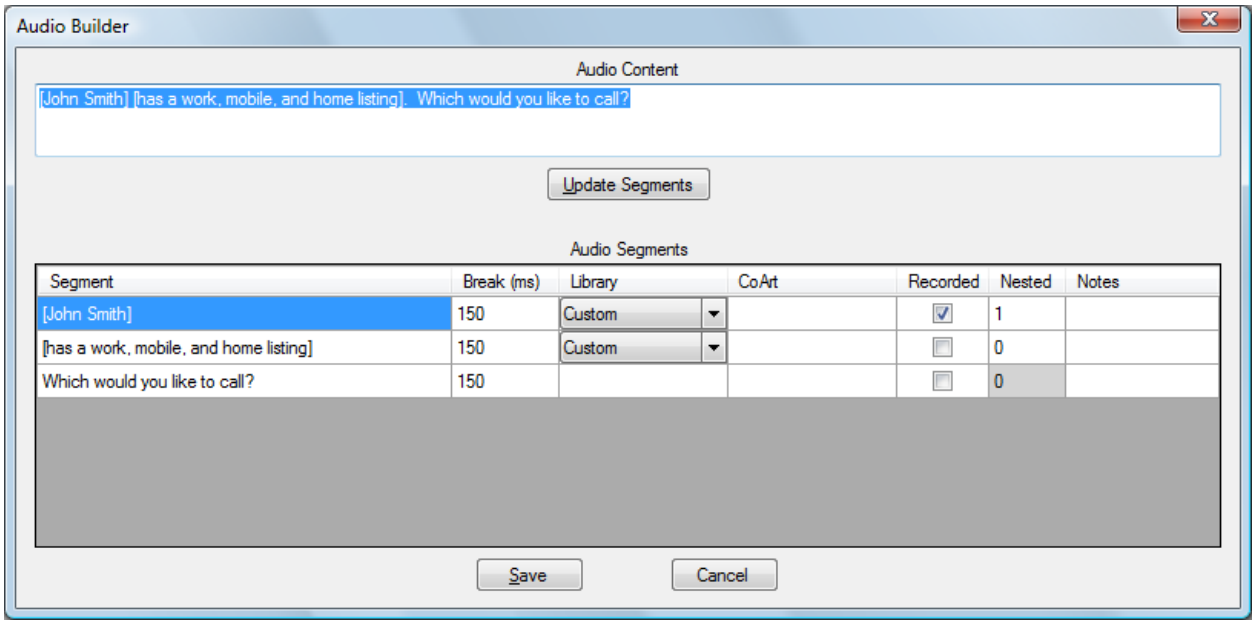
Confidence Based Confirmation

Icon	Blue Square
Occurrences (min,max)	0,1 (A CBC port is attached to a UserInput state if its **Type** property is set to **ConfidenceBasedConfirmation**)
Inbound Transitions (min,max)	0,0
Outbound Transitions (min,max)	0,0
Double-click	Opens/Creates the referenced module
Right-click	You will need to use the right-click menu to access the CBC handler properties

The confidence based confirmation (CBC) port works similarly to a module state. The main difference is that the return from the CBC module is handled by the UserInput state that the handler is attached to (either re-prompting the user or continuing on with the call flow). Deleting the CBC port will change the UserInput state **Type** back to **Menu**.

Audio Builder

The audio builder breaks up the audio content string into its individual wav file segments: static, concatenated, or co-articulated.



Audio Content Format

The audio content is the textual representation of the audio file; it is used for TTS and generation of the recording scripts. The audio content value has a special format supporting static, concatenated, and co-articulated audio.

Sample 1: Square brackets are used for concatenated variable audio:
Hi [Joe], welcome to the [Demo] application!

Sample 2: Curly braces are used for co-articulated audio:
Hi [Joe], welcome to the {Demo} application!

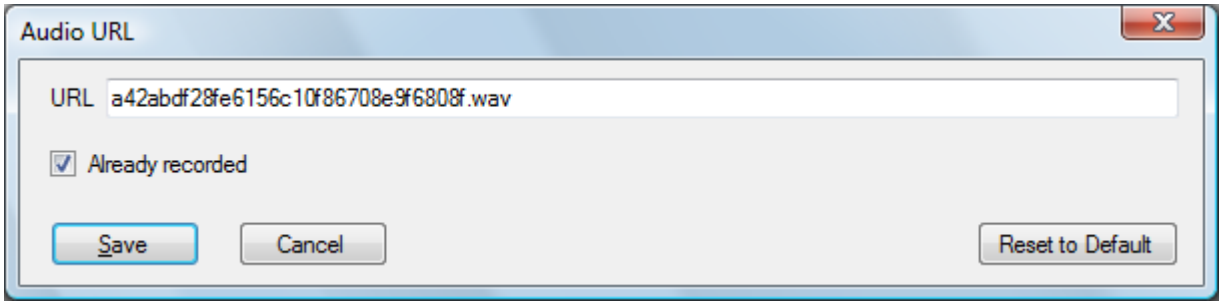
If there is no concatenation or co-articulation the audio file is treated as a single static audio file. In the first sample two JavaScript function stubs would be generated and a developer would have to add the logic to insert the correct user name and application name. In the second sample three JavaScript function stubs would be generated and a developer would have to add the logic to insert the correct user name, the correct phoneme version of "welcome to the" audio segment (based on the application name) and the application name.

Tips and Tricks

The JavaScript function names are a hash of the audio segment. If you provide the same example in all of your audio content, *e.g.*, [John Smith], only one playback function will be generated and the entire application can leverage it. Conversely, if you require slightly different playback, use a different example [John Q. Smith]. In this case a second JavaScript function will be generated allowing you to add the middle initial to the audio output array.

Audio Segments columns

- **Segment** (Read only) text segment of the audio content that will map to an individual wav file. For concatenated audio you can double-click on the segment and it will open/create a bulk audio file allowing you to define the list of variable audio prompts. Double-clicking on the [John Smith] segment shown above will create **BulkAudio\en-US\john smith.txt.** This file should contain the list of all names that can be played in the concatenation (one name per line). If the audio has already been recorded you should check the *Recorded* box and not worry about providing the bulk audio list.
- **Break** – The amount of time, in milliseconds, to pause before playing the next piece of audio.
- **Library** – If the segment is a variable piece of audio you can select a Tellme prerecorded audio library to use, or leave the default of *Custom* and supply your own library. The alpha release will not generate any code for the Tellme libraries, but the functionality will be added in a future release.
- **CoArt** – This is active for the static audio segment immediately preceding the co-articulated variable audio segment. You can use a predefined Tellme JavaScript library to select the correct phoneme or provide your own by selecting *Custom*. The alpha release will not generate any code for the Tellme libraries, but the functionality will be added in a future release.
- **Recorded** – Clicking the check box will bring up the Audio URL dialog shown below.



  - **URL** – The URL of the audio file. The default value will be shown but if the audio already exists a new URL can be entered in its place.
  - **Already Recorded** – Check this box if the audio has already been recorded; it will prevent it from appearing in the recording script.
- **Nested** – The nested value can be used on any variable audio segment. The number of items in the nested audio array is determined by the nested value plus the number of words in the example audio. The nesting level of one on [John Smith] will require the developer to provide three audio objects consisting of a wav file name, the TTS, and the pause time (FYI: only the TTS of the innermost nest is used). If the full name fails to play at the top level, the first nesting will attempt to play the first and last name as two separate files.
    1. Johnsmith.wav, John Smith, 100
        1. John.wav, John, 100
        2. Smith.wav, Smith, 100

Tips and Tricks

Since looping over an array is not supported within nested audio elements the final array has to be a fixed length. This means your audio content example should contain the longest playback string; *e.g.*, [John Q. Smith]. This will require the developer to provide four pieces of audio in the array. For the shorter playback of John Smith the fourth item would be "null, null, null". The extra 'slot' will allow you to add the middle initial when required.

- **Notes** – Documentation or recording instructions for the voice actor.

Validation

Full validation is performed every time the model is saved. However, you can also invoke it from the context menu by right-clicking on the canvas or a shape. Select **Validate** to run a check on the selected item or **Validate All** to run a check on the entire module. If there are validation errors you are still allowed to save the module but the application generation will not function properly until the errors are corrected.

Warning Messages

- `'StateX' <language> is missing audio.`
  - `Fix: Add audio to the the state/language specified.`
- `'StateX' <language> is missing a grammar.`
  - `Fix: add at least one grammar to the specified user input state/language.`

Error Messages

Double-clicking on an error message will select the state(s) containing the error.

- `State 'ID' has no name.`
  - `Fix: Update the name property on the state.`
- `State name: 'StateX' is used more than once.`
  - `Fix: Two states have the same name; rename one of them.`
- `'StateX' confirmation module must have a return and reprompt exit state.`
  - `Fix: Make sure the confidence based confirmation module for this UserInput state has two Exit states: one with the Type property set to Return, and the other with the property set to Reprompt.`
- `'StateX' must have a URL.`
  - `Fix: Add a value to the DataFetch state URL property.`
- `'StateX' transfer state must have a destination.`
  - `Fix: Add a value to the Transfer state Destination property.`
- `This module must have at least one Presentation or UserInput state.`
  - `Fix: A module must have some kind of user input/output; add a presentation or user input state.`
- `'StateX' contains a transition with no condition.`
  - `Fix: Add a textual description of the transition to the Condition property.`

- 'StateX' must have one default transition.
  - Select one of the transtions exiting the Decision or DataFetch state and set the Default property to True.
- 'StateX' must have at least x inbound transition(s).
- 'StateX' must have at least x outbound transition(s).
  - Fix: Add a transition to properly connect the state in the diagram.
- 'StateX' cannot have more than x outbound transition(s).
  - Fix: This will apply to a Module State or Error Handler Port. It indicates the module does not have an Exit state with the type property set to Return, therfore an oubound transition is not allowed. Remove the transition or modify the module to Return.
- Reco transition 'StateX' cannot use the same DTMF as 'StateY'.
  - Fix: Correct the DTMF property on one of the transitions so it no longer uses the same DTMF value as the other.
- Presentation state 'StateX'contains an extra language/modality 'en-US/mobile'.
  - Fix: Remove the extra language/modality or add the language modality to all the other presentation and user input states.
- Presentation state 'StateX' is missing a language/modality.
  - Fix: Add the missing modality or language to the state in error.
- 'StateX' cannot have more than x inbound transition(s).
  - The model should prevent this from happening so this error message should not be seen.
- State 'StateX's error handler cannot be connected to another error handler.
  - The model should prevent this from happening so this error message should not be seen.

Application Generation

Application Generation can be performed from anywhere on the canvas by using the right-click context menu. Select **Generate Module** to regenerate the code for the currently active module. This performs a partial generation; other modules that have been previously generated will not be purged. To generate the entire application from scratch, select **Generate Application.** This will delete the **GeneratedApplication** directory and rebuild all modules (tml files) in the **Modules** directory.

Templates

The entire generation process is controlled by a series of text templates located in the **Templates** directory (in the project root). Each set of templates is in its own subdirectory. The entry point is **main.tmltt.** To add your own templates, say for custom documentation, create a subdirectory called CustomDocs containing a template named main.tmltt. The next time generation is invoked that template will be processed. The output will reside in **GenerateApplication\CustomDocs**. By default the generation process will produce one file per module; *module_name.ext*. The file extension is controlled in the main template by using the output directive.

```
<#@ output extension=".txt" #>
```

If the template returns an empty/blank string, no file will be created. This allows the template code to open and create its own files outside the normal generation process (see the Grammar generation templates which can create multiple files per UI state).

If your main.tt template needs to be executed for each language in the model, make sure it includes a language substitution variable; **@@LANGUAGE@@** (see vxml/main.tt). Using the above example the output would now reside in **GenerateApplication*<language>*\CustomDocs.**

Configuration Variables

The **app-config.xml** file is located in the project root**.** Currently it consists of four elements which are used in the VXML source code comment headers. If you create your own templates that require global variables, they should be added here.

```
<?xml version="1.0" encoding="utf-8"?>
<app-config>
  <title>Cal Shuttle</title>
  <description>Vendor Exercise</description>
  <author>Trink</author>
  <company>Tellme Networks Inc</company>

  <!-- global event handlers -->
  <goback>enabled</goback>
</app-config>
```

Application Root

The **app-root.vxml.*<language>*** file is located in the project root. Add your VXML variables, properties, and event handlers **before** the **Generated Variables/Handlers** section. During the generation process this file is modified to include any global variables needed by the generated code and then it is

copied to the **GeneratedApplication*<language>*\vxml** directory. Partial generation (Generate Module) will only add variables that don't already exist. If there are variables that are no longer used they will not be purged until a full regeneration.

```
<?xml version="1.0"?>
<vxml version="2.1">
<!--*************************************************************
 Application Level JavaScript Support
*************************************************************-->
    <script src="../../js/app-root.js" />
<!--*************************************************************
 Global Properties
*************************************************************-->
    <property name="timeout" value="5.0s" />
    <property name="sensitivity" value="0.50" />
    <property name="confidencelevel" value="0.50" />
    <!--  For DTMF Buffering   -->
    <property name="termtimeout" value="1s" />
    <property name="interdigittimeout" value="3000" />
<!--*************************************************************
 Variables
*************************************************************-->
    <!-- Full Path to the root audio -->
    <var name="appsAudioRootPath" expr="'http://audio.en-us.tellme.com/'" />
<!--*************************************************************
 Generated Variables/Handlers
 DON'T ADD OR MODIFY ANY ELEMENTS IN THIS SECTION IT WILL BE
 DELETED AND REBUILT ON FULL APPLICATION GENERATION
*************************************************************-->
<!-- begin generated -->
<!— *snip* -->
<!-- end generated -->
</vxml>
```

Generated JavaScript

The **JavaScript** directory, in the project root, is where all the custom application logic is stored. All generated stub functions will contain a **// TODO** comment. Once the developer completes a stub function, the TODO comment should be removed; this will make it easier to find new work on subsequent generations.

All function names contain hash identifiers; this is a little disconcerting in the beginning, but the functions and their calls are well documented, so it is relatively easy to understand what each function actually does. Once a JavaScript stub function is written out the contents of the function are not modified by subsequent generations. This allows the developer to modify to the function as necessary and not lose the changes on regeneration. However, the comments preceding the function are always regenerated since they come directly from the model and should only be changed in the model.

So far the biggest issue with the hash naming convention is around bulk audio; if new audio is added to the list the concatenation function is not updated. In most cases this will be a problem. The quickest way to get the new audio file names is to rename the current function, let the generator create a new

stub function, and then extract the new audio. Obviously if you don't know that new audio was added then this is a real problem; Tellme is looking into ways to address this.

app-apis.js

All API (DataFetch) JavaScript functions are written to a single file **app-apis.js;** allowing reuse of the API code between modules. By default the API function will parse the XML result into a JavaScript object in application scope; *e.g.*, **appoAPI_*EncodedStateName*.** The generated file is output to **GeneratedApplication\js\app-apis.js**.

audio-<language> .js

All audio related functions are written to a single file per language **audio-*<language>* .js,** allowing common audio playback sequences to be shared between modules. The generated file is output to **GeneratedApplication\<language>\audiojs\audio.js**.

<modulename>.js

A ***<modulename>*.js** file will be created if there are user defined conditions, such as a custom decision state, that must be defined by a developer. The generated file is output to **GeneratedApplication\js*<modulename>*.js**.

Grammars

The **Grammars** directory, in the project root, is where all the custom grammars and item lists are stored. A language directory (*e.g.*, en-US) is created for each active language in the application. Every module with a Custom, ItemList, or ItemListBySpelling grammar, for that language, will have a module directory there. The module directory should contain the GRXML and txt files specified in the User Input state grammar configuration.

Error Messages

If the modules passed validation there should only be the potential for a single type of error message.

- `Running transformation: System.IO.FileNotFoundException: Could not find file 'C:\Visual Studio 2008\Projects\Tml\Debugging\Grammars\other\main_menu.grxml'.`
    - `Fix: The use of a custom grammar or item list was specified but the grammar/list was not provided. Create the file specified in the error message and regenerate.`

Publishing to Studio

Every studio user has a dedicated private workspace on https://publish.studio.tellme.com/<StudioID>. The workspace is a WebDAV enable folder that can be accessed directly from Voice Studio or even mounted as a network drive on your file system. The folder is protected by HTTP Basic Authentication and requires your Studio ID and Studio password to gain access. The storage space is limited to ten megabytes.

To publish an application from Voice Studio, right-click anywhere on the canvas and select **Publish Application** from the menu. The system will prompt you for your Studio ID and password. Publishing an application usually takes just a few seconds (for larger applications or a slow connection it could be much longer). In the alpha release there is no progress indication, but a dialog appears when the operation is complete. The **GeneratedApplication** directory is copied to the **VoiceStudio** folder in the root of your publishing workspace. Warning: **DO NOT** store or modify any files in the VoiceStudio tree; it is deleted and recreated every time publish is invoked.

To access you application from Studio you will need to set your scratchpad URL here: https://studio.tellme.com/mystudio/mystudio.pl?newmode=2&submit=yes (In the future there will be an option to do this from Voice Studio). Set the scratchpad URL to point at your application entry point e.g., https://publish.studio.tellme.com/<StudioID>/VoiceStudio/en-US/vxml/index.vxml and click the update icon. The Tellme Studio VXML validation will produce errors on the GRXML grammars; these can be ignored since the online validation hasn't been updated to support GRXML or you can disable the grammar checker by un-checking the 'Automatically run grammar checker' in your Studio preferences. Once your application is published you can access it at 888-534-2220 using your Studio ID and pin.

Debugging and Test

Metrics

TBD

Cyclomatic Complexity

TBD

Call Flow Coverage

TBD

Call Logs

TBD

Manual Testing

TBD

Automated Testing

TBD

Audio Creation/Services

TBD

Provisioning a Number

TBD

Publishing to Production

TBD

Reporting

TBD

Transcription and Utterance Capture

TBD

Tuning

TBD