Form

The `Form` voice element is used to capture any input from the caller, based on application designer-specified grammars. The valid caller inputs can be specified either directly in the voice element settings (which will create an inline grammar) or with external grammar files. Information returned by the grammar are saved in element data that then can be analyzed by developer-defined components. A Form voice element can be configured to listen for voice input only, DTMF input only, or both voice and DTMF input. In short, the Form element is the most flexible of included Unified CVP elements as it allows almost any custom information to be captured without requiring a separate voice element. If a Unified CVP or third-party voice element does not capture the information desired, one can always use a Form element before embarking on constructing a custom voice element.

The Form element provides support for custom control over the VoiceXML code generation. For example, the developer can decide what name to use for the VoiceXML field, whether or not to include a field-level slot attribute and how to name the slot attribute. The element also supports separate options for activating help prompts and the ability to set modality for Form.

Multiple DTMF and speech external grammars can be referenced within a single Form element, and the application designer has the ability to specify grammar weights for speech grammars and set MIME types for both speech and DTMF grammars. Additionally, the Form element can be used to capture multiple slots, and the developer can specify for which slot(s) they want the recognition values stored as element data. N-best processing can be enabled, and standard n-best results are stored in element data and the activity log.

Settings

Name (Label)	Type	Req'd	Single Setting Value	Sub. Allow	Default	Notes
`inputmode`	string enum	Yes	`true`	`false`	both	The type of entry allowed for input. Possible values are: `voice` | `dtmf` | `both`.

(Input Mode)						The adapter type Cisco DTMF is not compatible with input modes `voice` and `both`.
`noinput_timeout` (Noinput Timeout)	string	Yes	`true`	`true`	`5s`	The maximum time allowed for silence or no keypress before a noinput event is thrown. Possible values are standard time designations including both a non-negative number and a time unit, for example, 3s (for seconds) or 3000ms (for milliseconds). Default = 5s.
`form_max_noinput_count` (Form Max NoInput)	int ≥ 0	Yes	`true`	`true`	`3`	0 = infinite noinputs allowed.
`form_max_nomatch_count` (Form Max NoMatch)	int ≥ 0	Yes	`true`	`true`	`3`	0 = infinite nomatches allowed.
`confidence_level` (Form Confidence Level)	decimal (0.0 – 1.0)	Yes	`true`	`true`	`0.40`	The confidence level threshold to use for data capture.
`voice_grammar` (Voice Grammar)	string	*No	`false`	`true`	*None*	Defines an external voice grammar for Form, in a string format delimited with semi-colons specifying these values in the following order: **1.** The language context in which the current grammar should be used (optional). If omitted the language will be the same as the page-scoped language. **2.** The language code to assign to the `xml:lang` attribute of the parent `<grammar>` tag (optional). If omitted the attribute will not have an xml:lang attribute and the standard scoping rules apply. **3.** The grammar weight (optional) **4.** The grammar type (optional) **5.** URL of the grammar file (required) **6.** builtin: speech/transcribe The type can be left blank to use the adapter default or set to `null` to not include a type at all. If one of the optional parameters is defined, **four** semi-colons must be used, even if the other parameters are not used. For example: • en-US;en-US;0.6;application/srgs +xml;http://IP:PORT/ mygrammar.grxml • fr-FR;en-US;;application/srgs +xml;http://IP:PORT/ mygrammar.grxml

						• ;;0.6;;http://IP:PORT/mygrammar.grxml • ;fr-FR;0.6;null;http://IP:PORT/mygrammar.grxml • http://IP:PORT/mygrammar.grxml This setting is repeatable so multiple external grammar sources may be specified. None of the four settings - `voice_grammar`, `dtmf_grammar`, `voice_keyword` and `dtmf_keypress` - is required, but at least one must be specified since a form cannot be completed without a grammar.
`dtmf_grammar` (DTMF Grammar)	URI	*No	`false`	`true`	*None*	Defines an external DTMF grammar for Form, in a string format delimited with a semi-colon specifying four values in the following order: **1.** The language context in which the current grammar should be used (optional). If omitted the language will be the same as the page-scoped language. **2.** The language code to assign to the `xml:lang` attribute of the parent `<grammar>` tag (optional) . If omitted the attribute will not have an xml:lang attribute and the standard scoping rules apply. **3.** The grammar type (optional) **4.** URL of the grammar file (required) The type can be left blank to use the adapter default or set to *null* to not include a type at all. If one of the optional parameters is defined, **three** semi-colons must be used, even if the other parameters are not used. For example: • en-US;en-US;application/srgs +xml;http://IP:PORT/ mygrammar.grxml • ;fr-FR;null;http://IP:PORT/mygrammar.grxml • en-US;;;http://IP:PORT/mygrammar.grxml • http://IP:PORT/mygrammar.grxml This setting is repeatable so multiple external grammar sources may be specified. None of the four settings - `voice_grammar`, `dtmf_grammar`, `voice_keyword` and `dtmf_keypress` - is required, but at least one must be specified since a form cannot be completed without a grammar.
`voice_keyword`	string	*No	`false`	`true`	*None*	Defines the inline voice grammar for Form, with each configuration of this repeatable setting

(Voice Keyword)						specifying one option for the grammar. The valid format is a string separated with a semi-colon specifying four values in the following order: **1.** The language context in which the current input should be included in the inline grammar (optional). If omitted the language will be the same as the page-scoped language. **2.** The language code to assign to the `xml:lang` attribute of the `<item>` tag inside the inline grammar (optional) . If omitted the attribute will not have an `xml:lang` attribute and the standard scoping rules apply. **3.** The weight of the grammar item (optional) **4.** The grammar item (required) **Note** The grammar item may either contain the input itself followed by an optional return value, or just the input. If one of the optional parameters is defined, **three** semi-colons must be used, even if the other parameters are not used. Sample configurations values are: • en-US;en-US;0.6;news report [news] • ;fr-FR;0.6;news report • news report [news] • news report None of the four settings - `voice_grammar`, `dtmf_grammar`, `voice_keyword` and `dtmf_keypress` - is required, but at least one must be specified since a form cannot be completed without at least one grammar.
`dtmf_keypress` (DTMF Keypress)	character (0-9, #, *)	*No	`false`	`true`	*None*	Defines the inline DTMF grammar for Form, with each configuration of this repeatable setting specifying one option for the grammar. The valid format is a string separated with a semi-colon specifying three values in the following order: **1.** The language context in which the current input should be included in the inline grammar (optional). If omitted the language will be the same as the page-scoped language. **2.** The language code to assign to the `xml:lang` attribute of the `<item>` tag inside the inline grammar (optional) . If omitted the attribute

						will not have an `xml:lang` attribute and the standard scoping rules apply. **3.** A character (0-9, #, *) representing the keypress, followed by an optional return value. **Note** The grammar item may either contain the input itself followed by an optional return value, or just the input. If one of the optional parameters is defined, **two** semi-colons must be used, even if the other parameters are not used. Sample configurations values are: • en-US;en-US;1 [news] • ;fr-FR;1 • 1 [news] • 1 None of the four settings - `voice_grammar`, `dtmf_grammar`, `voice_keyword` and `dtmf_keypress` - is required, but at least one must be specified since a form cannot be completed without at least one grammar.
`help_voice_keyword` (Help Voice Keyword)	string	No	`false`	`true`	*None*	Specifies a custom inline voice grammar to activate the help audio group. Each value of this repeatable setting adds another valid utterance. The format is a string specifying just the utterance (for example, *news report*). If this setting is configured, a custom inline voice grammar will be generated, replacing the default help grammar used by a browser, and the custom grammar will be active only within the current Form element.
`help_dtmf_keypress` (Help DTMF Keypress)	character (0-9, #, *)	No	`false`	`true`	*None*	Specifies a custom inline DTMF grammar to activate the help audio group. Each value of this repeatable setting adds another valid DTMF keypress. The format is a character (0-9, #, *) representing just the keypress. If this setting is configured, a custom inline DTMF grammar will be generated, and it will be active only within the current Form element.
`modal` (Disable Hotlinks)	boolean	Yes	`true`	`true`	`false`	Whether or not to temporarily disable all hotlink grammars (global or local) and universal grammars. If set to true, only the current Form

						element grammars will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
`field_name` (Field Name)	string	Yes	`true`	`true`	foundation_fld	*foundation_fld* - The value to assign to the VXML field name attribute.
`slot_name` (Field Slot)	string	No	`true`	`true`	*None*	The name to assign to the VXML field slot attribute. If left unspecified, the field will not include a slot attribute.
`slot_element_data` (Slot Element Data)	string	No	`false`	`true`	*None*	Specifies for which grammar slot the return value should be stored as element data. This is a repeatable setting so multiple slot names can be specified. See notes below for further details.
`maxnbest` (Maxnbest)	int ≥ 1	Yes	`true`	`true`	1	The maximum number of speech recognition results that can be generated per voice input.
`secure_logging` (Secure Logging)	boolean	Yes	`true`	`true`	`false`	If set to true, user DTMF input for the element is considered secure and the attributes utterance, interpretation, value, nbestUtteranceX and nbestInterpretationX are masked in VXML server logs. The format used to render secure element attributes is to add a *_secureLogging* suffix. For example `nbestUtterance1_secureLogging,*****`.
`recordutterance`	boolean	Yes	`true`	`true`	`false`	When the property is set to *true* the wave-form-uri of the recorded audio is submitted to VXML server.
`dtmf_overlay` (DTMF Overlay)	Boolean	Yes	`true`	`true`	`false`	Setting this property to `true` will enable the generation of random DTMF digits tone at random duration while DTMF recognition is in progress. **Note** `dtmf_overlay`supports only the following VoiceXML Gateways, and one of these options must be selected before creating or deploying the Call Studio application. • Cisco DTMF • VoiceXML 2.1 Cisco DTMF
`dtmf_overlay_interval` (DTMF Overlay Interval)	String	Yes	`true`	`true`	`1000ms`	Time Interval (in ms) between the generation of two DTMF tones. The interval is a random number that is +/-25% of the duration that is mentioned. For example, if the duration mentioned is `1000ms`, the interval will be between between 750ms and 1250ms.

						Note The duration mentioned must be between 500ms (minimum) and 2000ms (maximum).

- VXML 2.0-compliant browsers typically require top-level slot names in the grammar (inline or external) to match the field-level slot attribute (if it exists) or the field name attribute, in order for the field name variable (and hence the *value* element data) to be defined. For inline grammars, the Form element automatically generates the grammar slot name to match the slot attribute (if available) or the field name. For custom grammars that are referenced from an external source, the application designer needs to set `Field Name` and `Field Slot` properly based on the slot name returned by the grammar.
- If a grammar returns different slots for different inputs or multiple slots per utterance, there are two ways to configure the Form element to store this data:
  - Leave the `slot_element_data` setting empty. The Form element will create element data named "nbestInterpretationX" (where X is from 1 to the length of the n-best list) that contains a string that uses delimiters "+" and ":" to separate the multiple slot names from their values. For example: "+Slot1:value1+Slot2:value2...". A developer would then need to parse this string in a subsequent element to obtain the different slot name and value pairs.
  - Configure the `slot_element_data` setting with the names for all the slots that can be returned. The Form element will create a new set of n-best element data to store the recognition results for each slot listed in that setting. The element data will be named as `<SLOT_ELEMENT_DATAX>` (where *SLOT_ELEMENT_DATA* is a string identical to the setting value and X is from 1 to the length of the n-best list). For example, if `slot_element_data` had two values *city* and *state* and there are three n-best results triggered, then six element data in the names of *city1*, *city2*, *city3*, *state1*, *state2*, and *state3* will be created to store each of the n-best values for the *city* and *state* slots. Note that if n-best processing is disabled by setting the maxnbest setting to 1, then only one interpretation result will be returned per recognition and thereby only one element data per slot (*city1* and *state1*) will be created.

Element Data

Name	Type	Notes
`value`	string	This stores the value of the VXML field name variable.
`value_confidence`	float	This stores the confidence score of the captured Form utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.

`<SLOT_ELEMENT_DATA1>`
`<SLOT_ELEMENT_DATA2>`
…
`<SLOT_ELEMENT_DATAX*>`
	string	A separate set of element data stores the interpretation values for each filled slot of captured n-best utterances. While the maximum number of `<SLOT_ELEMENT_DATAX>` values is equal to the `maxnbest` setting value, the actual number of these values available is dependent on speech recognition at runtime, where `<SLOT_ELEMENT_DATA1>` holds the slot value of the top hypothesis in the n-best list and `<SLOT_ELEMENT_DATAX>` holds the slot value of the last hypothesis.

		Note If the `slot_element_data` setting is blank, these sets of element data will not be created.
`nbestLength`	int ≥ 1	This stores the number of n-best hypotheses generated by the speech engine.

`nbestUtterance1`
`nbestUtterance2`
…
`nbestUtteranceX`
	string	This set of element data stores the captured n-best utterances. While the maximum number of `nbestUtteranceX` values is equal to the `maxnbest` setting value, the actual number of these values available is determined by speech recognition at runtime, where `nbestUtterance1` holds the utterance of the top hypothesis in the n-best list and `nbestUtteranceX` holds the utterance of the last hypothesis.

`nbestInterpretation1`
`nbestInterpretation2`
…
`nbestInterpretationX`
	string	This set of element data stores the interpretations of captured n-best utterances. While the maximum number of `nbestInterpretationX` values is equal to the `maxnbest` setting value, the actual number of these values available is determined by speech recognition at runtime, where `nbestInterpretation1` holds the interpretation of the top hypothesis in the n-best list and `nbestInterpretationX` holds the interpretation of the last hypothesis.

`nbestConfidence1`
`nbestConfidence2`
…
`nbestConfidenceX`
	float	This set of element data stores the confidence scores of captured n-best utterances. While the maximum number of `nbestConfidenceX`values is equal to the `maxnbest` setting value, the actual number of these values available is determined by speech recognition at runtime, where `nbestConfidence1` holds the confidence score of the top hypothesis in the n-best list and `nbestConfidenceX` holds the confidence score of the last hypothesis.

`nbestInputmode1`
`nbestInputmode2`
…
`nbestInputmodeX`
	string	This set of element data stores the input modes of captured n-best utterances. This stores the number of no input events that the browser returned during the collection phase of the VXML field name variable.

`collect_noinput_count`	int ≥ 0	This stores the number of no input events that the browser returned during the collection phase of the VXML field name variable.
`collect_nomatch_count`	int ≥ 0	This stores the number of no match events that the browser returned during the collection phase of the VXML field name variable.

* `SLOT_ELEMENT_DATA` is a string identical to the configuration value of the `slot_element_data` setting, and X is from 1 to the length of the n-best list. If more than one such value is configured, then multiple sets of element data using the same naming convention will be created.

Exit States

Name	Notes

`max_nomatch`	The maximum number of nomatch events has occurred. If the nomatch max count is 0, this exit state will never occur.
`max_noinput`	The maximum number of noinput events has occurred. If the noinput max count is 0, this exit state will never occur.
`done`	The caller input matched the grammar correctly.

Audio Groups

Form Data Capture

Name (Label)	Req'd	Max1	Notes
`initial_audio_group` (Initial)	Yes	Yes	Played when the voice element first begins.
`nomatch_audio_group` (NoMatch)	No	No	Played when a nomatch event occurs.
`noinput_audio_group` (NoInput)	No	No	Played when a noinput event occurs.
`help_audio_group` (Help)	No	No	Played when the caller asks for help. If not specified, help is treated as a nomatch event by default.

End

Name (Label)	Req'd	Max 1	Notes
`done_audio_group` (Done)	No	Yes	Played when the form data capture is completed, and the voice element exits with the done exit state.

Folder and Class Information

Studio Element Folder Name	Class Name
`Form`	`com.audium.server.voiceElement.form.MFoundationForm`

Events

Name (Label)	Notes
`Event Type`	You can select **Java Exception**, **VXML Event**, or **Hotlink** as event handler for this element.