Element Specifications for Cisco Unified CVP VXML Server and Cisco Unified Call Studio

Release 7.0(1)

February 2008

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Element Specificaitons for Cisco Unified CVP VXML Server and Cisco Unified Call Studio

Table of Contents

Preface

Purpose

This document provides specifications for the elements included with VXML Server.

Audience

This document is intended for voice application developers using Call Studio and VXML Server.

Organization

Chapter 1, "Introduction"
Introduces elements.
Chapter 2, "Application_Modifier"
Describes the Application_Modifier element.
Chapter 3, "Audio"
Describes the Audio element.
Chapter 4, "Counter"
Describes the Counter element.
Chapter 5, "Currency"
Describes the Currency element.
Chapter 6, "Currency_With_Confirm"
Describes the Currency_With_Confirm element.
Chapter 7, “CVP Subdialog Return”
Describes the CVP Subdialog Return element.
Chapter 8, “CVP Subdialog Start”
Describes the CVP Subdialog Start element.
Chapter 9, "Database"
Describes the Database element.
Chapter 10, "Date"

Describes the Date element.

Chapter 11, "Date_With_Confirm"
Describes the Date_With_Confirm element.

Chapter 12, "Digits"
Describes the Digits element.

Chapter 13, "Digits_With_Confirm"
Describes the Digits_With_Confirm element.

Chaper 14, "Email"
Describes the Email element.

Chapter 15, "Form"
Describes the Form element.

Chapter 16, "Form_With_Confirm"
Describes the Form_With_Confirm element.

Chapter 17, "Math"
Describes the Math element.

Chapter 18, "2_Option_Menu, 3_Option_Menu, . . . , 10_Option_Menu"
Describes the menu elements with from 2 to 10 options.

Chapter 19, "Number"
Describes the Number element.

Chapter 20, "Number_With_Confirm"
Describes the Number_With_Confirm element.

Chapter 21, "Phone"
Describes the Phone element.

Chapter 22, "Phone_With_Confirm"
Describes the Phone_With_Confirm element.

Chapter 23, "Record"
Describes the Record element.

Chapter 24, "Record_With_Confirm"
Describes the Record_With_Confirm element.

Chapter 25, "ReqICMLabel"
Describes the ReqICMLabel element.

Chapter 26, “Subdialog Invoke”
Describes the Subdialog Invoke element.

Chapter 27, “Subdialog Return”
Describes the Subdialog Return element.

Chapter 28, “Subdialog Start”

Describes the Subdialog Start element.

Chapter 29, "Time"

Describes the Time element.

Chapter 30, "Time_With_Confirm"

Describes the Time_With_Confirm element.

Chapter 31, "Transfer"

Describes the Transfer element.

Chapter 32, "Yes_No_Menu"

Describes the Yes_No_Menu element.

Obtaining Documentation, Obtaining Support, and Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Related Documentation

Note: Planning your Unified CVP solution is an important part of the process in setting up Unified CVP. Cisco recommends that you read the *Cisco Unified Customer Voice Portal Release 7.x Solution Reference Network Design (SRND)* guide *before* configuring your Unified CVP solution. With Unified CVP 7.x, the *Planning Guide for Cisco Unified Customer Voice Portal* has been incorporated into the SRND guide.

- *Cisco Security Agent Installation/Deployment for Cisco Unified Customer Voice Portal* provides installation instructions and information about Cisco Security Agent for the Unified CVP deployment. **We strongly urge you to read this document in its entirety.**

- *Cisco Unified Customer Voice Portal Release 7.x Solution Reference Network Design (SRND)* provides design considerations and guidelines for deploying contact center voice response solutions based on Cisco Unified Customer Voice Portal (Unified CVP) 7.x releases.

- *Configuration and Administration Guide for Cisco Unified Customer Voice Portal* describes how to set up, run, and administer the Cisco Unified CVP product, including associated configuration.

- *Installation and Upgrade Guide for Cisco Unified Customer Voice Portal* describes how to install Unified CVP software, perform initial configuration, and upgrade.

- *Operations Console Online Help for Cisco Unified Customer Voice Portal* describes how to use the Operations Console to configure Unified CVP solution components.

- *Port Utilization Guide for Cisco Unified Customer Voice Portal* describes the ports used in a Unified CVP deployment.

- *Programming Guide for Cisco Unified CVP VXML Server and Cisco Unified Call Studio* describes how to build components that run on the Cisco Unified CVP VXML Server.

- *Reporting Guide for Cisco Unified Customer Voice Portal* describes the Reporting Server, including how to configure and manage it, and discusses the hosted database.

- *Say It Smart Specifications for Cisco Unified CVP VXML Server and Cisco Unified Call Studio* describes in detail the functionality and configuration options for all Say It Smart plugins included with the software.

- *Troubleshooting Guide for Cisco Unified Customer Voice Portal* describes how to isolate and solve problems in the Unified CVP solution.

- *User Guide for Cisco Unified CVP VXML Server and Cisco Unified Call Studio* describes the functionality of Cisco Unified Call Studio including creating projects, using the Cisco Unified Call Studio environment, and deploying applications to the Cisco Unified CVP VXML Server.

Conventions

This manual uses the following conventions:

Convention	Description
boldface font	Boldface font is used to indicate commands, such as user entries, keys, buttons, and folder and submenu names. For example: ▪ Choose **Edit** > **Find**. ▪ Click **Finish**.
italic font	*Italic* font is used to indicate the following: ▪ To introduce a new term. Example: A *skill group* is a collection of agents who share similar skills. ▪ For emphasis. Example: *Do not* use the numerical naming convention. ▪ A syntax value that the user must replace. Example: IF (*condition, true-value, false-value*) ▪ A book title. Example: See the *Cisco CRS Installation Guide*.
`window font`	`Window font`, such as Courier, is used for the following: ▪ Text as it appears in code or that the window displays. Example: `<html><title>Cisco Systems,Inc. </title></html>` ▪ File names. Example: `tserver.properties`. ▪ Directory paths. Example: `C:\Program Files\Adobe`
< >	Angle brackets are used to indicate the following: ▪ For arguments where the context does not allow italic, such as ASCII output. ▪ A character string that the user enters but that does not appear on the window such as a password.

Chapter 1: Introduction

Every element included with Call Studio and VXML Server must be configured before it can be used. This reference file contains a detailed specification for each of the core Cisco Unified Customer Voice Portal (Unified CVP) elements, listing all the options available in the configuration. The specifications must be followed, or the element may complain with an error message or behave erratically.

Each element specification in this reference file presents information on some or all of the following six topics:

- **Overview.** Each specification starts with a brief description of the element's behavior including what it does, how it reacts to various settings and audio groups, and other miscellaneous behavior. This information should help the developer decide whether to use these elements in an application or to rely on custom elements.
- **Settings.** Settings contain information that affects how the element behaves. Each setting has the following attributes:
    - **Type**: The type of data accepted such as a boolean, integer, or enumeration.
    - **Required**: This defines whether the setting is required to have a value **if the setting is active** (available to be configured in Builder for Studio). Note: the definition of required in this case is that the setting must have an appropriate value for Builder for Studio to validate the voice element configuration.
    - **Single setting value**: This defines whether the setting can have multiple values. If set to true, then the setting may have only a single configuration value. Multiple value settings are created in Builder for Studio by right clicking on the setting and choosing the "add *setting name*" option.
    - **Substitution allowed**: This setting attribute determines if the setting value can include substitution.
    - **Default**: The initial value of an element setting when a new element is dragged to the workspace.
- **Element Data.** Some elements capture data or yield information that may be useful to other elements, or for logging purposes. The variables created by each element are listed here.
- **Exit States.** Each element may have one or more exit states that indicate the dialog status when the element execution has completed. Exit states do not appear in an element configuration and cannot be changed.

- **Audio Groups.** Voice elements define audio groups that define the different places within the element that audio can be played. Application designers configure the contents of audio groups as a list of audio items that are played one after the other. Audio items may be pre-recorded audio files, text-to-speech (TTS) phrases, and Say It Smart types (playback of formatted data such as dates, currency amounts, etc.). Each audio group can be required or optional and can also define multiple counts. Audio groups with multiple counts are used to define different audio to play each time a certain VoiceXML event occurs (often known as tapered prompts).

Chapter 2: Application_Modifier

The Application_Modifier action element is used to modify context variables and remove session data values at runtime in a voice application. It allows for a developer to change the application's environment anywhere in the callflow. A typical use for the Application_Modifier element would be for multi-language support because it can be used to change the application level **xml:lang** and encoding values. Visiting an Application_Modifier element instance will update the application for the current session only.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
maintainer **(Maintainer)**	string	No	`true`	`true`	*None*	This setting specifies the e-mail address of the voice application administrator. This value is set in a VoiceXML **<meta>** tag.
language **(Language)**	string	No	`true`	`true`	*None*	This setting specifies the language identifier to specify in each VoiceXML document's **"xml:lang"** attribute. This value is set in the **<vxml>** tag.
encoding **(Encoding)**	string	No	`true`	`true`	*None*	This setting specifies the encoding to use when creating VoiceXML documents. This value is set in the **<xml>** tag.
default_audio_path **(Default Audio Path)**	string	No	`true`	`true`	*None*	This setting specifies a partial URI to a path containing the audio content for this voice application.
remove_session_data **(Session Data to Remove)**	string	No	`false`	`true`	*None*	This setting specifies the names of session data values to remove from this voice application.

Exit States

Name	Notes
`done`	The application's context variables were modified and session data values were removed.

Studio Element Folder: Context
Class Name: `com.audium.server.action.context.ApplicationModifier`

Chapter 3: Audio

The Audio voice element simply outputs a VoiceXML page with the contents of a single audio group. The Audio element is used for greetings, error messages and any other time audio is to be played in a situation not associated with an input state.

Audio Groups

Audio Playback

Name (Label)	Max 1	Req'd	Notes
`initial_audio_group (Initial)`	Yes	Yes	The audio group containing the audio to play.

Studio Element Folder: Top Level
Class Name: `com.audium.server.voiceElement.audio.MAudio`

Chapter 4: Counter

The Counter action element is used to keep track of a count stored as element data. The initial value of the count is defined as a configuration setting. In addition, the element may be configured to increment or decrement with a user defined step size. A typical use for the Counter element would be in a loop in the call flow that increments the count until a decision element decides that the loop must end. Revisiting a Counter element instance will automatically update the count.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
initial **(Initial Count)**	int	Yes	`true`	`true`	*None*	This setting specifies at which integer value this counter should start.
type **(Type)**	string enum	Yes	`true`	`true`	*None*	This setting specifies whether the counter should be incremented or decremented. Possible values are: `decrement` | `increment`.
step **(Step Size)**	int	Yes	`true`	`true`	`1`	This setting specifies by how much this counter should be incremented or decremented.

Element Data

Name	Type	Notes
`count`	String	The current count.

Exit States

Name	Notes
`done`	The counter was updated.

Studio Element Folder: Calculation
Class Name: `com.audium.server.action.counter.CounterAction`

Chapter 5: Currency

The Currency voice element captures from the caller a currency amount in dollars and cents. The currency amount can be entered using the keypad or spoken. The captured value will be stored in element data as a decimal value (without the $ character).

There are several different formats for speaking a currency amount or entering it through the keypad. Voice browsers may use different grammars and therefore accept different utterances. However, the spoken formats listed below should result in the same behavior for all supported browsers. The tables below list each input and the value that is stored in the element variable as a result. If some data is left out, the system assumes a default value for the missing information.

Utterance	Stored Value	Example	Description
[dollar] "dollar(s)" ("and") [cent] "cent(s)"	D.CC	"thirteen dollars and fifty cents " = 13.50	Dollars are whole numbers >= 0. Cents are from 00 to 99. The word "and" is optional.
[dollar] "dollar(s) "[cent]	D.CC	"thirteen dollars five" = 13.05	Dollars are whole numbers >= 0. Cents are from 00 to 99.
[dollar] "dollar(s)"	D.00	“three hundred fifty” = 350.00	A plain whole number is interpreted as dollars with no cents.
[cent] "cent(s)"	0.CC	"three cents" = 0.03	To specify cents only, the word "cents" but be uttered. Cents are from 00 to 99.

DTMF Entry	Stored Value	Example	Description
[D]*[CC]	D.CC	3*99 = 3.99	The decimal is represented by the * button.

There are other formats that are possible, particularly when entering via DTMF and inputting incomplete amounts. These inputs may yield differing results on various voice browsers. The returned variable will always be a decimal value with the appropriate number of padded zeros if applicable.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
Inputmode **(Input Mode)**	string enum	Yes	true	false	both	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
max_noinput_count **(Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during currency input capture. 0 = infinite noinputs allowed.
max_nomatch_count **(Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during currency input capture. 0 = infinite nomatches allowed.
currency_confidence_level **(Currency Confidence Level)**	decimal (0.0 to 1.0)	Yes	true	true	0.40	The confidence level threshold to use during currency capture.
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 currency grammars will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
secure_logging **(Secure Logging)**	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Currency element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX† and nbestInterpretationX†. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.
maxnbest **(Maxnbest)**	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

†Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
Value	string	The currency amount captured. This will always be a decimal number with the appropriate number of padded zeros (up to 2).
value_confidence	float	This is the confidence value of the captured utterance. When n-best recognition is enabled, this

Name	Type	Notes
		stores the confidence score of the top hypothesis in the n-best list.
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.

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 currency capture was completed.

Audio Groups

Currency Capture

Name (Label)	Req'd	Max 1	Notes
initial_audio_group **(Initial)**	Yes	Yes	Played when the voice element first begins.

Name (Label)	Req'd	Max 1	Notes

`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 asked for help. If not specified, by default help is treated as a nomatch.

End

Name (Label)	Req'd	Max 1	Notes

`done_audio_group`
`(Done)`
	No	Yes	Played when the currency capture is completed and the voice element exits with the **`done`** exit state.

Studio Element Folder: Commerce

Class Name: `com.audium.server.voiceElement.currency.MBasicCurrency`

Chapter 6: Currency_With_Confirm

The Currency_With_Confirm voice element captures from the caller a currency amount in dollars and cents, and presents a confirmation menu allowing the caller to either accept their entry or re-enter the currency value. The currency amount can be entered using the keypad or spoken. The captured value will be stored in element data as a decimal value (without the $ character).

There are several different formats for speaking a currency amount or entering it through the keypad. Voice browsers may use different grammars and therefore accept different utterances. However, the spoken formats listed below should result in the same behavior for all supported browsers. The tables below list each input and the value that is stored in element data as a result. If some data is left out, the system assumes a default value for the missing information.

Utterance	Stored Value	Example	Description
[dollar] "dollar(s)" ("and") [cent] "cent(s)"	D.CC	"thirteen dollars and fifty cents " = 13.50	Dollars are whole numbers >= 0. Cents are from 00 to 99. The word "and" is optional.
[dollar] "dollar(s) "[cent]	D.CC	"thirteen dollars five" = 13.05	Dollars are whole numbers >= 0. Cents are from 00 to 99.
[dollar] "dollar(s)"	D.00	“three hundred fifty” = 350.00	A plain whole number is interpreted as dollars with no cents.
[cent] "cent(s)"	0.CC	"three cents" = 0.03	To specify cents only, the word "cents" but be uttered. Cents are from 00 to 99.

DTMF Entry	Stored Value	Example	Description
[D]*[CC]	D.CC	3*99 = 3.99	The decimal is represented by the * button.

There are other formats that are possible, particularly when entering via DTMF and inputting incomplete amounts. These inputs may yield inconsistent results on various voice browsers. The returned variable will always be a decimal value with the appropriate number of padded zeros if applicable.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
Inputmode **(Input Mode)**	string enum	Yes	true	false	both	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.
currency_max_noinput_count **(Currency Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during currency input capture. 0 = infinite noinputs allowed.
currency_max_nomatch_count **(Currency Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during currency input capture. 0 = infinite nomatches allowed.
confirm_max_noinput_count **(Confirm Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during currency input confirmation. 0 = infinite noinputs allowed.
confirm_max_nomatch_count **(Confirm Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during currency input confirmation. 0 = infinite nomatches allowed.
max_disconfirmed_count **(Max Disconfirmed Count)**	int ≥ 0	Yes	true	true	3	The maximum number of times a caller is allowed to disconfirm a captured input. 0 = infinite disconfirmations allowed.
currency_confidence_level **(Currency Confidence Level)**	decimal (0.0 to 1.0)	Yes	true	true	0.40	The confidence level threshold to use during currency capture.
confirm_confidence_level **(Confirm Confidence Level)**	Decimal (0.0 to 1.0)	Yes	true	true	0.50	The confidence level threshold to use during confirmation.
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 grammars of the current Currency_With_Confirm element (the

						currency and boolean grammars) will be enabled for the duration of the element. Otherwise all active grammars will be enabled.

`secure_logging`
`(Secure Logging)`
	boolean	Yes	`true`	`true`	`false`	Whether or not to enable logging of potentially sensitive data of the Currency_With_Confirm element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX[†] and nbestInterpretationX[†]. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.

`maxnbest`
`(Maxnbest)`
	int ≥ 1	Yes	`true`	`true`	`1`	The maximum number of speech recognition results that can be generated per voice input.

[†] Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
`Value`	string	The currency amount captured. This will always be a decimal number with the appropriate number of padded zeros (up to 2).
`value_confidence`	float	This is the confidence value of the captured currency utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
`confirm_confidence`	float	This is the confidence value of the captured confirm utterance.
`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`	float	This set of element data stores the confidence scores of captured n-best utterances. While the

`nbestConfidence2`
…
`nbestConfidenceX`
		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.

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.
`max_disconfirmed`	The maximum number of disconfirmation has occurred. If the max disconfirmed count is set to 0, this exit state will never occur.
`done`	The currency captured was confirmed.

Audio Groups

Currency Capture

Name (Label)	Req'd	Max 1	Notes

`currency_initial_audio_group`
`(Currency Initial)`
	Yes	Yes	Played when the voice element first begins.

`currency_nomatch_audio_group`
`(Currency NoMatch)`
	No	No	Played when a nomatch event occurs during currency capture.

`currency_noinput_audio_group`
`(Currency NoInput)`
	No	No	Played when a noinput event occurs during currency capture.

`currency_help_audio_group`
`(Currency Help)`
	No	No	Played when the caller asks for help during currency capture. If not specified, help is treated as a nomatch by default.

Currency Confirm

Name (Label)	Req'd	Max 1	Notes
`confirm_initial_audio_group`	Yes	Yes	Played when confirmation first begins.

(Confirm Initial)			
confirm_nomatch_audio_group **(Confirm NoMatch)**	No	No	Played when a nomatch event occurs during confirmation. The nomatch event count corresponds to the audio group count.
confirm_noinput_audio_group **(Confirm NoInput)**	No	No	Played when a noinput event occurs during confirmation. The noinput event count corresponds to the audio group count.
confirm_help_audio_group **(Confirm Help)**	No	No	Played when a help event occurs during confirmation. The help event count corresponds to the audio group count. If not specified, by default help throws a nomatch.
disconfirmed_audio_group **(Disconfirmed)**	No	No	Played after the caller disconfirms a captured currency entry. Upon reaching the **max_disconfirmed_count**, the prompt should be about exiting with the **max_disconfirmed** exit state.

End

Name (Label)	Req'd	Max 1	Notes
yes_audio_group **(Yes)**	No	Yes	Played after the caller chooses the "yes" option. If not specified, no audio will be played when this option is chosen.

Studio Element Folder: Commerce

Class Name: `com.audium.server.voiceElement.currency.MBasicCurrencyWithConfirm`

Chapter 7: CVP Subdialog Return

For a Cisco Unified CVP Voice application invoked as a subdialog, the CVP Subdialog Return element must be used to return data back to the calling application. The element should be used in place of Hang Up elements throughout the call flow. Like a Hang Up element, the element has no exit states.

Note: There is one exception to the above description. If the voice application will *only ever* be called by a Subdialog Invoke element (i.e., never by ICM), then the Subdialog Start and Subdialog Return elements may be used instead. Refer to Chapter 26: Subdialog Invoke, Chapter 27: Subdialog Return and Chapter 28: Subdialog Start for details.

The settings for this element are used to define what data to pass back to the calling application. The "**`Caller Input`**" setting must be assigned a value in order for the application to validate, since it is required to have a value. Each element setting corresponds to an ICM ECC external variable name, and therefore the configuration values must conform to requirements associated with ICM ECC variables. Refer to the Unified CVP documentation for further details.

The CVP Subdialog Return element can be used to enable multiple types of transfer in call failure conditions. In case of a Hook Flash (HF) or Two B-Channel Transfer (TBCT) transfer, for example, the "**`Caller Input`**" should be set to the transfer destination number prefixed with "HF" or "TBCT" (as in HF800xxxxxxx or TBCT800xxxxxxx). An HF or TBCT transfer will be invoked after the "**`Caller Input`**" was passed back from the CVP Subdialog Return element.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
`caller_input` **`(Caller Input)`**	string	yes	true	true	*none*	Required return argument that holds a value to be returned to the calling application.
`FromExtVXML0` **`(External VXML 0)`**	string	no	true	true	*none*	Optional return argument that is returned to the calling application.
`FromExtVXML1` **`(External VXML 1)`**	string	no	true	true	*none*	Optional return argument that is returned to the calling application.
`FromExtVXML2` **`(External VXML 2)`**	string	no	true	true	*none*	Optional return argument that is returned to the calling application.
`FromExtVXML3` **`(External VXML 3)`**	string	no	true	true	*none*	Optional return argument that is returned to the calling application.

Exit States

Name	Notes
`done`	The element execution is complete

Studio Element Folder: Cisco
Class Name: `com.audium.server.voiceElement.internal.CiscoSubdialogReturnElement`

Chapter 8: CVP Subdialog Start

For a Cisco CVP voice application invoked as a subdialog, the CVP Subdialog Start element must be used, which receives data from a calling application and creates corresponding element data or session data. The element should be placed at the entrance point of the application, immediately after the Start of Call element.

Data can be passed to the VoiceXML application either as HTTP parameters or VoiceXML parameters (using the <param> tag). In the first case (i.e. as HTTP parameters), Cisco Unified CVP VoiceXML Server will automatically create session data using the name of the data received. In the second case (i.e. as VoiceXML parameters), the CVP Subdialog_Start element must be configured appropriately in order for the data to be available as element or session data for the duration of the call session. For each data passed as a VoiceXML parameter, the "**Parameter**" setting must be configured with the same exact name as the data. The "**Store As**" setting can be configured to store the passed data either as session or element data. The "**Enable Digits Bypass**" setting is used to activate a VoiceXML workaround to ensure expected functionality for a particular TDM or analog phone. When this setting is set to "true", a new setting named "**Audio Filler URI**" will be enabled in VoiceXML Studio and can be configured to set a reference to a silence wave file to be played in the digits field. For IP phones the "**Enable Digits Bypass**" setting should be set to "false".

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
Parameter (Parameter)	string	no	false	true	*none*	Holds the name of a parameter passed as input to the subdialog. It must match the exact value specified in the VoiceXML page that calls the subdialog. This is a repeatable setting, so multiple values can be specified.
Where (Store As)	string	no	true	false	Session Data	Determines whether the parameter passed to the subdialog will be stored as element data or session data. By making it element data, the information will "belong" only to this element, and so there is no chance that these variables will overwrite any other variables.
enable_digits_bypass (Enable Digits Bypass)	boolean	yes	true	true	false	Determines whether the digits field is used at the beginning of an application. By default this is disabled.
audio_filler_uri (Audio Filler URI)	string	no	true	true	*none*	Configures a URI for a silence wave file to be played in the above digits field.

Exit States

Name	Notes
`done`	The element execution is complete

Studio Element Folder: Cisco

Class Name: `com.audium.server.voiceElement.internal.CiscoSubdialogStartElement`

Chapter 9: Database

The database element provides the ability to execute an SQL command on external databases within a voice application call flow. The element requires JNDI to be configured in the Java application server to handle database connections. Only a single SQL statement can be executed per element. There are four types of commands that can be made:

- **Single.** This is used to run a SQL query that returns only a single row. Element data will be created with the variable names being the names of the columns returned and the value of that column as the element data value (as a string). If no row is returned, no element data will be set.
- **Multiple.** This is used to run a SQL query that returns multiple rows. A Unified CVP-defined Java data structure, the Java class `ResultSetList`, stores the full result and is placed in session data. If no rows are returned, the `ResultSetList` object in session data will be empty. For detail about the `ResultSetList` data structure, refer to the javadocs for this class.
- **Inserts.** This is used to run a SQL INSERT command that inserts information into the database.
- **Updates.** This is used to run a SQL UPDATE command that updates information in the database.

The developer can utilize substitution to create dynamic queries. The Database element is ideal for performing simple queries and updates. It may not be sufficient for performing complex database interactions such as multiple dependent queries or stored procedure calls. One would use a custom configurable or generic action element for these tasks. Also note that in order to avoid performance issues creating database connections, implementing database pooling on the application server is highly recommended.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
type **(Type)**	string enum	Yes	true	true	single	The type of query: `single, multiple, insert` or `update`.
jndiName **(JNDI Name)**	string	Yes	true	true	*None*	This JNDI name for the SQL datasource of the database.
key **(Session Data Key)**	string	Yes	true	true	*None*	For queries of type multiple, the name of the session variable which the results of the query will be stored.
query **(SQL Query)**	string	Yes	true	true	*None*	The SQL query to be executed.

Element Data

Element data is created *only* when the “type” setting is set to “single”. Element data given the names of the return columns are created containing the respective return values. For example, if a query returned the following information:

foo	bar
123	456

The following element data will be created: “foo” with the value “123” and “bar” with the value “456”.

Session Data

Session data is created *only* when the “type” setting is set to “multiple”. In all other cases, no session data is created.

Name	Type	Notes
`[value of setting “key”]`	ResultSetList	The Java data structure that stores the returned values from a multiple type query. The name of the session data variable is specified by the developer in the “key” setting.

Exit States

Name	Notes
`done`	The database query successfully completed.

Studio Element Folder: Integration
Class Name: `com.audium.server.action.database.DatabaseAction`

Chapter 10: Date

The Date voice element captures a date input from the caller. The date can be entered using DTMF input (in the YYYYMMDD format). It can also be spoken in natural language including a month, day and year. The captured value will be stored in element data as a fixed-length date string in the YYYYMMDD format. If the year is not specified in the input, YYYY is stored as "????". And if the month or the day is not specified, MM and DD will be stored as "??".

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	*both*	The type of entry allowed for input. Possible values are: voice | dtmf | both.
Noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	*5s*	The maximum time length 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 300ms (for milliseconds). Default = 5s.
collect_max_noinput_count **(Date Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events. 0 = infinite noinputs allowed.
collect_max_nomatch_count **(Date Max NoMatch Count)**	int ≥ 0	Yes	true	false	3	The maximum number of nomatch events allowed. 0 = infinite nomatches allowed.
collect_confidence_level **(Date Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.40	The confidence level threshold to use during date capture.
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 grammars of the current Date element will be enabled for the duration of the element. Otherwise all active grammars will be enabled.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
secure_logging **(Secure Logging)**	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Date element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX[†] and nbestInterpretationX[†]. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****
maxnbest **(Maxnbest)**	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

[†] Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
value	string	The date stored in the YYYYMMDD format.
value_confidence	float	This is the confidence value of the captured date utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
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.

Exit States

Name	Notes
max_nomatch	The maximum number of nomatch events has occurred. If the max nomatch count is 0, this exit state will never occur.
max_noinput	The maximum number of noinput events has occurred. If the max noinput count is 0, this exit state will never occur.
done	The date capture was completed.

Audio Groups

Date Capture

Name (Label)	Req'd	Max 1	Notes
collect_initial_audio_group **(Date Initial)**	Yes	Yes	Played when the voice element first begins.
collect_noinput_audio_group **(Date NoInput)**	No	No	Played when a noinput event occurs during date input. The noinput event count corresponds to the audio group count.
collect_nomatch_audio_group **(Date NoMatch)**	No	No	Played when a nomatch event occurs during date input. The nomatch event count corresponds to the audio group count.
collect_help_audio_group **(Date Help)**	No	No	Played when a help event occurs during date input. The help event count corresponds to the audio group count. If not specified, a help event is treated as nomatch.

End

Name (Label)	Req'd	Max 1	Notes
done_audio_group **(Done)**	No	Yes	Played after the date capture is completed. If not specified, no audio will be played.

Studio Element Folder: Date & Time
Class Name: `com.audium.server.voiceElement.date.MBasicDate`

Chapter 11: Date_With_Confirm

The Date_With_Confirm voice element captures a date input from the caller, and presents a confirmation menu allowing the caller to either accept their entry or re-enter the date. The date can be entered using DTMF input (in the YYYYMMDD format). It can also be spoken in natural language including a month, day and year. The captured value will be stored in element data as a fixed-length date string in the YYYYMMDD format. If the year is not specified in the input, YYYY is stored as “????”. If the month or the day is not specified, MM and DD will be stored as “??”.

Settings

Name (Label)	Type	Req’d	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	*both*	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	*5s*	The maximum time length 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 300ms (for milliseconds). Default = 5s.
collect_max_noinput_count **(Date Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during date input capture. 0 = infinite noinputs allowed.
collect_max_nomatch_count **(Date Max NoMatch Count)**	int ≥ 0	Yes	true	false	3	The maximum number of nomatch events allowed during date input capture. 0 = infinite nomatches allowed.
confirm_max_noinput_count **(Confirm Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during date input confirmation. 0 = infinite noinputs allowed.
confirm_max_nomatch_count **(Confirm Max NoMatch Count)**	int ≥ 0	Yes	true	false	3	The maximum number of nomatch events allowed during date input confirmation. 0 = infinite nomatches allowed.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
max_disconfirmed_count **(Max Disconfirmed Count)**	int ≥ 0	Yes	true	false	3	The maximum number of times a caller is allowed to disconfirm a captured input. 0 = infinite disconfirmations allowed.
collect_confidence_level **(Date Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.40	The confidence level threshold to use during date capture.
confirm_confidence_level **(Confirm Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.50	The confidence level threshold to use during confirmation.
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 grammars of the current Date_With_Confirm element (the builtin date and boolean grammars) will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
secure_logging **(Secure Logging)**	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Date_With_Confirm element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX† and nbestInterpretationX†. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.
maxnbest **(Maxnbest)**	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

†Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
value	string	The date stored in the YYYYMMDD format.
value_confidence	float	This is the confidence value of the captured date utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
confirm_confidence	float	This is the confidence value of the captured confirm utterance.

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.

Exit States

Name	Notes
max_nomatch	The maximum number of nomatch events has occurred. If the max nomatch count is 0, this exit state will never occur.
max_noinput	The maximum number of noinput events has occurred. If the max noinput count is 0, this exit state will never occur.
max_disconfirmed	The maximum number of disconfirmation occurred. If the **max_disconfirmed_count** is set to 0, this exit state will never occur.
done	The date captured was confirmed.

Audio Groups

Date Capture

Name (Label)	Req'd	Max 1	Notes
collect_initial_audio_group **(Date Initial)**	Yes	Yes	Played when the voice element first begins.

Name (Label)	Req'd	Max 1	Notes

`collect_noinput_audio_group`
`(Date NoInput)`
	No	No	Played when a noinput event occurs during date input. The noinput event count corresponds to the audio group count.

`collect_nomatch_audio_group`
`(Date NoMatch)`
	No	No	Played when a nomatch event occurs during date input. The nomatch event count corresponds to the audio group count.

`collect_help_audio_group`
`(Date Help)`
	No	No	Played when a help event occurs during date input. The help event count corresponds to the audio group count. If not specified, a help event is treated as a nomatch.

Date Confirm

Name (Label)	Req'd	Max 1	Notes

`confirm_initial_audio_group`
`(Confirm Initial)`
	Yes	Yes	Played when the captured date is confirmed.

`confirm_nomatch_audio_group`
`(Confirm NoMatch)`
	No	No	Played when a nomatch event occurs during date confirmation. The nomatch event count corresponds to the audio group count.

`confirm_noinput_audio_group`
`(Confirm NoInput)`
	No	No	Played when a noinput event occurs during date confirmation. The noinput event count corresponds to the audio group count.

`confirm_help_audio_group`
`(Confirm Help)`
	No	No	Played when a help event occurs during date confirmation. The help event count corresponds to the audio group count. If not specified, by default help is treated as nomatch.

`disconfirmed_audio_group`
`(Disconfirmed)`
	No	No	Played after the caller disconfirms a date entry.

End

Name (Label)	Req'd	Max 1	Notes

`yes_audio_group`
`(Yes)`
	No	Yes	Played after the caller chooses the "yes" option. If not specified, no audio will be played when this option is chosen.

Studio Element Folder: Date & Time
Class Name: `com.audium.server.voiceElement.date.MBasicDateWithConfirm`

Chapter 12: Digits

The Digits voice element captures a string of numerical digits. It may be used to collect small or large strings of digits. The digit string can be spoken or entered using the keypad. The captured value will be stored in element data as a string. The string cannot contain any non-numerical characters. Using speech input, the number is spoken one digit at a time (i.e. 49678 is spoken "four nine six seven eight"). DTMF input can be terminated by a # keypress if desired (if not used, the entry is considered terminated when the input timeout has been reached).

With the Digits voice element, the application designer has the ability to set length restrictions on the digit string. A minimum and maximum length can be given to narrow the criteria. If a string of a specific length is required, the minimum and maximum lengths should be set to the same value. If fewer digits are entered, a nomatch event will be thrown. A string of digits with length greater than the maximum length cannot be entered.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	both	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.
max_noinput_count **(Digits Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during digits input capture. 0 = infinite noinputs allowed.
max_nomatch_count **(Digits Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during digits input capture. 0 = infinite nomatches allowed.
digits_confidence_level **(Digits Confidence Level)**	decimal (0.0 to 1.0)	Yes	true	true	0.40	The confidence level threshold to use during digits capture.
min_digit **(Min Digits)**	int > 0	Yes	true	true	*None*	Minimum number of digits allowed.

max_digit **(Max Digits)**	int > 0	Yes	true	true	*None*	Maximum number of digits allowed.
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 grammars of the current Digits element will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
secure_logging **(Secure Logging)**	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Digits element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX[†] and nbestInterpretationX[†]. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.
maxnbest **(Maxnbest)**	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

[†] Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
Value	string	The digit string value captured.
value_confidence	float	This is the confidence value of the captured utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
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.

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 digit string capture was completed.

Audio Groups

Digits Capture

Name (Label)	Req'd	Max 1	Notes
digits_initial_audio_group **(Digits Initial)**	Yes	Yes	Played when the voice element first begins.
digits_nomatch_audio_group **(Digits NoMatch)**	No	No	Played when a nomatch event occurs.
digits_noinput_audio_group **(Digits NoInput)**	No	No	Played when a noinput event occurs.
digits_help_audio_group **(Digits Help)**	No	No	Played when the caller asked for help. If not specified, help is treated as a nomatch by default.

End

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

Studio Element Folder: Number Capture
Class Name: `com.audium.server.voiceElement.digit.MBasicDigit`

Chapter 13: Digits_With_Confirm

The Digits_With_Confirm voice element captures a string of numerical digits, and presents a confirmation menu allowing the caller to either accept their entry or re-enter the digits. It may be used to collect small or large strings of digits. The digit string can be spoken or entered using the keypad. The captured value will be stored in element data as a string. The string cannot contain non-numerical characters. Using speech input, the number is spoken one digit at a time (i.e. 49678 is spoken "four nine six seven eight"). DTMF input can be terminated by a # keypress if desired (otherwise, the entry is considered terminated when the input timeout is reached).

With the Digits_With_Configm voice element, the application designer has the ability to set length restrictions on the digit string. A minimum and maximum length can be given to narrow the criteria. If a string of a specific length is required, the minimum and maximum lengths should be set to the same value. If fewer digits are entered, a nomatch event will be thrown. A string of digits with length greater than the maximum length cannot be entered.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	both	The type of entry allowed for input (during digits capture and confirmation). Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.
digits_max_noinput_count **(Digits Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during digits input capture. 0 = infinite noinputs allowed.
digits_max_nomatch_count **(Digits Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during digits input capture. 0 = infinite nomatches allowed.
confirm_max_noinput_count **(Confirm Max NoIput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during digits input confirmation. 0 = infinite

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
						noinputs allowed.

`confirm_max_nomatch_count`
`(Confirm Max NoMatch Count)`
	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during digits input confirmation. `0` = infinite nomatches allowed.

`max_disconfirmed_count`
`(Max Disconfirmed Count)`
	int ≥ 0	Yes	true	true	3	The maximum number of times a caller is allowed to disconfirm a captured digits input. `0` = infinite disconfirmations allowed.

`digits_confidence_level`
`(Digits Confidence Level)`
	decimal (0.0 to 1.0)	Yes	true	true	0.40	The confidence level threshold to use during digits capture.

`confirm_confidence_level`
`(Confirm Confidence Level)`
	decimal (0.0 to 1.0)	Yes	true	true	0.50	The confidence level threshold to use during confirmation.

`min_digit`
`(Min Digits)`
	int > 0	Yes	true	true	*None*	Minimum number of digits allowed.

`max_digit`
`(Max Digits)`
	int > 0	Yes	true	true	*None*	Maximum number of digits allowed.

`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 grammars of the current Digits_With_Confirm element (the builtin digits and boolean grammars) will be enabled for the duration of the element. Otherwise all active grammars will be enabled.

`secure_logging`
`(Secure Logging)`
	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Digits_With_Confirm element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX[†] and nbestInterpretationX[†]. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.

`maxnbest`
`(Maxnbest)`
	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

[†] Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
`Value`	string	The digit string captured.
`value_confidence`	float	This is the confidence value of the captured digit string utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
`confirm_confidence`	float	This is the confidence value of the captured confirm utterance.
`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.

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.
`max_disconfirmed`	The maximum number of disconfirmation has occurred. If the max disconfirmed count is set to 0, this exit state will never occur.
`done`	The digit string captured was confirmed.

Audio Groups

Digits Capture

Name (Label)	Req'd	Max 1	Notes
digits_initial_audio_group **(Digits Initial)**	Yes	Yes	Played when the voice element first begins.
digits_nomatch_audio_group **(Digits NoMatch)**	No	No	Played when a nomatch event occurs during digits capture.
digits_noinput_audio_group **(Digits NoInput)**	No	No	Played when a noinput event occurs during digits capture.
digits_help_audio_group **(Digits Help)**	No	No	Played when the caller asks for help during digits capture. If not specified, by default help is treated as a nomatch.

Digits Confirm

Name (Label)	Req'd	Max 1	Notes
confirm_initial_audio_group **(Confirm Initial)**	Yes	Yes	Played when confirmation first begins.
confirm_nomatch_audio_group **(Confirm NoMatch)**	No	No	Played when a nomatch event occurs during confirmation. The nomatch event count corresponds to the audio group count.
confirm_noinput_audio_group **(Confirm NoInput)**	No	No	Played when a noinput event occurs during confirmation. The noinput event count corresponds to the audio group count.
confirm_help_audio_group **(Confirm Help)**	No	No	Played when a help event occurs during confirmation. The help event count corresponds to the audio group count. If not specified, by default help throws a nomatch.
disconfirmed_audio_group **(Disconfirmed)**	No	No	Played after the caller disconfirms a captured digits entry. Upon reaching the **max_disconfirmed_count**, the prompt should be about exiting with the **max_disconfirmed** exit state.

End

Name (Label)	Req'd	Max 1	Notes
yes_audio_group **(Yes)**	No	Yes	Played after the caller chooses the "yes" option. If not specified, no audio will be played when this option is chosen.

Studio Element Folder: Number Capture
Class Name: `com.audium.server.voiceElement.digit.MBasicDigitWithConfirm`

Chapter 14: Email

The Email action element sends messages using the Javamail package supplied by the application server to send messages to the provided email address. Additionally the message can include attachments. The application server must be configured to set a JNDI datasource for mail sessions. The *to* and *tolist* fields are not individually required however at least one must be defined. Email addresses are not verified for syntax or validity. Attachments that do not exist will be skipped but the message will still be sent. Repeated email addresses are sent the message multiple times. The *toList*, *ccList* and *bccList* settings must refer to session data variables that holds a `ResultSetList` Java class holding a list of email addresses (retrieved from a Database element).

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`jndiName`
`(JNDI Name)`
	string	Yes	`true`	`true`	*None*	The configured JNDI datasource for mail sessions under the java application server.

`to`
`(To)`
	string	No	`false`	`true`	*None*	The email address this message will be sent to. This setting is repeatable so that each setting value contains a separate email address.

`toList`
`(To List)`
	string	No	`true`	`true`	*None*	The name of a session data variable containing a `ResultSetList` object holding a list of email addresses as retrieved from a Database element. The email will be sent to every address in this list.

`from`
`(From)`
	string	Yes	`true`	`true`	*None*	The email address this message will be sent from.

`cc`
`(Cc)`
	string	No	`false`	`true`	*None*	The email address this message will be carbon copied to. This setting is repeatable so that each setting value contains a separate email address.

`ccList`
`(Cc List)`
	string	No	`true`	`true`	*None*	The name of a session data variable containing a `ResultSetList` object holding a list of email addresses as retrieved from a Database element. The email will be carbon copied to each address in this list.

`bcc`
`(Bcc)`
	string	No	`false`	`true`	*None*	The email address this message will be blind carbon copied to. This setting is repeatable so that each setting value contains a separate email address.

`bccList`
`(Bcc List)`
	string	No	`true`	`true`	*None*	The name of a session data variable containing a `ResultSetList` object holding a list of email addresses as retrieved from a Database element. The email will be blind carbon copied to each address in this list.

`subject`
`(Subject)`
	string	No	`true`	`true`	*None*	Subject field of the email.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
attachment (Attachment)	string	No	false	true	*None*	Full local path of the file to be attached. This setting is repeatable so that each setting value contains a reference to separate attachments.
messageBody (Message Body)	string	Yes	true	true	*None*	The message body of the email.

Exit States

Name	Notes
done	The database query successfully completed.

Studio Element Folder: Notification
Class Name: `com.audium.server.action.email.EmailAction`

Chapter 15: 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	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	both	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (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.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
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 five 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) 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**, are required, but at least one must be specified since a form cannot be completed without a grammar.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`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`**, are required, but at least one must be specified since a form cannot be completed without a grammar.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
voice_keyword **(Voice Keyword)**	string	*No	false	true	*None*	Defines the inline voice 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 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 that 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**, are required, but at least one must be specified since a form cannot be completed without at least one grammar.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
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 follwing 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 that 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**, are 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 (e.g. "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.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
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*	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	Whether or not to enable logging of potentially sensitive data of the Form element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, <SLOT_ELEMENT_DATAX>, nbestUtteranceX, and nbestInterpretationX. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.

Notes:

- 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 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 that if the **`slot_element_data`**

Name	Type	Notes
		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.
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	Max 1	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.

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

Chapter 16: Form_With_Confirm

The Form_With_Confirm voice element is used to capture and confirm 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_With_Confirm voice element can be configured to listen for voice input only, DTMF input only, or both voice and DTMF input. In short, the Form_With_Confirm element is the most flexible of included elements that have confirmation menus as it allows almost any custom information to be captured and confirmed without requiring a separate voice element. If a Unified CVP or third-party voice element does not capture and confirm the information desired, one can always use a Form_With_Confirm element before embarking on constructing a custom voice element.

The Form_With_Confirm 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_With_Confirm 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_With_Confirm 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	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	both	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
form_max_noinput_count **(Form Max NoInput)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during form input capture. 0 = infinite noinputs allowed.
form_max_nomatch_count **(Form Max NoMatch)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during form input capture. 0 = infinite nomatches allowed.
confirm_max_noinput_count **(Confirm Max NoInput)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during form input confirmation. 0 = infinite noinputs allowed.
confirm_max_nomatch_count **(Confirm Max NoMatch)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during form input confirmation. 0 = infinite nomatches allowed.
max_disconfirmed_count **(Max Disconfirmed Count)**	int ≥ 0	Yes	true	true	3	The maximum number of times a caller is allowed to disconfirm a captured input. 0 = infinite disconfirmations allowed.
form_confidence_level **(Form Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.40	The confidence level threshold to use for capture of the form data.
confirm_confidence_level **(Confirm Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.50	The confidence level threshold to use for confirmation of the form data.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
voice_grammar **(Voice Grammar)**	string	*No	false	true	*None*	Defines an external voice grammar for Form_With_Confirm, in a string format delimited with semi-colons specifying five 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) 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**, are required, but at least one must be specified since a form cannot be completed without a grammar.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
dtmf_grammar **(DTMF Grammar)**	URI	*No	false	true	*None*	Defines an external DTMF grammar for Form_With_Confirm, 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**, are required, but at least one must be specified since a form cannot be completed without a grammar.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`voice_keyword`
`(Voice Keyword)`
	string	`*No`	`false`	`true`	*None*	Defines the inline voice grammar for Form_With_Confirm, 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 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 that 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`**, are required, but at least one must be specified since a form cannot be completed without at least one grammar.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`dtmf_keypress`
`(DTMF Keypress)`
	character (0-9, #, *)	`*No`	`false`	`true`	*`None`*	Defines the inline DTMF grammar for Form_With_Confirm, 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 follwing 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 that 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`**, are required, but at least one must be specified since a form cannot be completed without at least one grammar.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
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 (e.g. "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_With_Confirm 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_With_Confirm 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_With_Confirm element grammars (including the builtin boolean grammar for confirmation) 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*	The value to assign to the VXML field-level name attribute.
slot_name **(Field Slot)**	string	No	true	true	*None*	The name to assign to the VXML field-level slot attribute. If left unspecified (i.e. the default value), the field will not have 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.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`secure_logging`
`(Secure Logging)`
	boolean	`Yes`	`true`	`true`	`false`	Whether or not to enable logging of potentially sensitive data of the Form_With_Confirm element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, <SLOT_ELEMENT_DATAX>, nbestUtteranceX, and nbestInterpretationX. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.

Notes:

- 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_With_Confirm 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_With_Confirm element to store this data:

  - Leave the the "**`slot_element_data`**" setting empty. The Form_With_Confirm 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_With_Confirm 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_With_Confirm 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 that 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.
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.
confirm_noinput_count	int ≥ 0	This stores the number of no input events that the browser returned during the confirmation phase of the VXML field name variable.
confirm_nomatch_count	int ≥ 0	This stores the number of no match events that the browser returned during the confirmation 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.
`max_disconfirmed`	The maximum number of disconfirm events has occurred. If the disconfirm 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	Max 1	Notes

`form_initial_audio_group`
`(Form Initial)`
	Yes	Yes	Played when the voice element first begins.

`form_nomatch_audio_group`
`(Form NoMatch)`
	No	No	Played when a nomatch event occurs during form data capture.

`form_noinput_audio_group`
`(Form NoInput)`
	No	No	Played when a noinput event occurs during form data capture.

`form_help_audio_group`
`(Form Help)`
	No	No	Played when the caller asks for help during form data capture. If not specified, help is treated as a nomatch event by default.

Form Data Confirm

Name (Label)	Req'd	Max 1	Notes

`confirm_initial_audio_group`
`(Confirm Initial)`
	Yes	Yes	Played after the caller enters a value, requesting the caller's confirmation of that value.

`confirm_nomatch_audio_group`
`(Confirm NoMatch)`
	No	No	Played when a nomatch event occurs during confirmation.

`confirm_noinput_audio_group`
`(Confirm NoInput)`
	No	No	Played when a noinput event occurs during confirmation.

`confirm_help_audio_group`
`(Confirm Help)`
	No	No	Played when the caller asks for help during confirmation.

`disconfirmed_audio_group`
`(Disconfirmed)`
	No	No	Played when the caller disconfirms the value.

End

Name (Label)	Req'd	Max 1	Notes

`yes_audio_group`
`(Yes)`
	No	Yes	Played after the caller chooses the "yes" option. If not specified, no audio will be played when this option is chosen.

Studio Element Folder: Form

Class Name: `com.audium.server.voiceElement.form.MFoundationFormWithConfirm`

Chapter 17: Math

The Math action element is used to evaluate basic mathematical expressions. The mathematical expression is composed of operators and functions in the form of a string which is passed as a setting to the element, parsed and evaluated at runtime. The result is a double value stored as a string in either element data or session data. All common arithmetic operators are supported. Boolean operators are also fully supported. Boolean expressions are evaluated to be either 1.0 or 0.0 (true or false respectively).

Examples

Expression: 2 * 4
Result: 8.0

Expression: sqrt(16)
Result: 4.0

Expression: {Data.Session.myNumber} == 4
Result: 1.0

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
Type **(Type)**	string enum	Yes	`true`	`false`	`Element`	This setting specifies the type of data that will store the result of the mathematical expression. Possible values are: `Element` | `Session`. Default = `Element`.
Name **(Name)**	string	Yes	`true`	`true`	*None*	This setting specifies the name to assign to the data that will store the result of the mathematical expression.
Expression **(Expression)**	string	Yes	`true`	`true`	*None*	This setting specifies the mathematical expression to parse and evaluate. For supported operators and functions see tables below.

Operators

Operation	Operator
Power	^
Boolean Not	!
Unary Plus, Unary Minus	+x, -x
Modulus	%
Division	/
Multiplication	*
Addition, Subtraction	+, -
Less or Equal, More or Equal	<=, >=
Less Than, Greater Than	<, >
Not Equal, Equal	!=, ==
Boolean And	&&
Boolean Or	||

Functions

Function	Syntax
Sine	sin(x)
Cosine	cos(x)
Tangent	tan(x)
Arc Sine	asin(x)
Arc Cosine	acos(x)
Arc Tangent	atan(x)
Arc Tangent (with 2 parameters)	atan2(y, x)
Hyperbolic Sine	sinh(x)
Hyperbolic Cosine	cosh(x)
Hyperbolic Tangent	tanh(x)
Inverse Hyperbolic Sine	asinh(x)
Inverse Hyperbolic Cosine	acosh(x)
Inverse Hyperbolic Tangent	atanh(x)
Natural Logarithm	ln(x)
Logarithm base 10	log(x)
Exponential	exp(x)
Absolute Value / Magnitude	abs()
Modulus	mod()
Square Root	sqrt()
Sum	sum()
If	if()

Element Data

Element data is created *only* when the “type” setting is set to “Element”. In all other cases, no element data is created.

Name	Type	Notes
`[value of setting “name”]`	String	The result of the mathematical expression.

Session Data

Session data is created *only* when the “type” setting is set to “Session”. In all other cases, no session data is created.

Name	Type	Notes
`[value of setting “name”]`	String	The result of the mathematical expression.

Exit States

Name	Notes
`done`	The mathematical expression was evaluated and the result was stored as either element data or session data.

Studio Element Folder: Math
Class Name: `com.audium.server.action.math.MathAction`

Chapter 18: 2_Option_Menu, 3_Option_Menu, . . . , 10_Option_Menu

These voice elements define menus that support from 2 to 10 options. The Menu voice elements are similar to the Form voice element, however the number of choices is fixed and all grammars are defined in the voice element itself. Additionally, there is an exit state for each option, therefore the captured value does not have to be analyzed afterwards to determine the next dialog in the call flow. Use Menu elements when the situation defines a fixed number of choices where each choice does something different in the call flow.

Because the number of exit states is fixed for a voice element, there are separate voice elements for Menu voice elements with 2 to10 options. For each additional option, three additional settings are added to handle the spoken keyword, DTMF entry, and interpretation value for each option. The audio groups and element data saved are the same for all Menu voice elements.

Each option must be assigned an interpretation value that the element will return as element data named **`value`** when any of the keywords or DTMF key presses assigned to that option are captured. The element variable (**`value`**) will contain the same value regardless of the input mode (speech or DTMF).

The audio groups are identical to those of the Form voice element. The **`done_audio_group`** group may be used for a message that is to be played regardless of what option is chosen. If an **`option`** specific message is desired, it is recommended that the **`done_audio_group`** not be configured and an Audio voice element be used after the particular choice is made.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`noinput_timeout`
`(Noinput Timeout)`
	string	Yes	`true`	`true`	`5s`	The maximum time length 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 `300ms` (for milliseconds). Default = `5s`.

`max_noinput_count`
`(Max NoInput Count)`
	int ≥ 0	Yes	`true`	`true`	`3`	The maximum number of noinput events allowed during input capture. `0` = infinite

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
						noinputs allowed.
max_nomatch_count **(Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during input capture. 0 = infinite nomatches allowed.
confidence_level **(Confidence Level)**	decimal (0.0 to 1.0)	Yes	true	true	0.40	The confidence level threshold to use.
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 grammars of the current X_Option_Menu element will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
optionX_dtmf **(Option X DTMF)**	Character (0-9, #, *)[†]	No	true	true	*None*	This setting defines the DTMF grammar that can be used to select the menu **optionX**. The valid format is a string separated with a semi-colon specifying two values in this order: 1) The language context in which the current input should be included in the menu grammar (optional). If omitted the language used will be the same as the page-scoped language. 2) The dtmf keypress or keypresses that is included in the menu DTMF grammar (required) Sample configurations values are: en-US;1 1 Additional **optionX_dtmf** settings may be used to define multiple dtmf keypresses corresponding to the same return value. Note that 1) at minimum, one of the two settings: **optionX_dtmf** or **optionX_voice** *must* be specified; and 2) keypresses are currently limited to single digits.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`optionX_voice`
`(Option X Voice)`
	string	No	`true`	`true`	`None`	This setting defines the voice grammar that can be used to select the menu **`optionX`**. Each configuration of this setting specifies an option for the grammar. The valid format is a string separated with semi-colons specifying three values in this order:
						1) The language context in which the current input should be included in the menu grammar (optional). If omitted the language used will be the same as the page-scoped language.
						2) "exact" or "approximate" (optional) for the accept attribute value, where if "exact", the spoken utterance must match the expected value exactly; and where if "approximate", the spoken utterance may match one of several words
						3) The voice keyword or keywords (required) that is included in the menu voice grammar.
						If one of the optional parameters is defined, two semi-colons must be used, even if the other parameter is not used. Sample configurations values are:
						en-US;exact;news report
						;approximate;news report
						fr-FR;;news report
						news report
						Additional **`optionX_voice`** settings may be used to define multiple matching voice keywords corresponding to the same return value. Note that at the minimum, one of the two settings: **`optionX_dtmf`** or **`optionX_voice`** *must* be specified.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`optionX_value`
`(Option X Value)`
	string	Yes	`false`	`true`	*None*	The value to be stored in the element data **`value`** for this voice element when the caller selects **`optionX`**. Note that only a single value is allowed for each option.

where X is 2 – 10 as applicable.

† Some voice browsers may not support menu options using * or #.

Element Data

Name	Type	Notes
`value`	string	The value associated with the keyword or DTMF keypress inputted by the caller is stored in this variable.
`value_confidence`	float	This is the confidence value of the matched utterance.

Exit States

Name	Notes
`max_nomatch`	The maximum number of nomatch events has occurred. If the **`max_nomatch_count`** is `0`, this exit state will never occur.
`max_noinput`	The maximum number of noinput events has occurred. If the **`max_noinput_count`** is `0`, this exit state will never occur.
`optionX`	The utterance or DTMF entry matched **`optionX.`**

where X is 2 – 10 as applicable.

Notes:

- Each option can react on just a spoken keyword, just DTMF keypresses, or both, but at least one method must be specified or an error will be reported.
- All options in the menu must have a consistent input mode. For example, a menu cannot be configured so that option 1 is chosen through both voice and DTMF but option 2 is chosen only through voice.
- There are no menus with more than 10 options. In cases where more are needed, a Form voice element is recommended.

Audio Groups

Menu Option Capture

Name (Label)	Req'd	Max 1	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 asked for help. If not specified, by default help is treated as a nomatch.

End

Name (Label)	Req'd	Max 1	Notes
`done_audio_group (Done)`	No	Yes	Played when the voice element completes any of the option exit states.

Studio Element Folder: Menu
Class Name: `com.audium.server.voiceElement.menu.MFoundationXOptionMenu`

Chapter 19: Number

The Number voice element captures a number input from the caller. The number can be spoken or entered using the keypad. The resulting value will be stored in element data as a decimal value. The number can be negative or positive and can contain a decimal point. Using DTMF entry, however, the number is restricted to being positive and the decimal point is entered by pressing the * key. Using speech input, the number may be spoken naturally.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	String enum	Yes	true	false	both	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.
max_noinput_count **(Number Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during number input capture. 0 = infinite noinputs allowed.
max_nomatch_count **(Number Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during number input capture. 0 = infinite nomatches allowed.
number_confidence_level **(Number Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.40	The confidence level threshold to use during number capture.
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 grammars of the current Number element will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
secure_logging **(Secure Logging)**	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Number element. If set to

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
						true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX[†] and nbestInterpretationX[†]. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.
maxnbest **(Maxnbest)**	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

[†] Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
Value	string	The number captured and stored as a whole or decimal number with an optional minus sign.
value_confidence	float	This is the confidence value of the captured utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
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	string	This set of element data stores the input modes of captured n-best utterances.

Name	Type	Notes
`nbestInputmode2` … **`nbestInputmodeX`**		

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 number capture was completed.

Notes:

- If the number to be captured is a positive whole number and the input is via DTMF, the number can be entered using this voice element or the **Digits** voice element.

Audio Groups

Number Capture

Name (Label)	Req'd	Max 1	Notes

`number_initial_audio_group`
`(Number Initial)`
	Yes	Yes	Played when the voice element first begins.

`number_nomatch_audio_group`
`(Number NoMatch)`
	No	No	Played when a nomatch event occurs.

`number_noinput_audio_group`
`(Number NoInput)`
	No	No	Played when a noinput event occurs.

`number_help_audio_group`
`(Number Help)`
	No	No	Played when the caller asked for help. If not specified, by default help is treated as a nomatch.

End

Name (Label)	Req'd	Max 1	Notes

`done_audio_group`
`(Done)`
	No	Yes	Played when the number capture is completed and the voice element exits with the **`done`** exit state.

Studio Element Folder: Number Capture
Class Name: `com.audium.server.voiceElement.number.MBasicNumber`

Chapter 20: Number_With_Confirm

The Number_With_Confirm voice element captures a standard number, and presents a confirmation menu allowing the caller to either accept their entry or re-enter the number. The number can be spoken or entered using the keypad. The resulting value will be stored in element data as a decimal value. The number can be negative or positive and can contain a decimal point. Using DTMF entry, however, the number is restricted to being positive and the decimal point is entered by pressing the * key. Using speech input, the number may be spoken naturally.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	both	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.
number_max_noinput_count **(Number Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during number input capture. 0 = infinite noinputs allowed.
number_max_nomatch_count **(Number Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during number input capture. 0 = infinite nomatches allowed.
confirm_max_noinput_count **(Confirm Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during number input confirmation. 0 = infinite noinputs allowed.
confirm_max_nomatch_count **(Confirm Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during number input confirmation. 0 = infinite nomatches allowed.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
max_disconfirmed_count **(Max Disconfirmed Count)**	int ≥ 0	Yes	true	true	3	The maximum number of times a caller is allowed to disconfirm a captured input. 0 = infinite disconfirmations allowed.
number_confidence_level **(Number Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.40	The confidence level threshold to use during number capture.
confirm_confidence_level **(Confirm Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.50	The confidence level threshold to use during confirmation.
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 grammars of the current Number_With_Confirm element (the builtin number and boolean grammars) will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
secure_logging **(Secure Logging)**	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Number_With_Confirm element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX† and nbestInterpretationX†. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.
maxnbest **(Maxnbest)**	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

†Refer to the Element Data table for information about nbestUtteraceX and nbestInterpretationX.

Element Data

Name	Type	Notes
Value	string	The number captured and stored as a whole or decimal number with an optional minus sign.
value_confidence	float	This is the confidence value of the captured number utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.

confirm_confidence	float	This is the confidence value of the captured confirm utterance.
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.

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.
max_disconfirmed	The maximum number of disconfirmation has occurred. If the max disconfirmed count is set to 0, this exit state will never occur.
done	The number captured was confirmed.

Notes:

- If the number to be captured is a positive whole number and the input is via DTMF, the number can be entered using this voice element or the **Digits_With_Confirm** voice element.

Audio Groups

Number Capture

Name (Label)	Req'd	Max 1	Notes

`number_initial_audio_group`
`(Number Initial)`
	Yes	Yes	Played when the voice element first begins.

`number_nomatch_audio_group`
`(Number NoMatch)`
	No	No	Played when a nomatch event occurs during number capture.

`number_noinput_audio_group`
`(Number NoInput)`
	No	No	Played when a noinput event occurs during number capture.

`number_help_audio_group`
`(Number Help)`
	No	No	Played when the caller asks for help during number capture. If not specified, by default help is treated as a nomatch.

Number Confirm

Name (Label)	Req'd	Max 1	Notes

`confirm_initial_audio_group`
`(Confirm Initial)`
	Yes	Yes	Played when confirmation first begins.

`confirm_nomatch_audio_group`
`(Confirm NoMatch)`
	No	No	Played when a nomatch event occurs during confirmation. The nomatch event count corresponds to the audio group count.

`confirm_noinput_audio_group`
`(Confirm NoInput)`
	No	No	Played when a noinput event occurs during confirmation. The noinput event count corresponds to the audio group count.

`confirm_help_audio_group`
`(Confirm Help)`
	No	No	Played when a help event occurs during confirmation. The help event count corresponds to the audio group count. If not specified, help throws a nomatch by default.

`disconfirmed_audio_group`
`(Disconfirmed)`
	No	No	Played after the caller disconfirms a captured number entry.

End

Name (Label)	Req'd	Max 1	Notes

`yes_audio_group`
`(Yes)`
	No	Yes	Played after the caller chooses the "yes" option. If not specified, no audio will be played when this option is chosen.

Studio Element Folder: Number Capture
Class Name: `com.audium.server.voiceElement.number.MBasicNumberWithConfirm`

Chapter 21: Phone

The Phone voice element captures a phone number input from the caller. The phone number can be spoken or entered using the keypad. The captured value will be stored in element data as a string. The string may contain a number of digits and an optional character "x" to indicate a phone number with an extension. Using speech input, the entire phone number (including the extension) may be spoken in natural language. Using DTMF entry, the caller can enter an extension by pressing the * keypress followed by the extension.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	*both*	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	*5s*	The maximum time length 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 300ms (for milliseconds). Default = 5s.
collect_max_noinput_count **(Phone Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during phone input capture. 0 = infinite noinputs allowed.
collect_max_nomatch_count **(Phone Max NoMatch Count)**	int ≥ 0	Yes	true	false	3	The maximum number of nomatch events allowed during phone input capture. 0 = infinite nomatches allowed.
collect_confidence_level **(Phone Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.40	The confidence level threshold to use during phone capture.
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 grammars of the current Phone element will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
secure_logging	boolean	Yes	true	true	false	Whether or not to enable logging of potentially

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
(Secure Logging)						sensitive data of the Phone element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX† and nbestInterpretationX†. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.
maxnbest (Maxnbest)	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

†Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
Value	string	The phone number captured.
value_confidence	float	This is the confidence value of the captured phone utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
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.

Name	Type	Notes

`nbestInputmode1`
`nbestInputmode2`
…
`nbestInputmodeX`
	string	This set of element data stores the input modes of captured n-best utterances.

Exit States

Name	Notes
`max_nomatch`	The maximum number of nomatch events has occurred. If the max nomatch count is 0, this exit state will never occur.
`max_noinput`	The maximum number of noinput events has occurred. If the max noinput count is 0, this exit state will never occur.
`done`	The phone number capture was completed.

Audio Groups

Phone Capture

Name (Label)	Req'd	Max 1	Notes

`collect_initial_audio_group`
`(Phone Initial)`
	Yes	Yes	Played when the voice element first begins.

`collect_noinput_audio_group`
`(Phone NoInput)`
	No	No	Played when a noinput event occurs.

`collect_nomatch_audio_group`
`(Phone NoMatch)`
	No	No	Played when a nomatch event occurs.

`collect_help_audio_group`
`(Phone Help)`
	No	No	Played when the caller asked for help. If not specified, help is treated as a nomatch by default.

End

Name (Label)	Req'd	Max 1	Notes

`done_audio_group`
`(Done)`
	No	Yes	Played after phone capture is completed.

Studio Element Folder: Number Capture
Class Name: `com.audium.server.voiceElement.phone.MBasicPhone`

Chapter 22: Phone_With_Confirm

The Phone_With_Confirm voice element captures a phone number input from the caller, and presents a confirmation menu allowing the caller to either accept their entry or re-enter the phone number. The phone numbercan be spoken or entered using the keypad. The captured value will be stored in element data as a string. The string may contain a number of digits and an optional character "x" to indicate a phone number with an extension. Using speech input, the entire phone number (including the extension) may be spoken in natural language. Using DTMF entry, the caller can enter an extension by pressing the * keypress followed by the extension.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	*both*	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	*5s*	The maximum time length 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 300ms (for milliseconds). Default = 5s.
collect_max_noinput_count **(Phone Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during phone input capture. 0 = infinite noinputs allowed.
collect_max_nomatch_count **(Phone Max NoMatch Count)**	int ≥ 0	Yes	true	false	3	The maximum number of nomatch events allowed during phone input capture. 0 = infinite nomatches allowed.
confirm_max_noinput_count **(Confirm Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during phone input confirmation. 0 = infinite noinputs allowed.
confirm_max_nomatch_count **(Confirm Max NoMatch Count)**	int ≥ 0	Yes	true	false	3	The maximum number of nomatch events allowed during phone input confirmation. 0 = infinite nomatches allowed.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
max_disconfirmed_count **(Max Disconfirmed Count)**	int ≥ 0	Yes	true	false	3	The maximum number of times a caller is allowed to disconfirm a captured input. 0 = infinite disconfirmations allowed.
collect_confidence_level **(Phone Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.40	The confidence level threshold to use during phone capture.
confirm_confidence_level **(Confirm Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.50	The confidence level threshold to use during confirmation.
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 grammars of the current Phone_With_Confirm element (the builtin phone and boolean grammars) will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
secure_logging **(Secure Logging)**	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Phone_With_Confirm element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX[†] and nbestInterpretationX[†]. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.
maxnbest **(Maxnbest)**	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

[†] Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
Value	string	The phone number captured.
value_confidence	float	This is the confidence value of the captured phone number utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
confirm_confidence	float	This is the confidence value of the captured confirm utterance.

`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.

Exit States

Name	Notes
`max_nomatch`	The maximum number of nomatch events occurred. If the max nomatch count is 0, this exit state will never occur.
`max_noinput`	The maximum number of noinput events occurred. If the max noinput count is 0, this exit state will never occur.
`max_disconfirmed`	The maximum number of disconfirmation occurred. If the max disconfirmed count is set to 0, this exit state will never occur.
`done`	The phone number captured was confirmed.

Audio Groups

Phone Capture

Name (Label)	Req'd	Max 1	Notes

`collect_initial_audio_group`
`(Phone Initial)`
	Yes	Yes	Played when the voice element first begins.

`collect_noinput_audio_group`	No	No	Played when a noinput event occurs during phone number capture.

Name (Label)	Req'd	Max 1	Notes

`collect_initial_audio_group`
`(Phone Initial)`
	Yes	Yes	Played when the voice element first begins.

`(Phone NoInput)`			

`collect_nomatch_audio_group`
`(Phone NoMatch)`
	No	No	Played when a nomatch event occurs during phone number capture.

`collect_help_audio_group`
`(Phone Help)`
	No	No	Played when the caller asked for help during phone number capture. If not specified, help is treated as a nomatch by default.

Phone Confirm

Name (Label)	Req'd	Max 1	Notes

`confirm_initial_audio_group`
`(Confirm Initial)`
	Yes	Yes	Played when confirmation first begins.

`confirm_noinput_audio_group`
`(Confirm NoInput)`
	No	No	Played when a noinput event occurs during confirmation. The noinput event count corresponds to the audio group count.

`confirm_nomatch_audio_group`
`(Confirm NoMatch)`
	No	No	Played when a nomatch event occurs during confirmation. The nomatch event count corresponds to the audio group count.

`confirm_help_audio_group`
`(Confirm Help)`
	No	No	Played when a help event occurs during confirmation. The help event count corresponds to the audio group count.

`disconfirmed_audio_group`
`(Disconfirmed)`
	No	No	Played after the caller disconfirms a captured phone entry. Upon reaching the **`max_disconfirmed_count`**, the prompt content should be about exiting with the **`max_disconfirmed`** exit state.

End

Name (Label)	Req'd	Max 1	Notes

`yes_audio_group`
`(Yes)`
	No	Yes	Played after the caller chooses the "yes" option. If not specified, no audio will be played when this option is chosen.

Studio Element Folder: Number Capture
Class Name: `com.audium.server.voiceElement.phone.MBasicPhoneWithConfirm`

Chapter 23: Record

The Record voice element makes a recording of the caller's voice. A prompt is played to the caller then the voice element records the caller’s voice until a termination key is inputted, the recording time limit has been reached, or (if the configuration specifies so) the caller hung up. An audio cue (beep) may be activated to signal to the caller that the system is ready to record the caller’s voice. Different voice browsers may have varying default maximum lengths for voice recording.

The recording is sent to the Record element by the voice browser and is stored in an audio file in the location specified by the developer. Any pre-existing file with the same name will be overwritten. The element can be configured to produce a non-repeating filename so all recordings can be retained. The format for this filename is *audioN.wav* where N is the number of milliseconds since midnight January 1, 1970 (GMT). All recordings are saved in the WAV format.

Settings

Name (Label)	Type	Req’d	Single setting value	Substitution allowed	Default	Notes
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.
max_noinput_count **(Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during input capture. 0 = infinite noinputs allowed.
start_with_beep **(Start With Beep)**	boolean	Yes	true	true	true	Whether or not to play a beep before recording begins.
terminate_on_dtmf **(Terminate On DTMF)**	boolean	Yes	true	true	true	Whether or not the caller can end the recording by pressing a touchtone key.
keep_recording_on_hangup **(Keep Recording On Hangup)**	boolean	Yes	true	true	false	Whether or not the recording is stored if the caller hung up while making the recording. Default = false

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
max_record_time **(Max Record Time)**	int ≥ 0	Yes	true	true	180s	The maximum time that the recording is allowed to last. Possible values are standard time designations including both a non-negative number and a time unit, for example, 30s (for seconds) or 300ms (for milliseconds). Default = 180s.
final_silence **(Final Silence)**	int ≥ 0	Yes	true	true	4s	The interval of silence that indicates the end of speech. Possible values are standard time designations including both a non-negative number and a time unit, for example, 3s (for seconds) or 300ms (for milliseconds). Default = 4s.
filename **(Filename)**	string	No	true	true	*None*	The filename of the recording (without extension). If left blank, an auto-generated filename will be used.
file_type **(File Type)**	string enum	Yes	true	true	wav	This specifies the audio type of the file that will hold the recording. Possible values are: wav | vox | au | other.
mime_type **(Mime Type)**	string	Yes	true	true	*none*	This specifies the MIME type of the file that will hold the recording, if **file_type** is set to other.
file_extension **(File Extension)**	string	No	true	true	*None*	This specifies the file extension to use for the recorded file. A file extension different from the file type can be used. For example, with a mime type of vox, the file extension could be set to "ulaw".
path **(Path)**	string	No	true	true	*None*	The path to the file that will hold the recording. If left blank, the file is assumed to be sent via ftp.
ftp_host **(FTP Host)**	string	No	true	true	*None*	The domain name of the host to ftp the recording. If left blank, the recording is assumed to be stored in a local file.
ftp_user **(FTP User)**	string	No	true	true	*None*	The user name to use while FTPing the recording.
ftp_password **(FTP Password)**	string	No	true	true	*None*	The password to use while FTPing the recording.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`ftp_path`
`(FTP Path)`
	string	No	`true`	`true`	*None*	The directory in which to FTP the recording.

`ftp_in_background`
`(FTP In Background)`
	boolean	Yes	`true`	`true`	`true`	Whether or not the FTP is to be performed in the background.

Notes:

- Nomatch events cannot be thrown in this voice element. Since all audio is recorded (except DTMF key presses), there is no reaction on spoken commands (including hotlinks).
- A noinput event is possible if the voice browser detects no audio once recording has started. If the input timeout has been reached, the noinput event is thrown.
- The path setting does not require a trailing slash. The voice element will determine the appropriate destination. The path may be specified in operating system specific format (for example, on Windows it might be specified as *C:\directory\subdirectory\* and on UNIX it might be */usr/local/directory/*).
- If **`terminate_on_DTMF`** is "false" or off, recording will stop only after the voice browser reaches the input timeout.
- Note that not all voice browsers support the ability to retain a recording if the caller hung up while making the recording.
- Some voice browsers may not accept all options provided for the **`file_type`** and **`mime_type`** settings. Check your voice browser documentation for information on supported audio types.
- It is important to ensure that VXML Server has permission to save audio files to the specified path.

Element Data

Name	Type	Notes
`filename`	string	This stores the filename of the recording (without the path).
`filepath`	string	This stores the path to the file holding the recording (including the filename).
`hungUpWhileRecording`	boolean	This stores a "true" if the caller hung up while making the recording, "false" if not.

Exit States

Name	Notes
max_noinput	The maximum number of noinput events has occurred. If the **max_noinput_count** is set to 0, this exit state will never occur.
done	The message was recorded.

Audio Groups

Record Capture

Name (Label)	Req'd	Max 1	Notes
initial_audio_group **(Initial)**	Yes	Yes	Played when the voice element first begins.
noinput_audio_group **(No Input)**	No	No	Played when a noinput event occurs.

Studio Element Folder: Record
Class Name: com.audium.server.voiceElement.record.MRecord

Chapter 24: Record_With_Confirm

The Record_With_Confirm voice element combines the functionality of the Record voice element with that of the MenuYesNo voice element. The voice element records the caller's voice, then prompts the caller to confirm that the recording is acceptable. The caller can then accept or reject the confirmation or ask to have the message replayed. If the caller accepts the recording, the voice element saves the file just as the Record voice element does. This voice element contains all settings and audio groups from both the Record and MenuYesNo voice elements, however audio groups that are found in both voice elements (nomatch, noinput, and help) are now named differently for them to be distinguished.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	true	both	The type of entry allowed for input during confirmation. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.
record_max_noinput_count **(Record Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during input capture. 0 = infinite noinputs allowed.
confirm_max_noinput_count **(Confirm Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during confirmation. 0 = infinite noinputs allowed.
confirm_max_nomatch_count **(Confirm Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	The maximum number of nomatch events allowed during confirmation. 0 = infinite nomatches allowed.
max_disconfirmed_count **(Max Disconfirmed Count)**	int ≥ 0	Yes	true	true	3	The maximum number of times a caller is allowed to reject a recording. 0 = infinite disconfirmations allowed.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
confirm_confidence_level **(Confirm Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.50	The confidence level threshold to use for the confirmation.
start_with_beep **(Start With Beep)**	boolean	Yes	true	true	true	Whether or not to play a beep before recording begins.
terminate_on_dtmf **(Terminate On DTMF)**	boolean	Yes	true	true	true	Whether or not the caller can end the recording by pressing a touchtone key.
keep_recording_on_hangup **(Keep Recording On Hangup)**	boolean	Yes	true	true	false	Whether or not the recording is stored if the caller hung up while making the recording or during the confirmation menu. Default = false.
max_record_time **(Max Record Time)**	int ≥ 0	Yes	true	true	180s	The maximum time that the recording is allowed to last. Possible values are standard time designations including both a non-negative number and a time unit, for example, 30s (for seconds) or 300ms (for milliseconds). Default = 180s.
final_silence **(Final Silence)**	int ≥ 0	Yes	true	true	4s	The interval of silence that indicates the end of speech. Possible values are standard time designations including both a non-negative number and a time unit, for example, 3s (for seconds) or 300ms (for milliseconds). Default = 4s.
replay **(Replay)**	boolean	Yes	true	true	false	Adds an option to replay the confirm initial audio groups.
filename **(Filename)**	string	No	true	true	*None*	The filename of the recording (without extension). If left blank, an auto-generated filename will be used.
file_type **(File Type)**	string enum	Yes	true	true	wav	This specifies the audio type of the file that will hold the recording. Possible values are: wav | vox | au | other.
mime_type **(Mime Type)**	string	Yes	true	true	*none*	This specifies the MIME type of the file that will hold the recording, if **file_type** is set to other.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes

`file_extension`
`(File Extension)`
	string	No	`true`	`true`	*None*	This specifies the file extension to use for the recorded file. A file extension different from the file type can be used. For example, with a mime type of vox, the file extension could be set to "ulaw".

`path`
`(Path)`
	string	No	`true`	`true`	*None*	The path to the file that will hold the recording. If left blank, the file is assumed to be sent via ftp.

`ftp_host`
`(FTP Host)`
	string	No	`true`	`true`	*None*	The domain name of the host to ftp the recording. If left blank, the recording is assumed to be stored in a local file.

`ftp_user`
`(FTP User)`
	string	No	`true`	`true`	*None*	The user name to use while FTPing the recording.

`ftp_password`
`(FTP Password)`
	string	No	`true`	`true`	*None*	The password to use while FTPing the recording.

`ftp_path`
`(FTP Path)`
	string	No	`true`	`true`	*None*	The directory in which to FTP the recording.

`ftp_in_background`
`(FTP In Background)`
	boolean	Yes	`true`	`true`	`true`	Whether or not the FTP is to be performed in the background.

Notes:

- The path setting does not require a trailing slash. The voice element will determine the appropriate destination. The path may be specified in operating system specific format (for example, on Windows it might be specified as *C:\directory\subdirectory\* and on UNIX it might be */usr/local/directory/*).
- If **`terminate_on_DTMF`** is "false" or off, recording will stop only after the voice browser reaches the input timeout.
- Note that not all voice browsers support the ability to retain a recording if the caller hung up while making the recording.
- Some voice browsers may not accept all options provided for the **`file_type`** and **`mime_type`** settings. Check your voice browser documentation for information on supported audio types.
- It is important to ensure that VXML Server has permission to save audio files to the specified path.

Element Data

Name	Type	Notes
`filename`	string	This stores the filename of the recording (without the path).
`filepath`	string	This stores the path to the file holding the recording (including the filename).
`confirm_confidence`	float	This is the confidence value of the utterance for the confirmation menu.
`hungUpWhileRecording`	boolean	This stores a "true" if the caller hung up while making the recording or the confirmation menu, "false" if not.

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.
`max_disconfirmed`	The maximum number of disconfirms has occurred. If the maximum disconfirm count is 0, this exit state will never occur.
`done`	The recorded message was confirmed.

Audio Groups

Record Capture

Name (Label)	Req'd	Max 1	Notes

`record_initial_audio_group`
`(Record Initial)`
	Yes	Yes	Played when the voice element first begins.

`record_noinput_audio_group`
`(Record NoInput)`
	No	No	Played when a noinput event occurs during recording.

Record Confirm

Name (Label)	Req'd	Max 1	Notes

`before_confirm_audio_group`
`(Before Confirm)`
	No	Yes	Played before the recording is played back. The recording will be played back after this audio group is done playing.

`after_confirm_audio_group`
`(After Confirm)`
	No	Yes	Played after the recording is played back. At least one of the two confirm prompts must be specified.

`confirm_nomatch_audio_group`
`(Confirm NoMatch)`
	No	No	Played when a nomatch event occurs during confirmation.

`confirm_noinput_audio_group`	No	No	Played when a noinput event occurs during confirmation.

Name (Label)	Req'd	Max 1	Notes
`(Confirm NoInput)`			

`confirm_help_audio_group`
`(Confirm Help)`
	No	No	Played when the caller asks for help during the confirmation menu. If not specified, help is treated as a nomatch by default.

`max_disconfirmed_audio_group`
`(Max Disconfirmed)`
	No	Yes	Played after the caller disconfirms the recorded entry, upon reaching the **`max_disconfirmed_count`**. The prompt should be about exiting with the **`max_disconfirmed`** exit state.

Studio Element Folder: Record

Class Name: `com.audium.server.voiceElement.record.MRecordWithConfirm`

Chapter 25: ReqICMLabel

The ReqICMLabel element allows a Call Studio script to pass caller input, Call Peripheral Variables, and External Call Context (ECC) variables to an ICM script. The ReqICMLabel must be inserted into a Call Studio script as a decision element. In Call Studio, the returned ICM label contains a result, which can be used by other elements in the same application, such as the Transfer or Audio element.

After the ReqICMLabel exits its done path, you can retrieve the values set by the ICM script by selecting the Element Data tab for the ReqICMLabel element. The element data value is {Data.Element.*ReqICMLabelElement*.result}. ReqICMLabelElement is the name of the ReqICMLabel element in the Studio script. The default name for this element is ReqICMLabel_<*n*>. For example, if you changed ReqICMLabel to GetICMLabel, the value returned from ICM would be {Data.Element.GetICMLabel.result}, where result is the variable of the ReqICMLabel element that contains the ICM label.

For more information on using the ReqICMLabel, refer to the *Configuration and Administration Guide for Cisco Unified Customer Voice Portal*.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
Call Peripheral Variables 1 – 10 **(callvar1 – callvar10)**	String	No	true	true	*None*	Call Peripheral Variables passed by the Studio script to the ICM Server. Each of these settings can be a maximum length of 210 characters. The ICM Server returns a name-value pair for up to 10 Call Peripheral Variables in a result. Any value that is placed in callvar<n> from a Call Studio script is returned unchanged, if the ICM Script does not change it.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
Call Peripheral Variables Return 1 - 10 (callvarReturn1 - callvarReturn10)	String	No	true	true	*None*	Call Peripheral Variables created upon the return of the ICM Label request, regardless of whether or not these variables are filled by the ICM Script. The reason we need two sets of these variables is to keep reporting the To ICM Call Peripheral Variables separate from what is returned from the ICM.
FromExtVXML0 - 3 (External VXML 0 – External VXML 3)	String Array	No	true	true	*None*	External Call Context (ECC) variables passed by the Studio script to the ICM Server. Each variable is a string of name-value pairs, separated by semicolons, for up to 4 external VXML variables. Each of these settings can be a maximum length of 210 characters.
ToExtVXML0 - 4 (External VXML 0 – External VXML 4)	String Array	No	true	true	*None*	External Call Context (ECC) variables received from the ICM script. The ICM Server returns a string of name-value pairs, separated by semicolons, for up to 5 external VXML variables.
Timeout	Integer	Yes	true	true	3000 (ms)	The number of milliseconds the transfer request waits for a response from the ICM Server before timing out. Note: This value can only be increased or decreased by increments of 500 ms.
caller_input (Caller Input)	String	No	true	true	*None*	This setting can be a maximum length of 210 characters. The value of this setting will be sent from VXML Server to ICM at runtime. Should a response from ICM be needed, the Call Peripheral Variables or ToExtVXML settings should be used.

Element Data

Name	Type	Notes
result	String	ICM Label returned from an ICM server.
callvar<n>	String	Call Peripheral Variables that the Studio scripts passes to the ICM Server. Valid Call Peripheral

Name	Type	Notes
		Variables are callvar1 – callvar10.
`callvarReturn<n>`	String	Call Peripheral Variables that the ICM script returns to the VXML Server. Valid Call Peripheral Variables are callvarReturn1 – callvarReturn10. For example, if an ICM script contains call peripheral variable 3 with the string value “CompanyName=Cisco Systems, Inc”, you can access the value of CompanyName that is returned by the ICM script by using: Data.Element.ReqICMLabelElement.callvarReturn3. The returned value is “Cisco Systems, Inc.”

Session Data

Name	Type	Notes
`name`	String	Value for a name-value pair contained in a ToExtVXML variable returned in the ICM label. You must know which name-value pairs are set in the ICM script to retrieve the correct value from the Call Studio script. For example, if an ICM script contains a user.microapp.ToExtVXML0 variable with the string value “CustomerName=Mantle”, specify Data.Session.CustomerName. If the same ICM script contains a user.microapp.ToExtVXML0 variable with the string value “BusinessType=Manufacturing”, you can access the customer business type returned by the ICM script by using Data.Session.BusinessType.

Exit States

Name	Notes
`done`	The element execution is complete and the value was successfully retrieved.
`error`	The element failed to retrieve the value.

Studio Element Folder: Cisco

Class Name: `com.cisco.cvp.vxml.custelem.ReqICMLabel`

Chapter 26: Subdialog Invoke

The Subdialog Invoke element initiates a subdialog invocation to another VoiceXML application, and handles passing data to and from the application. For the entire duration while a subdialog application is handling a call, the calling application waits in a dormant state for the subdialog to return. The goal of the Subdialog Invoke element is to allow voice applications to be invoked across multiple servers, as well as giving temporary control of the call to a voice application (such as flat VoiceXML and JSPs) created outside Call Studio.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
subdialog_uri (Subdailog URI)	string	yes	true	true	*none*	Specifies the URI of the subdialog to invoke. This may either be a relative or absolute URI, but must be accessible to the voice browser at runtime.
local_application (Local Application)	boolean	yes	true	true	*none*	Specifies whether or not the subdialog application is running on the same application server as the application in which the current element appears.
parameter (Parameter)	string	no	false	true	*none*	Holds the name and value of a parameter to pass to the subdialog. The format is the name of the parameter followed by an equal sign (=) followed by the value of the parameter. For example: "name=John Doe". The element will use the text up to the first equal sign as the name of the parameter and the remaining text as the value .
return_value (Return Value)	string	no	false	true	*none*	Holds the name of a return value from the subdialog. For example: "result". The names specified here must match the variable names returned by the subdialog. Return values will be stored as element data, in a variable of the name specified here.

Exit States

Name	Notes
Done	The element execution is complete

Class Name: com.audium.server.voiceElement.internal.SubdialogInvoke

Chapter 27: Subdialog Return

In most situations, the CVP Subdialog Return element (see Chapter 7: CVP Subdialog Return) should be used instead of this one, to offer full compatibility with ICM. However, there is one exception to this. If the voice application will *only ever* be called by a Subdialog Invoke element (i.e., never by ICM), then the Subdialog Start and Subdialog Return elements may be used instead. In this scenario, using this element allows an arbitrary number of return values to be retrieved from the subdialog, whereas the CVP Subdialog Return element allows only four.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
return_value (Return Value)	string	no	false	true	*none*	Optional return argument that holds a name/value pair to be returned to the calling application. The format should be: the name of the argument followed by an equal sign and the value of the argument. For example; "name=John Doe". The element will take the text up to the first equal sign to be the name of the argument and the text following the equal sign to the value.

Exit States

Name	Notes
done	The element execution is complete

Studio Element Folder: General
Class Name: com.audium.server.voiceElement.internal.DefaultSubdialogReturnElement

Chapter 28: Subdialog Start

In most situations, the CVP Subdialog Start element (see Chapter 8: CVP Subdialog Start) should be used instead of this one, to offer full compatibility with ICM. However, there is one exception to this. If the voice application will *only ever* be called by a Subdialog Invoke element (i.e., never by ICM), then the Subdialog Start and Subdialog Return elements may be used instead.

Data can be passed to the VoiceXML application either as HTTP parameters or VoiceXML parameters (using the <param> tag). In the first case (i.e. as HTTP parameters), Call Services will automatically create session data using the name of the data received. In the second case (i.e. as VoiceXML parameters), the Subdialog Start element must be configured appropriately in order for the data to be available as element or session data for the duration of the call session. For each data passed as a VoiceXML parameter, the repeatable "**Parameter**" setting must be configured with the same exact name as the data.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
Parameter (Parameter)	string	no	false	true	*none*	Holds the name of a parameter passed as input to the subdialog. It must match the exact value specified in the calling dialog. This is a repeatable setting, so multiple values can be specified.
Store As (Store As)	String	No	true	false	*Session Data*	Set to "Session Data" to store the listed parameters in Session data, or to "Element Data" to store them in Element data.

Exit States

Name	Notes
done	The element execution is complete

Studio Element Folder: General
Class Name: com.audium.server.voiceElement.internal.DefaultSubdialogStartElement

Chapter 29: Time

The Time voice element captures a time input from the caller. The time input can be entered using spoken inputs (including hours and minutes) or DTMF inputs (in the HHMM format). The captured value will be stored in element data as a five character string in the format HHMMX, where X is one of four possible values: "a" for AM, "p" for PM, "h" for a military time, or "?" for an ambiguous time. Using speech input, the time input may be spoken in natural language.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	*both*	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	*5s*	The maximum time length 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 300ms (for milliseconds). Default = 5s.
collect_max_noinput_count **(Time Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during time input capture. 0 = infinite noinputs allowed.
collect_max_nomatch_count **(Time Max NoMatch Count)**	int ≥ 0	Yes	true	false	3	The maximum number of nomatch events allowed during time input capture. 0 = infinite nomatches allowed.
collect_confidence_level **(Time Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.40	The confidence level threshold to use during time capture.
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 grammars of the current Time element will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
secure_logging **(Secure Logging)**	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Time element. If set to true,

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
						the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX[†] and nbestInterpretationX[†]. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.
maxnbest (Maxnbest)	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

[†]Refer to the Element Data table for information about nbestUtteranceX and nbestIInterpretationX.

Element Data

Name	Type	Notes
Value	string	The time stored in the HHMMX format, where X is one of a, p, h, or ?.
value_confidence	float	This is the confidence value of the captured time utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
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	string	This set of element data stores the input modes of captured n-best utterances.

nbestInputmode2 … **nbestInputmodeX**		

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 time capture was completed.

Audio Groups

Time Capture

Name (Label)	Req'd	Max 1	Notes
collect_initial_audio_group **(Time Initial)**	Yes	Yes	Played when the voice element first begins.
collect_noinput_audio_group **(Time NoInput)**	No	No	Played when a noinput event occurs. The noinput event count corresponds to the audio group count.
collect_nomatch_audio_group **(Time NoMatch)**	No	No	Played when a nomatch event occurs. The nomatch event count corresponds to the audio group count.
collect_help_audio_group **(Time Help)**	No	No	Played when a help event occurs. The help event count corresponds to the audio group count. If not specified, a help event is treated as nomatch.

End

Name (Label)	Req'd	Max 1	Notes
done_audio_group **(Done)**	No	Yes	Played after the time capture is completed. If not specified, no audio will be played.

Studio Element Folder: Date & Time
Class Name: `com.audium.server.voiceElement.time.MBasicTime`

Chapter 30: Time_With_Confirm

The Time_With_Confirm voice element captures a time input from the caller, and presents a confirmation menu allowing the caller to either accept their entry or re-enter the time. The time input can be entered using spoken inputs (including hours and minutes) or DTMF inputs (in the HHMM format). The captured value will be stored in element data as a five character string in the format HHMMX, where X is one of four possible values: “a” for AM, “p” for PM, “h” for a military time, or “?” for an ambiguous time. Using speech input, the time input may be spoken in natural language.

Settings

Name (Label)	Type	Req’d	Single setting value	Substitution allowed	Default	Notes
inputmode **(Input Mode)**	string enum	Yes	true	false	*both*	The type of entry allowed for input. Possible values are: voice | dtmf | both.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	*5s*	The maximum time length 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 300ms (for milliseconds). Default = 5s.
collect_max_noinput_count **(Time Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during time input capture. 0 = infinite noinputs allowed.
collect_max_nomatch_count **(Time Max NoMatch Count)**	int ≥ 0	Yes	true	false	3	The maximum number of nomatch events allowed during time input capture. 0 = infinite nomatches allowed.
confirm_max_noinput_count **(Confirm Max NoInput Count)**	int ≥ 0	Yes	true	true	3	The maximum number of noinput events allowed during time input confirmation. 0 = infinite noinputs allowed.
confirm_max_nomatch_count **(Confirm Max NoMatch Count)**	int ≥ 0	Yes	true	false	3	The maximum number of nomatch events allowed during time input confirmation. 0 = infinite nomatches allowed.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
max_disconfirmed_count **(Max Disconfirmed Count)**	int ≥ 0	Yes	true	false	3	The maximum number of times a caller is allowed to disconfirm a captured input. 0 = infinite disconfirmations allowed.
collect_confidence_level **(Time Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.40	The confidence level threshold to use during time capture.
confirm_confidence_level **(Confirm Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.50	The confidence level threshold to use during confirmation.
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 grammars of the current Time_With_Confirm element (the builtin time and boolean grammars) will be enabled for the duration of the element. Otherwise all active grammars will be enabled.
secure_logging **(Secure Logging)**	boolean	Yes	true	true	false	Whether or not to enable logging of potentially sensitive data of the Time_With_Confirm element. If set to true, the following potentially sensitive data of the element will not log: utterance, interpretation, value, nbestUtteranceX[†] and nbestInterpretationX[†]. Instead, the above will be logged as the field name appended with the suffix "_secureLogging" and with the value "*****", for example nbestUtterance1_secureLogging,*****.
maxnbest **(Maxnbest)**	int ≥ 1	Yes	true	true	1	The maximum number of speech recognition results that can be generated per voice input.

[†]Refer to the Element Data table for information about nbestUtteranceX and nbestInterpretationX.

Element Data

Name	Type	Notes
Value	string	The time stored in the HHMMX format, where X is one of a, p, h, or ?.
value_confidence	float	This is the confidence value of the captured time utterance. When n-best recognition is enabled, this stores the confidence score of the top hypothesis in the n-best list.
confirm_confidence	float	This is the confidence value of the captured confirm utterance.

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.

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.
max_disconfirmed	The maximum number of disconfirmation has occurred. If the **max_disconfirmed_count** is set to 0, this exit state will never occur.
done	The time captured is confirmed.

Audio Groups

Time Capture

Name (Label)	Req'd	Max 1	Notes
collect_initial_audio_group **(Time Initial)**	Yes	Yes	Played when the voice element first begins.
collect_noinput_audio_group	No	No	Played when a noinput event occurs during time input. The

Name (Label)	Req'd	Max 1	Notes
(Time NoInput)			noinput event count corresponds to the audio group count.
collect_nomatch_audio_group (Time NoMatch)	No	No	Played when a nomatch event occurs during time input. The nomatch event count corresponds to the audio group count.
collect_help_audio_group (Time Help)	No	No	Played when a help event occurs during time input. The help event count corresponds to the audio group count. If not specified, a help event throws a nomatch event.

Time Confirm

Name (Label)	Req'd	Max 1	Notes
confirm_initial_audio_group (Confirm Initial)	Yes	Yes	Played when confirmation of the captured time first begins.
confirm_nomatch_audio_group (Confirm NoMatch)	No	No	Played when a nomatch event occurs during time confirmation. The nomatch event count corresponds to the audio group count.
confirm_noinput_audio_group (Confirm NoInput)	No	No	Played when a noinput event occurs during time confirmation. The noinput event count corresponds to the audio group count.
confirm_help_audio_group (Confirm Help)	No	No	Played when a help event occurs during time confirmation. The help event count corresponds to the audio group count. If not specified, by default help throws a nomatch.
disconfirmed_audio_group (Disconfirmed)	No	No	Played after the caller disconfirms a time entry captured.

End

Name (Label)	Req'd	Max 1	Notes
yes_audio_group (Yes)	No	Yes	Played after the caller chooses the "yes" option. If not specified, no audio will be played when this option is chosen.

Studio Element Folder: Date & Time
Class Name: `com.audium.server.voiceElement.time.MBasicTimeWithConfirm`

Chapter 31: Transfer

The **Transfer** voice element performs a call transfer to a phone number specified by a configuration setting. Depending on how the voice browser is configured, the call transfer can be a bridge transfer or a blind transfer. For a bridge transfer, the voice browser makes an outbound call while maintaining the original call and acts as a bridge between the two calls. The advantage of this is that once the secondary call ends, the original call can still continue with the IVR. The disadvantage is that two separate phone lines are used. For a blind transfer, the voice browser makes an outbound call and when connected, links the original call to the new caller through the use of a telephony switch. At this point, the voice browser (and as a result VXML Server) is no longer in control of the call. Blind transfers involve only one line.

The Transfer element defines exit states for the different ways bridge transfers can end such as the person being called hung up, there was no answer, there was a busy signal, or some other phone-related error occurred. Since blind transfers take the call away from the voice browser and VXML Server, a Transfer element performing a blind transfer would never return an exit state. Instead, an special event would be thrown by the voice browser, caught in the root document for the call, and VXML Server would terminate the session by interrupting the Transfer element.

The number to transfer to can be any phone number allowed by the voice browser telephony provider (some may place restrictions on outbound dialing). Please note that different voice browsers may or may not accept certain kinds of phone numbers. Check your voice browser documentation for specific requirements and restrictions for call transfer.

Settings

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
transfer_destination (Transfer Destination)	string	Yes	true	true	*None*	The phone number to transfer to. It may contain non-numerical characters to allow support for phone extensions.
destination_type (Destination Type)	string	No	true	True	tel	The type of transfer destination to which the voice element is to connect. Possible values are: tel | sip.
connect_timeout (Connect Timeout)	int > 0	Yes	true	true	60s	The maximum time that the voice element is allowed to wait for an answer, before exiting with a "noanswer" exit state. Possible values are standard time designations including both a number and a time unit, for example, 10s (for seconds) or 300ms (for milliseconds). Default = 60s.

Name (Label)	Type	Req'd	Single setting value	Substitution allowed	Default	Notes
max_transfer_time **(Max Transfer Time)**	int > 0	Yes	true	true	0s	The maximum duration that the transfer is allowed to last. Possible values are standard time designations including both a non-negative number and a time unit, for example, 30s (for seconds) or 300ms (for milliseconds). Default = 0s (means no limit). This setting only applies when bridge is set to true.
bridge **(Bridge)**	binary	Yes	true	true	false	Determines whether the application remains connected to the caller after the transfer is initiated. Possible values are: true | false. Default = false. When set to false (i.e. a blind transfer), the application redirects the caller to the callee without remaining in the connection; the transfer outcome is completely unsupervised. When set to true (i.e. a bridge transfer), the application stays connected to the caller and adds the callee to the connection for the duration of the transferred call.
transfer_audio **(Transfer Audio)**	string	No	true	true	*None*	The URI location of the audio file to be played while connecting the call.
aai **(Application-to-application Information)**	string	No	true	true	*None*	A string containing Application-to-Application Information data to be sent to an application on the far-end.

Element Data

Name	Type	Notes
result	string	The value returned by the transfer field. This is dependent on the voice browser.

Exit States

Name	Notes
busy	The number was busy.
noanswer	There was no answer.
phone_error	There was some sort of phone-related error.
done	The call transfer completed successfully.

Notes:

- Hosting voice browsers may disable call transfers for developer accounts. You should verify with your provider that transfer is enabled for your application.
- Some voice browsers use a code to indicate which call transfers will be allowed. This code appears before the phone number.
- Some voice browsers support the inclusion of an extension in the phone number so that the system can transfer to a particular extension. It is up to the developer to pass this voice element a string containing the appropriate format. Check the platform specific documentation for support of extension dialing in transfer.

Audio Groups

Transfer Audio

Name (Label)	Req'd	Max 1	Notes
initial_audio_group **(Initial)**	No	Yes	Played to introduce the transfer. If there is none, the transfer occurs immediately.
busy_audio_group **(Busy)**	No	Yes	Played when there is a busy signal, right before the voice element exits with the "busy" exit state.
noanswer_audio_group **(No Answer)**	No	Yes	Played when there is no answer, right before the voice element exits with the "noanswer" exit state.
phone_error_audio_group **(Phone Error)**	No	Yes	Played when there is some kind of phone-related error, right before the voice element exits with the "phone_error" exit state.

End

Name (Label)	Req'd	Max 1	Notes
done_audio_group **(Done)**	No	Yes	Played when the call transfer completes with the party called hanging up and the caller staying on the line.

Studio Element Folder: Call Control
Class Name: `com.audium.server.voiceElement.transfer.MTransfer`

Chapter 32: Yes_No_Menu

The Yes_No_Menu voice element presents a yes/no menu. It can be configured to accept DTMF entry (1 for yes and 2 for no) or spoken input ("yes" or "no" and other synonymous utterances, however this is dependent on the voice browser). There is an optional feature that allows the word "replay" to be spoken (or DTMF button 3) that replays the **initial_audio_group**. The voice element uses the browser specific VoiceXML builtin grammar for the boolean field type. A separate exit state exists for the yes and no choices (there is no exit state for replay since dialog execution is still contained within the confines of the voice element).

Settings

Name (Label)	Type	Req'd	Single setting value	Sugstitution allowed	Default	Notes
max_noinput_count **(Max NoInput Count)**	int ≥ 0	Yes	true	true	3	0 = infinite noinputs allowed.
max_nomatch_count **(Max NoMatch Count)**	int ≥ 0	Yes	true	true	3	0 = infinite nomatches allowed.
inputmode **(Input Mode)**	string enum	Yes	true	false	both	The type of entry allowed for input (using speech recognition, DTMF entry, or both). Possible values are: voice | dtmf | both.
replay **(Replay)**	boolean	Yes	true	true	false	True adds a "replay" option which replays the initial prompt.
noinput_timeout **(Noinput Timeout)**	string	Yes	true	true	5s	The maximum time length 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 300ms (for milliseconds). Default = 5s.
confidence_level **(Confidence Level)**	decimal (0.0 – 1.0)	Yes	true	true	0.50	The confidence level threshold to use.
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 boolean builtin grammar will be enabled for the duration of the element. Otherwise all active grammars will be enabled.

Element Data

Name	Type	Notes
`value`	string	This is the value chosen by the caller. Can be: “yes” or “no”.
`value_confidence`	float	This is the confidence value of the utterance.

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.
`yes`	The utterance was recognized as “yes”.
`no`	The utterance was recognized as “no”.

Notes:

- The replay option, when activated, resets all the event counts (noinput and nomatch).

Audio Groups

Yes / No Capture

Name (Label)	Req’d	Max 1	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 asked for help. If not specified, by default help is treated as a nomatch.

End

Name (Label)	Req’d	Max 1	Notes

`yes_audio_group`
`(Yes)`
	No	Yes	Played when the caller chose the "yes" option. If not present, no audio will play when this option is chosen.

Studio Element Folder: Menu

Class Name: `com.audium.server.voiceElement.menu.MYesNoMenu`

Index