Dynamic Element Configurations

Configurable voice, action, and decision elements used in an application must have configurations. Usually, the configuration will be fixed, that is, it acts the same every time a caller visits it. In this case, the configuration itself exists as an XML file stored on the system. Builder for Call Studio creates this file when the application is deployed. Programming is required when a dynamic element configuration is desired, that is, one which is thrown at runtime each time a caller visits it.

The manner in which configurations are used warrants closer examination. A configuration functions based on its pre-built elements. Since configurable elements are constructed with Java, the configuration for the element must be given to it in the form of a Java class. The API provides a set of Java classes that encapsulate an entire element configuration. The Java classes are Java expressions of the visual Builder for Call Studio's Configuration Pane consisting of three tabs: *General*, *Settings*, and *Data* for action and decision elements, and a fourth tab, *Audio*, for voice elements.

When a static configuration is used, this information is stored as an XML file thrown by Builder for Call Studio. VXML Server converts this XML file to one of the Java configuration classes and then passed it on to the element.

A dynamic configuration adds an additional step in this process. Once VXML Server loads the static representation of the configuration (known as the base configuration), it will pass this to the dynamic configuration Java class or URI for modification, instead of passing it directly to the element. The class or URI adds to or changes the base configuration depending on the application business logic and returns a complete configuration. VXML Server then passes this new configuration to the element.

Java API Use

Dynamic voice, action, and decision element configurations are constructed in the Java API by implementing the Java interfaces `VoiceElementInterface`, `ActionConfigInterface` and `DecisionConfigInterface` respectively, all found in the `com.audium.server.proxy` package.

Note The name of the voice element interface is not consistent with the others due to backwards compatibility concerns. Each of these interfaces contains a single method named `getConfig` that receives three arguments:

- The name of the element as a `String`.

- An instance of `ElementAPI` or `ActionAPI` (for dynamic action element configurations). These classes belong to the Session API and are used to access session information. (See Session API for more information on this API.)
- An instance of `VoiceElementConfig`, `ActionElementConfig` or `DecisionElementConfig` (found in the `com.audium.server.xml` package) that contains the base configuration for the element (or `null` if there is no base configuration).

The method must return an instance of the configuration object (`VoiceElementConfig`, `ActionElementConfig` or `DecisionElementConfig`). This can be a modified version of the object passed as input to the method or one built from scratch. It is expected that should an unrecoverable error occur, the dynamic configuration class should throw an `AudiumException`.

Due to the fact that most dynamic configurations involve only a few changes to the static configuration, obtaining a base configuration as input to the method saves significant coding effort since the dynamic configuration class simply needs to modify this object in order to create the final configuration object then return it.

All three configuration classes extend a common base class, `ElementConfig`. This class defines those features common to all three element configurations: settings, element and session data created, custom log content, and associating the call with a UID. `ActionElementConfig` and `DecisionElementConfig` are essentially identical, separate classes are used for design considerations and for possible future differentiation. `VoiceElementConfig`, however, expands upon the `ElementConfig` class by introducing voice element only features: local hotlinks, VoiceXML properties and audio groups. The three configuration classes allow the developer to obtain everything about a configuration as well as change or add to the configuration in any way.

To handle audio groups, `VoiceElementConfig` introduces inner classes that define an audio group (`AudioGroup`) and a generic audio item (`AudioItem`). Two additional inner classes define audio item types that extend the `AudioItem` class to define a Say It Smart audio item (`SayItSmart`) and a static audio item (`StaticAudio`). The `AudioGroup` class encapsulates any number of `AudioItem` objects of either type. A developer can create new audio groups separately and call a method in `VoiceElementConfig` to add the audio group to the configuration, or an existing `AudioGroup` object can be obtained, modified, and then reinserted into the configuration.

To handle local hotlinks, which are supported on voice elements only and add page-scoped VoiceXML links to the pages generated by the voice element, `VoiceElementConfig` introduces an inner class called `LocalHotlink`.

The Javadocs provide much more detail regarding these classes and their methods.

Remote Execution of Dynamic Element Configuration

For remote execution of the Dynamic Element Configuration, choose the option **`URI`** in drop-down for the dynamic configuration option.

The following syntax for URI is to be used:

For HTTP and RPC call:
`remote://system/?classurl=<fully_qualified_java_class_path>`

For example: example- remote://system/?classurl=com.cisco.cvp.callstudio.Action.CustomVoiceConfig

This is valid for Voice, Decision and Action Elements dynamic configuration. `remote://system` indicates that the configurations will be fetched from the **Remote Url Settings** property tab which is application-specific.

Note If direct remote server URI is provided, then that `IP:Port` will be used and not fetched from the **Remote Url Settings** property tab.

For example:
`http://<IP>:<Port>/<target_path>/?classurl=<fully_qualified_java_class_path>`.

Add Events in the Element Configuration to handle any particular or general exception gracefully.

XML API Use

Dynamic element configurations using the XML API send four HTTP POST arguments to the URI specified:

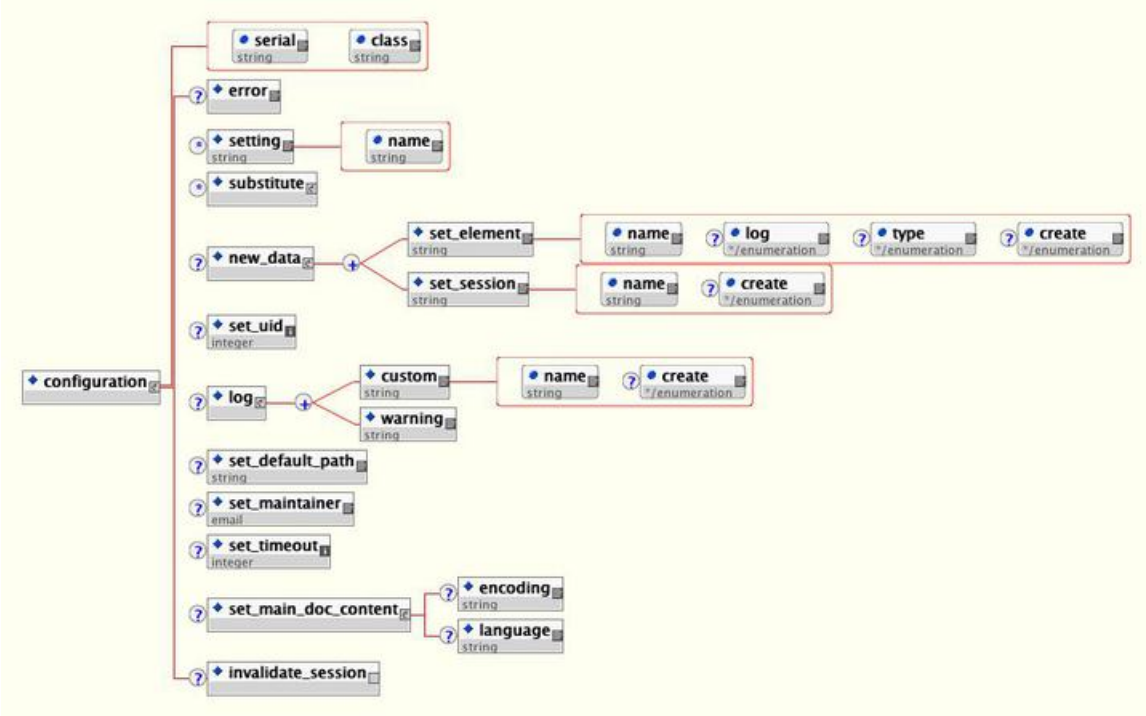
- **name** – The name of the element whose configuration is dynamic as a `string`.
- **inputs** – One of the standard arguments passed to all components utilizing the XML API as described in Session API.
- **settings** – One of the standard arguments passed to all components utilizing the XML API as described in Session API.
- **defaults** – The base configuration for the element represented as an XML document. If there is no base configuration, this argument is not included. There are two possible DTDs for this argument. One is used if the dynamic configuration is for a voice element and the other is if the dynamic configuration is for decision and action elements.

The response must contain the final configuration to use, which follows the same DTD as the base configuration XML document. Incidentally, this DTD is the same one used for the fixed element configuration XML files created by Builder for Call Studio.

Decision and Action Element Configuration DTD

The following figure shows the DTD for decision and action element configurations sent in the argument *defaults*. The DTD for decision element configurations is defined in the file `DecisionElementConfiguration.dtd` and the DTD for action element configurations is defined in the file `ActionElementConfiguration.dtd`, both located in the VXML Server `dtds` folder. Each DTD is stored as a separate file despite being syntactically identical, to allow for future divergence.

Figure 1: Decision and Action DTD



The tags in this XML document are:

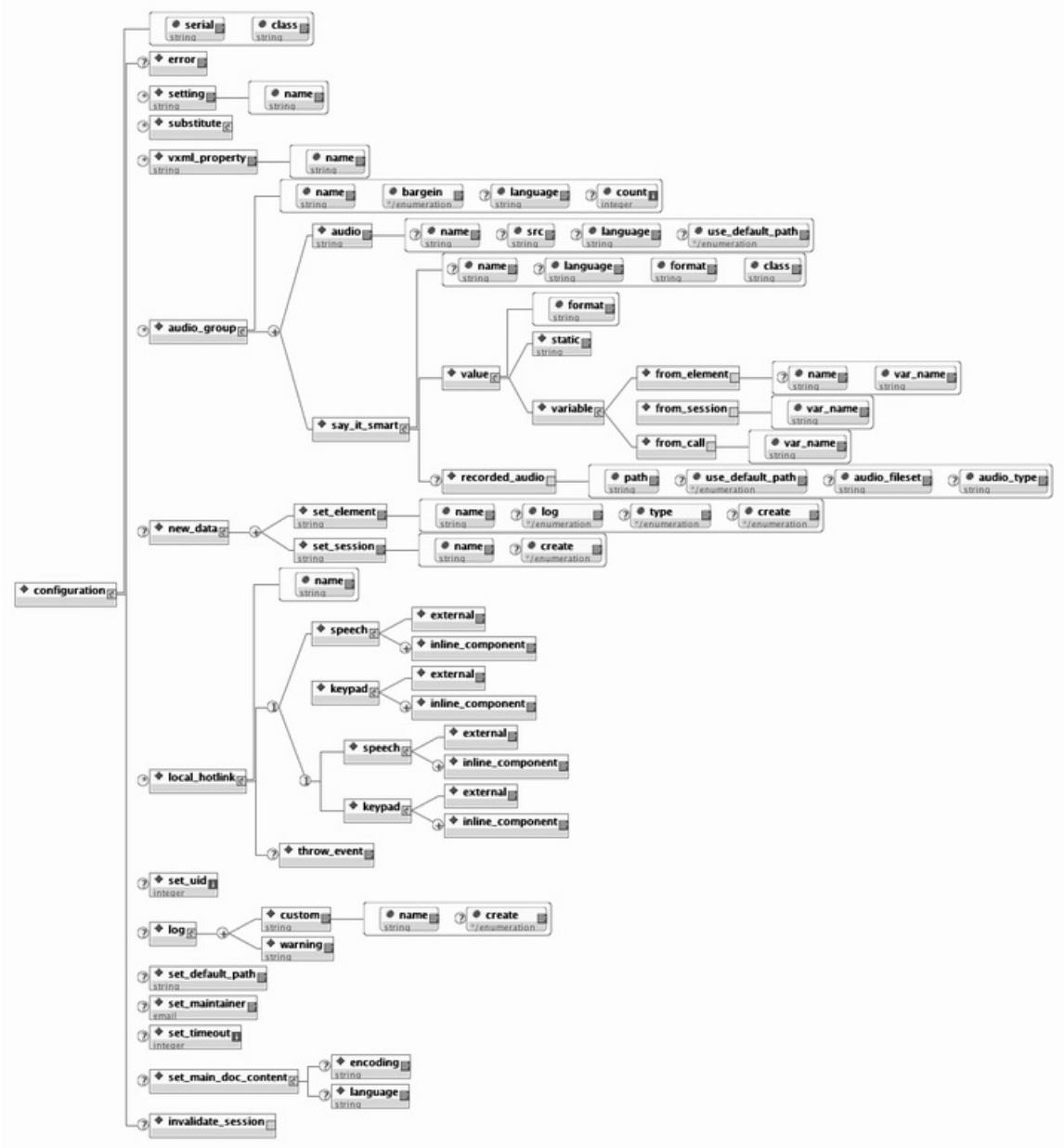
- **configuration** – The root tag. The `class` attribute refers to the Java class defining the configurable action or decision element whose configuration is being dynamically produced. Refer to the Element Specifications for Cisco Unified CVP VXML Server and Unified Call Studio document for the full Java class names of all Unified CVP elements. The `serial` attribute is used by Call Studio and can be safely ignored here.
- **error** – This tag reports to VXML Server that an error occurred while running the dynamic configuration. VXML Server will then throw an exception whose message is contained in the `<error>` tag. This allows the XML API to throw exceptions just as the Java API does.
- **setting** – This tag holds an element setting, the name appearing in the name attribute and the value of the setting contained within the `<setting>` tag. It is repeated for each setting included in the base configuration. No `<setting>` tags appear if the base configuration contains no settings or the element itself defines no settings.
- **substitute** – This tag holds information on substitution. Substitution is typically used in static configurations and since static and dynamic configuration XML documents share the same DTDs, it appears here. Substitution usually would not be used with dynamic configurations. The substitution tag contents are fully described in Substitution XML Format, on page 8.
- **new_data** – This tag holds the element and session data this dynamic element configuration is to create. Any number of `<set_element>` and `<set_session>` tags can appear, one for each element and session data variable to be created. The `log` attribute of `<set_element>` sets whether the value of the variable is stored in the activity log. The optional `type` attribute is used to specify the data type of the variable and can be *string*, *int*, *float*, or *boolean*. The `create` attribute found in both tags determines when the variable is created, before the element is entered (*before_enter*), or after the element exits (*after_exit*).

- **set_uid** – This tag is used to associate the call with a UID in the user management system. The content of the tag should be the integer UID.
- **log** – This tag is used to trigger logger events when this dynamic configuration is run. Any number of `<custom>` tags can appear, denoting the triggering of a custom event. The `name` attribute holds the name of the data, and the `<custom>` tag encapsulates the value. Any number of `<warning>` tags can appear, denoting the triggering of a warning event. The `<warning>` tag encapsulates the warning message.
- **set_default_path** – This tag is used to change the default audio path from this point onwards for this call.
- **set_maintainer** – This tag is used to change the maintainer e-mail address from this point onwards for this call.
- **set_timeout** – This tag allows the timeout length set for this session to be changed. The contents of the tab must be an integer representing the number of minutes in the timeout.
- **set_main_doc_content** – This tag enables the encoding and language settings for the application to be changed for this call. The `<language>` tag content is formatted according to the specification for using languages in VoiceXML (for example, *en-US*). The `<encoding>` tag content is formatted according to the specification for encoding XML pages (for example, *UTF-8*).
- **invalidate_session** – This tag, if included in the XML, will prompt VXML Server to invalidate the call session it retains in memory, call the end of call class or URI (if defined), and free up the VXML Server port utilized by the call. The session is invalidated only after the dynamic configuration method is run. This tag is rarely used and would be needed in a few circumstances where some external process takes the call away from VXML Server such as when using a CTI system to transfer the call to an agent.

Voice Element Configuration DTD

The following figure shows the DTD diagram for the voice element configuration XML document sent in the argument *defaults*. The DTD is defined in the file `VoiceElementConfiguration.dtd` found in the VXML Server `dtds` folder.

Figure 2: Voice Element Configuration DTD



The tags in this XML document are:

- **configuration** – The root tag. The `class` attribute refers to the Java class defining the configurable voice element whose configuration is being dynamically produced. Refer to the Element Specifications for Cisco Unified CVP VXML Server and Unified Call Studio document for the full Java class names of all Unified CVP elements. The `serial` attribute is used by Call Studio and can be safely ignored here.
- **error** – This tag reports to VXML Server that an error occurred while running the dynamic configuration. VXML Server will then throw an exception whose message is contained in the `<error>` tag. This allows the XML API to throw exceptions just as the Java API does.

- **setting** – This tag holds an element setting, the name appearing in the `name` attribute and the value of the setting contained within the `<setting>` tag. It is repeated for each setting included in the base configuration. No `<setting>` tags appear if the base configuration contains no settings or the element itself defines no settings.
- **substitute** – This tag holds information on substitution. Substitution is typically used in static configurations and as static and dynamic configuration XML documents share the same DTDs, it appears here. Substitution usually would not be used with dynamic configurations. The substitution tag contents are fully described in the section entitled Substitution XML Format at the end of this chapter.
- **vxml_property** – This tag holds a VoiceXML property, the name appearing in the name attribute and the value of the property contained within the `<vxml_property>` tag. It is repeated for each VoiceXML property referred to in the base configuration.
- **audio_group** – This tag holds all the audio items for a single audio group. Attributes to `<audio_group>` set its name, bargein preference and count (for those audio groups that can have counts greater than 1), and the language that it encapsulates. Each audio item is represented as a single `<audio>` or `<say_it_smart>` tag.
  - The `<audio>` tag defines a name for the audio item, the source of the audio file (optional if no audio file is being referenced), whether to use the default audio path (the `use_default_path` attribute may be *true* or *false*), and encapsulates the TTS backup message.
  - The `<say_it_smart>` tag's attributes define the name of the audio item, the output format to represent the data, and Java class name of the Say It Smart plugin. Its contents encapsulate a `<value>` tag representing either a static value or a value from a variable. The `format` attribute of `<value>` defines the input format of the data. The `<variable>` tag contains tags for obtaining the data from element data, session data or call data. The `var_name` attribute can contain the following values: *ani*, *dnis*, *iidigits*, *uui*, *start_date*, *start_time*, and *application_name*.

Note You can avoid using the `<variable>` tag by referring to a substitution string in the contents of the `<value>` tag. This also allows for the substitution of content in addition to element, session, and call data. The `<variable>` tag remains for backwards compatibility and for those not willing to use substitution.

- **new_data** – This tag holds the element and session data this dynamic element configuration is to create. Any number of `<set_element>` and `<set_session>` tags can appear, one for each element and session data variable to be created. The `log` attribute of `<set_element>` sets whether the value of the variable is stored in the activity log. The optional `type` attribute is used to specify the data type of the variable and can be *string*, *int*, *float*, or *boolean*. The `create` attribute found in both tags determines when the variable is created, before the element is entered (*before_enter*), or after the element exits (*after_exit*).
- **local_hotlink** – This tag is used for local hotlink configurations. The `name` attribute defines the local hotlink's name and must be unique within the element configuration.
  - The child `<speech>` tag indicates whether the inline or external speech grammar was set for this hotlink. The `<external>` tag should contain the URI to the external speech grammar. The `<inline_component>` tags encapsulate each utterance that activates the hotlink.
  - The child `<keypad>` tag indicates whether the inline or external DTMF grammar was set for this hotlink. The `<external>` tag should contain the URI to the external DTMF grammar. The `<inline_component>` tags encapsulate each DTMF entry that activates the hotlink.

  - The optional child `<throw_event>` tag is used if this hotlink throws an event.

- **set_uid** – This tag is used to associate the call with a UID in the user management system. The content of the tag should be the integer UID.
- **log** – This tag is used to trigger logger events when this dynamic configuration is run. Any number of `<custom>` tags can appear, denoting the triggering of a custom event. The name attribute holds the name of the data, and the `<custom>` tag encapsulates the value. Any number of `<warning>` tags can appear, denoting the triggering of a warning event. The `<warning>` tag encapsulates the warning message.
- **set_default_path** – This tag is used to change the default audio path from this point onwards for this call.
- **set_maintainer** – This tag is used to change the maintainer e-mail address from this point onwards for this call.
- **set_timeout** – This tag allows the timeout length set for this session to be changed. The contents of the tab must be an integer representing the number of minutes in the timeout.
- **set_main_doc_content** – This tag allows the encoding and language settings for the application to be changed from this point onwards for this call. The `<language>` tag content is formatted according to the specification for using languages in VoiceXML (for example, *en-US*). The `<encoding>` tag content is formatted according to the specification for encoding XML pages (for example, *UTF-8*).
- **invalidate_session** – This tag, if included in the XML, will prompt VXML Server to invalidate the call session it retains in memory, call the end of call class or URI (if defined), and free up the VXML Server port utilized by the call. The session is invalidated only after the method of the dynamic configuration is completed. This tag is rarely used and would be needed in a few circumstances where some external process takes the call away from VXML Server such as when using a CTI system to transfer the call to an agent.

Substitution XML Format

The DTD for element configuration XML documents contain a tag `<substitute>` that is used to define substitution. Substitution is the process of constructing a value from a combination of static and dynamic content. It is used as a way for a developer to use dynamic content in an element configuration without having to resort to a dynamic configuration. Substitution can be used throughout an element's configuration such as settings, audio, VoiceXML properties, etc.

See User Guide for Cisco Unified CVP VXML Server and Unified Call Studio for more information on substitution.

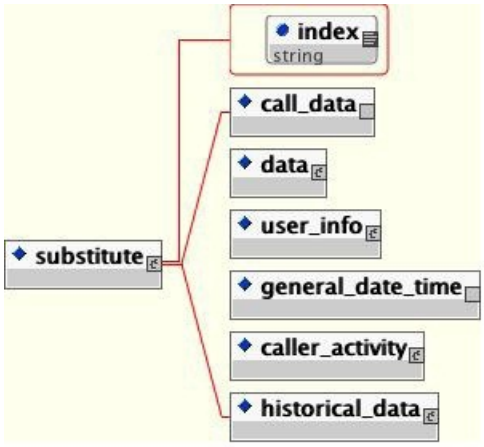
Since the DTDs of the documents returned by the XML API are the same as those for static element configurations produced by Builder for Call Studio, dynamic configurations may also utilize substitution. Using substitution in dynamic configurations, however, makes little sense as the dynamic configuration is produced by programming code which could just as easily set the appropriate value rather explicitly rather than assemble it using substitution. To be comprehensive, this section briefly describes the contents of the `<substitute>` tag.

A value for a setting, audio source, audio TTS or any other configuration option that supports substitution contains static content combined with integer values encapsulated by braces. When this format is detected, VXML Server knows to replace (substitute) the parts encapsulated in braces with the dynamic data. For example, `http://{0}/grammar/{1}` as a value for a setting indicates to substitute some dynamic content for

"{0}" and "{1}", where the indices are used for uniqueness (the same index can be used multiple times in the same value or in separate values if applicable).

This is where the `<substitute>` tag comes in. Each `<substitute>` tag specifies what dynamic data to substitute for a particular number surrounded by braces. The `index` attribute must be an integer that matches the number to substitute. A diagram of what it can contain is shown in the following figure.

Figure 3: Content for the Substitute Tag



The content to substitute can be one of the six possible tags:

- **call_data** – Represents call information such as the ANI.
- **data** – Represents element or session data.
- **user_info** – Represents information about the user associated with the call (available only when the user management system is turned on and the call is associated with a particular UID).
- **general_date_time** – Represents the current time or the start of the call.
- **caller_activity** – Represents the activity taken by the caller in this call.
- **historical_data** – Represents past actions taken by the user associated with this call (available only when the user management system is turned on and the call is associated with a particular UID).

These tags are identical to tags of the same name used within the XML decision format. These tags are described in the User Guide for Cisco Unified CVP VXML Server and Unified Call Studio.

For example, in the above situation where a setting has the value `http://{0}/grammar/{1}`, the following substitute tag can represent index 0 coming from element data:

```
<substitute index="0">
   <data>
       <element name="AnElementName" variable="SomeValue"/>
   </data>
</substitute>
```