Call End Action

VXML Server can be configured to run code once a call has ended. Unlike the call start action, the call end action can occur at any time in the call and there are several different situations that would trigger the call end action. The following lists those situations:

- The caller hangs up normally.
- The application hangs up on the caller. This includes any errors that are caught by the system that yield a hang-up, or places in the application when the call's purpose is over.
- A blind telephony transfer takes place. Blind transfers connect the caller with the party called using telephony switching equipment, removing the voice browser (and hence VXML Server) from the calling context. Even though the physical phone call continues, the role of the automated system ends and so for it, the call has ended.

  Note The availability of blind transfers is determined by the voice browser's functionality and network setup.

- The application performs a transfer to another Unified CVP application. This is not a telephony transfer, but the results are very similar. Since the call leaves the source application, it is considered the end of the *call* to that application.
- VXML Server times out a session. This occurs only rarely, it would be seen only when some error prevented VXML Server from receiving a request in the middle of a call and it waited a certain amount of time before timing out the session. This could be due to a voice browser going down or if the request coming from the voice browser is malformed and VXML Server cannot determine which call that request was supposed to be for.
- The session is invalidated by a custom element. Standard and configurable elements have the ability to invalidate the session for situations where some process ends the call that would not prompt VXML Server to be notified that the call ended. This functionality is described in more detail in the chapters on custom elements.

The call end action can be implemented with either the Java API or the XML API. Unlike the call start action using the XML API, the call end action does not have an option to perform it in the background. In fact, one need not worry about performing time consuming tasks in the call end action because it will not affect the performance of the call since it has ended. One must still be careful not to perform tasks that maximize CPU usage since that would aversely affect the handling of other calls.

Like the call start action, the call end action can modify the session such as creating session data or changing the default audio path, though these actions would not make sense as there is no more call flow to visit. The call end action can access everything that occurred within the call, including how the call ended (hangup, call transfer, etc.) This is useful for activities such as creating CDR records which must list everything a caller did.

A unique feature of the call end action is to optionally send back a final VoiceXML page to the voice browser. Some voice browsers will actually interpret a VoiceXML page sent back in response to a request triggered by a disconnect or hang-up event. Since the caller is no longer interacting with the IVR, this page would obviously only be useful for limited functionality that had nothing to do with interacting with the caller, such as executing `<log>` tags.

Note This final page applies only to when the caller hangs up on the application or the application hangs up on the caller.

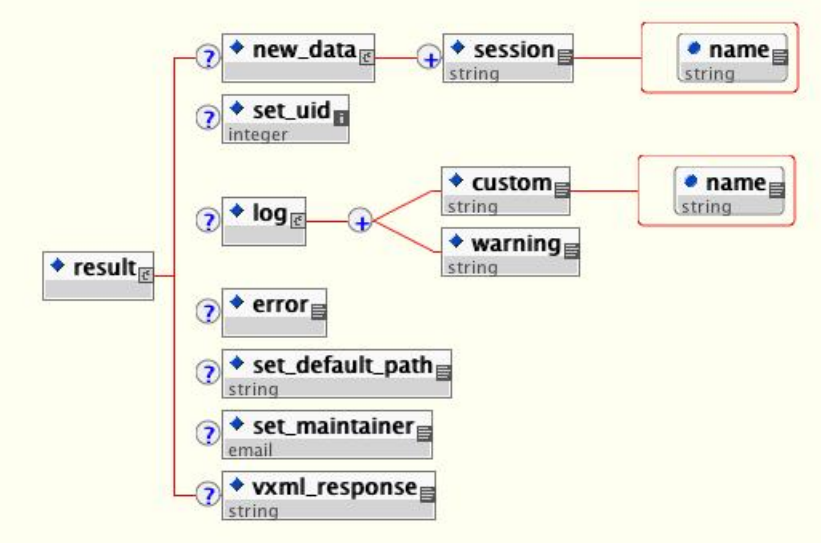
Java API Use

The end of call action is built in Java by implementing the Unified CVP class `EndCallInterface` found in the `com.audium.server.proxy` package. It contains a single method named `onEndCall` that is the execution method for the call end class. This method receives a single argument, an instance of `CallEndAPI`. This class belongs to the Session API and is used to access session information such as session data (See Session API for more information on this API). The method does not have a return value. It is expected that should an unrecoverable error occur, the call end action will throw an `AudiumException`.

If the call end action is to return a final VoiceXML page to the voice browser, this is done by using the Voice Foundation Classes (VFCs) (See Voice Foundation Classes for more on the VFCs) and accessing methods in the `CallEndAPI` Session API class.

XML API Use

As described in Session API, the standard *inputs* and *settings* XML documents are sent via POST to the call start URI. The following shows the DTD diagram of the XML document that must be sent in response. The DTD for the end of call action response is defined in the file `CallEndResponse.dtd` found in the VXML Server `dtds` folder.



The tags in these XML documents are:

- **new_data** – This tag holds the session data to be created. Any number of **<session>** tags can appear, one for each session data variable to be created.

  Note Element data cannot be created because the call end action is not an element.

- **set_uid** – This tag is used to associate the call to 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 for this application. 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.
- **error** – This tag reports to VXML Server that an error occurred while executing the call end action. 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.

  Note Since the call has ended, there would be no adverse affect to the call itself, though an error event will be thrown.

- **set_default_path** – This tag is used to change the default audio path.
- **set_maintainer** – This tag is used to change the maintainer e-mail address.
- **vxml_response** – This optional tag encapsulates the VoiceXML page that is to be passed to the voice browser for the final response. It is expected to contain a CDATA tag that encapsulates the entire VoiceXML document as it is to be returned to the voice browser (including the first line starting with `<?xml`).The developer is responsible for ensuring the VoiceXML is correct as VXML Server does no validation of the VoiceXML before returning it to the browser.

Note Since the VFCs are not used to throw the VoiceXML like the Java API, the developer is responsible for ensuring the VoiceXML is compatible with the voice browser(s) being deployed to.

Note All the tags are optional, there is no tag required except for the root `<result>` tag. Since the XML API requires a document in response, it is acceptable to return an XML document whose `<result>` tag is empty.