Genesys Interaction Recording Solution Guide

IVR Recording

4/7/2026

Contents

IVR Recording

The following table lists the DN types used with GVP and specifies how recording is supported for each DN type. There are two different ways to record a GVP-based IVR application:

- Record the entire IVR application, using the agent recording method with the SIP Server DN configuration.
- Record part or all of the IVR application with VoiceXML application-level control.

DN Type	Usage	Can be recorded using the agent recording method with SIP Server configuration	Can be recorded using VoiceXML recording control
VoIP Service DN (MSML)	PlayApplication treatment to execute a VoiceXML application.	No	Yes
Voice Treatment Port (VTP)	Legacy IVR ports for both inbound and outbound IVR calls	Yes	Yes
Trunk Group DN	Inbound GVP IVR calls and proactive notification (outbound GVP IVR calls)	Yes	Yes
Trunk DN	Inbound GVP IVR calls	Yes	Yes

Important

- Configuration of GVP is also required for IVR recording. For additional details, refer to the IVR Profile.
- You must also configure the `sip-enable-ivr-metadata` option so that SIP Server can pass its Application name to MCP for IVR Recording. For details about configuring this option, see Metadata Support for IVR Recording.

Controlling Recording with VoiceXML

The syntax to control recording within a VoiceXML application is described in VoiceXML Syntax, and the scope of when recording occurs within an IVR application is described in Recording Scope.

> **Important**
>
> This feature is applicable only for GIR and it is not applicable to environments with third-party recording solutions integrated with GVP.

VoiceXML Syntax

To control recording within a VoiceXML application, set the **dest** attribute of the **<log>** tag to **record**, anywhere that executable content is allowed. This value informs MCP that the contents of the **<log>** tag are commands, rather than text that should be logged. In the body of the tag, list the desired commands, delimited by semicolons. For example,

```
<?xml version="1.0"?>
<vxml version="2.1" xmlns:gvp="http://<url>/vxml21-extension">
..
<log gvp:dest="record">
   <command>
   [additional commands;]
</log>
```

Where **<command>** can be one of the following:

Command	Description
`start;`	This command starts recording, or restarts call recording (in a new recording file), if it has already been started. All audio data for the call from this point forward will be recorded into the newly opened recording file. This will continue until the call terminates, or a subsequent `start;` command is issued, or until a `stop;` command is issued.
`stop;`	This command stops recording and terminates recordings in progress. Audio will be recorded up until the point that this command is executed. If no recording are in progress at the time, then this is a no-operation.
`pause;`	This command pauses the recording in progress. The recording file will continue to be recorded with silent audio until the recording is resumed (with a `resume;` command), stopped, or the call is terminated. If no recording is in progress, then this is a no-operation.
`resume;`	This command resumes a paused recording in progress. The recording file will no longer be recording silent audio. If no recording is in progress, then this is a no-operation.

Command	Description
`additional commands;`	`parameter key=value` can be used as additional commands.

Command	Description
	Allows VoiceXML to inject an additional media file parameter into the recording file. More than one `parameter` command can be used to include multiple parameters in the recording. `key` is inserted as a property in the parameters object of the recording media file metadata, with `value` as the value of the property.

Example

```
<?xml version="1.0"?>
<vxml version="2.1" xmlns:gvp="http://<url>/vxml21-extension" >
  <form>
   <var name="Result" expr="'success'"/>
        <block>
      <log >
        START RECORDING
      </log>
        </block>
        <block>
      <prompt>
        Prompt from external vxml  - starting recording now
      </prompt>
    </block>
    <block>
      <log gvp:dest="record">
        start;
          </log>
    </block>
        <block>
      <log >
        RECORDING STARTED
      </log>
    </block>

        <block>
      <prompt>
        Prompt from external vxml  - recording started now
      </prompt>
    </block>


        <block>
     <return namelist="Result"/>
        </block>

     </form>
         <catch event="connection.disconnect.hangup telephone.disconnect.hangup" >
              <assign name="Result" expr="'hangup'"/>
              <log>Message is <value expr="typeof(_message)"/>.</log>
              <if cond="typeof(_message) == 'undefined' || _message == null  ||  _message ==
'undefined'  ||  (!_message)">
                     <assign name="ResultDetail" expr="_event"/>
              <else/>
                     <assign name="ResultDetail" expr="_event + ' - ' + _message"/>
              </if>
              <return namelist="Result ResultDetail ReturnedIntent ReturnedConfidenceScore
ReturnedUtterance"/>
    </catch>
         <catch event="error" >
               <if cond="typeof(_message) == 'undefined' || _message == null">
```

```
                    <assign name="ResultDetail" expr="_event"/>
            <else/>
                    <assign name="ResultDetail" expr="_event + ' - ' + _message"/>
            </if>
            <assign name="Result" expr="'error'"/>
            <return namelist="Result ResultDetail ReturnedIntent ReturnedConfidenceScore
ReturnedUtterance"/>
 </catch>
</vxml>
```

Recording Scope

The scope of the recording is within the scope of the VoiceXML session, and this is different from Full Call Recording using the agent recording method with the SIP Server DN configuration, where the scope of the recording is limited to the connection on the MCP. When the VoiceXML session terminates or **<exit>** or **<disconnect>** tags are called, the recording session is also considered terminated.

Each PlayApplication treatment will cause the MCP to generate a separate recording file. The recording files will be grouped together automatically by GIR as they share the same CallUUID.

The recording will be recorded from the perspective of the caller. It will follow the convention of the IVR Profile recording for mono or stereo recording. In stereo recording, the channels represent what the caller hears and what the caller says.

Access Control Considerations

GIR provides access control for recording files (refer to Agent Hierarchy), to allow recording files to be accessible only to specific GIR users. To enforce access control, each recording file is provided with an agent hierarchy or a set of partitions. Users must be a member of an Access Group with a matching agent hierarchy or partition to search and playback the recording.

IVR recordings have no agent hierarchy (as the IVR is not a user), and by default they have no partitions. This means that by default only users in the root access group (that have access to all recordings), or the default access group can access IVR recordings.

There are two ways to apply additional access control to IVR recordings:

1. By attaching the data key GRECORD_PARTITIONS to the call. In this case, all media files for the call (that is, the IVR segments and any agent segments) will receive the same set of partitions set in the attached data. For additional information, refer to the Partitions section.
2. You can individually apply a partition for a VoiceXML recording by using the additional command **parameter** and specifying the key **partitions**, with a value set to a comma-delimited list of partitions to apply to the recording. For example, to set partitions **/sales** and **/support** in VoiceXML:

```
// <?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form>
    <block>
      <log gvp:dest="record">
        start;
        parameter partitions=/sales,/support;
      </log>
```

```
      </block>
      <!-- ...more VoiceXML code -->
    </form>
  </vxml>
```