PureConnect®

2019 R3

Generated:

11-September-2019

Content last updated:

1-August-2019

See Change Log for summary of changes.

GENESYS™

VoiceXML

Technical Reference

Abstract

This document is a technical reference for the PureConnect VoiceXML system. VoiceXML, the Voice Extensible Markup Language, is an XML-based language used to create audio dialogs.

For the latest version of this document, see the PureConnect Documentation Library at: http://help.genesys.com/cic.

Table of Contents

Introduction to VoiceXML

The *VoiceXML Technical Reference* provides information about the Genesys PureConnect VoiceXML system.

VoiceXML, the Voice Extensible Markup Language, is an XML-based language used to create audio dialogs. These audio dialogs feature synthesized speech (TTS) or digitized audio (pre-recorded audio) to prompt the user, and they accept spoken words or DTMF key input. The VoiceXML application - or script - contains the logic that controls the flow of the dialog, and it's the script that prompts the caller, accepts the caller's input, and determines the next step for the caller.

The Genesys VoiceXML Interpreter uses Commetrex's BladewareVXML library which is written to the VoiceXML 2.1 specification published by the World Wide Web Consortium (W3C).

For the latest VoiceXML 2.0 specification (dated 16 March 2004 as of this writing), see http://www.w3.org/TR/voicexml20/.

For the latest VoiceXML 2.1 specification (an addendum to the VoiceXML 2.0 specification, dated 19 June 2007 as of this writing), see http://www.w3.org/TR/voicexml21/.

Other VoiceXML Documentation

The following documentation provides more information about Genesys's VoiceXML system.

PureConnect Installation and Configuration Guide

This document assists you in installing and configuring Interaction Center.

VoiceXML Installation and Configuration Guide

This document assists you in installing and configuring the IC VoiceXML Interpreter Server.

CIC VoiceXML Integration with Nuance Dialog Modules Version 5.2

This document describes updates required to allow Nuance Dialog Modules version 5.2 to work with the IC VoiceXML Interpreter Server.

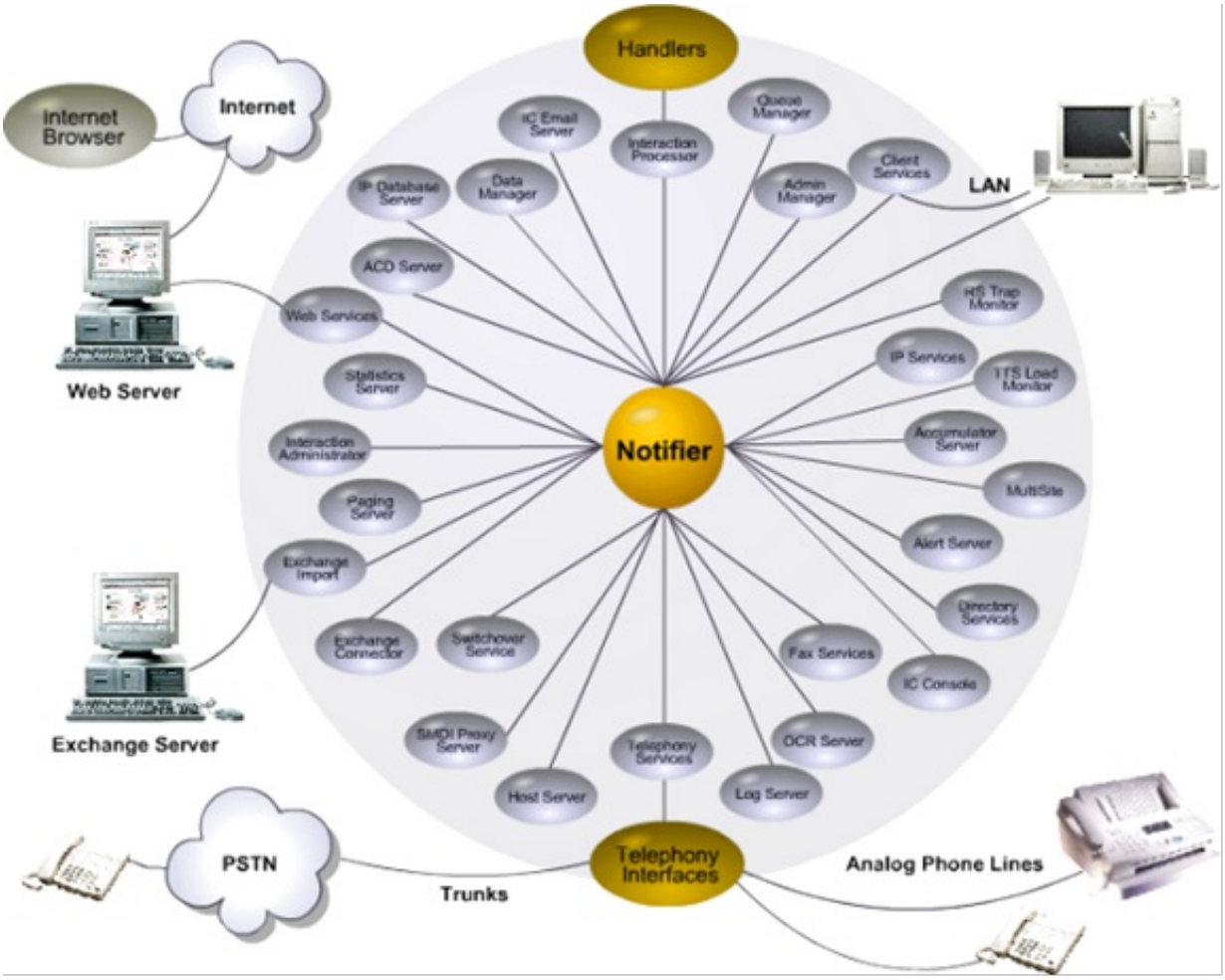
CIC VoiceXML Integration with Nuance Dialog Modules Version 6.1

This document describes updates required to allow Nuance Dialog Modules version 6.1 to work with the IC VoiceXML Interpreter Server.

These documents are available in the Documentation Library on your CIC system. In addition, the latest versions of these documents can be accessed from the PureConnect Documentation Library at http://help.genesys.com.

Customer Interaction Center Architecture

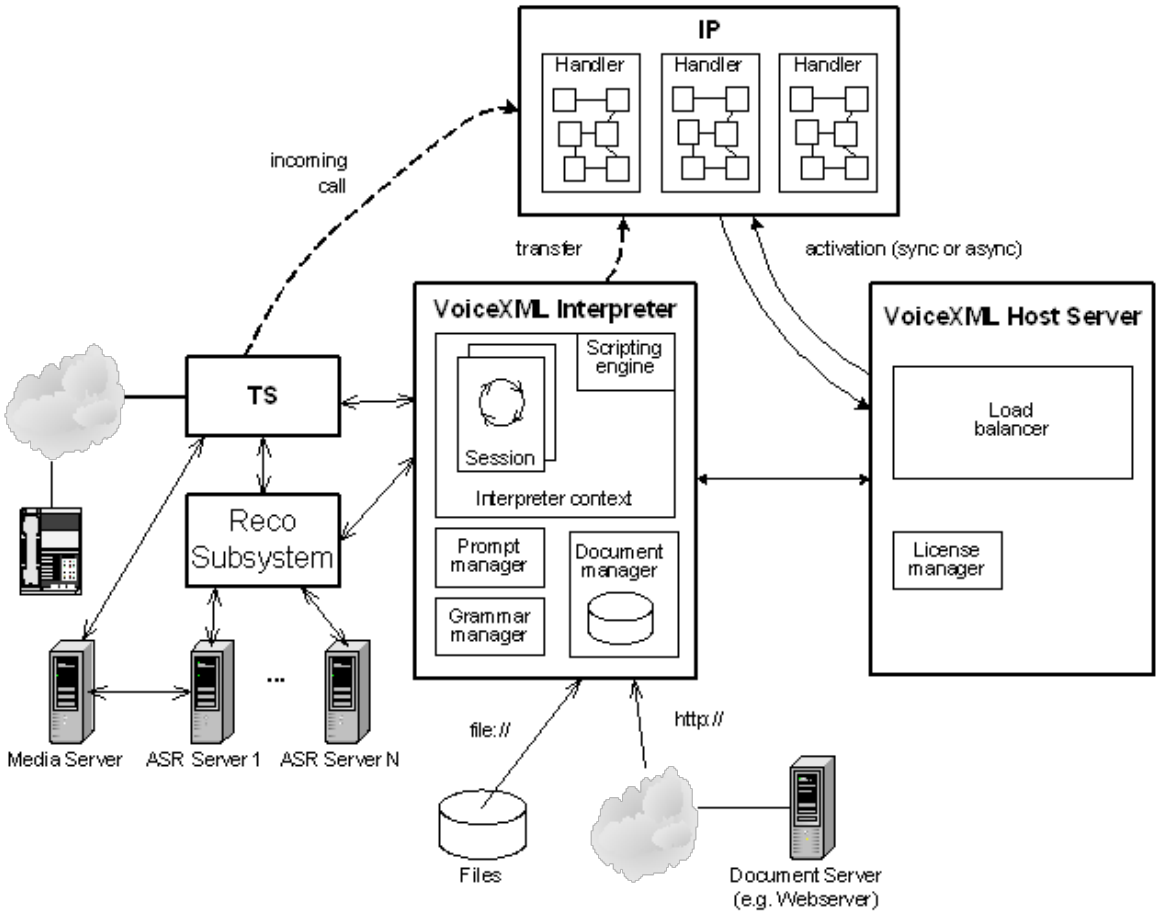
The CIC system is a collection of processes that communicate with one another through the Notifier process. Processes connect to the Notifier to either make requests, to service requests, or both.



The Interaction Processor (IP) process allows users to access many of CIC's capabilities by constructing handlers (using a set of handler tools) that communicate with the Notifier. These handlers in turn provide access to CIC's VoiceXML facilities.

VoiceXML Architecture

CIC's VoiceXML architecture consists of two main components: A VoiceXML Host Server and a VoiceXML Interpreter Server.



VoiceXML Component	Description
Handler Tools	The VoiceXML Handler Tools are the means to direct a call to use the VoiceXML Interpreter.
Host Server	The VoiceXML Host Server manages all of the VoiceXML sessions, in addition to the resources that the VoiceXML Interpreters need while interacting with CIC.
Interpreter Server	The VoiceXML Interpreter Server interprets the VoiceXML document and uses CIC's capabilities to perform the actions described in that VoiceXML document.The Interpreter can access grammar files, prompt files, and VoiceXML script files from local disc, network disc, or web server. The Interpreter works with the Telephony Services (TS) subsystem to provide telephony capabilities, and with the Voice Recognition (Reco) subsystem to provide voice recognition capabilities.

For more information about these components, see the following:

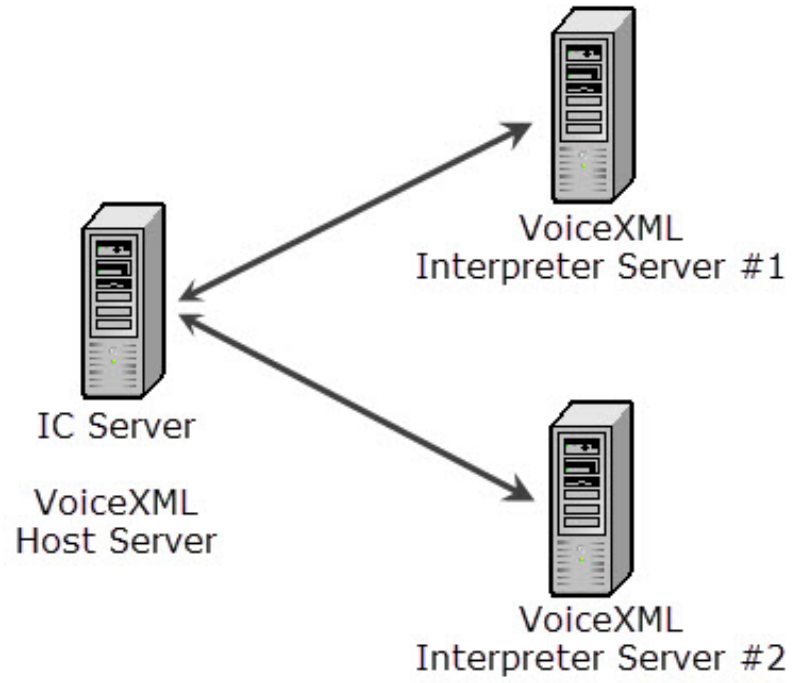
- Handlers and VoiceXML Tools
- VoiceXML Host Server
- VoiceXML Interpreter Server

1xN Architecture

The PureConnect VoiceXML feature consists of two main components:

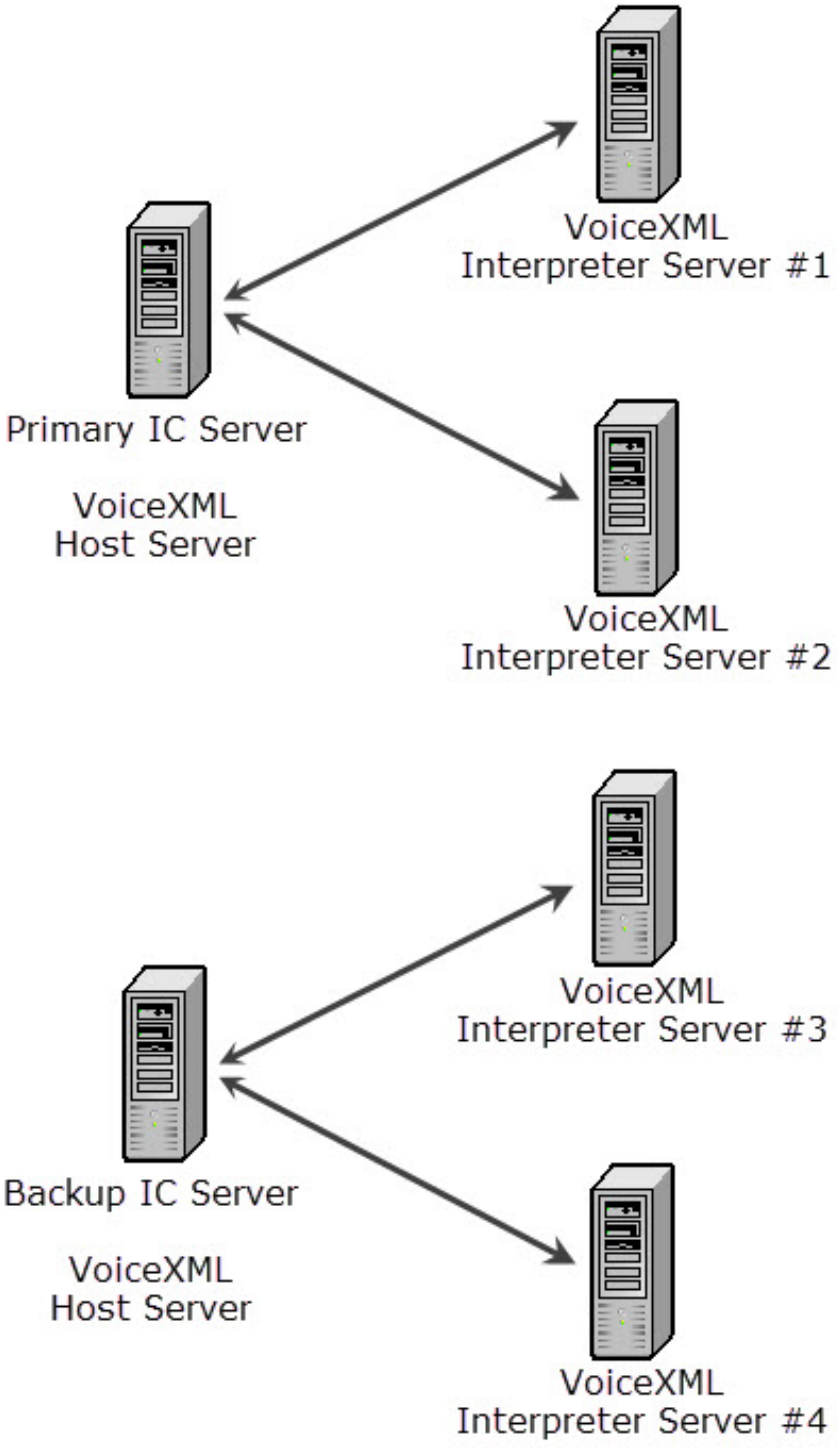
- VoiceXML Host Server (which runs on the CIC Server)
- VoiceXML Interpreter Server(s)

Each VoiceXML Interpreter Server can only be connected to one CIC server at a time, but more than one VoiceXML Interpreter Server can be associated with any given CIC server. As more traffic is routed through the VoiceXML Interpreters, more VoiceXML Interpreters can be brought online and used. Bringing more Interpreters online does not automatically increase the number of VoiceXML licenses in use. In fact, the number of licenses in use is independent of the number of VoiceXML Interpreters in use. Instead, bringing more interpreters online just brings more processing power to bear in servicing VoiceXML document requests.



Switchover Environment

Because a VoiceXML Interpreter can be set up to communicate with a maximum of only one CIC server at a time, using VoiceXML in a switchover environment requires special configuration. Each VoiceXML system must be duplicated for each CIC server in a switchover pair. If the user VoiceXML traffic merits having two VoiceXML Interpreters connected to the primary CIC Server, then two VoiceXML Interpreters must be configured to work with the backup CIC server.



Handlers and VoiceXML Tools

Handlers process incoming calls and decide whether a call goes through a VoiceXML application. If a handler decides an incoming call must go through a VoiceXML application, it initiates a tool that transfers the call to the VoiceXML subsystem. At this point, a VoiceXML session is then activated and the session activation tool specifies the URL of the VoiceXML document and which dialog to start. The VoiceXML Interpreter then attempts to load and parse the document, and if it is successful, the Interpreter takes ownership of the call.

A handler activates a VoiceXML session using one of the four tools available from the Interaction Designer Tools palette. This palette is on the VoiceXML page.

For more information about the VoiceXML tools, see the following:

- VoiceXML Initiate
- VoiceXML Async Initiate
- VoiceXML Initiate Document
- VoiceXML Async Initiate Document

For information about logging and tracing, see VoiceXML Handler Tools Logging/Tracing.

VoiceXML Initiate

The VoiceXML Initiate tool sends a request to the VoiceXML subsystem to start a session with the specified interaction and the initial VoiceXML document URL is given as an argument. This tool issues a synchronous Notifier request (**eVoiceXMLRequest_InterpretURL**), and then waits for a response from the VoiceXML subsystem.

The VoiceXML Initiate tool can use the parameters shown in the following table:

Parameter	Dir.	Type	Description
Interaction	IN	ID	Interaction to send to VoiceXML interpreter.
Document URL	IN	String	URL of the initial VoiceXML document.
Queued Plays Processing	IN	Integer	Optional. Not used. Reserved for future use.
Argument Names	IN	StringList	Optional. List of argument names.
Argument Values	IN	StringList	Optional. List of argument values corresponding to the arguments named in the 'Argument Names' parameter. These arguments are available in VoiceXML through 'session.name' or 'name'
Force Interaction Ownership	IN	Check box	Check box for the VoiceXML interpreter to force the interaction ownership. Unselected: Default. The VoiceXML interpreter only acquires ownership if the handler currently owns the interaction. Selected: The VoiceXML interpreter always acquires the interaction ownership.
Return Value	OUT	String	Result of the 'expr' expression of the <exit> element. Empty string if no return value.
Result Data	OUT	XMLNode	Element of the result data of the session. ("namelist" argument of the <exit> element) NULL node if no result data.
Event Name	OUT	String	VoiceXML event that caused the interpreter to exit, or reason for session initiation failure. Empty string if the session exited through <exit> element.
Event Message	OUT	String	Additional message text associated with the VoiceXML event ('_message' property).

The VoiceXML Initiate tool can take the exit paths shown in the following table:

Exit	Description
Success	VoiceXML session completed successfully and the return value and/or result data (XML node) are valid.
Disconnected	The interaction disconnected during the session.
Transferred	The interaction was transferred to another destination during the session through the <transfer> element.
Lost Ownership	Some other entity has taken the ownership of the interaction away during the VoiceXML session.
Invalid Interaction	VoiceXML interpreter does not support the interaction type.
Unhandled Event	This exit is taken if the session quit because some event other than an error is thrown during the session and not handled by the VoiceXML application.The Event Name output parameter contains the name of the event and Event Message may contain more information.
Failure	An error occurred. Typically an error event was thrown but not caught in the VoiceXML document.The name of the event is returned in Event Name.If the initial document fails to load, an error.badfetch event with potentially more decorations, such as error.badfetch.http.404, is returned. The Event Message parameter usually contains more information about what went wrong.

VoiceXML Async Initiate

The VoiceXML Async Initiate tool sends a request to the VoiceXML subsystem to start a session with the specified interaction, and the initial VoiceXML document URL is given as an argument. This tool issues an asynchronous Notifier request (**eVoiceXMLRequest_InterpretURL**), and does not wait for a response from the VoiceXML subsystem.

The VoiceXML Async Initiate tool can use the parameters shown in the following table:

Parameter	Dir.	Type	Description
Interaction	IN	ID	Interaction to send to VoiceXML interpreter.
Document URL	IN	String	URL of the initial VoiceXML document.
Queued Plays Processing	IN	Integer	Optional. Not used. Reserved for future use.
Argument Names	IN	StringList	Optional. List of argument names.
Argument Values	IN	StringList	Optional. List of argument values corresponding to the arguments named in the 'Argument Names' parameter These arguments are available in VoiceXML through 'session.name' or 'name'.
Force Interaction Ownership	IN	Check box	Check box for the VoiceXML interpreter to force the interaction ownership. Unselected: Default. The VoiceXML interpreter only acquires ownership if the handler currently owns the interaction. Selected: The VoiceXML interpreter always acquires the interaction ownership.
Event Name	OUT	String	VoiceXML event that caused the interpreter to exit, or reason for session initiation failure.
Event Message	OUT	String	Additional message text associated with the VoiceXML event ('_message' property).

The Async Initiate tool can take the exit paths shown in the following table:

Exit	Description
Success	VoiceXML session completed successfully and the return value and/or result data (XML node) are valid.
Disconnected	Interaction has already been disconnected before the session started.
Lost Ownership	The VoiceXML interpreter could not obtain ownership of the interaction as another entity has already taken it away.
Invalid Interaction	VoiceXML interpreter does not support the interaction type.
Failure	An error occurred.If the initial document fails to load, an 'error.badfetch' event with potentially more decorations, such as 'error.badfetch.http.404', is returned. The 'Event Message' parameter usually contains more information about what went wrong.

VoiceXML Initiate Document

This tool sends a request to the VoiceXML subsystem to start a session with the specified interaction, and the initial VoiceXML document is given as an argument. This tool is most useful to interpret dynamically generated VoiceXML documents or documents read from a database. This tool issues a synchronous Notifier request (**eVoiceXMLRequest_InterpretDocument**), and then waits for a response from the VoiceXML subsystem.

The VoiceXML Initiate Document tool can use the parameters shown in the following table:

Parameter	Dir.	Type	Description
Interaction	IN	ID	Interaction to send to VoiceXML interpreter.
VoiceXML Document	IN	XMLNode	XML DOM node of the VoiceXML document to interpret.The argument is either the document node or the <vxml> element.
Queued Plays Processing	IN	Integer	Optional. Not used. Reserved for future use.
Argument Names	IN	StringList	Optional. List of argument names.
Argument Values	IN	StringList	Optional. List of argument values corresponding to the arguments named in the 'ArgumentNames' parameter These arguments are available in VoiceXML through 'session.name' or 'name'.
Force Interaction Ownership	IN	Check box	Check box for the VoiceXML interpreter to force the interaction ownership. Unselected: Default. The VoiceXML interpreter only acquires ownership if the handler currently owns the interaction. Selected: The VoiceXML interpreter always acquires the interaction ownership.
Return Value	OUT	String	Result of the 'expr' expression of the <exit> element. Empty string if no return value.
Result Data	OUT	XMLNode	Element of the result data of the session.("namelist" argument of the <exit> element). NULL node if no result data.
Event Name	OUT	String	VoiceXML event that caused the interpreter to exit or reason for session initiation failure.
Event Message	OUT	String	Additional message text associated with the VoiceXML event ('_message' property).

The Initiate Document tool can take the exit paths shown in the following table:

Exit	Description
Success	VoiceXML session completed successfully and the return value and/or result data (XML node) are valid.
Disconnected	The interaction disconnected during the session.
Transferred	The interaction was transferred to another destination during the session through the <transfer> element.
Lost Ownership	Some other entity has taken the ownership of the interaction away during the VoiceXML session.
Invalid Interaction	VoiceXML interpreter does not support the interaction type.
Unhandled Event	This exit is taken if the session quit because some event other than "error" is thrown during the session and not handled by the VoiceXML application. The 'Event Name' output parameter contains the name of the event and 'Event Message' may contain more information.
Failure	An error occurred.That usually means an "error" event was thrown but not caught in the VoiceXML document.The name of the event is returned in 'Event Name'.If the initial document fails to load, an error.badfetch event with potentially more decorations, such as error.badfetch.http.404, is returned. The 'Event Message' parameter usually contains more information about what went wrong.

VoiceXML Async Initiate Document

The VoiceXML Async Initiate Document tool sends a request to the VoiceXML subsystem to start a session with the specified interaction and the initial VoiceXML document is given as an argument. This tool is most useful to interpret dynamically generated VoiceXML documents or documents read from a database. This tool issues an asynchronous Notifier request (**eVoiceXMLRequest_InterpretDocument**) and does wait for a response from the VoiceXML subsystem.

The VoiceXML Async Initiate Document tool can use the parameters shown in the following table:

Parameter	Dir.	Type	Description
Interaction	IN	ID	Interaction to send to VoiceXML interpreter.
VoiceXML Document	IN	XMLNode	XML DOM node of the VoiceXML document to interpret.The argument is either the document node or the <vxml> element.
Queued Plays Processing	IN	Integer	Optional. Not used. Reserved for future use.
Argument Names	IN	StringList	Optional. List of argument names.
Argument Values	IN	StringList	Optional. List of argument values corresponding to the arguments named in the 'ArgumentNames' parameter These arguments are available in VoiceXML through 'session.name' or 'name'.
Force Interaction Ownership	IN	Check box	Check box for the VoiceXML interpreter to force the interaction ownership. Unselected: Default. The VoiceXML interpreter only acquires ownership if the handler currently owns the interaction. Selected: The VoiceXML interpreter always acquires the interaction ownership.
Event Name	OUT	String	VoiceXML event that caused the interpreter to exit or reason for session initiation failure.
Event Message	OUT	String	Additional message text associated with the VoiceXML event ('_message' property).

The Async Initiate Document tool can take the exit paths shown in the following table:

Exit	Description
Success	VoiceXML session completed successfully and the return value and/or result data (XML node) are valid.
Disconnected	Interaction has already been disconnected before the session started.
Lost Ownership	Some other entity has taken the ownership of the interaction away during the VoiceXML session.
Invalid Interaction	VoiceXML interpreter does not support the interaction type.
Failure	An error occurred. If the initial document fails to load, an 'error.badfetch' event with potentially more decorations, such as'error.badfetch.http.404', is returned. The 'Event Message' parameter usually contains more information about what went wrong.

VoiceXML Handler Tools Logging/Tracing

The VoiceXML Handler Tools generate log and trace messages to help you to track down and troubleshoot issues.

ININ Tracing

The VoiceXML Handler Tools generate diagnostic trace messages in the `ip.ininlog` file located in the PureConnect logging directory of the CIC server. These messages all have a topic of `I3_VoiceXmlTools`.

System event log

The VoiceXML Handler Tools can generate the following messages in the Windows System Event log on the CIC server system:

Message	Category
MSG_VXML_TOOLS_STD_EXCEPTION (43500)	VoiceXML Tools (IC 3.0 99 / IC 4.0 -100)
MSG_VXML_TOOLS_COM_EXCEPTION (43501)	VoiceXML Tools (IC 3.0 99 / IC 4.0 -100)
MSG_VXML_TOOLS_MFC_EXCEPTION (43502)	VoiceXML Tools (IC 3.0 99 / IC 4.0 -100)
MSG_VXML_TOOLS_UNKNOWN_EXCEPTION (43503)	VoiceXML Tools (IC 3.0 99 / IC 4.0 -100)
MSG_VXML_TOOLS_NOTIFIER_ERROR (43504)	VoiceXML Tools (IC 3.0 99 / IC 4.0 -100)

VoiceXML Host Server

The VoiceXML Host Server (VXMLHostSvr) is a server process that initially receives all VoiceXML interpret document requests and forwards the requests to a VoiceXML Interpreter Server for processing. The VoiceXML Interpreter Server makes requests back to the VoiceXML Host Server to upload resources, such as prompts and grammars, to the CIC server.

The VoiceXML Host server resides on the CIC server and performs the following tasks:

- Accepts a logon from a VoiceXML Host Server and creates an Interpreter session
- Stops an Interpreter session
- Reserves a file name for use by the requesting VoiceXML Host Server
- Uploads files from the requesting VoiceXML Host Server to the VXML Host Server's cache
- Downloads files from the VXML Host Server's cache to the requesting VoiceXML Host Server
- Releases files from the VXML Host Server's cache
- Passes all VoiceXML interpret document requests to a VoiceXML Interpreter Server determined by a load balancing algorithm

Load balancing

The VoiceXML Host Server uses a simple form of load balancing to determine which VoiceXML Interpreter Server to use for a specified request to interpret a document. When a request is received, the VoiceXML Host Server looks through all of the interpreters currently accepting requests and determines how many requests each is processing. The VoiceXML Host Server then selects the interpreter which is processing the fewest number of requests to process the document. The selection does not account for how complicated any particular request is or for how much CPU load the interpreter is using.

Logging/tracing

The VoiceXML Host Server generates log and trace messages to help you to track down and troubleshoot issues.

ININ tracing

The VoiceXML Host Server generates diagnostic trace messages in the “voicexml host server.ininlog” file located in the ININ logging directory of the CIC server.

System event log

The VoiceXML Host Server can generate the following messages in the Windows System Event log of the CIC server system:

Message	Category
MSG_VXML_HOST_STARTED_INFO (43600)	VoiceXML Host Server (IC 3.0 100 / IC 4.0 -101)
MSG_VXML_HOST_START_ERROR (43601)	VoiceXML Host Server (IC 3.0 100 / IC 4.0 -101)
MSG_VXML_HOST_STOPPED_INFO (43602)	VoiceXML Host Server (IC 3.0 100 / IC 4.0 -101)
MSG_VXML_HOST_STD_EXCEPTION (43610)	VoiceXML Host Server (IC 3.0 100 / IC 4.0 -101)
MSG_VXML_HOST_MFC_EXCEPTION (43611)	VoiceXML Host Server (IC 3.0 100 / IC 4.0 -101)
MSG_VXML_HOST_UNKNOWN_EXCEPTION (43612)	VoiceXML Host Server (IC 3.0 100 / IC 4.0 -101)
MSG_VXML_HOST_NOTIFIER_ERROR (43613)	VoiceXML Host Server (IC 3.0 100 / IC 4.0 -101)

VoiceXML Host Server Notifier messages

The VoiceXML Host Server interacts with the Notifier process of a CIC system where it both services a number of Notifier messages and uses a number of Notifier messages.

Serviced Notifier messages

The VoiceXML Host Server services the Notifier messages described in the following table:

Notifier Message	Description
eVoiceXMLRequest_InterpreterLogin	Handles a request to log in an interpreter.
eVoiceXMLRequest_InterpreterLogout	Handles a request to log out an interpreter.
eVoiceXMLRequest_ExchangeStreamEndpoints	Handles a request to exchange Notifier stream endpoints.
eVoiceXMLRequest_AcceptSessions	Handles a request to set an interpreter as accepting sessions or as not accepting sessions.
eVoiceXMLRequest_ReserveFileName	Handles a request to reserve a file name on the local cache.
eVoiceXMLRequest_UploadFile	Handles a request to upload a file name to the local cache.
eVoiceXMLRequest_ReleaseFile	Handles a request to release a file from the local cache.
eVoiceXMLRequest_InterpretURL	Handles a request to process the VoiceXML document at a specified URL.
eVoiceXMLRequest_InterpretDocument	Handles a request to process the VoiceXML document in a specified string.

Utilized Notifier messages

The VoiceXML Host Server uses the following Notifier messages:

Notifier Message	Description
eVoiceXMLRequest_InitiateSessionURL	Handles a request with the URL specifying the document.
eVoiceXMLRequest_InitiateSessionDocument	Handles a request containing the initial VoiceXML document as (XML) data.

Determining connected VoiceXML Interpreter Servers

The only way to determine which VoiceXML Interpreter Servers are currently connected to the VoiceXML Host Server and are active is through the ININ trace log. Look under InterpreterMgrTopic in the log viewer.

License management

In addition to these other tasks, the VoiceXML Host Server process manages the usage of VoiceXML licenses. Each running VoiceXML session uses one VoiceXML license, no matter which VoiceXML interpreter is selected to process that session.

VoiceXML Interpreter Server

The VoiceXML Interpreter Server interprets VoiceXML documents. This server:

- Accepts requests to start a session
- Interprets the VoiceXML document
- Starts requests to play user prompts and accept user input
- Closes the session

For more information, see the following:

- VoiceXML Interpreter Server Configuration
- VoiceXML Interpreter Server Logging/Tracing
- VoiceXML Interpreter Server Notifier Messages
- VoiceXML Interpreter Server Web Configuration Interface
- Managing a VoiceXML Interpreter Server

VoiceXML Interpreter Server Configuration

The VoiceXML Interpreter Server gets its configuration parameters from a number of different sources.

Windows Registry

The Windows Registry contains configuration values that are set when the VoiceXML Interpreter is installed.

The values shown in the following table can be found under the key:

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Interactive Intelligence\VoiceXML\vxiserver

Value	Description
ConfigFile	A file containing user configurable configuration information.
WebConfigLoginName	The login name to use when entering the web configuration pages. The login name is not encrypted by the install, but it is encrypted if modified via the web configuration pages.
WebConfigLoginPassword	The login password to use when entering the web configuration pages. The login password is not encrypted by the install, but it is encrypted if modified via the web configuration pages.
WebConfigServerPort	The port used when entering the web configuration pages.

Configuration file

The configuration file (specified in the ConfigFile registry value) contains a number of configuration parameters, shown in the table below. While you can adjust the parameters by editing the file, we recommended that you use the VoiceXML Interpreter Server Web Configuration Interface to change these parameters.

Parameter	Description
cacheCleanupInterval	The number of seconds the Interpreter Server waits before deleting a locally cached VXML resource.The minimum that can be specified is 15 seconds and the maximum is 120 seconds. The default value (30 seconds) can remain unchanged.
cachepath	The directory used to store cached VXML resources. The default value can remain unchanged.
recMaxTimeSecs	The maximum time of a recording in seconds. This default value can remain unchanged.
recoValueSlotName	The element name to use for the value slot in returned NLSML recognition results. If none is specified, the default name is “value.”
ttsMRCP	Setting this parameter to true tells the VoiceXML Interpreter that the TTS engine supports the MRCP protocol. If the TTS engine does not support MRCP, set this parameter to false. The default setting is false.
ttsSSML	By default, this parameter is set to true since the default SAPI engine can support SSML and the MRCP servers must support SSML. When set to true, this parameter tells the VoiceXML Interpreter that the TTS engine supports SSML text handling. If the TTS engine does not support SSML, set this parameter to false.
vxiConfigFile	Path to the VoiceXML library configuration file. The default value can remain unchanged.

Optional parameters

There are four optional parameters, shown in the table below, that are not included in the default configuration file, but you can add them if you discover that they are necessary.

Parameters	Description
defaultGrammarMimeType	This parameter can be used to specify the default grammar type to use if no type is specified for a grammar.
ignoreGrammarFileMimeType	Setting this parameter to true tells the VoiceXML Interpreter to ignore a grammar file’s MimeType when determining a grammar file’s grammar type. If this parameter is set to false, then the grammar file’s MimeType is used as the grammar type.
keepRecoSession	When a VoiceXML session ends, VoiceXML calls RecoRelease. This means the ASR session is immediately closed and any additional ASR will require a new session.The assumption is that no further ASR will occur when the VoiceXML session ends, but that's not necessarily true if handlers will perform additional ASR work, or another VoiceXML session is created for additional processing. Setting this parameter to true tells the VoiceXML Interpreter to keep the Recognition session alive when a VoiceXML session ends. The Reco session will have to be released outside of the VoiceXML Interpreter.
grammarCaching	Setting this parameter to true instructs the VoiceXML interpreter to cache grammar files. Setting the parameter to false instructs the VoiceXML interpreter to NOT cache grammar files. If this parameter is not set, the default is true.

To add any of the optional parameters to the VoiceXML Interpreter Server, you must do so by editing the configuration file. Once you add an optional parameter to the configuration file and restart the VoiceXML Interpreter Server, the optional parameter will be accessible from the VoiceXML Interpreter Server Web Configuration Interface. If you must adjust an optional parameter at a later time, we recommend that you use the VoiceXML Interpreter Server Web Configuration Interface to do so.

I3runvxml.cfg file

The I3runvxml.cfg file is a configuration file for the BladewareVXML library used by the Genesys PureConnect VoiceXML Interpreter. We recommend leaving the settings in this file in their default state.

Some of the settings in this file include:

- Cache settings

- ECMAScript settings
- BladewareVXML logging settings
- The location of the VoiceXML defaults XML

I3defaults.xml file

The I3defaults.xml file contains some VoiceXML script defaults. The default values for a number of properties are set and some default error handlers are defined. This document is loaded before every document requested by the user for interpretation.

VoiceXML Interpreter Server Logging/Tracing

The VoiceXML Interpreter Server generates log and trace messages to help you to track down and troubleshoot issues.

ININ tracing

The VoiceXML Interpreter Server generates diagnostic trace messages in the “voicexmlserver.ininlog” file located in the ININ logging directory of the VoiceXML Server.

BladewareVXML tracing

The BladewareVXML library has various status messages that it can output. These messages are routed into the ININ trace log under the “VXILogTopic” topic. Here are the trace levels and the types of output associated with each trace level:

Trace Level	Description
20 (Error) or higher	Shows errors from VoiceXML.
60 (Status) or higher	Shows various VoiceXML events.
80 (Notes) or higher	Shows various VoiceXML "Content" messages.
90 (Verbose Notes) or higher	Shows various VoiceXML “Diagnostic” messages, and the output from any <log> tags (which VoiceXML treats as “Diagnostic” messages).

System event log

The VoiceXML Interpreter Server can generate the following messages in the Windows System Event log:

Message	Category
MSG_VXML_HOST_STARTED_INFO (43600)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)
MSG_VXML_INTERP_STARTED_INFO (43700)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)
MSG_VXML_INTERP_START_ERROR (43701)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)
MSG_VXML_INTERP_STOPPED_INFO (43702)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)
MSG_VXML_INTERP_STD_EXCEPTION (43710)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)
MSG_VXML_INTERP_UNKNOWN_EXCEPTION (43711)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)
MSG_VXML_INTERP_NOTIFIER_ERROR (43712)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)
MSG_VXML_INTERP_PROCESS_DOCUMENT_ERROR (43720)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)
MSG_VXML_INTERP_PROCESS_DOCUMENT_URL_ERROR (43721)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)
MSG_VXML_INTERP_PARSE_DOCUMENT_ERROR (43722)	VoiceXML Host Server (IC 3.0 101 / IC 4.0 -102)

VoiceXML Interpreter Server Notifier Messages

The VoiceXML Interpreter Server interacts with the Notifier process of a CIC system and both services a number of Notifier messages and uses a number of Notifier messages.

Serviced Notifier messages

The VoiceXML Interpreter Server services the Notifier messages as described in the following table.

Notifier Message	Description
eVoiceXMLRequest_InitiateSessionURL	Handles a request to process the VoiceXML document at a specified URL.
eVoiceXMLRequest_InitiateSessionDocument	Handles a request to process the VoiceXML document in a specified string.

Utilized Notifier messages

The VoiceXML Interpreter Server uses the Notifier messages listed in the following table.

Notifier Message	Description
eVoiceXMLRequest_InterpreterLogin	Handles a request by the VoiceXML Interpreter Server to log in to the VoiceXML Host Server.
eVoiceXMLRequest_InterpreterLogout	Handles a request by the VoiceXML Interpreter Server to log out of the VoiceXML Host Server.
eVoiceXMLRequest_ExchangeStreamEndpoints	Handles a request to exchange connection endpoints (for file transfers) between the VoiceXML Host Server and the requesting VoiceXML Interpreter Server.
eVoiceXMLRequest_AcceptSessions	Handles a request to set an interpreter as “accepting sessions” or as “not accepting sessions.”
eVoiceXMLRequest_ReserveFileName	Handles a request to reserve the specified file name for use by the requesting VoiceXML Interpreter Server.
eVoiceXMLRequest_UploadFile	Handles a request to upload the specified file from the requesting VoiceXML Interpreter Server to the VoiceXML Host Server's cache.
eCallMsg_GetInputEx	
eCallMsg_CancelPlays	
eCallMsg_SinglePartyRecord	
eRecoAPIRequest_Initialize	
eRecoAPIRequest_Release	
eRecoAPIRequest_Input	

VoiceXML Interpreter Server Web Configuration Interface

You can use the VoiceXML Interpreter Server Web Configuration interface to check the server's status as well as to view and change the server's current configuration settings.

Accessing the Web Configuration interface

To access the VoiceXML Interpreter Server Web Configuration interface:

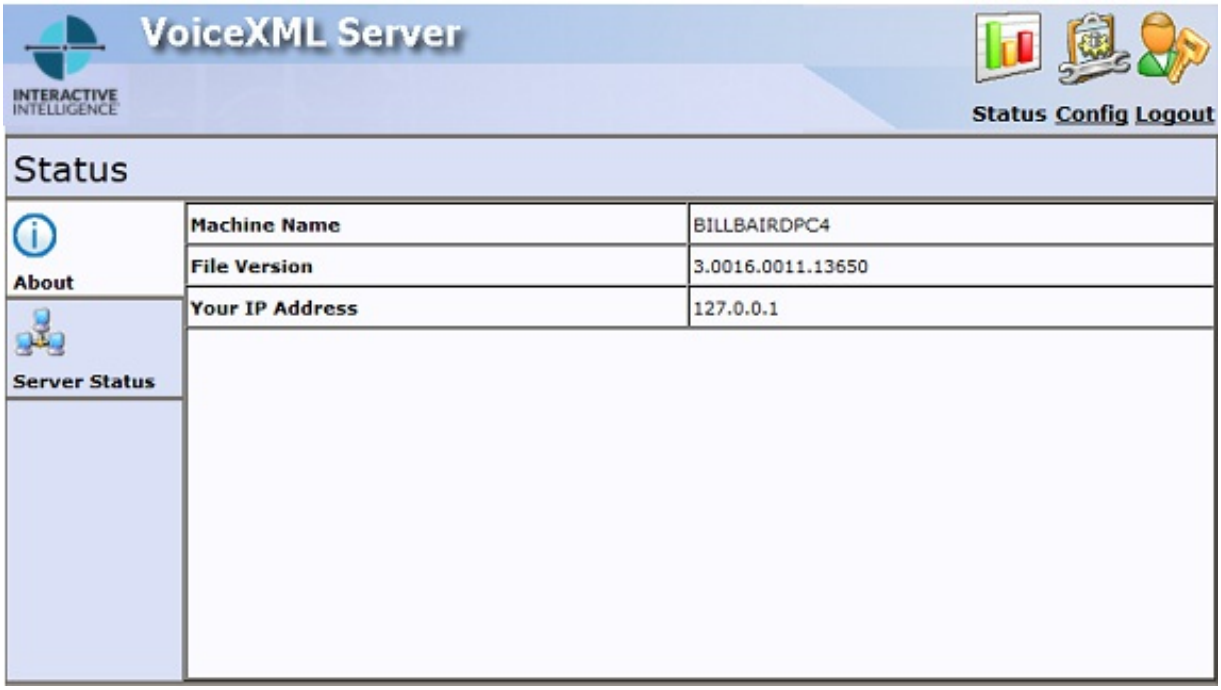
1. Ensure that the VoiceXML Interpreter Server is running.
2. Open your browser and access **http://{servername}:Port** where {servername} is the name of the system running the VoiceXML Interpreter Server and Port is the port number specified during installation as the Web Configuration Server Port. (The default port number is 8090.)

3. In the network authentication dialog box, type the **User Name** and **Password** that you specified during the VoiceXML Interpreter Server installation. The **Status** page for the appropriate type of VoiceXML Interpreter server appears.

> **Note:**
> If you forget the user name or password of the VoiceXML Interpreter Server Web Configuration interface, do the following on the VoiceXML Interpreter Server:
>
> a. Note the value in the registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Interactive Intelligence\VoiceXML\WebConfigLoginName`.
> b. Clear the value in the registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Interactive Intelligence\VoiceXML\WebConfigLoginPassword`.
> **Don't delete the registry entry; only clear its value.**
> c. Log on to the interface with the user name from "webConfigLoginName" and no password.
> d. Reset the password to restore the security of the web interface.

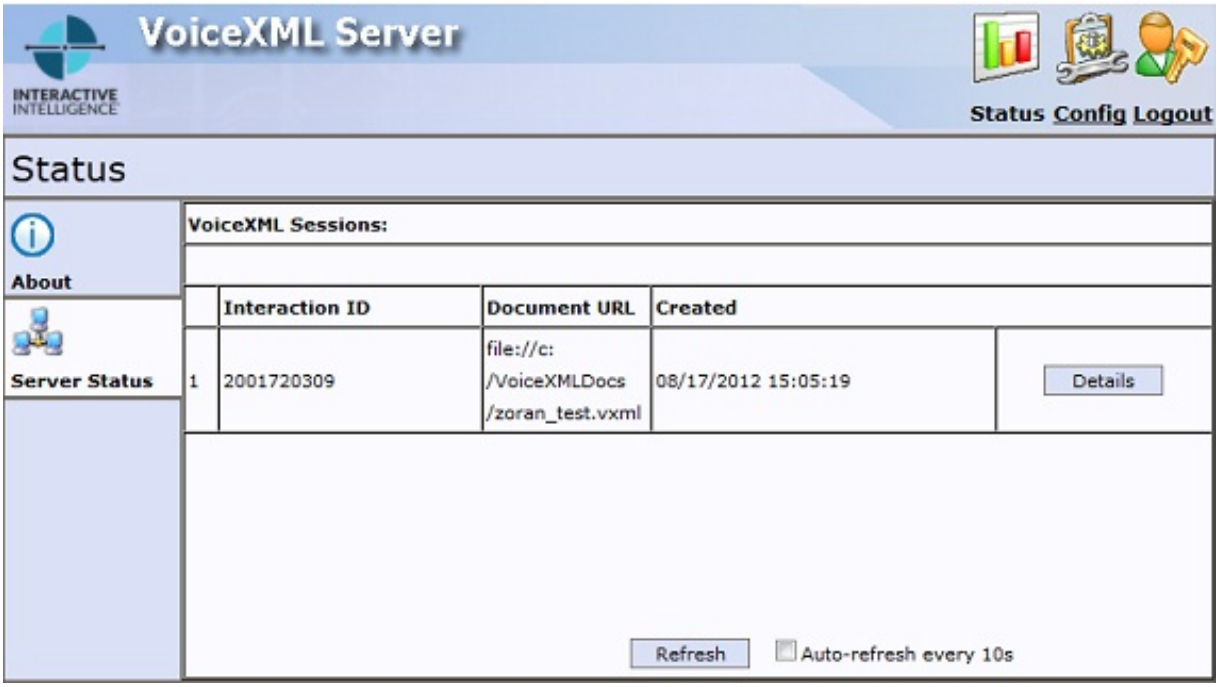
About

On the **About** tab of the **Status** page, you'll find the name of the VoiceXML Interpreter Server, the file version, and the IP address of the system you are using to access the Web Configuration interface.



Server Status

The **Server Status** tab of the **Status** page lists all the VoiceXML sessions that are running currently.



At any time, you can manually update the information on this screen by clicking **Refresh**. If you prefer, you can select the **Auto-refresh every 10s** check box and the information on this page will reload every 10 seconds.

Checking Active Document Sessions

You can check the active VoiceXML document sessions by clicking the **Details** button to display more detailed information for a specific interaction.

At any time, you can manually update the information on this screen by clicking **Refresh**. If you prefer, you can select the **Auto-refresh every 10s** check box and the information on this page will reload every 10 seconds.

The Sessions Details page displays information in the following fields:

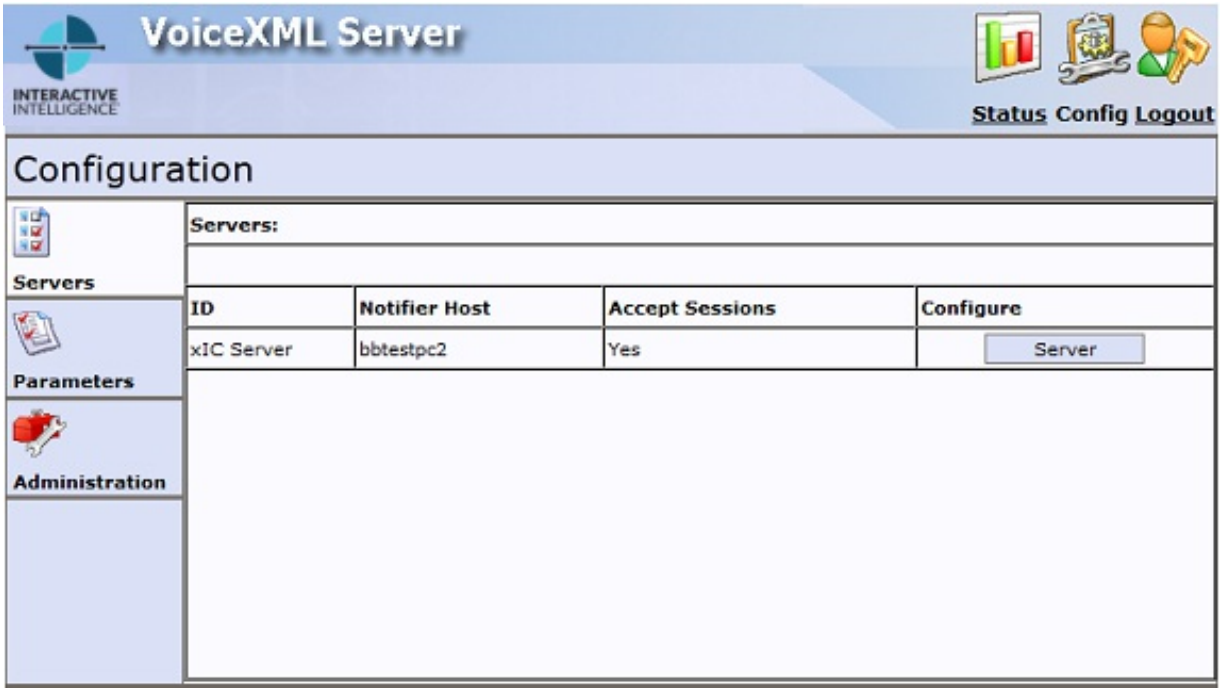
Field	Description
Session ID	The ID of the currently running document session.
Interaction ID	The ID of the interaction between the CIC server and the VoiceXML Interpreter server.
Status	The status of the connection.
Detailed Status	Details of exactly what is happening in the connection.
Current Document	The path to and name of the currently active document.
Created	The date and time that the document became active.

Configuring the VoiceXML Interpreter Server

You'll use the Configuration page to configure the VoiceXML Interpreter Server. To display the Configuration page, click **Config**. When you do, you'll see the Configuration page, which contains three tabs: Servers, Parameters, and Administration.

Servers

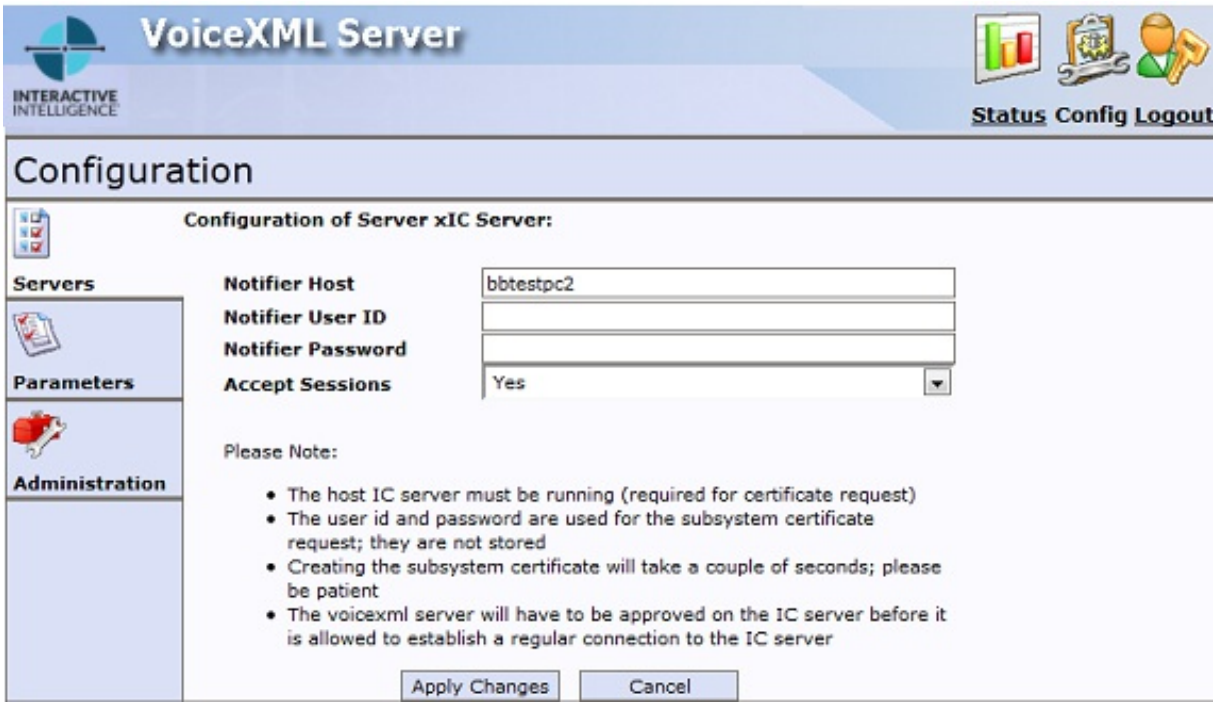
On the Servers tab you'll see the CIC server to which the VoiceXML Interpreter Server is connected and can determine whether the VoiceXML Interpreter Server is accepting VoiceXML sessions from that CIC server.



Clicking the **Server** button will allow you to connect a VoiceXML Interpreter Server to an CIC server as well as bring a VoiceXML Interpreter Server online or take it offline by changing the Accept Sessions setting.

Connecting a VoiceXML Interpreter Server to a CIC server

To connect a VoiceXML Interpreter Server to an CIC server, click the **Server** button. When you do, the Servers tab displays the Configuration of Server page.



The Configuration of Server page contains the following fields:

Field	Description
Notifier Host	Enter the name of a CIC server.
Notifier User ID	Enter the user name of the Customer Interaction Center user account that the VoiceXML Interpreter Server will use to log on to the CIC server.
Notifier Password	Enter the password for the user name specified above.
Accept Sessions	This field is set to Yes by default. The Yes setting configures the VoiceXML Interpreter Server to accept document sessions.

When you finish configuring the server, click **Apply Changes**.

Note: A VoiceXML Interpreter Server can only be associated with one CIC server at a time.

Taking a VoiceXML Interpreter Server Offline

To take a VoiceXML Interpreter Server offline so that it no longer accepts new document sessions, select **No** from the Accept Sessions dropdown and click **Apply Changes**.

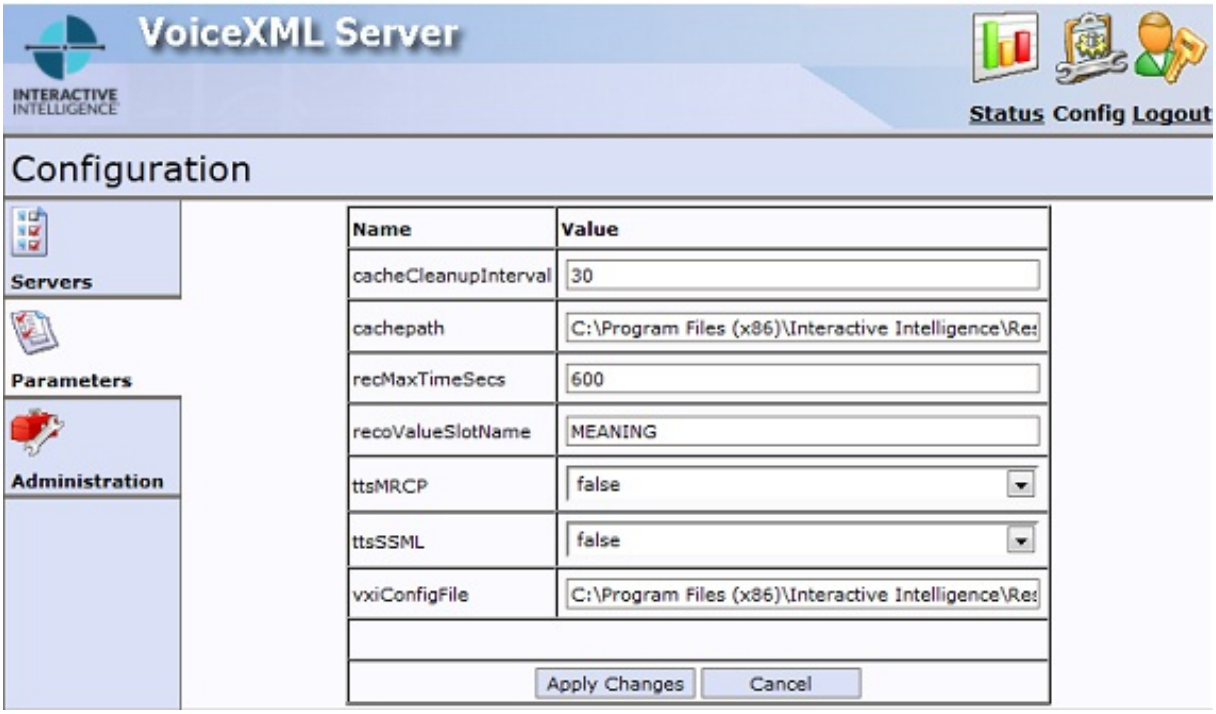
When you do, the VoiceXML Interpreter Server will prompt you to confirm that you want to change the status of the Accept Sessions to No. Click **OK** to continue. At this point, the VoiceXML Interpreter Server will finish processing all currently running document sessions, but will not accept new document sessions.

Bringing a VoiceXML Interpreter Server Online

To bring a VoiceXML Interpreter Server back online, so that it can accept new document sessions, select **Yes** from the Accept Sessions dropdown and click **Apply Changes**.

Parameters

To configure the VoiceXML Interpreter Server Parameters, select the **Parameters** tab. Once you make changes to the parameters, click **Apply Changes**.



The Parameters tab contains the following fields:

Parameter	Description
cacheCleanupInterval	The number of seconds the Interpreter Server waits before deleting a locally cached VXML resource.The minimum that can be specified is 15 seconds and the maximum is 120 seconds. The default value (30 seconds) can remain unchanged.
cachepath	The directory used to store cached VXML resources. The default value can remain unchanged.
recMaxTimeSecs	The maximum time of a recording in seconds. This default value can remain unchanged.
recoValueSlotName	The element name to use for the value slot in returned NLSML recognition results. If none is specified, the default name is “value.”
ttsMRCP	Setting this parameter to true tells the VoiceXML Interpreter that the TTS engine supports the MRCP protocol. If the TTS engine does not support MRCP, set this parameter to false. The default setting is false.
ttsSSML	Setting this parameter to true tells the VoiceXML Interpreter that the TTS engine supports SSML text handling. If the TTS engine does not support SSML, set this parameter to false. The default setting is false.
vxiConfigFile	Path to the VoiceXML library configuration file. The default value can remain unchanged.

If you add any of the available optional parameters to the configuration file and restart the VoiceXML Interpreter Server, those optional parameters will be accessible and can be changed from the VoiceXML Interpreter Server Web Configuration Interface. For more information, see "Optional parameters" in VoiceXML Interpreter Server Configuration.

Note: These are the same parameters that exist in the configuration file. For more information, see "Configuration file" in VoiceXML Interpreter Server Configuration.

Administration

You can use the controls on the Administration tab to change your login credentials and control external access.

To change your login credentials, you must first enter your old user name and password. Then, enter a new username and new password twice. To complete the operation, click **Apply**.

To change how you access the Web Configuration Interface, you'll use the control in the HTTP/HTTPS Server panel.

- If you want to be able to access Web Configuration Interface from a web browser running on any workstation on your network, make sure that the setting status shows

External access is currently enabled.

- If you only want to be able to access the Web Configuration Interface from a web browser running on the VoiceXML Interpreter Server, make sure that the setting status shows

External access is currently disabled.

- The label of the button in the HTTP/HTTPS Server panel will toggle between Enable and Disable depending upon the status of the setting.

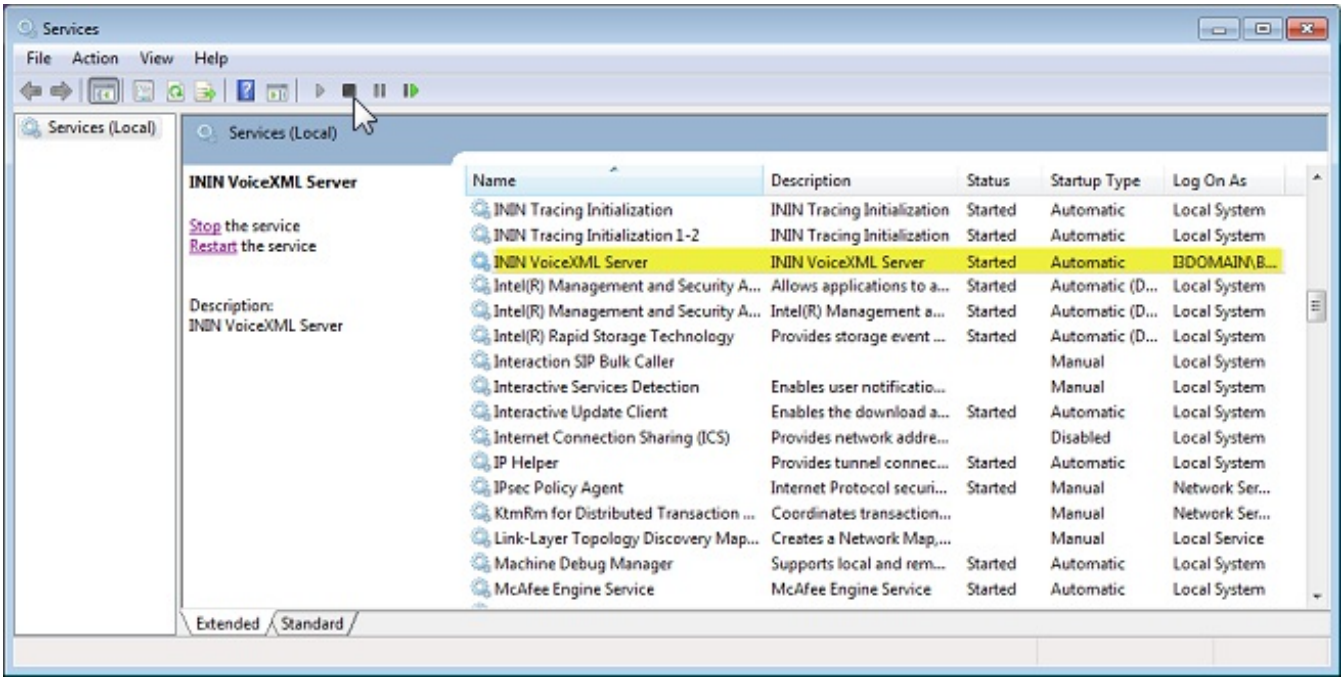
Managing a VoiceXML Interpreter Server

The VoiceXML Interpreter Server is installed as a service. As such, you must use the Services tool to stop and start the VoiceXML Interpreter Server.

Stopping VoiceXML Interpreter Server

Before stopping the VoiceXML Interpreter Server service, you will need to take the server offline as described in "Configuring the VoiceXML Interpreter Server" in VoiceXML Interpreter Server Web Configuration Interface. The reason that you need to do this first is to ensure that that all active VoiceXML sessions finish processing before the service stops.

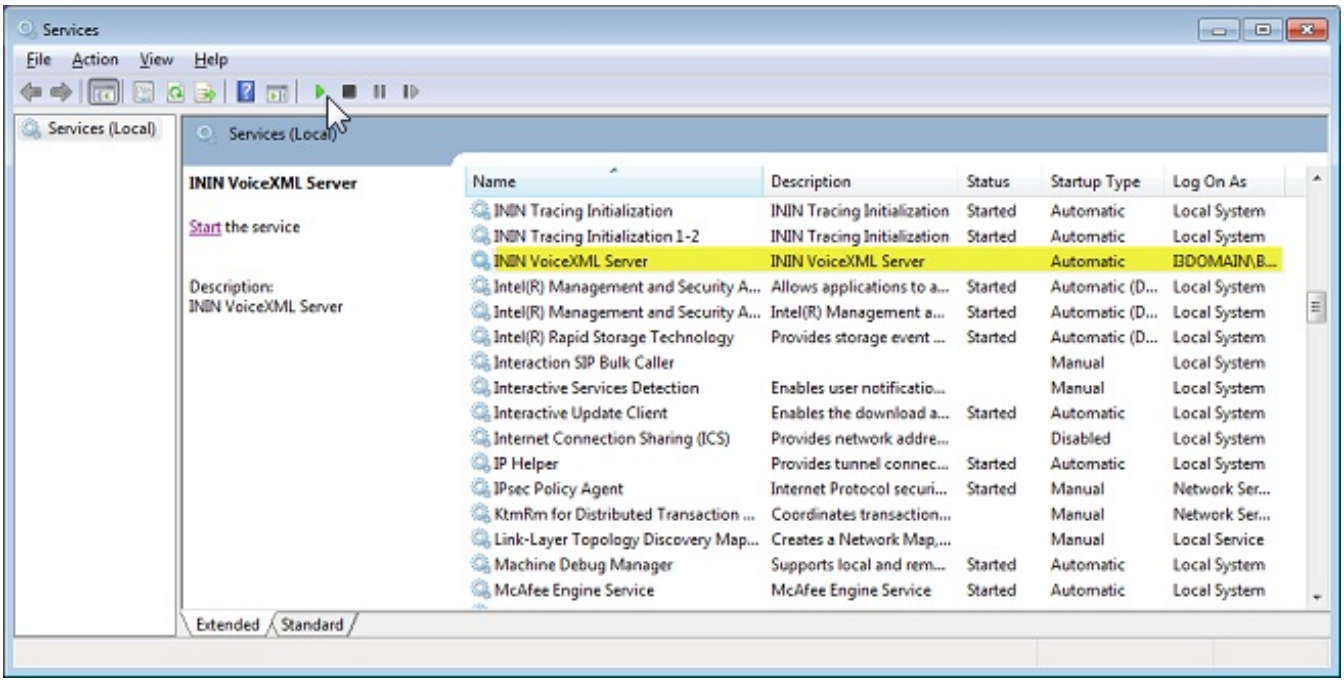
To stop the VoiceXML Interpreter Server in the Services tool, select the ININ VoiceXML Server service in the list of services and click the **Stop Service** icon on the toolbar.



Note: If you stop the VoiceXML Interpreter Server service before taking the server offline and documents are still being processed, the interpreter waits for a maximum of 5 minutes for the processing to finish. At the end of the 5 minutes, any running document sessions are forcibly closed, which could cause problems.

Starting VoiceXML Interpreter Server

To start the VoiceXML Interpreter Server in the Services tool, select the ININ VoiceXML Server service in the list of services and click the **Start Service** icon on the toolbar.



VoiceXML Data Security

By default, VoiceXML 2.0/2.1 specifcation doesn't provide a methodology for differentiating between normal and sensitive data. To add data security to Genesys PureConnect's implementation of VoiceXML, we have created a custom property named ***com.inin.securedata***. This new property will allow you to implement secure sessions for both recorded and logged data.

- If the property is set to **true**, the VoiceXML interpreter will assume that it is handling sensitive data, and so recognition handling and trace logging will be protected by obfuscating the data.
- If the property is set to **false** (or any other value but True), then the VoiceXML interpreter will assume that it is handling normal data.

Note: Keep in mind that for recordings to be properly secure during the input of sensitive data, you must Enable Secure Input in Interaction Administrator.

For more information, see the following:

- VoiceXML Interpreter Logging
- VoiceXML Interpreter's Use of Secure Sessions
- VoiceXML Host Server Logging
- IP Logging
- Enable Secure Input in Interaction Administrator
- Using the com.inin.securedata property

VoiceXML Interpreter Logging

If the com.inin.securedata property is set to True, then the methods listed below will protect their trace log messages.

Prompt-related traces

Confirmation prompts can contain sensitive data, so the majority of the traces containing prompt text are protected. The following methods contain such trace messages:

```
VXIPromptImpl::Queue()
PromptResource::queueToSystemLocalCache()
SessionResourceCache::queuePromptToSession()
GlobalResourceCache::getResource()
GlobalResourceCache::handleResFromRemote()
GlobalResourceCache::readResource()
GlobalResourceCache::uploadResToxIC()
GlobalResourceCache::cleanUpResources()
```

Special cases

```
cleanUpResources()
```

This method can trace out prompt file names from the cache, but they are not tied to a particular VoiceXML session. Furthermore, this method does not have access to the com.inin.securedata property, but it is not practical to completely remove these traces. As such, these log messages are only traced when the trace level is set to Verbose or higher.

Recognition-related traces

Speech recognition results can contain sensitive data, so traces containing the results from recognition requests are protected. The following methods contain such trace messages:

```
VXIRecognitionImpl::RecognizeMain()
VXILogImpl::ContentWrite()
```

Result-related traces

Information provided as VoiceXML document results can contain sensitive data, so traces containing such information are protected. The following method contains such trace messages:

`VXISession::translateDocResultToResults()`

Special cases

`VXISession::ProcessDocument( url )`

`VXISession::ProcessDocument( document )`

These two methods are only going to trace sensitive data if the handler passes the sensitive data into the document in the input name/value pairs.These traces are done before the VoiceXML document is actually entered, so we can't use the com.inin.securedata property to determine how to treat this data. As such, we will trace the incoming variables at the Verbose trace level and allow the logging trace level determine what gets logged.

BladewareVXML diagnostic traces

B ladewareVXML diagnostic messages can contain sensitive data, so traces containing such information are protected. The following method contains such trace:

`VXILogImpl::VDiagnostic()`

This method will only protect information from BladewareVXML diagnostic messages with the following tagID's:

TagIDs	Where they originate in BladewareVXML
3000, 3001, 3004, 3005	The .inet layer.
3500, 3501, 3502, 3503	The .cache layer.
4000	The .jsi layer.
8002	The .vxi layer.

<log> element traces

We will make no attempt to protect data from any <log> elements, as the inclusion of sensitive data in these logs is a VoiceXML application level decision. This includes Nuance NDM modules, which sometimes <log> sensitive information.

VoiceXML Interpreter's Use of Secure Sessions

When the com.inin.securedata property is set to True, the following methods will implement a secure session through the TSApi. More specifically, the secure session is enabled with the TSAPI::SecureSessionStart() method and disabled with the TSAPI::SecureSessionEnd() method. This applies to the following methods:

`VXIRecognitionImpl::RecognizeMain()`

`VXIPromptImpl::Wait()`

VoiceXML Host Server Logging

The VoiceXML Host Server, which runs on the CIC server, can trace sensitive data from several methods if the trace levels of the InterpreterSsnTopic and VXIHostSvcJobTopic are set to Notes or higher. Since the VoiceXML Host Server cannot access the com.inin.securedata property, the logging trace level determines what gets logged. This applies to the following methods:

`InterpreterSession::interpretDocument`

`InterpreterSession::interpretURL()`

`eVoiceXMLRequest_InterpretDocument`

`eVoiceXMLRequest_InterpretURL`

IP Logging

The IP subsystem that runs on the CIC server can trace sensitive data from several methods if the trace level of the I3_VoiceXmlTools topic is set to Notes or higher. Since the VoiceXML Host Server cannot access the com.inin.securedata property, the logging trace level determines what gets logged. This applies to the following methods:

`VoiceXmlInitiate()`

`VoiceXmlAsyncInitiate()`

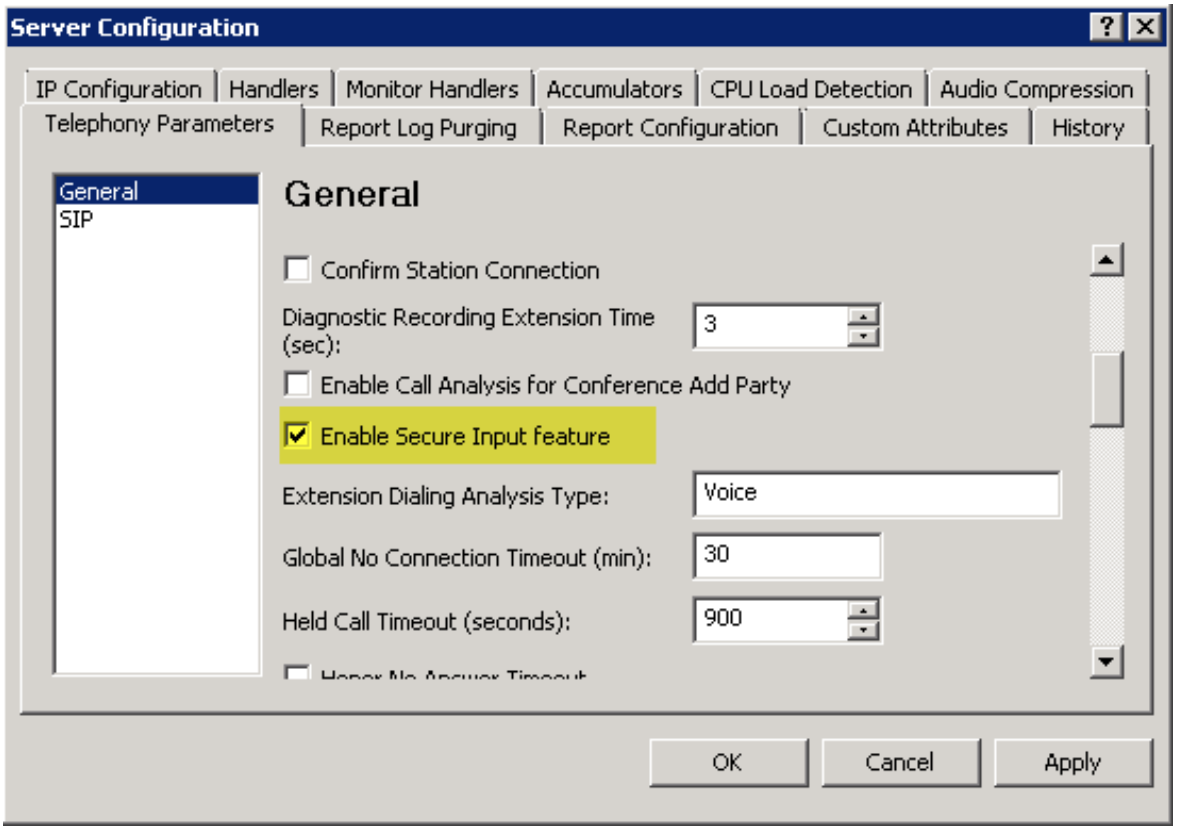
`VoiceXmlInitiateDoc()`

`VoiceXmlAsyncInitiateDoc()`

Enable Secure Input in Interaction Administrator

For recordings to be properly secure during the input of sensitive data, Secure Input needs to be enabled in Interaction Administrator. To enable secure input:

1. In Interaction Administrator, click the server name to display the Server container.
2. In the Server container, double-click Configuration to display the Server Configuration dialog.
3. In the Server Configuration dialog, click the Telephony Parameters tab and click General if both are not already visible.
4. Scroll down the General section and select the Enable Secure Input Feature check box.
5. Click OK.



Using the com.inin.securedata property

To use the com.inin.securedata property, you will set the value to **true** or **false** using the following syntax:

```
<property name="com.inin.securedata" value="true"/>
```

- If the property is set to **true**, the VoiceXML interpreter will assume that it is handling sensitive data, and so recognition handling and trace logging will be protected by obfuscating the data.
- If the property is set to **false** (or any other value but true), then the VoiceXML interpreter will assume that it is handling normal data.

The com.inin.securedata property adheres to the same rules set out for all VoiceXML properties as outlined in Section 6.3 Property Element in the VoiceXML 2.0 Specification. (http://www.w3.org/TR/voicexml20/#dml6.3)

> Properties may be defined for the whole application, for the whole document at the <vxml> level, for a particular dialog at the <form> or <menu> level, or for a particular form item. Properties apply to their parent element and all the descendants of the parent. A property at a lower level overrides a property at a higher level. When different values for a property are specified at the same level, the last one in document order applies. Properties specified in the application root document provide default values for properties in every document in the application; properties specified in an individual document override property values specified in the application root document.

Usage considerations

When using the com.inin.securedata property, there are several subtleties that you will need to be aware of:

- Any <prompt> contents that are queued in com.inin.securedata as true are not necessarily output right away. They can be output at the next user input stage or at the end of the document. As such, if the value of com.inin.securedata changes to false while the <prompt> contents are still in the queue and not yet output, then the contents of the prompt are not treated as sensitive data.Trace logs should be secure, but if the call is being recorded by the CIC server, that recording won't treat the data as sensitive.
- If sensitive data is passed back to the calling handler from an <exit> element, it is important to make sure that com.inin.securedata has a value of true when the <exit> is executed.Don't pass sensitive data in an <exit> element in a <form> where com.inin.securedata has a value of false.
- If sensitive information is handled in ECMAScript, it is important that this is done when com.inin.securedata has a value of true.
- Remember that a <subdialog> runs in a separate context from the calling document and the value of the com.inin.securedata property is not inherited from the calling document.The <subdialog> will either need to set the value of the com.inin.securedata property itself, or inherit it from a value set in the I3defaults.xml document. This can be inconvenient for those applications that use the Nuance NDM facilities, as they employ <subdialog> elements heavily.

Example

In order to help you effectively use the com.inin.securedata property in your VoiceXML documents, the following example script, which performs a simple output/input test while treating some data as secure, illustrates how the property is used.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
File:WhichDrink_secure.vxml
Date:07/03/2014
Author:WEB
Desc:This document does a simple output/input test whilst treating some data as secure.
Basic Steps:
- Play a text prompt
- Define an inline grammar
- Get input
- Play back the choice
-->
<vxml xmlns="http://www.w3.org/2001/vxml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/vxml
http://www.w3.org/TR/voicexml20/vxml.xsd"
version="2.0">
<var name="answer1"/>
<var name="answer2"/>
```

```
<var name="answer3"/>
<form>
<property name="com.inin.securedata" value="true"/>
<!--
We want to treat the "number", "drink" and "delight" information as sensitive data.
-->
<field name="number">
<prompt>
<audio src="http://billbairdpc4:8080/my-files/Suzanna.wav">
Alternative text for Suzanna prompt.
</audio>
Punch a two digit number.
</prompt>
<grammar src="builtin:dtmf/digits?minlength=2;maxlength=2"/>
</field>
<field name="drink">
<prompt>Would you like coffee, tea, milk, or nothing?</prompt>
<grammar version="1.0" mode="voice" root="drinklist">
<rule id="drinklist" scope="public">
<one-of>
<item>coffee</item>
<item>tea</item>
<item>milk</item>
<item>nothing</item>
</one-of>
</rule>
</grammar>
</field>
<field name="delight">
<prompt>What kind of tasty delight would you like?</prompt>
<grammar version="1.0" mode="voice" root="delightlist">
<rule id="delightlist" scope="public">
<one-of>
<item>cookies</item>
<item>crumpets</item>
<item>biscuits</item>
<item>nothing</item>
</one-of>
</rule>
</grammar>
</field>
<filled>
<prompt>You chose <value expr="number"/>, <value expr="drink"/> and <value
expr="delight"/>.</prompt>
<assign name="answer1" expr="drink"/>
```

```
<assign name="answer2" expr="delight"/>
<goto next="#weather"/>
</filled>
</form>
<form id="weather">
<property name="com.inin.securedata" value="false"/>
<!--
The weather information obtained in this form is not sensitive data.
Note that the <prompt> queued above ("You chose xxx and yyy.") is not actually issued until the
recognition input generated by the <form> below, and so is not treated as sensitive data.
Trace logs should be secure, but if the call is being recorded by the CIC server, then it won't
be secure.
-->
<field name="temperature">
<prompt>Is the weather hot or cold?</prompt>
<grammar version="1.0" mode="voice" root="templist">
<rule id="templist" scope="public">
<one-of>
<item>hot</item>
<item>cold</item>
</one-of>
</rule>
</grammar>
</field>
<field name="state">
<property name="com.inin.securedata" value="true"/>
<prompt>In which state do you live?</prompt>
<grammar version="1.0" mode="voice" root="statelist">
<rule id="statelist" scope="public">
<one-of>
<item>Indiana</item>
<item>Ohio</item>
</one-of>
</rule>
</grammar>
</field>
<field name="number2">
<prompt>
Punch a three digit number.
</prompt>
<grammar src="builtin:dtmf/digits?minlength=3;maxlength=3"/>
</field>
<filled>
<prompt>The weather is <value expr="temperature"/>.</prompt>
<assign name="answer3" expr="temperature"/>
```

```
<prompt>You entered <value expr="number2"/>.</prompt>
<goto next="#the_end"/>
</filled>
</form>
<form id="the_end">
<property name="com.inin.securedata" value="true"/>
<block>
<!--
answer1 and answer2 contain sensitive data, and since we want to pass this
information back to the handler, we must make sure that we <exit> from a
secure form.
-->
<prompt>Your answers are <value expr="answer1"/>, <value expr="answer2"/> and <value
expr="answer3"/>.</prompt>
<log>I am logging sensitive data (Suzanna): (<value expr="answer1"/> and <value
expr="answer2"/>)</log>
<exit namelist="answer1 answer2 answer3"/>
</block>
</form>
</vxml>
```

VoiceXML Example Pack

The VoiceXML Example Pack contains simple examples that illustrate different aspects of working with the IC VoiceXML Interpreter. The ReadMe.doc file, included in the VoiceXML Example Pack, describes the example files provided in the VoicexmlExampleFiles.zip file. The files included in the pack are illustrations of simple VoiceXML concepts and how to access VoiceXML in the PureConnect environment.

The following information describes the basic steps to run the VoiceXML scripts provided in the VoiceXML Example Pack. More detailed information about VoiceXML handler tools, host servers, interpreter servers, and other considerations are available in this document.

Prerequisites

The following are the prerequisites for running the VoiceXML scripts, available in the Example Pack:

- The VoiceXML feature is installed and configured, as documented in the *VoiceXML Installation and Configuration Guide*
- The user knows Interaction Designer and has permission to publish handlers
- The user has permission to run Interaction Attendant
- Access to the VoiceXML Example Pack zip file, VoicexmlExampleFiles.zip

Obtaining the Example Pack

The VoiceXML Example Pack is available from the PureConnect Speech FTP site at:

http://www3.inin.com/Products/ININSpeech/

This site requires a user name and password. To obtain these credentials, contact your Genesys partner or Genesys Customer Care. When you have logged on to the site, download the following file to your computer:

VoicexmlExampleFiles.zip

When you have downloaded the VoiceXML Example Pack, extract the contents of the file into a new folder. For example, you could extract the files into the C:\VoiceXMLDocs folder

Running a VoiceXML Script

The steps to run VoiceXML scripts included in the VoiceXML Example Pack are:

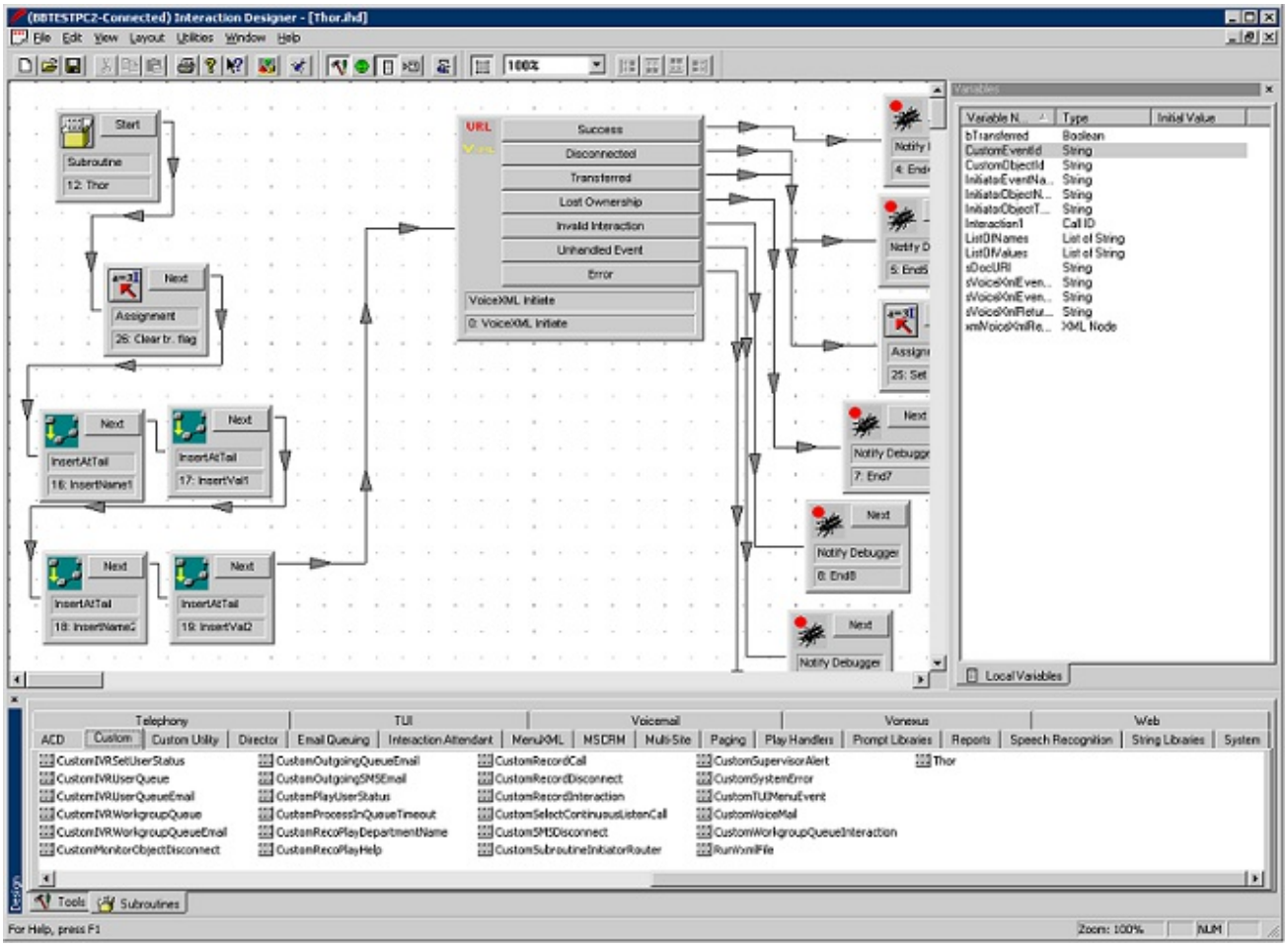
- Create a handler to access one of the VoiceXML tools
- Set up Interaction Attendant to activate the handler
- Call to access your script
- Experiment with different example files

Creating a handler

To create a Handler, you'll use Interaction Designer, which accesses one of the previous tools and calls a VoiceXML script.

To create a handler

1. Start Interaction Designer
2. From the **File** menu, select **Open**, and then navigate to the folder containing the Example Pack files
3. Select the file Thor.ihd and click **Open.**
    - Notice that this handler uses a Subroutine Initiator and calls the VoiceXML Initiate toolstep
4. Publish this handler by selecting the **Activate Handler** check box and choosing **Custom** as the Subroutine category
5. To verify that the Thor subroutine was successfully created, select the **Subroutines** tab at the bottom of the dialog, and choose the Custom page, as shown below, then, look for **Thor** in the list.

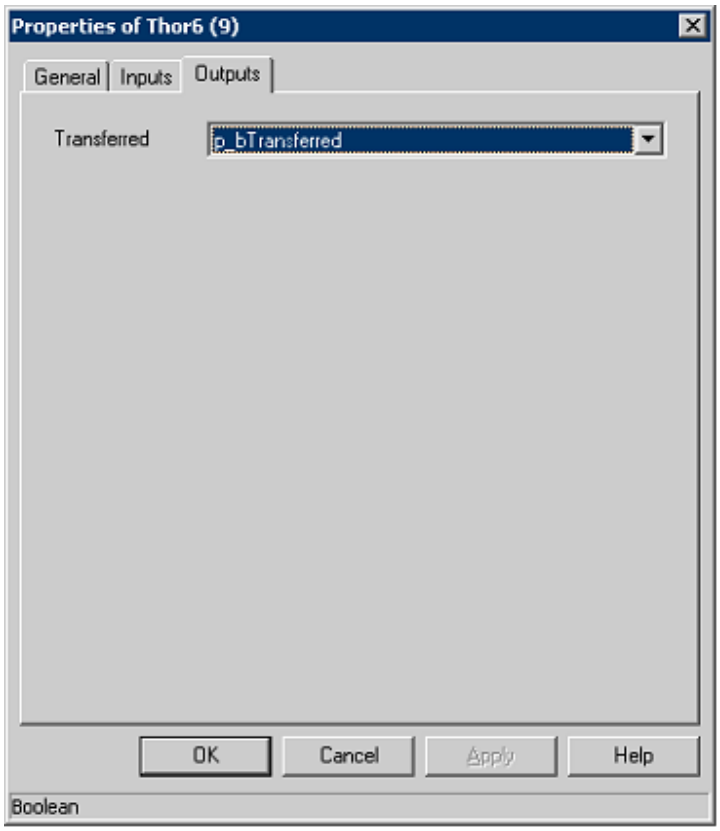


Modifying the custom subroutine handler

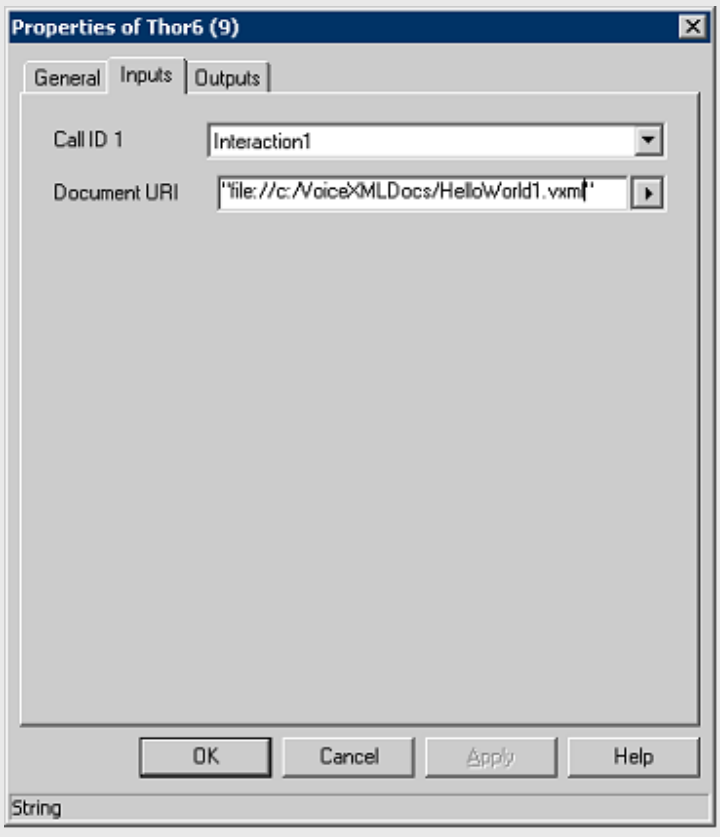
Next, modify the CustomSubroutineInitiatorRouter.ihd file to allow use of the Thor subroutine that was created in the previous steps.

1. From the File menu, click **Open**, and then navigate to the CustomSubroutineInitiatorRouter.ihd file, which can be found in the CIC install path in the \Handlers\40Handlers folder, and click **Open**.

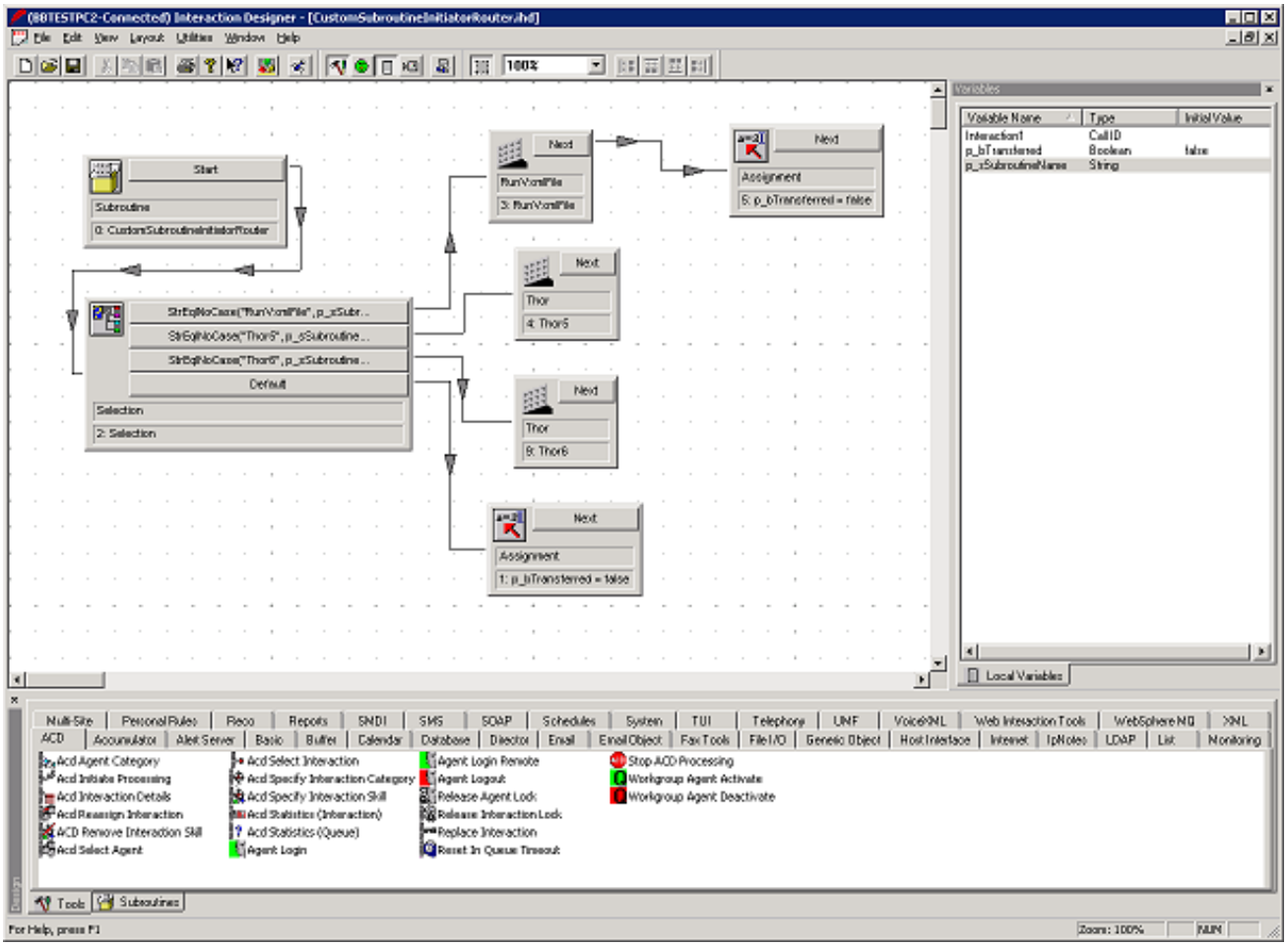
   Note:
   Your CIC install path will be either `\I3\IC` or `\Program Files (x86)\Interactive Intelligence`
2. In the Selection toolstep, add a **StrEqlNoCase("Thor6," p_sSubroutineName)** statement.
3. Add a Thor subroutine toolstep into the handler.
4. Connect the StrEqlNoCase statement with the Thor subroutine that you just added.
5. On the Outputs page, select **p_bTransferred** from the Transferred drop down.



On the Inputs page, select **Interaction1** from the Call ID 1 drop down and select `file://c:/VoiceXMLDocs/HelloWorld1.vxml` from Document URI. (The exact path depends on where files in the Example Pack have been extracted)



Publish the handler.



Testing the handler

To test the Thor Handler, make a call.

1. Dial into your main IVR (press * from your workstation)
2. At the main IVR prompt, press 6
3. The executing script plays, "Hello World"

Setting up Interaction Attendant

To access the Thor Handler, set up Interaction Attendant.

1. Start Interaction Attendant
2. From the Active Schedule, select Insert/New Operation/Run Subroutine
3. Set the Name to Subroutine Initiator – Thor6
4. Select the Digit 6 Six
5. Set the Subroutine to Thor6
6. Publish the schedule

Experimenting with other scripts

Now that you have created a Handler to run a VoiceXML script, you can start experimenting with other VoiceXML scripts available from the Example Pack. Or you can use your own VoiceXML scripts. Here's how to run more scripts:

1. Start Interaction Designer
2. From the File menu, click **Open** and then navigate to the CustomSubroutineInitiatorRouter.ihd file and click **Open.**
3. Edit the Thor toolstep as follows:

On the Inputs page, set Document URI to `file://c:/VoiceXMLDocs/XXX.vxml` (the exact path depends on where files in the Example Pack have been extracted)

4. Publish the Handler

General VoiceXML Considerations

Session variables

The following table lists the session variables that are supported, and from where their data is obtained:

Property	Data source
session.connection.local.uri	Eic_LocalTnRaw call attribute
session.connection.remote.uri	Eic_RemoteTnRaw call attribute
session.connection.protocol.name	Eic_ProtocolId call attribute
session.connection.protocol.version	' '
session.connection.redirect	Not currently set
session.connection.aai	Not currently set
session.connection.originator	session.connection.remote.uri if call is incoming session.connection.local.uri if call is outgoing ' ' if neither
session.inin.localMachineName	Host computer of VoiceXML interpreter
session.inin.interpreterName	VoiceXML interpreter instance on a host computer

ININ-specific errors

In addition to the "standard" errors that are specified in the VoiceXML 2.1 standard, VoiceXML Interpreter can return a number of ININ-specific errors, which are specific to the Genesys (formerly Interactive Intelligence) PureConnect implementation.

ININ-specific errors include:

- error.com.inin.lostownership
- error.com.inin.processdoc
- error.com.inin.initiation.session
- error.com.inin.hostsvr.deserialize.vxiinterpretdocreq
- error.com.inin.hostsvr.deserialize.vxiinterpreturlreq
- error.com.inin.hostsvr.interpretermgr.nosessions
- error.com.inin.hostsvr.interpretermgr.nullsession
- error.com.inin.hostsvr.interpretermgr.sessionnotfound
- error.com.inin.hostsvr.interpretermgr.loginfailed
- error.com.inin.hostsvr.interpretermgr.deserialize.vxiacceptsessionsreq
- error.com.inin.hostsvr.interpretermgr.deserialize.vxiloginreq
- error.com.inin.hostsvr.interpretermgr.deserialize.vxilogoutreq
- error.com.inin.hostsvr.interpreterssn.deserialize.vxiuploadfilereq
- error.com.inin.hostsvr.interpreterssn.deserialize.vxidownloadfilereq
- error.com.inin.hostsvr.interpreterssn.deserialize.vxiexchangestreamendpoints
- error.com.inin.hostsvr.interpreterssn.deserialize.vxireservefilenamereq
- error.com.inin.hostsvr.interpreterssn.deserialize.vxireleasefilereq
- error.com.inin.hostsvr.interpreterssn.nolicense
- error.com.inin.hostsvr.interpreterssn.interpreturl.vxiserver
- error.com.inin.hostsvr.interpreterssn.upload.fileNotReserved
- error.com.inin.hostsvr.interpreterssn.upload.streamopenfailure
- error.com.inin.hostsvr.interpreterssn.upload.createfilefailure
- error.com.inin.hostsvr.interpreterssn.upload.transferfailure

<log> messages

The output from VoiceXML <log> tags is sent to the voicexmlserver.ininlog trace log file “voicexmlserver.ininlog” under the “VXILogTopic.” Because BladewareVXML considers these <log> messages as “Diagnostic” messages, set the trace level to 90 (Verbose Notes) or higher to see these messages in the trace log file.

Passing data between VoiceXML and Handlers

Examples of passing data between VoiceXML and handlers are available in the VoiceXML Example Pack available from the Utilities and Downloads page at: https://my.inin.com/products/cic/Pages/Utilities-Downloads.aspx This site requires standard Genesys credentials, the same as logging in to the Customer Care portal.

Input parameters

The VoiceXML handler tools have two input parameters used to pass information into a VoiceXML script:

Parameter	Description
Argument Values	An optional string list of argument names.
Argument Names	An optional string list of argument values.

Together the arguments are taken to form a set of name/value pairs that are available to the VoiceXML script as variables of the form “session.name” or just “name.” So, if a handler loads the following strings into the Argument Names parameter:

```
“Person”
“Description”
```

and the following strings into the Argument Values parameter:

```
“Suzanna”
“a beautiful young lady”
```

before calling the VoiceXML Initiate tool, then the following two variables are available to the designated VoiceXML script:

```
session.Person(or just Person)
session.Description(or just Description)
```

Here is an example of a VoiceXML script using these variables:

```
<form>
<block>
Good morning, <value expr="Person"/>
is <value expr="Description"/>.
</block>
<block>
Good afternoon, <value expr="session.Person"/>
is <value expr="session.Description"/>.
</block>
</form>
```

The example handler Thor.ihd and the VoiceXML script file TestVars.vxml from the VoiceXML Example Pack illustrate this scenario.

Output parameters

The VoiceXML handler tools have two output parameters that are used to receive information from a VoiceXML script:

Parameter	Description
Return Value	A string that is the result of the expr attribute of the <exit> element
Result Data	An XMLNode created from the data specified in the namelist element of the <exist> element.

So if the following VoiceXML script is running:

```
<form>
<field name="drink">
<prompt>Would you like coffee, tea, milk, or nothing?</prompt>
<grammar type="application/srgs+xml" src="drink.grxml"/>
</field>
<filled>
<prompt>You chose <value expr="drink"/>.</prompt>
<exit expr="drink"/>
</filled>
</form>
```

then when the <exit> element is executed, the information in the drink variable-for example, "tea"-would be returned in the Return Value output parameter of the calling handler tool.

The VoiceXML script file WhichDrink2a.vxml (the Beverage Test) illustrates this concept.

If the following VoiceXML script is executed:

```
<filled>
<prompt>You have ordered <value expr="number"/> <value expr="size"/> pizzas with <value
expr="toppings"/>.
</prompt>
<exit namelist="number size toppings"/>
</filled>
```

then when the <exit> element is executed, the information in the number, size, and toppings variables is returned in XMLNode form. For example:

```
<ResultData>
<toppings>pepperoni</toppings>
<number>3</number>
<size>large</size>
</ResultData>
```

in the Result Data output parameter of the calling handler tool.

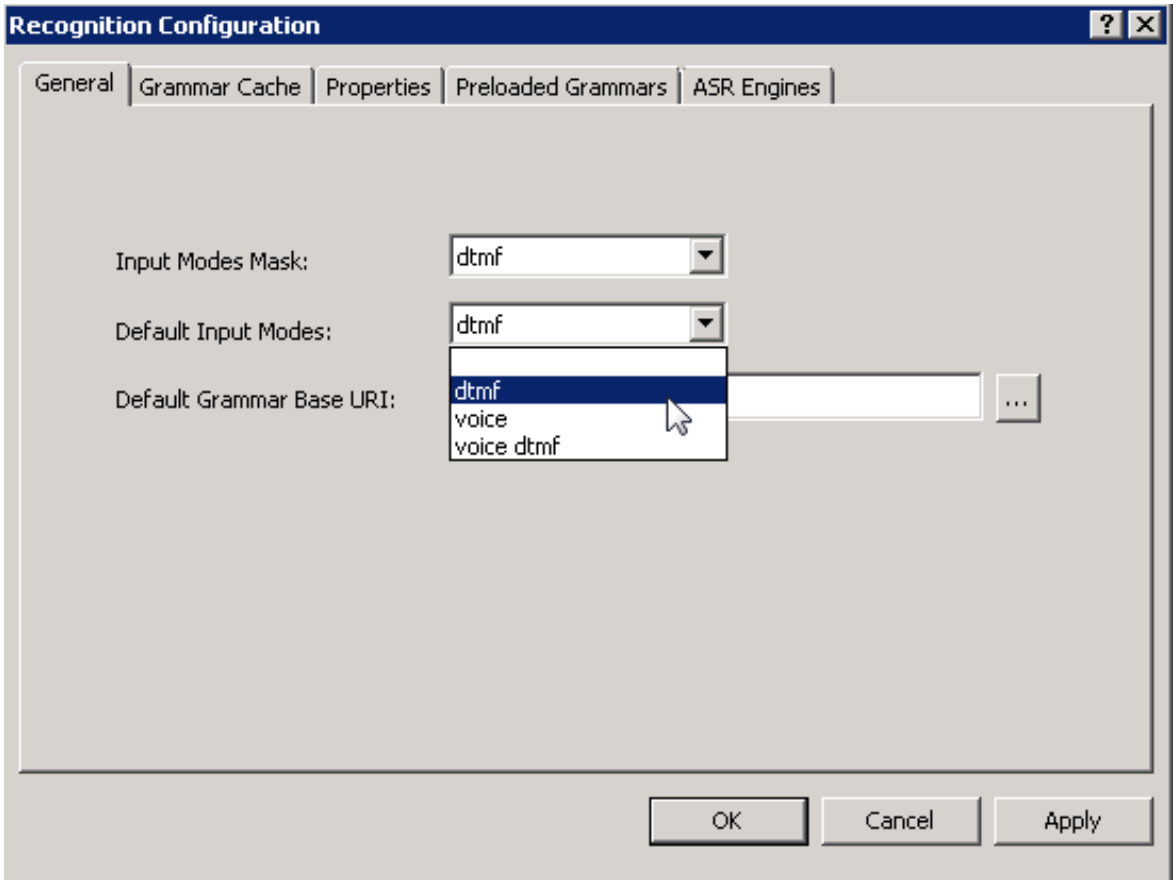
The VoiceXML script file zoran_test.vxml (the Pizza Test) illustrates this concept.

Using VoiceXML without ASR

If you are not using an ASR server, you can configure the VoiceXML interpreter to only accept DTMF input by changing a couple of input mode settings in Interaction Administrator and by altering the input mode setting in your VoiceXML scripts.

To set DTMF as the input mode in Interaction Administrator:

1. On the CIC server, open Interaction Administrator.
2. In the left pane of the Interaction Administrator window, select the **System Configuration\Recognition** container.
3. In the right pane, under the System Configuration column, double-click **Configuration**.
4. When you see the Recognition Configuration dialog box, select the **General** tab.
5. Select **dtmf** from both the Input Modes Mask and Default Input Modes drop downs.



Note: These settings affect all recognition, not just recognition related to VoiceXML.

To set DTMF as the input mode in VoiceXML scripts, set the inputmodes property to "dtmf":

```
<property name="inputmodes" value="dtmf"/>
<prompt>
For news press 1, For weather press 2, For sports press 3.
</prompt>
```

File-based Grammar Types

The grammar type that VoiceXML tells the Reco subsystem about for a file-based grammar is determined as follows:

1. If the type is specified in the <grammar> element in the VoiceXML script, then that type is used as the grammar type.
2. If ignoreGrammarFileMimeType is false or unspecified,
   a. If the file has a MimeType associated with it, that MimeType is used as the grammar type
   b. If the file does NOT have a MimeType associated with it, the configured defaultGrammarMimeType is used as the grammar type.
3. If ignoreGrammarFileMimeType is true, the configured defaultGrammarMimeType is used as the grammar type.

Note: If the Reco subsystem is familiar with the type of Voice grammar (application/srgs and application/srgs+xml), it tries to parse them. For types that it doesn't know, the Reco subsystem sends the grammars on to the recognition engine for parsing.

For DTMF grammars, the Reco subsystem tries to parse the grammar, and if the type is one with which it is not familiar, it objects.It never passes DTMF grammars on to the recognition engine.

Supported VXML Data Elements

The <data> element enables a VoiceXML application to fetch or post data to a document sever without transitioning to a new VoiceXML document. The VXML2.1 specification defines the attributes supported by the <data> element. The "enctype" attribute represents the encoding or MIME type of the document submitted to the document server.

PureConnect supports two JSON encoding types ("enctype" attributes) to be used with the <data> element: "application/json" and "application/json-inin". PureConnect also supports using the PUT HTTP request method with the <data> element. The content sent by the VoiceXML interpreter in this context is identical for the POST and PUT methods, which are described together.

JSON Data Elements

The two similar but different JSON enctype data elements provide support for the VoiceXML document to have complete control of the JSON in the body, depending on the requirements of the webservice the application is working with. The primary difference between these two encoding types is the following:

JSON enctype	Description
application/json	The JSON body is constructed automatically using the ECMAScript variable names specified in namelist attribute, along with the data in those variables. Use this for the simple case where the variable names in the script are also valid for urlencoding and can be passed through to the webservice. If the variable names should not be passed through, use the application/json-inin enctype.
application/json-inin	The JSON body consists of the data from the first ECMAScript variable named in the namelist attribute (which is assumed to be a string containing properly formatted JSON). Use this when you want to pass the data but not the script variable names to the webservice. In this case, you must provide the data in well formed JSON format.

Content for the "application/json" enctype

When the interpreter traverses a <data> element with "application/json" encoding type, it creates a single object containing a member for each variable, whether using the POST or PUT method. The member names are the variable names and member values are the corresponding variable values. When the interpreter sends the POST or PUT request, the request body contains that internal object converted to string. Note that variable names in the "namelist" attribute of the <data> element are submitted in this case to the document server.

For example, assume a script defines the following variables:

```
var person1 = {
    firstName: "Bob",
    lastName: "Smith",
    age: 32
};
var location1 = {
    city: "Indianapolis",
```

```
    state: ”Indiana”
};
```

When the interpreter traverses the following <data> element:

```
<data src=”http://www.example.org” method=”post” enctype=”application/json”
namelist=”person1 location1”/>
```

the interpreter sends to the server located at URI “http://www.example.org” an HTTP POST request with the following body:

```
{
    ”person1”: {
        ”firstName”: ”Bob”,
        ”lastName”: ”Smith”,
        ”age”: 32
    },
    ”location1”: {
        ”city”: ”Indianapolis”,
        ”state”: ”Indiana”
    }
}
```

Content for the "application/json-inin" enctype

When the interpreter traverses a <data> element with “application/json-inin” encoding type, it sends as content of the POST or PUT request the content of the variable specified, with a "namelist" attribute having the name of an ECMAScript variable. Note that the name of the variable in the “namelist” attribute of the <data> element is not submitted in this case to the server.

For example, assume a script defines the following variable:

```
var person2 = '{ ”firstName”: ”Bob”, ”lastName”: ”Smith”, ”age”: 32}';
```

When the interpreter traverses the following <data> element:

```
<data src=”http://www.example.org” name=”response” method=”put” enctype=”application/json-
inin” namelist=”person2”/>
```

the interpreter sends to the server located at URI “http://www.example.org” a PUT request with the body equivalent to:

```
{ ”firstName”: ”Bob”, ”lastName”: ”Smith”, ”age”:32 }
```

Change Log

The following table lists the changes to the *VoiceXML Technical Reference* since its initial release.

Date	Changes
13-August-2012	Document created-version 1.0
04-October-2012	Added descriptions of the `default GrammarMimeType` and `ignore GrammarFileMimeType` parameters
29-January-2013	Document Version: 1.0.1 released for CIC 4.0 SU3
19-July-2013	Merged Developer's TR with Overview TR, reformatted document design, updated cross-references, edited to conform to MMOS
05-June-2014	• Updated Copyright page. • Added information about the ability to disable external accessibility to the web configuration page of a VoiceXML server.
01-August-2014	Updated documentation to reflect changes required in the transition from version 4.0 SU# to CIC 2015 R1, such as updates to product version numbers, system requirements, installation procedures, references to Interactive Intelligence Product Information site URLs, and copyright and trademark information.
10-February-2015	• Updated Copyright page. • Reorganized the document layout. • Added a section containing information about adding Data Security to VoiceXML scripts via the `com.inin.securedata` custom property.
20-April-2015	• Added an optional parameter to the VoiceXML server configuration file. • Updated Copyright and Trademarks page. • Updated the document to reflect the CIC 2015R3 version.
12-June-2015	• Added optional `grammarCaching` parameter to the VoiceXML server configuration file. • Updated cover page to reflect new color scheme and logo. Updated copyright and trademark information.
09-October-2015	• Updated the description of the `ttsSSML` configuration parameter. • Updated documentation to reflect 2016 R1 Release
04-February-2016	• Updated Copyright and Trademarks for 2016. • Updated the document to reflect the CIC 2016 R2 version. • Added a link to the CIC Documentation Library at Help.inin.com.
24-August-2016	Added `session.inin.localMachineName` and `session.inin.interpreterName` session variables in "Session Variables" section.
10-May-2018	Rebranded from Interactive Intelligence to Genesys.
12-June-2019	Reorganized the content only, which included combining some topics and deleting others that just had an introductory sentence such as, "In this section...".
25-June-2019	Added a note to VoiceXML Interpreter Server Web Configuration Interface about how to reset the password for the VoiceXML Interpreter Server Web Configuration interface.
1-August-2019	Added a topic for Supported VXML Data Elements for new JSON enctype attribute support in data elements. Added an opening paragraph in the Introduction to VoiceXML. Change the link in the VoiceXML Example Pack to point to a download on the Product Information Site as the old FTP link was invalid.