GENESYS®
AN ALCATEL-LUCENT COMPANY

VoiceGenie 7.2.2

Media Platform

User's Guide

About Genesys

Genesys Telecommunications Laboratories, Inc., a subsidiary of Alcatel-Lucent, is 100% focused on software for call centers. Genesys recognizes that better interactions drive better business and build company reputations. Customer service solutions from Genesys deliver on this promise for Global 2000 enterprises, government organizations, and telecommunications service providers across 80 countries, directing more than 100 million customer interactions every day. Sophisticated routing and reporting across voice, e-mail, and Web channels ensure that customers are quickly connected to the best available resource—the first time. Genesys offers solutions for customer service, help desks, order desks, collections, outbound telesales and service, and workforce management. Visit www.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Technical Support website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other trademarks and trade names referred to in this document are the property of other companies. The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Technical Support at the following regional numbers:

Region	Telephone	E-Mail
North America and Latin America	+888-369-5555 or +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44-(0)-127-645-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Japan	+81-3-6361-8950	support@genesyslab.co.jp

Prior to contacting technical support, please refer to the Genesys Technical Support Guide for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the Genesys 7 Licensing Guide.

Released By

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com
Document Version: 10-2009

Table of Contents

Chapter 1

Chapter 2

Chapter 3

Chapter

1 Introduction

This is the *User's Guide* for the VoiceGenie 7.2 Media Platform product. It is provides an overview of the VoiceGenie 7.2 media platform architecture and capabilities, and describes how to provision, monitor and maintain the system.

1.1 Terminology

The following table gives definitions of some acronyms that are used throughout this document:

Acronyms	Full Definitions
ASR	Automated Speech Recognition (Engines/Technologies)
CLC	Command Line Console – A command line interface that can be used to query information and issue commands
MRCP	Media Resource Control Protocol – Adopted by the VoiceGenie Media Platform to control ASR and TTS resources
SRM	Speech Resource Management – A component integrated into the VoiceGenie Media Platform to provide Speech Recognition and Synthesis functionalities to the application developers
SMC	System Management Console – A web based tool for administering clusters of VoiceGenie VoiceXML Platforms
OA&M	Operation, Administration and Management
TTS	Text To Speech (Engines/Technologies)

The following sections may contain references to terminology that has become:

Historical Terms	New Terms
PhoneWeb Software / NeXusPoint 6.4.x Software	VoiceGenie 7.2 Software
Cluster Management Platform (CMP)	OA&M Framework
Voice Resource Manager (VRM)	Speech Resource Management (SRM)
VoiceGenie Management Console (VMC)	System Management Console (SMC)

1.2 Document Structure

This document is intended to provide a complete resource for information regarding the VoiceGenie 7.2 platform. The remainder of this document is structured as follows:

- **Introduction** – A brief introduction to the VoiceGenie Media Platform
- **VoiceGenie 7.2 System Overview** – Explains the overall VoiceGenie 7.2 System Architecture, and provides information for various modules within the Media Platform as well as how the Media Platform interacts with other VoiceGenie components.
- **Running Applications on the Media Platform** – Instructs users on how to use the Media Platform to run their desired applications. The section explains how applications are mapped based on the DNIS information in incoming calls on the platform, as well as the caching/fetching mechanisms of the call manager component.
- **Network Interfaces** – A description of the telephony interfaces to the platform and supported telephony technologies, including PSTN, VoIP and Hybrid setups of the two. Specific features of each line manager will be discussed in details. Application provisioning is covered in this section.
- **Call Control** – A description of the platform basic call control support capabilities, including VoIP and PSTN interfaces.
- **Call Transfer** – A description of the platform advanced call control support capabilities and an overview of extended features
- **Other Features** – Covers other Media Platform capabilities such as Remote Dial, Call Analysis, Full Call Recording and Conferencing.
- **Operations** – Provides information about system logging/billing ("application metrics"), alarming, health status checking and system maintenance.

1.3 Further Information

1.3.1 VoiceGenie Documents

The following VoiceGenie documents provide additional information regarding the VoiceGenie Gateway.

- *VoiceGenie 7.2 Installation Guide* – Information regarding the installation and deployment of the VoiceGenie platform.
- *VoiceGenie 7.2 Media Platform System Reference Guide* – A guide that provides in-depth references to configuration information, metrics and alarm entries of the VoiceGenie 7.2 Media Platform.
- *VoiceGenie 7.2 OA&M Framework User's Guide* – A guide to the use of OA&M Framework and the user interfaces. Along with this guide, additional guides are available for each component in the OA&M Framework, including the Cluster Management Platform (CMP); the System Management Console (SMC) which is a web based tool for administering clusters of VoiceGenie VoiceXML Platforms; the Command Line Console (CLC) which is a command line interface that can be used to query information and issue commands and the SNMP Agent.
    - *VoiceGenie 7.2 OA&M Framework – SMC User's Guide*
    - *VoiceGenie 7.2 OA&M Framework – CLC User's Guide*
    - *VoiceGenie 7.2 OA&M Framework – SNMP User's Guide*
- *VoiceGenie 7.2 Speech Resource Management User's Guide* and *VoiceGenie 7.2 Speech Resource Management System Reference Guide* – The guides that covers all the information for the VoiceGenie 7.2 Speech Resource Management components.
- *VoiceGenie 7.2 Media Platform Release Notes* – Contains information regarding known issues, fixed problems, and behavioral information regarding the VoiceXML Gateway.
- *VoiceGenie 7.2 Application Migration Guide* – A guide that outlines the new features and changes in the VoiceGenie 7.2 release that may require changes to existing VoiceXML applications.

Additional documentation may be provided with particular system releases.

1.3.2 Third Party Documents

Each supported speech engine may include vendor provided documentation. These documentation sets are available from Genesys.

1.3.3 Genesys Web Sites

Genesys maintains a number of web sites to support developers and customers:

- `http://developer.voicegenie.com` – A free online resource for documentation, testing of VoiceXML applications, and collaboration between VoiceXML developers.
- `http://support.voicegenie.com` – The support resource for Genesys customers, including documentation, tutorials, and other information.
- `http://www.genesyslab.com` – Genesys's corporate web site, providing case studies, white papers, and general information regarding Genesys.

1.3.4 Third Party Web Sites

There are a number of third party web resources that provide useful information regarding Genesys and VoiceXML.

- `http://www.w3c.org/Voice` – The Voice Browser Working Group of the World Wide Web Consortium (W3C) – this group holds primary responsibility for the evolution of Voice related technologies such as VoiceXML, and the Speech Recognition Grammar and Speech Synthesis language specifications.
- `http://www.voicexml.org` – The VoiceXML Forum web site – The VoiceXML Forum is an industry consortium of VoiceXML supporters, responsible for activities such as education and conformance, as well as the initial evolution of the VoiceXML specification.
- `http://www.voicexmlreview.org` – An on-line magazine providing information regarding VoiceXML and VoiceXML applications, published by the VoiceXML Forum.

1.3.5 Related Standards and Specifications

The following specifications are published and maintained by the W3C Voice Browser Working Group:

- VoiceXML 2.0 Specification
- Speech Recognition Grammar Specification
- Speech Synthesis Markup Language Specification

The VoiceGenie platform is based on open standards – as a result, the platform provides complete or subset support for many Requests for Comments (RFCs) published and maintained by the Internet Engineering Task Force (IETF – `http://www.ietf.org`). These include:

- RFC 1738 Uniform Resource Locators

- RFC 1808 Relative Uniform Resource Locators
- RFC 1867 Form-based File Upload in HTML
- RFC 2109 HTTP State Management Mechanism
- RFC 2190 RTP Payload Format for H.263 Video Streams
- RFC 2388 Returning Values from Forms: multipart/form-data
- RFC 2326 Real Time Streaming Protocol (RTSP)
- RFC 2396 Uniform Resource Identifiers (URI): Generic Syntax
- RFC 2429 RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H263+)
- RFC 2616 Hypertext Transfer Protocol – HTTP/1.1 (subset)
- RFC 2806 URLs for Telephone Calls
- RFC 2833 RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals
- RFC 2964 Use of HTTP State Management
- RFC 2965 HTTP State Management Mechanism
- RFC 2976 The SIP INFO Method
- RFC 3261 SIP: Session Initiation Protocol (subset)
- RFC 3264 An Offer/Answer Model with the Session Description Protocol (SDP)
- RFC 3267 Real Time Transport Protocol (RTP) Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs
- RFC 3515 The SIP REFER Method
- RFC 3550 RTP: A Transport Protocol for Real-Time Applications
- Most extensions from proposed IETF draft “A SIP Interface to VoiceXML Dialog Servers” `(http://tools.ietf.org/id/draft-burke-vxml-02.txt)`
- Some extensions from proposed IETF draft “Basic Network Media Services with SIP” (`http://www.ietf.org/rfc/rfc4240.txt`)

The platform also includes complete support for many network related (i.e. TCP/IP, SNMP) and other protocols. Contact Genesys for additional details.

Chapter

2 VoiceGenie 7.2 System Overview

2.1 The Media Platform

The VoiceGenie Media Platform provides the infrastructure for supporting interpreted dialog management services suitable for deployment by carriers and enterprises, including related operational and management tools.

The platform provides a number of services:

- Call control and state management
- Media routing and negotiation
- Interpreter dialog execution and state management, using standard languages such as VoiceXML
- Complete HTTP document management, including advanced caching and proxy support

2.1.1 Call Manager

The Call Manager is a major subcomponent of media platform responsible for call state management. The Call Manager is responsible for call control and media routing for the platform. The Call Manager is designed to support multiple 'line managers' (call control interfaces – i.e. SIP, and H.323) simultaneously.

The Call Manager provides call signaling and media transport across a number of underlying protocols and networks. Call signaling interfaces are used to establish and terminate calls, and to perform advanced call control functions such as call transfer, and call conferencing. Media transport functionality allows media to be exchanged with the end user on established calls, enabling playback of stored audio/video and TTS, recording, and ASR.

The following summarizes the basic functionalities:

- Call setup
- Call teardown
- Messaging to the network infrastructure
- In-band and/or out-of-band reporting of DTMF
- Overall channel state management
- Possibly test/maintenance functionality
- Media routing and characteristic negotiation

The Call Manager subsystem includes the following components:

- **CMAPI (Call Manager API) and Applications** – an interface to the call manager that is intended to enable the integration of arbitrary interpreters and applications with an underlying platform abstracted by the Call Manager. This API encapsulates the entire functionality of the VoiceGenie platform, providing an application (or interpreter delivering an application) with the following key groups of capabilities:
    - Basic, Q.931-like call control (accepting or rejecting calls, placing outbound calls)
    - Enhanced call control (call join, call transfer, network-based conferencing, etc).
    - DTMF detection and generation
    - Media playback (locally stored, cached, streamed via HTTP)
    - Media recording (multiple concurrent recordings, background mixed recording)
    - Media switching between calls (join/release, TLT, bridge transfers)
    - Advanced media services (e.g. local conferencing)
    - Speech recognition via VoiceGenie's SRM component
    - Text-to-speech output via VoiceGenie's SRM component

  In addition to providing the above capabilities, the Call Manager API also provides application management capabilities that allow applications created using different languages and technologies to interact with a call, concurrently in some cases. In future releases of VoiceGenie platforms, this will enable VoiceXML applications and technologies such as SALT to be delivered by a single instance of the platform, and will enable CCXML applications to invoke and control VoiceXML applications.
- **Line Managers** – each Line Manager provides a protocol specific interface to an underlying call control mechanism. Currently supported Line Managers include SIP and H323.
- **Media Transports** – media transport is available using RTP streams.

2.1.2 The VoiceXML Interpreter

The VoiceXML Interpreter is a major subcomponent of the Media Platform responsible for parsing, understanding, and executing application written in VoiceXML, a dialog description language. The VoiceGenie 7.2 Media Platform supports two VoiceXML Interpreters: the Legacy Interpreter (VXMLi) and the Next Generation Interpreter (NGI).

The Legacy Interpreter is the latest version of the VoiceXML Interpreter that has been used in pre-7.2 VoiceGenie Media Platforms. It is a mature software component that handles millions of calls every day in deployments around the world.

The Next Generation Interpreter is the latest VoiceXML Interpreter with a number of improvements over the Legacy Interpreter. It is re-architected to leverage today's multi-core, multi-processor systems. The new architecture is also well suited for the forthcoming standards such as VoiceXML 3.

Both Interpreters implement the VoiceXML 2.1 language as specified by the Voice Browser Working Group of the W3C (`http://www.w3.org/Voice`). VoiceXML applications and the Interpreter allow the application authors to write applications in a high-level, portable language without the need to understand the various complicated low-level telephony protocols. In addition, the VoiceXML language and the Interpreter uses file-based and HTTP protocols to deliver applications from the application server to the Interpreter. This allows the user of the VoiceGenie Media Platform to re-use their existing web infrastructure for their Voice Gateway.

The following summarizes basic functionalities:

- Issues commands to the Fetching Module to acquire VoiceXML applications
- Parse and executes VoiceXML applications, according to the VoiceXML specificcation
- Issues commands to the Call Manager to execute call and media operations, such as prompt playing, recording, speech recognition, call transfer, etc.
- Handles Call Manager responses and continues the application execution.
- Performs DTMF recognition

2.1.3 The Fetching Module

The Fetching Module is a major subcomponent of the Media Platform responsible for performing fetching. It uses shared memory to communicate between itself and the interpreter and the call manager. The Interpreter issues HTTP requests to the fetching module, and after the fetching module acquires the resource it will respond back to the Interpreter. The fetching module connects to an external http proxy (squid) for HTTP fetching. The fetching module also does in-memory caching and sharing of these HTTP requests, which allows for more efficient operation.

The following summarizes basic functionalities:

- Receives HTTP and File-based requests from the Interpreter, and returns the results in the same format back to the Interpreter.
- Connects to squid for HTTP requests, while accessing the file directly for file-based requests.
- Performs shared-memory caching.

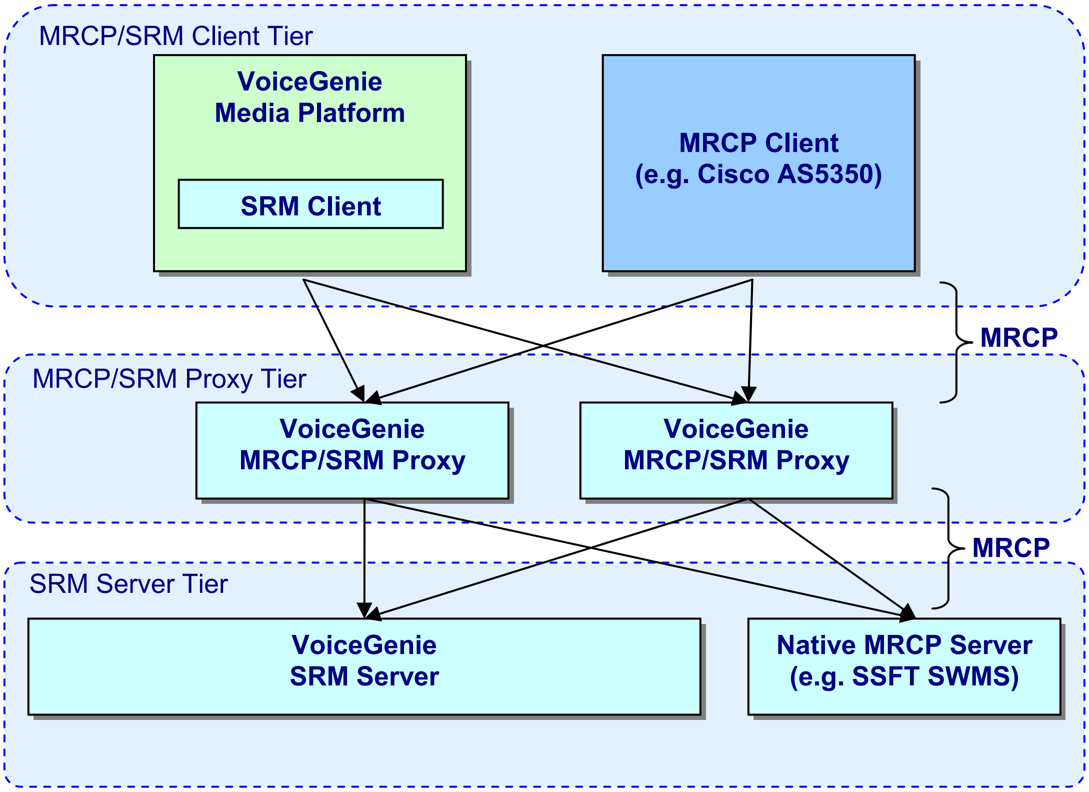
2.2 Speech Resource Manager (SRM)

With modern day telephony applications, it is no longer sufficient to provide services to customers using only touch-tones input and pre-recorded audio. VoiceGenie Media Platform can accept speech as user input and can provider dynamically generated prompts by interacting with the Speech Resource Management (SRM) component.

The SRM is used to manage 3rd party Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) engines. It consists of three components: the SRM client component, the SRM server component and MRCP proxy component. Using SRM, VoiceGenie Media Platform can work with the different TTS/ASR products provided by different vendors.

The SRM client component is integrated to the VoiceGenie Media Platform as a Dynamic Link Library. The Speech Resources (i.e. the TTS/ASR servers) are provisioned to Media Platform via System Management Console. The Media Platform uses the SRM client library to access ASR/TTS functionality based on the Media Resource Control Protocol (MRCP).

The diagram below depicts the Media Platform's position in a the SRM architecture, with the Media Platform being a SRM client:


MRCP/SRM Client Tier
VoiceGenie Media Platform
SRM Client
MRCP Client (e.g. Cisco AS5350)
MRCP
MRCP/SRM Proxy Tier
VoiceGenie MRCP/SRM Proxy
VoiceGenie MRCP/SRM Proxy
MRCP
SRM Server Tier
VoiceGenie SRM Server
Native MRCP Server (e.g. SSFT SWMS)

For details of the SRM components, please refer to the following documents:

- *VoiceGenie 7.2 Speech Resource Management User's Guide*
- *VoiceGenie 7.2 Speech Resource Management System Reference Guide*
- *VoiceGenie 7.2 MRCP Proxy User's Guide*
- *VoiceGenie 7.2 MRCP Proxy System Reference Guide*

2.3 Operation, Administration and Management (OA&M)

The VoiceGenie Media Platform relies upon a combination of on board and distributed OAM&P tools to provide support for management of the platform, including logging, provisioning, alarming and other maintenance operations. These components include:

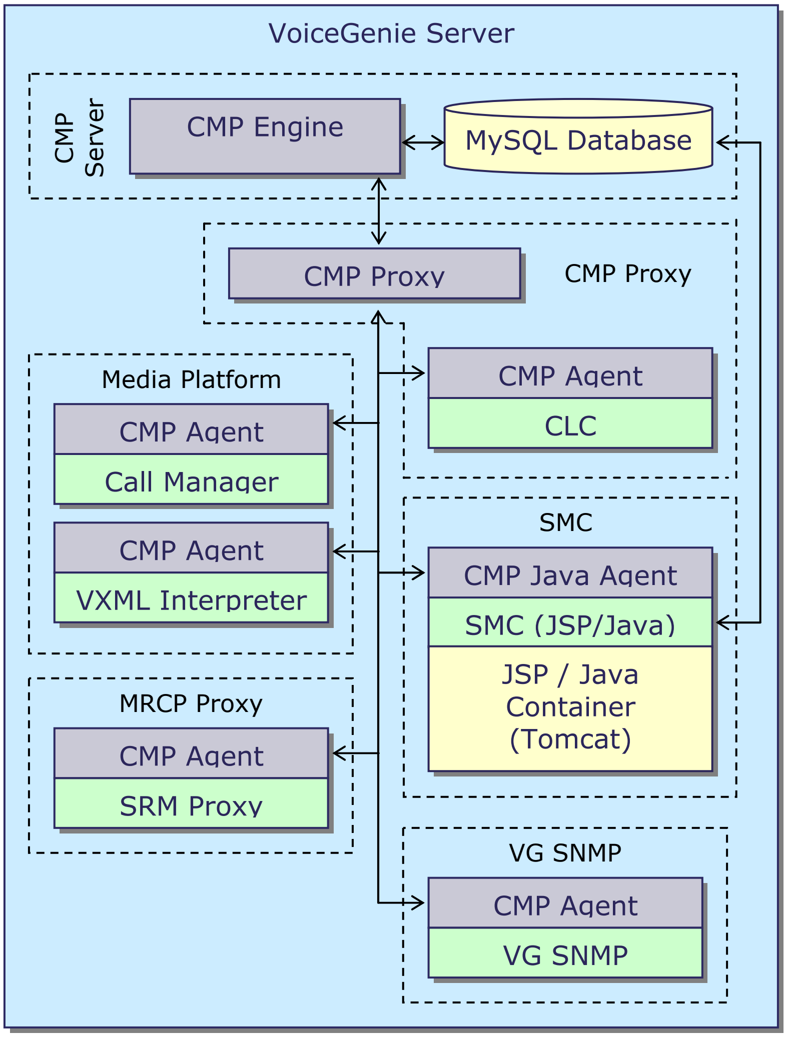
- **The Cluster Management Platform (CMP) Server** – it is responsible for all centralized logging and configuration capability. All CMP Proxies in the VoiceGenie network of servers connect to a CMP Server.
- **The CMP Proxy** – which must run on every server that is managed or monitored by the OA&M Framework. It acts as a single point of communication for all VoiceGenie software running on that server. The CMP Proxy is responsible for server level logging; this includes the

metrics logs, alarms and system level logging. Also, the CMP Proxy is responsible for starting and stopping all VoiceGenie software components. In addition, the CMP Proxy monitors the disk, CPU and memory utilization of the system, as well as the CPU and memory utilization of all VoiceGenie processes and can restart them if required.

- **The Command Line Console (CLC)** – a command line interface to the OA&M Framework. Through this interface, users can query information about the components that are part of the VoiceGenie network of servers. Also, the CLC allows users to inject commands into the OA&M Framework to carry out various tasks.
- **The System Management Console (SMC)** – consists of a web interface that can be used to access various monitoring, operations, installation, configuration, and administration capabilities. Through the web interface users can access both real time and historical information about the VoiceGenie software, as well as perform various operations and carry out configuration and provision changes.
- **The VoiceGenie SNMP** – the SNMP agent for all VoiceGenie components. Via the VoiceGenie SNMP Agent users can receive SNMP traps whenever an alarm condition occurs, also, SNMP gets and sets are supported.

Each platform includes facilities suitable for managing the platform as a standalone unit, and for integrating into a cluster managed by a distributed management console, or an existing network management solution.

The following diagram illustrates the architecture and distribution of the various OA&M components as well as the Media Platform in an “all-in-one” setup, i.e. the CMP Proxy & CLC, CMP Server, SMC, VoiceGenie SNMP and the rest of the VoiceGenie components are installed on a single server:



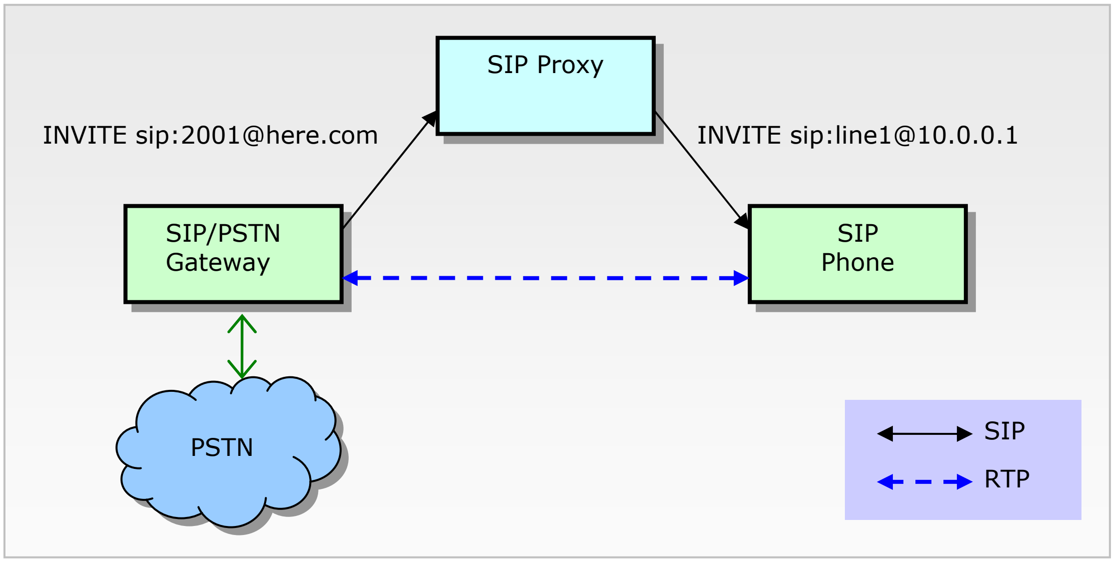
For details about the OA&M architecture and component details, please refer to the following documents:

- *VoiceGenie 7.2 OA&M Framework User's Guide*
- *VoiceGenie 7.2 OA&M Framework – SMC User's Guide*
- *VoiceGenie 7.2 OA&M Framework – CLC User's Guide*
- *VoiceGenie 7.2 OA&M Framework – SNMP User's Guide*

2.4 Other VoiceGenie Components

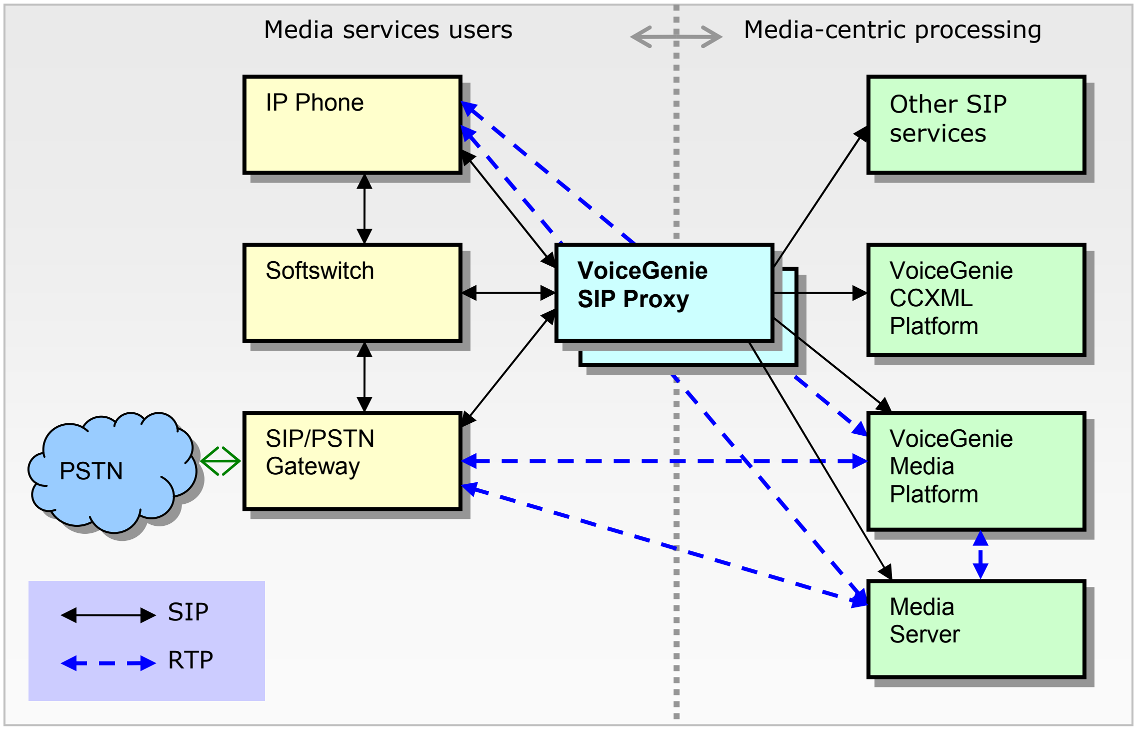
2.4.1 SIP Proxy

SIP proxies provide a variety of services in VOIP networks that are based on the use of SIP, such as authorization and access control, validation and security, call routing, accounting, user location, and others. A variety of SIP proxies are available, some of which are commercial (e.g. Cisco) and some of which are open source (e.g. Vovida, iptel.org). Fundamentally, any SIP proxy delivers services by controlling how requests and responses are routed between a SIP client (UAC) and a SIP server (UAS), generally by deciding on how requests get routed, as well as by manipulating headers in the request or response. A simple example of this is shown in the figure below:



VoiceGenie's SIP proxy provides a very specific set of services that are not available from general-purpose SIP proxies that are widely available. These services are specific to providing media-centric SIP services, such as VoiceXML dialogs, and conferencing capabilities. VoiceGenie's SIP Proxy is also designed for scalability and has redundant architecture to protect against server failures.

From a logical perspective, the purpose of VoiceGenie's SIP proxy is to act as an interface to a collection of media processing resources, such as VoiceGenie's media platform, CCXML platform, audio/video conferencing or other resources. SIP devices and applications can then make use of media-centric services through the proxy, without having to know the actual location of those resources or how to manage various routing decisions. The services provided by the VoiceGenie SIP proxy are used not only by clients such as media gateways or softswitches, but may also be used by internal media resources to co-ordinate interactions with one another. For instance, VoiceGenie's CCXML platform offers the ability to manage a VoiceXML dialog that actually executes on VoiceGenie's media platform; the CCXML platform may make use of proxy capabilities to locate an appropriate VoiceXML platform. The following diagram illustrates this logical architecture:



Although the above diagram shows a number of elements, both users of media services managed by the proxy as well as the underlying media-centric services that the proxy routes to, the actual configuration used in a deployment is typically much simpler. For instance, in a contact centre environment providing automated self-service or call routing, the deployment might consist solely of a number of SIP/PSTN gateways to handle incoming calls, a cluster of VoiceGenie media platforms that provide the actual treatment of calls through touch-tone or speech applications, and a redundant pair of VoiceGenie SIP proxies that provide load balancing across the available VoiceGenie platforms. In more complex next-generation network architectures, the VoiceGenie SIP proxy essentially acts as a *logical media server,* aggregating the capabilities of heterogeneous array of processing resources and acting as a single interface point for media services users.

2.4.2 CTI Connectors

The VoiceGenie platform provides a number of ways to support CTI integration. The two most common are:

- Application Server Side Integration
- Media Platform Integration

Application Server Integration

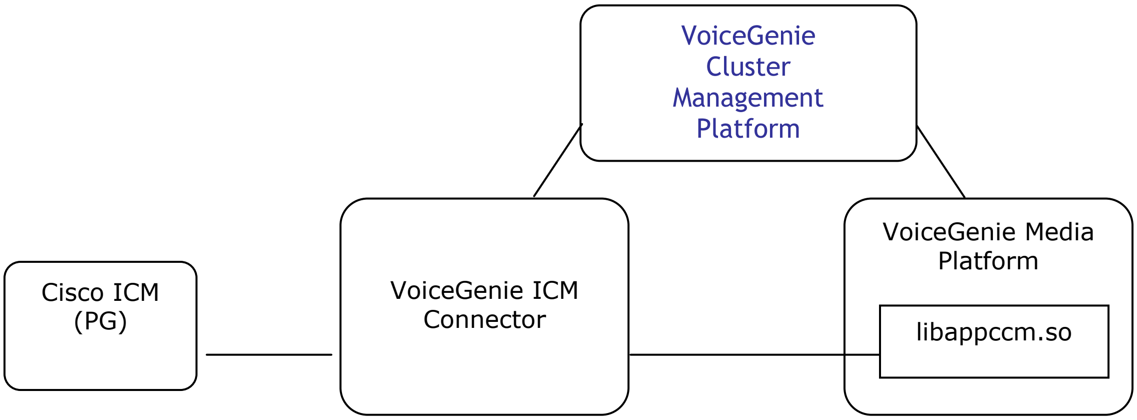
The VoiceGenie platform delivers call-related data to the application server as part of the initial page fetch related to a VoiceXML call. This data includes the information required to interface to an external CTI server – information such as port number, session identifier, and so on. This allows the application server to manage interaction with the CTI infrastructure.

Media Platform Integration

Media platform integration allows for CTI infrastructure interaction to take place on the media platform itself. The most common features are provided completely transparently to the application. This method has benefits to the application developer. The VoiceGenie media platform has integrated with the Cisco ICM package.

Architectural Overview – VoiceGenie and ICM

When ICM support is configured, the component architecture is as shown in the following figure.


VoiceGenie Cluster Management Platform
Cisco ICM (PG)
VoiceGenie ICM Connector
VoiceGenie Media Platform
libappccm.so

The VoiceGenie Media Platform loads an optional dynamic library named `libappccm.so`. CCM is an acronym for Call Control Module. When there is an incoming call to the Media Platform, control of the call is routed to the CCM instead of to a VoiceXML script. The CCM establishes a dialog with the Call Control Platform, using an internal CCI (Call Control Interface) protocol. The CCP uses the CCI protocol to control the call. For example, the CCP can select and launch VoiceXML scripts to handle the call, and can then initiate a call transfer when the VoiceXML script has completed.

The VoiceGenie Call Control Platform loads an optional dynamic library named `libccpicm.so`. This module acts as a protocol converter, between the CCI protocol, and the Cisco ICM/VRU Interface protocol which is supported by ICM. A VRU (Voice Response Unit) is Cisco's terminology for a media server platform. A VoiceGenie CCP, and the Media Platforms to which it's connected, look like one VRU as far as ICM is concerned. The VoiceGenie CCP can drive one or more media platforms.

The Cisco ICM protocol supports two different interfaces:

- **"Routing" and "Event Data Feed" interface** – VoiceGenie can use these to tell ICM when a call is connected and disconnected, and to request a route (a transfer destination address) for a given call. Using these interfaces, it helps to think of the VRU as being the client, and the ICM as being a server: the VRU issues route requests, and gets route responses from ICM.
- **"Service Control" interface** – ICM can use this to tell VoiceGenie what VoiceXML scripts it should run, as well as when and where to transfer calls. Using this interface, it helps to think of ICM as being the client, and the VRU as being the server: the ICM issues requests to start a VoiceXML script or to transfer a call, which are obeyed by the VRU.

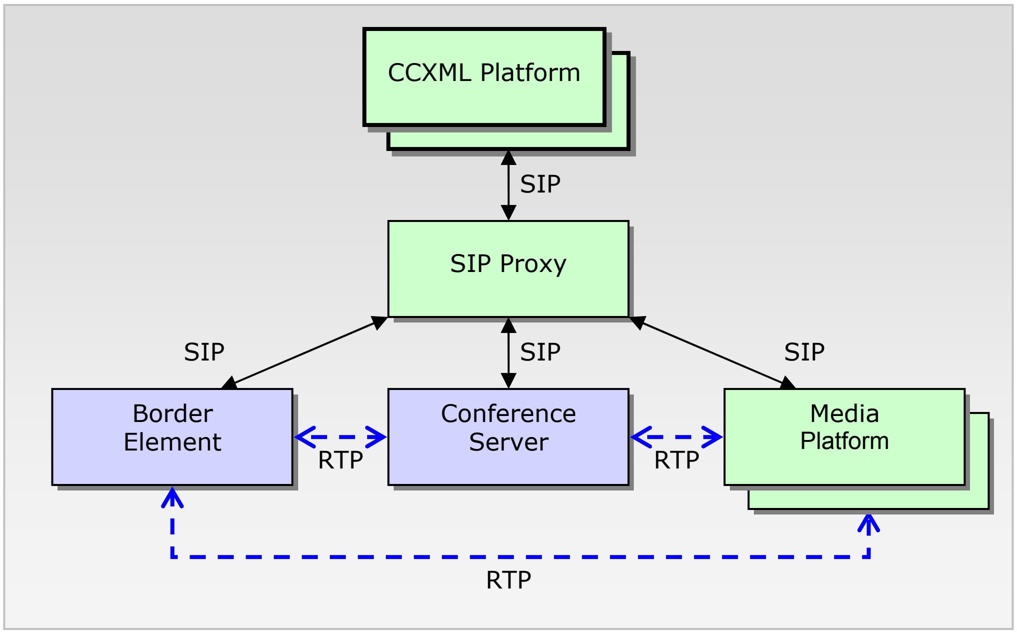
VoiceGenie supports both of these interfaces. However, at run-time ICM supports one or the other of these interfaces for a given VRU (you cannot use both interfaces simultaneously).

2.4.3 Call Control XML (CCXML) Platform

VoiceGenie CCXML Platform provides a CCXML interpreter that integrates with existing VoiceGenie infrastructure such as the Media Platform and SIP Proxy. The underlying network protocol for CCXML Platform is SIP; this means that CCXML Platform can interoperate with other conferencing server or dialog server.

Although VoiceGenie has traditionally provided extended call control capabilities through proprietary extensions to VoiceXML, the development of Call Control XML (CCXML) provides a standard, XML-based language for scripting call control logic. Like VoiceXML, CCXML is independent of the environment in which it operates, and can run in environments ranging from VOIP-based softswitch products to integrated residential gateways that manage a single telephone call.

In a clustered environment, SIP Proxy manages multiple instances of CCXML Platforms and Media Platforms. External elements send to the SIP Proxy to forward the SIP requests to the appropriate SIP service. The following diagram, similar to the architecture diagram above, shows elements managed by the SIP Proxy:



Other all other components, the CCXML Platform resides within VoiceGenie OA&M framework that allows CCXML Platform to be deployed, configured, monitored, and managed in a consistent manner with other VoiceGenie software components.

Chapter

3 Running Applications on the Media Platform

The VoiceGenie 7.2 Media Platform delivers speech enabled services and applications through a variety of means, which includes VoiceXML applications and conferencing applications (which are all CMAPI application modules). When an incoming call is received from the telephony network, an application is selected to handle the call, and the selection criteria is based on the dialed number (DNIS) information associated with the call from the telephony network.

The mappings are stored as provisioning data in the file `cm_provision.dat`, which is located in `/usr/local/phoneweb/config` for Linux systems and `($INSTALLROOT)\mp\config` in Windows systems. This file is managed by the OA&M Framework and cannot be changed manually. All changes must be made via the SMC Interface.

3.1 Application Provisioning (DNIS – URL Mapping)

To edit the application provisioning, log into SMC and click on the `Configuration` tab. Under the `Media Platform` there should be a link for `DNIS - URL Mapping`.

A user must first select the CMAPI application module to handle the incoming call (Note that the list of CMAPI application modules is defined in call manager configurations under the parameters `sessmgr.modules` and `sessmgr.appmodules`. Here is a description for each of the available options:

- **VoiceXML** – This application module interfaces with Next Generation Interpreter for the required VXML-defined application logic for a call. See 3.2 VoiceXML Applications and the VoiceXML Interpreterrs for more information.

- **VoiceXML – Pre 7.2 Compatibility Mode** – This application module interfaces with the Legacy Interpreter for the required VXML-defined application logic for a call. See 3.2 VoiceXML Applications and the VoiceXML Interpreter for more information.
- **ICM Call Control (CCM)** – This application module is designed to work with VoiceGenie ICM Connector module. This application module is hardwired as a "parent" of VXML application module (and forward requests to VXML application module if necessary). For more information on ICM integration, please see the following documents:
  - *VoiceGenie 7.2 ICM Connector User's Guide*
  - *VoiceGenie 7.2 ICM Connector System Reference Guide*
- **Policy Client (PolicyClient)** – This application module is designed to work with an external DCL (Data Connection) outgoing/transferring policy server in a customized project. This application module is hardwired as a "parent" of VXML application module. Hence, besides transfer operations where this application module needs to perform policy checking with the policy server, all other operations will pass through directly between the VXML application module and the call manager. This application module is using proprietary extensions for a customized project and is not available for general usage by other customers.
- **Continuity Check (ContCheck)** – This application module is designed to work with VoiceGenie CCP-SS7 platform to perform continuity check functionality. When accepting an incoming call, the application performs a media loopback operation and waits for a disconnect request.
- **Conferencing (Conference)** – This application module joins a call to a conference session (based on information in the incoming SIP message – see 3.3 Conferencing for more information), and waits for a disconnection request to remove the call from conference. The application module automatically manages creation and destruction of conference sessions.
- **VoiceXML w/ PortCount (PortCount)** – This application module is required for performing the application port count service (see 3.4 Application Count Service (Partition Definition) for details). The application module is hardwired as a "parent" of the VXML application module. It only intercepts incoming call and outgoing call requests to perform application port count checking. All operations after this point will pass through this module, and will be directly executed between the VXML application module and the call manager.

To create a new DNIS – URL Mapping entry, enter the `DNIS` and `URL` for the new entry:

- `DNIS` – The Dialed Number presented to the platform by the network (Dialed Number Identification Service). This is the key for the DNIS to URL lookup.

Note that `XXXX` is a special default DNIS. If the incoming DNIS does not match any other entries, it will be handled by the `XXXX` entry. Also, wildcard suffix is supported. For example, `123*` will match all DNIS with prefix=`123`. The Media Platform will perform a longest prefix match if multiple entries are matched (ie, if both `123*` and `1234` are defined as DNIS keys, then incoming call with `DNIS=1234` will match the `1234` DNIS key entry).

- `URL` – The initial URL of the VoiceXML application. This parameter is only mandatory for the VXML application module (or all application modules that are hardwired as “parent” of the VXML application module).
- `VoiceXML Defaults` – The default properties page of the VoiceXML application. This parameter is only required for the VXML application module (and all application modules that are hardwired as “parent” of the VXML application module).
- `Parameter Name/Value` – additional parameters can be passed to each application module to provide additional information about handling of the incoming call (which will be passed to the VXML Interpreter).

Here is an example:

In the DNIS – URL Mapping entry, specify `Parameter Name = FOO` and `Value = BAR`.

For Legacy Interpreter, add `session.connection.foo|FOO|0 to session_var` under the `VoiceXML Interpreter Configuration` section via SMC.

For Next Generation Interpreter, add `session.connection.foo|FOO|0` to `session_var` under the `Call Manager Configuration` section via SMC.

In the vxml page, a check such as the following can be performed:

```
<if cond="session.connection.foo == 'BAR'">
```

In Legacy Interpreter, the following parameters change the behavior of the application, particularly the behavior of the initial page (note that the names are not case-sensitive):

Parameter Name	Value
INIT_URL.maxage	-2 (and INIT_URL.maxstale also set to -2): Does not cache the initial page at all. The Pragma header in the corresponding http request will be set to no-cache. Any positive number represents the maxage time in seconds in the corresponding http request, if it is a VoiceXML 2.0 application. Default: -2
INIT_URL.maxstale	-2 (and INIT_URL.maxage also set to -2): Does not cache the initial page at all. The Pragma header in the corresponding http request will be set to no-cache. Any positive number represents the maxstale time in seconds in the corresponding http request, if it is a VoiceXML 2.0 application. Default: -2
INIT_URL.method	POST: Posts additional session data that is specified in INIT_URL.postbody to the application server. Default: Not set.
INIT_URL.postbody	The session data (in addition to the data specified in session_vars in the voicexml.cfg) that is included in the request of the initial page in the request body. Note: The content of each name and value must be URL encoded. In order to include the session data, the INIT_URL.method must be set to POST. Default: Not set.

Parameter Name	**Value**
PROP.<VoiceXML Property name>	The corresponding VoiceXML property value. Only on the initial page, the specified VoiceXML properties take precedence over the same properties in `defaults.vxml`. The document fetch/cache properties specified here do not affect the initial page fetch/cache behavior. Default: Not set.
VXMLI.default_xmllang	A valid language that is listed in the `supported_language` item of `voicexml.cfg`. The specified language is used as the `vxmli.default_xmllang` for the application. Default: Not set.
VXMLI. default_transfer_connect_timeout	Integer that is greater than `5` and less than `2147483648`. The specified value (in seconds) is used as the `vxmli.default_transfer_connect_timeout` for the application. Default: Not set.

Click on `Create` to create the DNIS – URL Mapping entry.

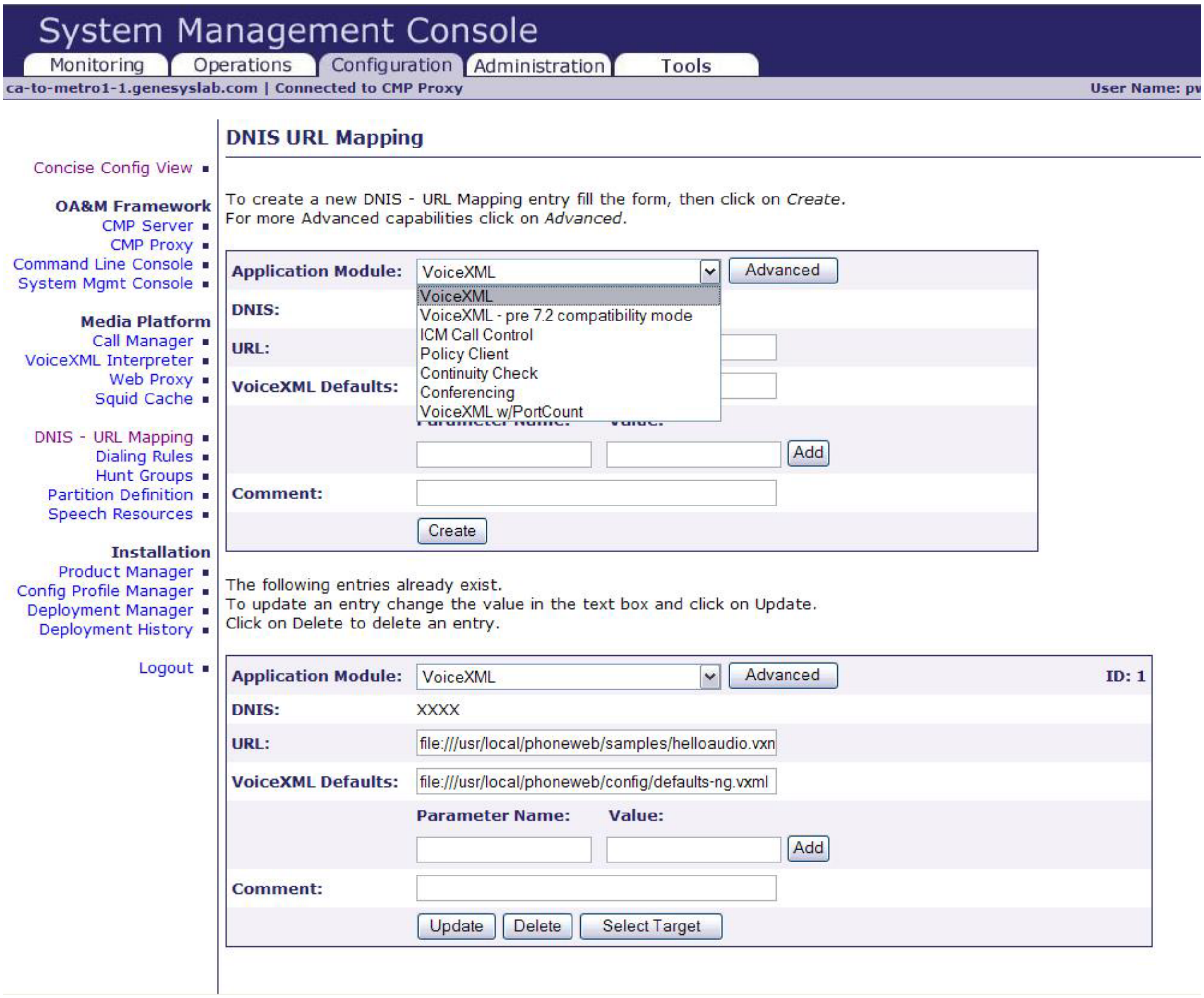
Once the entry has been created users can click on `Select Target` to select the systems to receive this mapping.

To update the contents of an entry, make changes to any of the parameters and click on `Update`. This will send all changes to the platforms that are selected as targets.

To delete an entry simply click on `Delete`.

Advanced users may make use of the `Advanced` button to further customize information related to the handling of the incoming call. By clicking on it a XML based editing tool will be launched allowing the user to manually edit the parameters and settings. The following is an example of a VoiceXML application module entry:

```
<key name="DNIS" value="4321"/>
<application module="VXML">
<param name="url" value="file:///usr/local/phoneweb/msh/test.vxml"/>
<param name="default" value="defaults.vxml"/>
</application>
```



DNIS URL Mapping

To create a new DNIS - URL Mapping entry fill the form, then click on *Create*.
For more Advanced capabilities click on *Advanced*.

Application Module: VoiceXML
DNIS:
URL:
VoiceXML Defaults:
Parameter Name: Value:
Comment:

The following entries already exist.
To update an entry change the value in the text box and click on Update.
Click on Delete to delete an entry.

Application Module: VoiceXML ID: 1
DNIS: XXXX
URL: file:///usr/local/phoneweb/samples/helloaudio.vxn
VoiceXML Defaults: file:///usr/local/phoneweb/config/defaults-ng.vxml
Parameter Name: Value:
Comment:

3.1.1 Choosing Defaults for Each Application

The DNIS – URL Mapping entries are used by the VoiceGenie platform to associate an application with each DNIS or VoIP address that can be used to call in to the platform. (For details on using DNIS – URL Mapping entries with a VoIP-based platform, see Calling In – #1 and 2, in the VoIP tutorial.)

Also, the entry determines the appropriate defaults file to use for the call. This is specified in the `VoiceXML Defaults` field.

The following sections explain how multiple default settings can be defined and how a default setting is chosen for each different application.

3.1.2 Multiple Default Settings

The VoiceGenie platform uses the default settings to specify property values and event handlers, in case the developer does not specify them in their VoiceXML page.

The VoiceGenie platform supports the use of multiple default settings, so that a different set of values can be used for each application running on the same platform. This is beneficial, for example, on a platform that has multiple ASR and TTS engines installed. In this case, any applications using ASR engine #1 could use default settings which are appropriate for that ASR engine, such as:

```
<property name="ASRENGINE" value="ASR1"/>
<property name="CONFIDENCELEVEL" value="0.35"/>
... [other speech-related properties] ...
```

while applications using ASR engine #2 could use default settings which are appropriate for that other ASR engine, such as:

```
<property name="ASRENGINE" value="ASR2"/>
<property name="CONFIDENCELEVEL" value="0.5"/>
... [other speech-related properties].
```

Note: In this example, all properties and event handlers unrelated to speech would be the same as the original default setting.

Having multiple default settings can also be beneficial if a set of related applications running on the same platform is using a different set of event handlers from the other applications on the same platform. Rather than redefining them in every application, this set of applications could use a different default setting, which redefines the event handlers once.

3.2 VoiceXML Applications and the VoiceXML Interpreters

The VoiceGenie 7.2 Media Platform supports two VoiceXML Interpreters: the field proven Legacy Interpreter (VXMLi) and the Next Generation Interpreter (NGI).

When an incoming call’s DNIS is mapped to a VXML application, upon receiving the call, the Call Manager will submit a new call request to the VoiceXML Interpreter (VXMLi). The VXMLi will first initiate a fetching request via the fetching module (iproxy), which makes use of 3.5 HTTP/HTTPS Support and 3.6 Caching in VoiceGenie 7.2 Media Platform.

If the page can be successfully fetched, the Legacy Interpreter will compile the page and upon success, it will instruct the Call Manager to accept the call. If any problem is encountered when fetching the page, the Legacy Interpreter will try to fetch and compile the alternate page. The alternate page can be specified by the parameter `ALTER_URL` or by configuration. Please see Application Provisioning (DNIS-URL Mapping) for details on how the `ALTER_URL` parameter can be specified as part of a DNIS-URL mapping. For the Legacy Interpreter, the alternate page is specified using the `alternate_initial_page` parameter in the `VoiceXML Interpreter Configuration` section. For the Next Generation Interpreter, it is specified using the `vxmli.default.alternate_uri` parameter in the `Call Manager Configuration` section. If an alternate page is specified by both the `ALTER_URL` parameter and through configuration, the value of the `ALTER_URL` parameter will be used.

For details regarding the VoiceXML language, please visit:

- http://support.voicegenie.com
- http://developer.voicegenie.com

3.2.1 URL Reference Syntax Supported by the VoiceXML Interpreter

The valid forms of URL supported by VoiceXML Interpreter are:

```
http[s]://<host>[:<port>][<absolute_path>[?<query string>][#<anchor>]]
file:///<absolute path>[#anchor]
```

Example:

```
http://vxml.example.com:8080/testbed/a1.jsp#welcome
https://secure.example.com:8080/testbed/b1.jsp?d=5&r=test
file:///usr/local/phoneweb/samples/helloasr.vxml
file:///C:/VoiceGenie/mp/samples/helloaudio.vxml#form1
```

A URL that does not begin with a valid scheme will be treated as a relative URL.

3.2.2 Communicating with the Legacy Interpreter via CLC

By issuing the `sendevent` command at the CLC prompt, it's possible to send messages to active VoiceXML sessions. The command has the following format:

```
sendevent vxmli [host] [instance] [recipient_session_id]
[sender_address] [message]
```

where all parameters are required. The parameters are defined as follows:

- `[host]` – the host where the Legacy Interpreter is running. For localhost, a - can be used
- `[instance]` – the instance of the interpreter, `-` implies instance 1
- `[recipient_session_id]` – the unique session ID of the interpreter session the message will be sent to. A typical session ID looks as follows:

  `0C9703E2-0C0028ED-0001`

  The session ID of a VoiceXML session can be determined in different ways, including:

  - The value of the `session.com.voicegenie.instance.myself` session variable of that session.
  - The value of the `X-Session-Id HTTP` header in the HTTP requests performed by that session.
- `[sender_address]` – can also be specified as a valid session ID of an interpreter session, or any character string whose length is under 127 characters. If the `sender_address` is a valid session ID, the recipient session would be able to send back a message to the session with the ID specified by the `sender_address,` which is dependent on the VoiceXML application.
- `[message]` – any combination of characters, with a maximum length of 2999 characters.

Example: The following is an example using the sendevent command:

```
CLC> sendevent vxmli - - 0C9703E2-0C0028ED-0001 1234 Hello World!
```

The following table outlines the return values and their meanings:

Result	Meaning
Sending Message from <sender_address> to <recipient_address>	Success.
Usage: sendevent [service] [hostname] [inatnce] [recipient_address] [sender_address] [message]	Failed, invalid format command.
Can not deliver message	Failed.
Failed to send message	Failed.

3.2.3 Limitations

- When the available disk space has reached a point so low that the Legacy Interpreter process cannot write to the disk any more (e.g. creating new promptsfile), the process will exit. At this point, the following actions should be performed:
  - Shutdown the Media Platform
  - Clean up the disk space
  - Restart the Media Platform
- When the Media Platform is being shut down, all unsent maintainer emails would be discarded
- Special characters `<`, `>`, `?`, `:` and `+` in DTMF grammars are not supported
- The DTMF digits `A`, `B`, `C` and `D` are not supported
- With the Legacy Interpreter, rule names in SRGS DTMF grammars are concatenated to form qualified ECMAScript variable names. If a grammar contains very long rule names, their concatenation may hit an internal limit (which is roughly 500 characters), causing a grammar processing error.
- With the Legacy Interpreter, property `scriptfetchhint` should always be set to `prefetch`. The value `safe` is not supported. This is also applicable to the `fetchhint` attribute in the `<script>` tag.

The custom application log file that is specified by the `dest` attribute in the `log` element is managed by the VoiceXML Interpreters. It has the following limitations:

NGI:

- Writing to the custom application log file is not thread safe. When under load, some log entries may be missing.
- The log file is rotated without saving the previous data after the file size exceeds its configured value, which is specified by `vxml.max_application_logfile_size` in `callmgr.cfg`.

VGI:

- There is no built-in mechanism to rotate the custom application log file. The user must manage the file size. If the file size exceeds 2GB in Linux, the pwvxmli process will crash.

Due to the above limitations, the custom log file is not recommended to be used in a production environment. The `pw_metrics` file should be used for VoiceXML application logging.

3.3 Conferencing

Many interesting voice applications require the use of conferencing capability in order to deliver the intended user experience. One of the functions that VoiceGenie product is expected to perform, in addition to supporting VoiceXML, prompt playback, SIP/RTP, and speech, is conferencing. Examples of applications that can make use of conferencing including enhanced voice activated dialers that allow multiple people to be called simultaneously, voicemail and other applications that allow outbound calls while staying on the line, delivery of IP Centrex conferencing services (in conjunction with a soft-switch application), and numerous others. Conferencing capability is included in the VoiceGenie 7.2 Media Platform product. There are two interfaces to access the conferencing feature. One is via SIP interface, and the other is via VXML interface.

3.3.1 Conferencing via SIP Interface

An incoming SIP call can be routed to a conference directly without using any VXML application. For this scenario, a Conference CMAPI application will be used to handle the call and manage the call's interactions with the conference bridge. In order to use this feature, the following Call Manager parameters (accessible via SMC) must be configured properly in call manager configuration:

- `Conference` must be included in `sessmgr.modules`
- `Conference:Conference` must be included in `sessmgr.appmodules`
- `Sessmgr.Conference.Conference` must be set as `Conference`

In addition, there are a few parameters in call manager configuration that can be used to define default properties of all calls joining the conference via SIP interface: `conference.limit`, `conference.initial_gain`, `conference.input_delay`, `conference.confdir`, `conference.highest_input`, `conference.suppress_silence`, `conference.silence_fill` and `conference.audio_format`.

Once the Conference CMAPI application is set up, a SIP call can be routed to join a conference (provided that the conference ID is known) directly using one of the following ways. Note that a new conference session will be created implicitly if there are no existing participants for the given conference ID:

1. Netann IETF draft

   Make a SIP call to the VoiceGenie platform with:

   `sip:conf=<XXX>@<host> in the request URI.`

This is a request to join a conference with ID `<XXX>` on the VoiceGenie platform `<host>`. For details, see section 5 of `http://www.ietf.org/rfc/rfc4240.txt`.

2. DNIS-URL mapping with request parameters

   Under SMC's `DNIS-URL Mapping Configuration` section, create a `Conferencing` mapping entry with the `DNIS` set to `WWWW`. There is no need to set any `URL`; the SMC will set it automatically. (See 3.1 Application Provisioning (DNIS – URL Mapping) for details on how to add `DNIS-URL Mapping` entries through SMC.)

   Make a SIP call to the VoiceGenie platform with `WWWW` in the user portion of the SIP request URI and with `confinstid=<XXX>` in the SIP request parameters. The call will be routed to the Conference CMAPI application as a request to join a conference with ID `<XXX>`. So calling `sip:WWWW@<host>;confinstid=123` will join the user to conference `123` on `<host>`.

3. DNIS-URL mapping without request parameters

   In the SMC `DNIS-URL Mapping Configuration`, create a `Conferencing` mapping entry with the `DNIS` set to `WWWW*`. There is no need to set any `URL`; the SMC will set it automatically. (See 3.1 Application Provisioning (DNIS – URL Mapping) for details on how to add DNIS-URL Mapping entries through SMC.) Then, when a SIP call is made to the VoiceGenie platform with a `DNIS` of the format `WWWWconfid<XXX>[confdir<Y>]`, it will be taken as a request to join a conference with ID `<XXX>` on the VoiceGenie platform being called, with mode optionally specified by `<Y>` (`0` for talk-only, `1` for listen-only, `2` for talk/listen). If `<Y>` is not specified, the value defined by the `conference.confdir` parameter in the Call Manager configuration will be used.

When the SIP call disconnects from the platform (eg, via a SIP `BYE`), it will exit from the conference. When there are no more participants in a conference, the conference session will be destroyed implicitly.

3.3.2 Conferencing via VXML Interface

Conference participation can also be controlled from the VXML application using `<join>` and `<release>` tags. There is no need to configure Conference CMAPI application module as described in the previous section. The default conference session properties can be defined using the following VXML properties:

- `com.voicegenie.conference.createnew`
- `com.voicegenie.conference.limit`
- `com.voicegenie.conference.initialgain`
- `com.voicegenie.conference.inputdelay`

- com.voicegenie.conference.highestinput
- com.voicegenie.conference.suppresssilence
- com.voicegenie.conference.silencefill
- com.voicegenie.conference.audioformat

The conference session will be identified using conf:<conferenceID>. The conference session identifier can be used directly in from, to, listen, talk, chan attributes in <join> and <release> tags.

For example, if the variable caller contains a reference to the original inbound caller's session ID, and the variable conf is set to 'conf:123', then:
<join name="j1" from="caller" to="conf"/>

would join the caller to the conference with ID 123, so that they can speak to the conference, and also listen to the conference output. If the contents of the from and to attributes are switched, the behaviour will remain the same.

Currently the Next Generation Interpreter does not support <join> and <release> tags for conferencing.

3.4 Application Count Service (Partition Definition)

Paritition in this context is a group of VoiceGenie media platforms viewed as one entity in executing a group of VoiceXML applications. VoiceGenie media platforms existing in the same network (reachable through multi-casting) and same CMP cluster can join the same partition. A partition associates a group of VoiceGenie media platforms to a group of VoiceXML applications. This allows the partition definition to limit the maximum concurrent applications count over a group of VoiceGenie media platforms. VoiceXML application is associated with partition through DNIS-URL mapping configuration where DNIS is associated with a partition through a parameter name called partition whose value is the partition name. In order to use the Application Count Service, configuration must be setup in three steps: 1) Activate PortCount CMAPI application, 2) Associate PortCount application with partition name in DNIS-URL mapping, and 3) Define partition.

3.4.1 Activate PortCount CMAPI application

PortCount application module must be enabled from call manager configuration.

1. Add PortCount to sessmgr.modules list.
2. Add PortCount:PortCount to sessmgr.appmodules list.
3. Add PortCount to the sessmgr.portcount.portcount list.

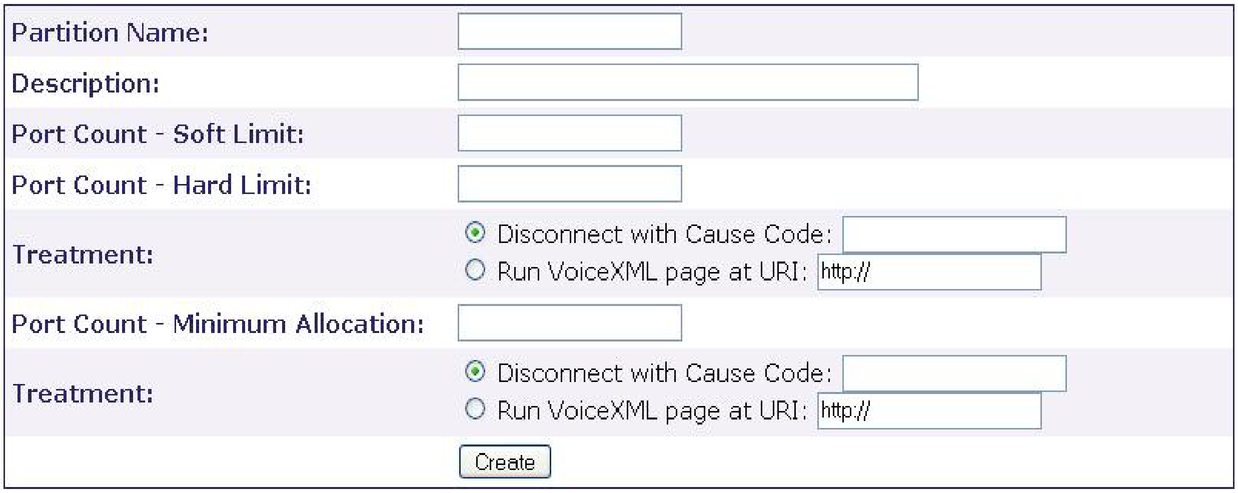
3.4.2 Associate PortCount application with partition name in DNIS-URL mapping

In DNIS-URL mapping, choose `PortCount` from the `Application Module` list. Insert the initial VoiceXML URL into the rest of the fields just as a regular VoiceXML app would be configured. On `Parameter Name/Value` field, type in `partition` and the name of partition to associate respectively.

To specifiy the VoiceXML Interpreter to be used, add a new parameter ‘gvp.appmodule’ in the Parameter Name/Value field. The value should be ‘VXML’ for the Legacy Interpreter or ‘VXML-NG’ for the Next Generation Interpreter. If gvp.appmodule is not specified, the default VoiceXML Interpreter specified by the Call Manager configuration parameter sessmgr.default_vxml_interpreter will be used.

3.4.3 Define Partition

The following is a snippet from the `Partition Definition` section of VoiceGenie SMC interface.



Partition Name:	
Description:	
Port Count - Soft Limit:	
Port Count - Hard Limit:	
Treatment:	● Disconnect with Cause Code: ○ Run VoiceXML page at URI: http://
Port Count - Minimum Allocation:	
Treatment:	● Disconnect with Cause Code: ○ Run VoiceXML page at URI: http://
	Create

- `Partition Name` should be the partition name associated in DNIS-URL mapping.
- `Description` can be any text comment for the partition.
- `Port Count - Soft Limit` defines the limit after which alarms will be generated to the O&AM framework. Limit is based on the maximum combined number of estimated incoming calls to the associated DNIS applications associated to the partition.
- `Port Count - Hard Limit` defines the limit after which calls will either be rejected or redirected to an alternate VoiceXML page. Limit is based on the maximum combined number of estimated incoming calls to the associated DNIS applications associated to the partition.

- `Treatment` following `Port Count - Hard Limit` is whether to reject or redirect when `Port Count - Hard Limit` is exceeded. Disconnect with Cause Code parameter applies only to PSTN environments and the `Cause Code` is the ISDN cause code. In the SIP case, calls will be declined with a 503 response.
- `Port Count - Minimum Allocation` defines the minimum number of calls that must exist in this partition.
- `Treatment` following `Port Count - Minimum Allocation` is whether to reject or redirect when `Port Count - Minimum Allocation` is not met. The incoming calls affected are the calls coming into other partitions. Disconnect with Cause Code parameter applies only to PSTN environments and the `Cause Code` is the ISDN cause code. In the SIP case, calls will be declined with a 503 response.

3.4.4 Alarms and Metrics

Whenever a limit is exceeded, an alarm will be raised. Please refer to the *VoiceGenie 7.2 Media Platform System Reference Guide* for information on the alarms related to the Application Count Service:

- Soft limit exceeded
- Hard limit exceeded
- Minimum required limit exceeded

If a call is rejected by the `Disconnect with Cause Code` treatment, the reject reason in metrics logs will be `interrupt`, for example

```
2005-12-02/17:58:10.529 METRIC 00020039-10002E26 incall_initiated
   0:0
2005-12-02/17:58:10.531 METRIC 00020039-10002E26 call_reference
   00000000-D5EDD98C-4443@10.0.0.208
2005-12-02/17:58:10.531 METRIC 00020039-10002E26 incall_reject
   sip:1234@10.0.0.104:5060|sip:VoiceGenie@10.0.0.208:4443|20051202
   564290013|N/A|N/A|N/A|interrupt
```

Note: When used in combination with the SS7 Connector a call will be shown being rejected with the default cause code `41`, instead of the cause code defined by the Call Manager configuration parameter `sessmgr.disconnect_cause.decline`.

3.5 HTTP/HTTPS Support

The VoiceGenie platform provides support for HTTP 1.0, with many of the most useful features of HTTP 1.1, including:

- Connection keepalive

- HTTP/1.1 cache control
- RFC 2109 state management (cookies)

The platform also supports Secure Socket Layer (SSL) connections using the conventional ‘https’ scheme.

Note: Complete use of SSL will have an impact on platform capacity, depending upon the amount of data transferred using SSL. In addition to this, data fetched with https will not be cached.

When using the Next Generation Interpreter (NGi), the Content-Type: header in the HTTP response from a web server must be present. Although some values are recommended by W3C, e.g. application/voicexml+xml for vxml pages, NGi places no limitations on the exact type, so for vxml pages application/vxml, text/plain, text/xml will all be accepted.

The Legacy Interpreter supports a number of configuration items related to HTTP and HTTPS. In particular, the following items can be configured in the `voicexml.cfg` configuration file:

- `USER_AGENT` – This parameter allows the user to override the user agent HTTP request header sent by the VoiceGenie platform as part of HTTP requests. It is advisable to extend the default parameter (`VoiceGenie NXP/$v` where `$v` is the version of the VoiceGenie platform, e.g. `7.0.0`) rather than completely overriding it. Third party application servers rely upon this to detect the VoiceGenie platform.
- `HTTP_ACCEPT` – This parameter allows the definition of the content types to be sent as part of the HTTP `Accept:` HTTP request header line. This is useful when the user cannot control the headers returned by a particular web server, and wishes to advertise the appropriate headers to the server for delivery.
- `HTTP_VERSION` – This parameter allows the version header value to be specified – Use this with caution however, as it may result in the server using an HTTP/1.1 capability not fully supported by the platform.

Some configuration changes may be required to use SSL for connections. This is done in the Web Proxy configuration in the SMC. Note that changes made to the Web Proxy configuration are applied to all modules that perform fetching (ie. the Legacy Interpreter and the NextGen Interpreter). The following are the parameters relevant for SSL configuration:

SSL Configuration Setting	
iproxy.ssl_cert	The file name of your certificate. The default format is `PEM` and can be changed with the configuration parameter `iproxy.ssl_cert_type.`

SSL Configuration Setting	
iproxy.ssl_cert_type	The format of the certificate. Possible values: `PEM, DER` Default: `PEM`
iproxy.ssl_key	The file name of the private key. The default format for the key is `PEM` and may be changed by the parameter `iproxy.ssl_key_type.`
iproxy.ssl_key_type	The format of the private key. Possible values: `PEM, DER, ENG` Default: `PEM`
iproxy.ssl_key_passwd	The password required to use the `iproxy.ssl_key.`
iproxy.ssl_engine	The identifier for the crypto engine you want to use for your private key.
iproxy.ssl_engine_default	Sets the actual crypto engine as the default for (asymetric) crypto operations.
iproxy.ssl_version	Set what version of SSL to attempt to use. By default, the SSL library will try to solve this by itself although some servers make this difficult why you at times may have to use this option. Possible values: `2, 3` Default: `2`
iproxy.ssl_verify_peer	Do you want verify the peer's certificate. When this option is set, you should set one of `iproxy.ssl_ca_info` or `iproxy.ssl_ca_path.` Possible values: `0, 1` Default: `0`
iproxy.ssl_ca_info	The file name holding one or more certificates to verify the peer with.
iproxy.ssl_ca_path	The path holding multiple CA certificates to verify the peer with. The certificate directory must be prepared using the `openssl c_rehash utility.`
iproxy.ssl_random_file	The path to a file which is read from to seed the random engine for SSL.
iproxy.ssl_verify_host	Should the Common name from the peer certificate in the SSL handshake be verified? Possible values: `0, 1, 2` Default: `0`

SSL Configuration Setting	
iproxy.ssl_cipher_list	The list of ciphers to use for the SSL connection. The list must be syntactly correct, it consists of one or more cipher strings sepa- separated rated by colons. Commas or spaces are also acceptable separators but colons are normally used, , - and + can be used as operators. Valid examples of cipher lists include `RC4-SHA`, `SHA1+DES`, `TLSv1` and `DEFAULT`. You'll find more details about cipher lists on this URL: `http://www.openssl.org/docs/apps/ciphers.html`. Default: `0`

If a specific port number is specified for https, for example:

`https://mozart.voicegenie.com:8553/test.vxml`

Then this port number must be configured in the proxy configuration file in order to allow the SSL connection to be established. Add port number (`8553`) to the two lines below. `443` and `563` are the defaults.

```
    acl SSL_ports port 443 563 8553
    acl Safe_ports port 443 563 8553 # https, snews
```

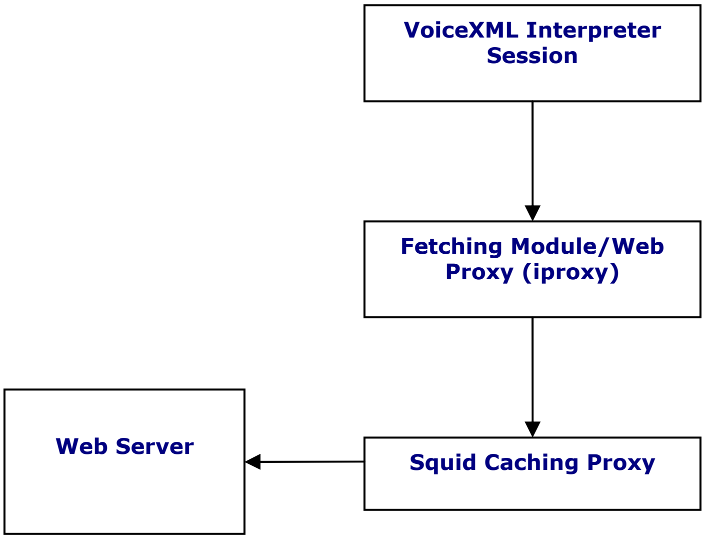
Additional caching proxy configuration is described in the following sections.

3.6 Caching in VoiceGenie 7.2 Media Platform

The VoiceXML interpreter context, like visual browsers, can use caching to improve performance in fetching documents and other resources; audio/video recordings (which can be quite large) are as common to VoiceXML documents as images are to HTML pages. In a visual browser it is common to include end user controls to update or refresh content that is perceived to be stale. This is not the case for the VoiceXML interpreter context, since it lacks equivalent end user controls. Thus enforcement of cache refresh is at the discretion of the document through appropriate use of the maxage, and maxstale attributes. The most common uses of these attributes are shown in 3.6.4 How to use maxage and maxstale attributes.

3.6.1 Caching Architecture

In the VoiceGenie Platform, caching is performed at multiple levels. The following diagram describes the levels of caching:


VoiceXML Interpreter Session
Fetching Module/Web Proxy (iproxy)
Squid Caching Proxy
Web Server

When a VoiceXML session needs to fetch a resource, it performs the fetch via the Web Proxy. This is a module built by VoiceGenie, which performs http fetches on behalf of the VoiceXML sessions. It also performs some limited in-memory caching (to be described in the section below) which is not HTTP/1.1 compliant. If the Web Proxy determines that it cannot serve the request from its in-memory cache, it will go to the Squid Caching Proxy to try to fetch the content. The Squid Caching Proxy performs HTTP/1.1 compliant caching, to be described in 3.6.6 Squid Caching Proxy. If Squid determines that it cannot serve the content from its cache, it will go to the Web Server to try to fetch the content.

3.6.2 Web Proxy caching

Even though the Web Proxy is a separate process from the VoiceXML interpreter, the communication between these processes is via share memory. Therefore, it is very efficient for the Web Proxy to pass fetch results and other information back to the VoiceXML interpreter. For performance reasons (especially for mostly static content such as large audio/video files), the web proxy performs caching (which is not HTTP/1.1 compliant). The following are the configuration parameters affecting Web Proxy caching:

Parameter	Description
iproxy.cache_max_age	Maximum age for data cached in iproxy in seconds (default is `60`). It applies only if data is cacheable. iproxy caching could be turned off by setting this to `0`. Default: `60`
iproxy.cache_error_max_age	Maximum age of cache for failed fetches in seconds. Default: `0`
iproxy.no_cache_url_substr	If a URL contains any one of the sub-strings in this list, it will not be cached. Default: `cgi-bin`
iproxy.use_strict_caching_rules	When set to `true`, the Fetching Module performs strictly HTTP/1.1-conformant caching. Setting this parameter to `false` results in better performance. Possible values: `true`, `false` Default: `true`

If an application is a dynamic document, meaning that its content always changes, it cannot be cached. The application URL or sub-string of the application URL must be added to `iproxy.no_cache_url_substr`. For example, `jsp` can be listed in `iproxy.no_cache_url_substr` for all the applications that have “jsp” in their URL.

If strict conformance to HTTP/1.1 caching behavior is not necessary, it is suggested to set `iproxy.use_strict_caching_rules` to `false` for better performance.

When the VoiceXML interpreter requests the Web Proxy to perform a fetch to URI, it uses the following algorithm to determine whether it will use the cached version within its own memory:

```
if ( URI contains one of the items in the iproxy.no_cache_url_substr
   list )
 re-fetch the item from the squid proxy
else
 if (the URI matches exactly [including all parameters] with a URI
   in Web Proxy cache)
 if (the previous was a fetch error)
 if (the previous result was fetched within
   iproxy.cache_error_max_age seconds)
 return result from cache;
 end if
 else if (the previous was a successful fetch)
 if (the previous result was fetched within iproxy.cache_ max_age
   seconds)
 return result from cache;
 end if
 end if
 end if
 re-fetch the item from the squid proxy
end if
```

This level of caching is non-compliant, and should be used carefully. If HTTP/1.1 compliance is desired, this should be turned off, by either adding more to the `iproxy.no_cache_url_substr` list, or by changing the `iproxy.cache_max_age` and `iproxy.cache_error_max_age` values to `0`.

Note that the preceding Web Proxy caching algorithm has the following exceptions that relate to the initial document and to the root documents of a VoiceXML application:

- For both the Next Generation and the Legacy VoiceXML interpreters, the initial document is not cached by the Web Proxy if the session data is included in the request.
- For the Legacy VoiceXML interpreter, if the parameters `INIT_URL.maxage` and `INIT_URL.maxstale` in the DNIS-URL mapping are not set or are both set to `-2`, the initial document is not cached by the Web Proxy, and the `Pragma` header in the http request is set to `no-cache`. For details about `INIT_URL.*` parameters, refer to Section 3.1: Application Provisioning (DNIS – URL Mapping).
- For the Legacy VoiceXML interpreter and VMXL 2.0+ application, the `documentmaxage` and `documentmaxstale` properties in `defaults.vxml` determine whether root documents are fetched, and also determine the cache behavior. If neither `documentmaxage` nor `documentmaxstale` is set, root documents are cached by the Web Proxy based on the `iproxy` cache configuration. If `documentmaxage` = `-2` and `documentmaxstale` = `-2`, root documents are fetched for every call. The http request will have the `Pragma no-cache` header. If either property is set to a positive number, the corresponding values of `documentmaxage` and `documentmaxstale` are reflected in the cache control header of the http request.
- For the Legacy VoiceXML interpreter and VMXL 1.0 application, the `caching` property in `defaults.vxml` determines whether root documents are fetched. If it is set to `FAST`, root documents are cached by `iproxy` based on the `iproxy` cache configuration. If the property is set to `SAFE`, root documents are fetched for every call.
- For the Next Generation interpreter, the root document in the initial page is always cached by the Web Proxy based on the `iproxy` cache configuration;

neither documentmaxage nor documentmaxstale property values affect it. The root documents in the other VoiceXML document (not the initial document) are cached, and the documentmaxage or documentmaxstale property value affects the cache behavior of the root document the same way as it affects the cache behavior of the VoiceXML document that owns the root document.

3.6.3 Caching policies

Here is a summary of caching in VoiceXML 2.1 on the VoiceGenie platform:

- The application server maintainer/content provider can provide guidelines for content expiry using the Cache-Control and Expires HTTP response headers
- If these headers are not present, the caching proxy will use heuristics to generate expiry times
- The application developer can deterministically control the caching behaviour of application resources using the maxage and maxstale attributes for each URI-related VoiceXML tag, *including forcing a validation of the current cache contents (using* maxage*), and accepting expired cache contents (using* maxstale*)*
- The platform maintainer can control cache resource usage using the caching proxy configuration
- The caching proxy generates HTTP/1.0 requests, but supports HTTP/1.1 caching functionality if iproxy.use_strict_caching_rules is set to true.

The primary impact of this is that the client has control over what it will accept from the cache, even if the server has specified an Expires header or maxage/maxstale attributes, or if the caching proxy has generated an expiry time itself.

VoiceXML 2.1 Caching Algorithms

The caching policy used by the VoiceXML interpreter context must adhere to the cache correctness rules of HTTP 1.1. In particular, the Expires and Cache-Control headers must be honored.

Documents from the web server will be delivered with zero, one, or both of the response headers. If an Expires header is present, it is used to set the expiry time of the object in the cache. If the Expiry header is not present, the caching proxy will apply a heuristic to set an expiry time. If a Cache-Control header is in the response, it will be used to control expiry times, and will override an Expiry time if also provided.

The following algorithm summarizes these rules and represents the interpreter context behaviour when requesting a resource:

```
If the resource is not present in the cache, fetch it from the
server using get.

If the resource is in the cache,
   If a maxage value is provided,
      If age of the cached resource <= maxage,
         If the resource has expired,
            Perform maxstale check.
         Otherwise, use the cached copy.
      Otherwise, fetch it from the server using get.
   Otherwise,
      If the resource has expired,
         Perform maxstale check.
      Otherwise, use the cached copy.
```

Here is the algorithm for the “`maxstale` check”:

```
If maxstale is provided,
   If cached copy has exceeded its expiration time by no more than
maxstale seconds,
      then use the cached copy.
   Otherwise, fetch it from the server using get.
Otherwise, fetch it from the server using get.
```

Note: It is an optimization to perform a “get if modified” (the request includes an `If-Modified-Since` (`IMS`) header) on a document still present in the cache when the policy requires a fetch from the server. The caching proxy in use on the VoiceGenie platform does perform this optimization.

VoiceXML allows the author to control this caching policy for each use of each resource.

Each resource-related element may specify `maxage` and `maxstale` attributes. Setting `maxage` to a non-zero value can be used to get a fresh copy of a resource that may not have yet expired in the cache. A fresh copy can be unconditionally requested by setting `maxage` to zero.

Using `maxstale` enables the author to state that an expired copy of a resource, that is not too stale (according to the rules of HTTP 1.1) may be used. This can improve performance by eliminating a fetch that would otherwise be required to get a fresh copy. It is especially useful for authors who may not have direct server-side control of the expiration dates of large static files.

Note: Note the fact that caching proxies using these techniques will not delete items from the cache after their expiry time, unless other cache requirements (i.e., memory or disk usage limits) dictate such action. The reason for this is that the client may specify that an expired resource is acceptable; this is done with the `maxstale` attribute.

While the `maxage` and `maxstale` attributes are drawn from and are directly supported by HTTP 1.1, some resources may be addressed by URIs that name protocols other than HTTP. If the protocol does not support the notion of resource age, the interpreter context shall compute a resource's age from the time it was received. If the protocol does not support the notion of resource staleness, the interpreter context shall consider the resource to have expired immediately upon receipt.

3.6.4 How to use maxage and maxstale attributes

Using the `maxage` and `maxstale` attributes can provide the document author with fine-grained control over when documents are returned from the cache, or fetched from the origin server. However, these do interact with server-provided expiry times as well. In order to 'force' particular behaviour, you can use `maxage` and `maxstale` to achieve your goals.

Here are some sample behaviours you might find interesting:

Desired Behaviour	maxage	maxstale	Notes
VoiceXML 1.0 caching="safe"	maxage="0"	maxage="0"	Caching based on `Expires` header; will use IMS for each reference
VoiceXML 1.0 caching="fast"	maxage="large_value"	maxstale="0"	Caching based on `Expires` header; will not consult server until expiry. On expiry, will use IMS.
Client control over Expiry	maxage="desired_expiry"	maxstale="0"	Caching based on `Expires` header; refetch based on `maxage` and `maxstale;` uses IMS.
Expired document acceptable	maxage="large_value"	maxstale="desired_maxstale"	Caching based on `Expires` header: refetch after `Expiry` time plus `maxstale;` uses IMS.

3.6.5 Determination of an Expiry Time

Web servers may or may not return an `Expires` response header to the client. In the event that they do, this expiry time is used in the cache refresh algorithm. If this information is instead provided as part of a `Cache-Control` header (using `maxage/maxstale`), this information will be used to control cache expiry.

The caching proxy used by VoiceGenie uses a Refresh-Rate model, rather than a time-based expiration model. Objects are no longer purged from the cache when they expire. Instead of assigning a ‘time-to-live’ when the object enters the cache, freshness requirements are checked when objects are requested. If an object is “fresh” it is given directly to the client. If it is “stale” then the caching proxy will make an `If-Modified-Since` request for it.

In the event that no `Expiry` header is returned, and that a relevant `Cache-Control` header is absent in the response, the following algorithm is used to calculate an expiry time.

Note: HTTP 1.1 does not mandate this algorithm, beyond noting that some heuristic is often used (Section 13.2.2). However, this algorithm, or one similar to it, is used by a number of such caching proxies.

3.6.6 Squid Caching Proxy

Squid Configuration

The Squid configuration file controls configuration of the caching proxy:

`/usr/local/squid/etc/squid.conf` (linux)

`C:\squid\etc\squid.conf` (windows)

This is a text file, which includes keywords and values. For example:

```
http_port 3128
```

defines the TCP port number, which the caching proxy will use for receiving requests (note the absence of an equal sign (=) here).

Changes to this file are not reflected in the running configuration immediately. It is necessary to issue the following command on the platform:

`/usr/local/squid/bin/squid -k reconfigure` (linux)

`C:\squid\sbin\squid.exe -k reconfigure -n squidNT` (windows)

in order to force a re-read of the configuration file.

In general, the default squid configuration file should be suitable for most installations. However, there are a number of common configuration elements that are addressed here. For details regarding all squid configuration items, see the *Squid Configuration Guide* (`http://squid.visolve.com/squid24s1/contents.htm`).

Using a Second-level Proxy Server

In order to configure for a second level proxy, add the following lines to the `squid.conf` file:

```
cache_peer parentcache.yourdomain.com parent 3128 0 no-query
default
acl local-servers dstdomain yourdomain.com
acl all src 0.0.0.0/0.0.0.0
never_direct deny local-servers
never_direct allow all
```

The bold items will need to be altered for the particular caching infrastructure. The required information includes the next proxy in the chain (`parentcache.yourcomain.com`), identification of domains that should not go through the parent proxy (`yourdomain.com`) and the port number on which the parent cache is listening (`8080`).

Squid Expiry Time Generation

When checking the object freshness, the following values are calculated:

- `AGE` is how much the object has aged since it was retrieved:

  `AGE = NOW - OBJECT_DATE`
- `LM_AGE` is how old the object was when it was retrieved:

  `LM_AGE = OBJECT_DATE - LAST_MODIFIED_TIME`
- `LM_FACTOR` is the ratio of `AGE` to `LM_AGE:`

  `LM_FACTOR = AGE / LM_AGE`
- `CLIENT_MAX_AGE` is the (optional) maximum object age the client will accept as taken from the `Cache-Control` request header.
- `EXPIRES` is the (optional) expiry time from the server reply headers. These values are compared with the parameters of the `refresh_pattern` rules (see Squid Specific Configuration Elements).

The refresh parameters are:

- `URL regular expression`
- `MIN_AGE`
- `PERCENT`
- `MAX_AGE`

The URL regular expressions are checked in the order listed until a match is found. Then this algorithm is applied for determining if an object is fresh or stale:

```
if (CLIENT_MAX_AGE)
    if (AGE > CLIENT_MAX_AGE)
       return STALE
if (AGE <= MIN_AGE)
    return FRESH
if (EXPIRES) {
    if (EXPIRES <= NOW)
       return STALE
    else
       return FRESH
}
if (AGE > MAX_AGE)
    return STALE
if (LM_FACTOR < PERCENT)
    return FRESH
return STALE
```

Note: the Max-Age in a client request takes the highest precedence. The MIN value should normally be set to zero since it has higher precedence than the server‘s Expires: value. But if you wish to override the Expires: headers, you may use the MIN value.

Squid Specific Configuration Elements

Squid allows control over refresh behaviour based on regular expression matching of request URIs. These would likely only be used for very specific situations, and it is unlikely that these need to be (or in fact should be) modified. The one exception could be a situation where you cannot configure your server to deliver Expires headers, and wish to change the defaults provided by squid.

Configuration elements include:

```
Tag Name refresh_pattern
Usage refresh_pattern [-i] regex min percent max [options]
```

- Min is the time (in minutes) an object without an explicit expiry time should be considered fresh. The recommended value is 0; any higher values may cause dynamic applications to be erroneously cached unless the application designer has taken the appropriate actions.
- Percent is a percentage of the objects age (time since last modification age) an object without explicit expiry time will be considered fresh.
- Max is an upper limit on how long objects without an explicit expiry time will be considered fresh.

Options:

- `override-expire`
- `override-lastmod`
- `reload-into-ims`
- `ignore-reload`
- `override-expire` enforces min age even if the server sent a `Expires:` header. Doing this *violates* the HTTP standard. Enabling this feature could make you liable for problems, which it causes
- `override-lastmod` enforces min age even on objects that was modified recently.
- `reload-into-ims` changes client `no-cache` or “reload'' to `If-Modified-Since` requests. Doing this *violates* the HTTP standard. Enabling this feature could make you liable for problems, which it causes.
- `ignore-reload` ignores a client `no-cache` or “reload'' header. Doing this *violates* the HTTP standard. Enabling this feature could make you liable for problems, which it causes.

A cached object is basically:

```
FRESH if expires < now, else STALE
STALE if age > max
FRESH if lm-factor < percent, else STALE
FRESH if age < min
else STALE
```

The `refresh_pattern` lines are checked in the order listed here. The first entry that matches is used. If none of the entries match, then the default will be used.

The default for `refresh_pattern` is set as: `refresh_pattern. 0 20% 4320`

The caching proxy logs can provide useful information when attempting to identify performance issues or resolve application problems. Following is a description of the contents of the proxy log files, and some guidelines on how to interpret the information in these files.

- Access Log
- Caching Proxy Log

Access.log field definitions

The squid `access.log` file can be found at either `/usr/local/squid/var/logs/` (linux) or `C:\squid\var\logs\` (windows). The native `access.log` has ten (10) fields. There is one entry for each HTTP (client) request and each ICP Query. HTTP requests are logged when the client socket is closed. A single dash (-) indicates unavailable data.

Timestamp

The time when the client socket is closed. The format is “Unix time” (seconds since Jan 1, 1970) with millisecond resolution. This can be modified to visible format by:

```
cat access.log | perl -nwe 's/^(\d+)/localtime($1)/e; print'
```

Elapsed Time

The elapsed time of the request, in milliseconds. This is time between the `accept()` and `close()` of the client socket.

Client Address

The IP address of the connecting client, or the fully qualified domain name (FQDN) if the `log_fqdn` option is enabled in the configuration file. This parameter is normally turned off for performance reasons.

Log Tag/HTTP Code

The Log Tag describes how the request was treated locally (hit, miss, etc). All the tags are described below. The HTTP code is the reply code taken from the first line of the HTTP reply header. Non-HTTP requests may have zero reply codes.

Size

The number of bytes written to the client.

Request Method

The HTTP request method, or `ICP_QUERY` for ICP requests.

URL

The requested URL.

Ident

If `ident_lookup` is on, this field may contain the username associated with the client connection as derived from the ident service. This lookup is typically turned off for performance reasons.

Hierarchy Data/ Hostname

A description of how and where the requested object was fetched.

Content Type

The `Content-type` field from the HTTP reply.

Access Log Tag/HTTP Code

`TCP_` refers to requests on the HTTP port.

TCP_HIT	A valid copy of the requested object was in the cache.
TCP_MISS	The requested object was not in the cache

TCP_REFRESH_HIT	The object was in the cache, but STALE. An If-Modified-Since request was made and a 304 Not Modified reply was received.
TCP_REF_FAIL_HIT	The object was in the cache, but STALE. The request to validate the object failed, so the old (stale) object was returned.
TCP_REFRESH_MISS	The object was in the cache, but STALE. An If-Modified-Since request was made and the reply contained new content.
TCP_CLIENT_REFRESH	The client issued a request with the no-cache pragma.
TCP_CLIENT_REFRESH_MISS	The client issued a no-cache pragma, or some analogous cache control command along with the request. Thus, the cache has to refetch the object from origin server. It is users pushing that reload-button forcingthe proxy to check for a new copy (also triggered by selecting a bookmark some browser versions). In short, the browser forced the proxy to check for a new version
TCP_IMS_HIT	The client issued an If-Modified-Since request and the object was in the cache and still fresh. TCP_HIT and TCP_IMS_HIT are hits, the only difference is that in the TCP_IMS_HIT case the browser already had an up to date version so there was no need to send the Squid cached copy to the requestor.
TCP_IMS_MISS	The client issued an If-Modified-Since request for a stale object.
TCP_NEGATIVE_HIT	A previously failed request is satisfied from the cache, as the proxy believes it still be a problem.
TCP_SWAPFAIL	The object was believed to be in the cache, but could not be accessed.
TCP_DENIED	Access was denied for this request

Inter-Cache Protocol Entries

UDP_ refers to requests on the ICP port

UDP_HIT	A valid copy of the requested object was in the cache.
UDP_HIT_OBJ	Same as UDP_HIT, but the object data was small enough to be sent in the UDP reply packet. Saves the following TCP request.
UDP_MISS	The requested object was not in the cache.
UDP_DENIED	Access was denied for this request.
UDP_INVALID	An invalid request was received.
UDP_RELOADING	The ICP request was “refused” because the cache is busy reloading its metadata.

Refreshing an Object in the Cache

From the System Managemant Console (SMC), click on the `Operations` tab and click on `View Cache`, select the platform and click on either `View In Memory Cache` or `View All Cache`. Find the cached object you would like to reload and click on the `Reload` link.

For Linux:

From the Linux shell, issue the following command, replacing the `<uri>` with the full URI of the object:

```
/usr/local/squid/bin/client -s -r <uri>
```

For Windows:

From the cmd Console Window, issue the following command and replace the `<uri>` with the full URI of the object:

```
C:\VoiceGenie\cmp\cmp-proxy\scripts\cacheclient.bat fetch <uri>
```

Purging an Object from the Cache

From the System Management Console (SMC), click on the `Operations` tab and click on `View Cache`, select the platform and click on either `View In Memory Cache` or `View All Cache`. Find the cached object you would like to purge and click on the `Purge` link.

For Linux:

From the Linux shell, issue the following command, replacing the `<uri>` with the full URI of the object

```
/usr/local/squid/bin/client -s -m <PURGE> <uri>
```

For Windows:

From the cmd Console Window, issue the following command and replace the `<uri>` with the full URI of the object:

```
C:\VoiceGenie\cmp\cmp-proxy\scripts\cacheclient.bat purge <uri>
```

Clearing the entire Cache

From the System Management Console (SMC), click on the `Operations` tab and click on `Start/Stop Cache`, select `Restart` and the platform on which you would like to clear the cache. Then, select `Yes` for `Purge All at Start`, then click on the `Execute` button.

For Linux:

From the Linux shell, issue the following commands:

```
/usr/local/squid/bin/squid -k shutdown
echo "" > /usr/local/squid/cache/swap.state
/usr/local/squid/bin/squid
```

For Windows:

From the cmd Console Window, issue the following command:

```
C:\VoiceGenie\cmp\cmp-proxy\scripts\startcache.bat restart purgeall
```

Chapter

4 Network Interfaces

For 7.2, the Media Platform can be deployed in VoIP-only telephony configurations:

VoIP Only: no telephony hardware required; both call control and media operations are over IP. The Media Platform uses SIP and/or H.323 for call control and RTP for media operation signaling. Call control and media traffic transit one of the LAN interfaces.

4.1 SIP

4.1.1 Standards

The Media Platform SIP implementation of VoIP is compliant to the following standards:

- RFC 3261 Session Initiation Protocol (SIP)
- RFC 2327 Session Description Protocol (SDP)
- RFC 1889 Real-time Transport Protocol (RTP)
- RFC 2833 RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals
- RFC 2976 The SIP INFO Method
- RFC 3515 The SIP REFER Method
- RFC 3264 An Offer/Answer Model with the Session Description Protocol (SDP)
- Most extensions from proposed IETF draft “A SIP Interface to VoiceXML Dialog Servers” (http://tools.ietf.org/id/draft-burke-vxml-02.txt)
- Some extensions from proposed IETF draft “Basic Network Media Services with SIP” (http://www.ietf.org/rfc/rfc4240.txt)

Please refer to 4.3 RTP Support to see the list of supported codecs.

4.1.2 SIP Call Connection Mechanisms

When both the physical network and necessary call routing parameters have been properly configured, the Media Platform will be able to receive calls using SIP in the following ways:

- **Direct Connect** – Directly from another SIP endpoint (i.e. IP phone), at a SIP address of sip: `<dnis>@machine.voicegenie.com [: 5060]`. This assumes that the VoiceGenie software is not configured to run on an alternate endpoint or to restrict incoming calls.
- **Via the SIP proxy server** – if the VoiceGenie software is configured to register with the SIP proxy server at a particular address, and the SIP proxy server is configured to accept this registration.
- **Integrated media and signalling gateways** – The Media Platform can receive SIP and RTP data from PSTN/VoIP gateways such as the Cisco AS5300 gateway. On some gateways it is possible to configure any of the above two addresses to be referred by a PSTN phone number.
- **Softswitch architecture** – The Media Platform can also work with a softswitch controller and SIP application server to participate in a fully distributed softswitch architecture
- **CCXML Platform** – The Media Platform can be directly controlled by a CCXML Platform

Once connected, and with the exception of call transfer, there should be no notable differences between the operation of a SIP based platform and a PSTN-based platform.

4.1.3 Interoperability

VoiceGenie has performed interoperability testing with the following devices:

- Cisco Universal Access Gateways, including the AS5300, AS5350, AS5400 and AS5850
- Cisco 7960 IP Phone
- Cisco ATA-186 analog gateway (residential gateway)
- AudioCodes media gateways, including the IPMedia 2000, Mediant 2000 and MP-104 gateways
- Pingtel xpressa IP phone
- Pingtel instant xpressa soft-phone
- Vovida SIP Proxy Server
- X-lite

- And various other soft-phone clients

4.1.4 SIP INFO support

One way for SIP User Agents to communicate with each other within a call dialog session is via the SIP `INFO` method. The VoiceGenie 7.2 Media Platform is capable of sending and receiving SIP `INFO` messages.

Sending SIP INFO messages with the Legacy Interpreter

A VXML application can trigger the sending of a SIP `INFO` message by using the `<log>` tag with `dest="callmgr"` within a call session. This will generate an application event from the VXML Interpreter to the Call Manager component in the Media Platform. Call Manager will then generate a SIP `INFO` message to be sent to the remote end of this SIP call dialog. By default, the content type of this message is `application/text.` Currently, the content type is configurable using the platform-wide Call Manager configuration parameter sip.info.contenttype.

For example, the following `<log>` tag can be defined to send an application event to Call Manager:

```
<block>
 <log dest="callmgr">abc=123;def=567</log>
</block>
```

When the above application event is received, Call Manager will send the following SIP `INFO` message:

```
INFO sip:12345@205.150.90.93 SIP/2.0
Via: SIP/2.0/UDP 10.0.0.254:5060;branch=z9hG4bK0842fa306569b6
From: sip:unknown@10.0.0.254:5060;tag=AFCD0B00-34F6-F636-AE8D-
   3CB2ED51A556
To: <sip:12345@205.150.90.93>;tag=AB230400-CE35-4817-C313-
   3F55A485C57
Max-Forwards: 70
Call-ID: AFCD0B00-34F6-206F-5109-41DD81235E8F@205.150.90.93
Contact: sip:VoiceGenie@10.0.0.254:5060
CSeq: 2 INFO
Content-Type: application/text
Content-Length: 15

abc=123;def=567
```

Receiving SIP INFO messages with the Legacy Interpreter

Call Manager is responsible for receiving SIP `INFO` messages and passing the content of these messages to the VXML application. After a SIP `INFO` message is received, Call Manager will examine the content type of the message. By default, all non-DTMF (SIP `INFO` DTMF events have content `type="application/dtmf-relay"`) SIP `INFO` messages will be notified to VXML Interpreter. Filtering can be applied using call manager configuration parameter, `sip.sipinfoallowedcontenttypes,` to provision a list of content types that are allowed. Call Manager will pass the SIP `INFO` content body to VXML Interpreter as a SIP `INFO` event.

At the application level, VoiceGenie proprietary Call Control extension already provides a well-defined mechanism to receive external messages. When a VoiceXML Interpreter session receives a message sent from another VoiceXML session or from `clc` command (`sendevent`), event `com.voicegenie.message` will be generated.

SIP `INFO` events will be handled using the same mechanism. When a SIP `INFO` message is received, an event `com.voicegenie.message` will be generated which can be thrown to the application immediately or be queued, depending on the property `com.voicegenie.messagehandling` (`queue/immediate/discard`). The queued event will be thrown before executing a form item or a menu. If the parameter `com.voicegenie.processmessagequeue` is set to `true,` the queued event will be thrown before executing a form item or a menu, otherwise sip info can only be received using synchronous `<receive>`.

There are four shadow variables available for `<receive>`.

Property	Description
name$.msgsourcetype	One of `vxmlisession`, `CLC` or `sipinfo.`
name$.msgsource	The instance ID of the sender; undefined if `msgsourcetype` is `sipinfo.`
name$.contenttype	The content type of the sip info. It’s undefined if `name.msgsourcetype` is not `sipinfo.`
name$.msgcontent	The contents of the message.

When `event com.voicegenie.message` is thrown, the application must use `<receive>` tag to access the message through shadow variables. Below is an example of using `<receive>` when `com.voicegenie.messagehandling = immediate` or `queue` and `com.voicegenie.processmessagequeue = true:`

```
<catch event="com.voicegenie.message">
<receive name="info" mode="first"/>
<log> The message type :<value expr="info$.msgsourcetype"/> </log>
<log> The content is : <value expr="info$.msgcontent"/> </log>
```

```
 <log> content type is : <value expr="info$.contenttype"/> </log>
 </catch>
```

An example of using synchronous `<receive>` to receive sip info:

```
<block>
<receive name="msg" mode="first" maxtime="20"/>
 <if cond="msg=='success'">
 The message type :<value expr="info$.msgsourcetype"/>
 The content is : <value expr="msg$.msgcontent"/>.
 The content type is : <value expr="msg$.contenttype"/>.
 </if>
</block>
```

When synchronous `<receive>` is executed, vxmli would be waiting for message within certain time (specified by `maxtime`).

Sending SIP INFO messages with the Next Generation Interpreter

A VXML application can send a SIP INFO message by using the <vg:send> tag. The <vg:send> tag allows the VXML Interpreter session to send a request to the Call Manager. Call Manager would then generate a SIP INFO message that is sent to the caller.

If either the body, bodyexpr, event or eventexpr attribute of the <vg:send> tag is specified, then the content of that attribute will appear as content of the SIP INFO message. If the namelist attribute of the <vg:send> tag is specified, and the namelist contains a single element which is a number or a string, then this element's value will appear as content of the SIP INFO message. Otherwise, the ECMAscript variables of the namelist are converted into a string of name/value pairs, and this string will appear as content of the SIP INFO message. The format of the string is:

"<name1>=<value1>;<name2>=<value2>;<name3>=<value3>"…

The value of an ECMAScript object is output with the following format,: "{<property_name1>=<value1>;<property_name2>=<value2>;<property_name2>=<value2>…}"

If the contenttype or contenttypeexpr attribute of the <vg:send> tag is specified, then the content of that attribute will appear as the Content-Type header of the SIP INFO message. If the target or targetexpr attribute is specified, it is ignored by the platform, and the message is always sent to the caller of the page. Note that there is no confirmation response for a send request.

The following example demonstrates the use of <send>:

```
<vxml version="2.1"
```

```
  xmlns="http://www.w3.org/2001/vxml">

  <form>
  <field name="user_pin" type="digits">
    <prompt>please enter pin</prompt>
    <filled>
      <vg:send async="false"
            bodyexpr="'&lt;pin&gt;' + user_id +
'&lt;/pin&gt;'"
            contenttype="text/xml"/>
    </filled>
  </field>
  </form>
</vxml>
```

The <vg:send> tag also supports the async boolean attribute, but it currently has no effect

Receiving SIP INFO messages with the Next Generation Interpreter

Call Manager is responsible for receiving SIP `INFO` messages and passing the content and content-type of these messages to the VXML application. After a SIP `INFO` message is received, Call Manager will pass the information to the VXML Interpreter. The SIP INFO message content will be exposed via the VXML shadow variable application.lastmessage$.content, and the Content-Type SIP message header will be exposed as application.lastmessage$.contenttype.

A VXML application can receive SIP INFO messages in asynchronous or synchronous mode. The VXML property com.voicegenie.externalevents.enable boolean controls the mode. When set to true, asynchronous mode is used and when set to false (default) synchronous mode is used.

Synchronous Mode:

In synchronous mode the <vg:receive> tag is used to receive a SIP INFO message within a VXML application. When a <vg:receive> element is executed it checks to see if any external SIP INFO messages are available. If available, the oldest SIP INFO message is processed. Otherwise <vg:receive> waits until an external event is received and then processes it. The time specified by the "maxtime/maxtimeexpr" attribute can be used to limit the maximum time that <vg:receive> will wait. If this time elapses, then an error.badfetch is thrown. When more than one external SIP INFO message is available, <vg:receive> must be called separately for each SIP INFO message. The <vg:receive> tag processes SIP INFO messages by populating the application.lastmessage$.content and application.lastmessage$.contenttype shadow variables.

The com.voicegenie.externalevents.queue boolean property controls whether SIP INFO messages are queued. When this property is set to false, incoming SIP INFO messages will only be processed when <vg:receive> is actively waiting at the time the message arrives; otherwise, the incoming SIP INFO message is discarded. When set to true, SIP INFO messages will be queued and made available for the next <vg:receive> call.

Example:

```
<vxml version="2.1"
  xmlns="http://www.w3.org/2001/vxml">

  <property name="com.voicegenie.externalevents.queue"
value="true"/>
  <form>
    <catch event="error.badfetch">
      <log>timed out waiting for SIP INFO message</log>
    </catch>

    <block>
      One moment...
      <receive maxtime="10s”/>
      <log>SIP INFO msg: <value
expr="application.lastmessage$.content"/></log>
    </block>
  </form>
</vxml>
```

Asynchronous Mode:

When the com.voicegenie.externalevents.enable property is set to true, external messages will be handled asynchronously. All incoming external events will be queued up by the VXML session. Just before the VXML application enters a waiting state (when executing <field>, <initial>, <transfer>, or <record>), it checks to see if an external SIP INFO message is pending. If at least one received message is pending, the application will throw an 'externalmessage' VoiceXML event. This received message event is no longer considered pending. In addition, the application.lastmessage$.content and application.lastmessage$.contenttype shadow variables will be populated.

In asynchronous mode, when a VXML application attempts to enter a waiting state while an external SIP INFO message is pending, there are certain cases when the application will not throw an ‘externalmessage’ event. For instance, when a 'connection.disconnect' event is pending, it takes precedence over the 'externalmessage' and the latter is not thrown. When the VXML session is already in the final processing state, it must carry out an implicit exit when it attempts to enter a waiting state, and the 'externalmessage' event is not thrown.

In asynchronous mode, a SIP INFO message could be received while the application is waiting for input. This would cause the ‘externalmessage' VoiceXML event to be thrown and cause the active input request to be aborted.

When 'externalmessage ' is thrown, it is thrown in the scope of the form item. After the ‘externalmessage’ event is handled, FIA resumes as normal by selecting the next form item to visit. If another message is pending on the external message queue, it will be processed before the next waiting state. Also, in asynchronous mode the <vg:receive> tag behaves the same as it would in synchronous mode with com.voicegenie.externalevents.queue set to true.

Example:

```
<vxml version="2.1"
  xmlns="http://www.w3.org/2001/vxml">
  <property name="com.voicegenie.externalevents.enable"
value="true"/>
  <catch event="externalmessage">
      <log>received <value expr="lm.content"/></log>
    <script>
  </catch>

  <form>
  <field name="num" type="digits">
    <prompt>pick a number</prompt>
    <catch event="noinput nomatch">
      Try again
      <reprompt/>
    </catch>
    <filled>
      You said the number <value expr="num"/>
      <clear/>
    </filled>
  </field>
  </form>
</vxml>
```

For more details about <send> and <receive>, please refer to http://developer.voicegenie.com.

4.1.5 SIP customizable headers and parameters

The SIP version (using SIP2 line manager) of VoiceGenie 7.2 Media Platform supports propagation of SIP headers, parameters and request URI parameters to the VXML applications for incoming SIP messages, and customization of SIP headers, parameters and request URI parameters for outgoing SIP messages.

The VoiceGenie 7.2 Media Platform can be configured to pass incoming SIP INVITE requests' URI's parameters, headers, and parameters of any headers to the VXML application as session variables.

The Call Manager configuration parameters sip.in.invite.headers and sip.in.invite.params can be defined to abstract information from incoming SIP messages. They will generate variables to be sent from the Call Manager to the VXML Interpreter in Sip.Invite.<headername> and Sip.Inivte.<headername>.<paramname> formats respectively. Sip.Invite.<headername> will contain the header value, as well as all its parameters. Sip.Invite.<headername>.<paramname> will contain the value of a specific header parameter.

Detailed information for the above two parameters, along with examples, can be found in the:

VoiceGenie 7.2 Media Platform System Reference Guide, under the "Customizable Headers and Parameters" section of the SIP line manager.

The Legacy Interpreter configuration parameter session_vars and the Next Generation Interpreter configuration parameter vxmli.session_vars can be defined to provide the session variables to the VXML applications. For details please see the *VoiceGenie 7.2 Media Platform System Reference Guide,* under the "Next Generation Interpreter" section.

Here is an example configuration for exposing request URI's paramA, request URI's paramB, From header, and To header's paramC:

Call Manager:

```
sip.in.invite.headers=From
sip.in.invite.params=RequestURI To
```

Next Generation Interpreter:

```
Vxmli.session_vars=...|session.connection.protocol.sip.invite.from
   |Sip.Invite.From|0|session.connection.protocol.sip.invite.reques
   turi.paramA|Sip.Invite.RequestURI.paramA|0|session.connection.pr
   otocol.sip.invite.requesturi.paramB|Sip.Invite.RequestURI.paramB
   |0|session.connection.protocol.sip.invite.to.paramC|Sip.Invite.T
   o.paramC|0
```

Legacy Interpreter:

```
session_vars=...|session.connection.protocol.sip.invite.from
   |Sip.Invite.From|0|session.connection.protocol.sip.invite.reques
   turi.paramA|Sip.Invite.RequestURI.paramA|0|session.connection.pr
   otocol.sip.invite.requesturi.paramB|Sip.Invite.RequestURI.paramB
   |0|session.connection.protocol.sip.invite.to.paramC|Sip.Invite.T
   o.paramC|0
```

With the configuration above and the following SIP `INVITE` message:

```
INVITE sip:test1@10.0.0.25;paramA=valueA;paramB=valueB SIP/2.0
Via: SIP/2.0/UDP 205.150.90.207:5060;branch=z9hG4bK0809fb404f9bcd
From: <sip:VoiceGenie@205.150.90.207:5060>;tag=9FB30200-B96C-01D0-
5052-C114EBCA0416
To: <sip:test1@10.0.0.25>;paramC=valueC
Max-Forwards: 70
CSeq: 1 INVITE
Call-ID: 9FB30200-B96C-C781-2A00-F3B654BEA9AD@205.150.90.207:5060
Contact: sip:VoiceGenie@205.150.90.207:5060
Content-Length: 190
Content-Type: application/sdp

v=0
o=Cisco-SIPUA 2455 9673 IN IP4 205.150.90.208
s=SIP Call
c=IN IP4 205.150.90.208
t=0 0
m=audio 30400 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

the following session variables will be defined:

Session variable	Value
session.connection.protocol.sip.invite.from	<sip:VoiceGenie@205.150.90.207:5060>;tag=9FB30200-B96C-01D0-5052-C114EBCA0416
session.connection.protocol.sip.invite.requesturi.paramA	valueA
session.connection.protocol.sip.invite.requesturi.paramB	valueB
session.connection.protocol.sip.invite.to.paramC	valueC

The VoiceGenie 7.2 Media Platform can also be configured to set outgoing SIP `INVITE` or `REFER` requests' URI's parameters, headers, and parameters of any headers (limitations) using signaling variables from VXML application. This feature is supported for `<transfer>` and remdial calls, and can be enabled by configuring `sip.out.invite.headers`, `sip.out.invite.params`, sip.out.refer.headers and sip.out.refer.params. Again please see the VoiceGenie 7.2 Media Platform regarding the details for these parameters.

The following is an example Call Manager configuration for customizing request URI's paramA, request URI's paramB and HeaderC in outgoing INVITE messages (for <transfer> involving two call legs and remdial calls):

```
sip.out.invite.headers=HeaderC
sip.out.invite.params=RequestURI
```

If the following signaling variables are defined (or the equivalent name/value list is defined and appended to the remdial call request):

```
    Sip.Invite.RequestURI.paramA=valueA
    Sip.Invite.RequestURI.paramB=valueB
    Sip.Invite.HeaderC=valueC
```

Then, the following SIP INVITE message will be generated for the outgoing call:

```
INVITE sip:test1@10.0.0.25;paramA=valueA;paramB=valueB SIP/2.0
Via: SIP/2.0/UDP 205.150.90.207:5060;branch=z9hG4bK0809fb404f9bcd
From: <sip:VoiceGenie@205.150.90.207:5060>;tag=9FB30200-B96C-01D0-
5052-C114EBCA0416
To: <sip:test1@10.0.0.25>
Max-Forwards: 70
CSeq: 1 INVITE
Call-ID: 9FB30200-B96C-C781-2A00-F3B654BEA9AD@205.150.90.207:5060
Contact: sip:VoiceGenie@205.150.90.207:5060
HeaderC: valueC
Content-Length: 190
Content-Type: application/sdp

v=0
o=Cisco-SIPUA 2455 9673 IN IP4 205.150.90.208
s=SIP Call
c=IN IP4 205.150.90.208
t=0 0
m=audio 30400 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

4.1.6 Codec Negotiation

SIP systems can be configured to support multiple codec simultaneously using the parameter mpc.codec accessible via SMC under the Call Manger Configuration section.

mpc.codec allows a space-delimited list that specifies the list of codec that can be supported. For SIP calls, the VoiceGenie 7.2 Media Platform performs media capability exchange using the SDP Offer/Answer model (RFC3264). During SIP/SDP media negotiation, the codec list will be used as the offering media capabilities from the Media Platform, or used to match the remote's offer to generate the answering media capabilities from the Media Platform. Normally in SIP, the media capabilities are exchanged in the `INVITE/200OK/ACK` sequence. The caller usually includes a SDP offer in the `INVITE` message. However, the VoiceGenie Media Platform can also support incoming `INVITE` without a SDP so that the platform will generate a SDP offer in the `200OK` response. As well, the VoiceGenie Media Platform can handle re-`INVITE/200OK/ACK` for in-call media information updates. For outgoing calls, the VoiceGenie Media Platform also supports incoming SDP in the `183 Session Progress` response. Note that multiple audio codec can be used within a single SIP call if both VoiceGenie Media Platform and the remote endpoints are configured to negotiate multiple codec for a call session.

If media negotiation returns more than one supported codecs, the parameter `mpc.transmitmultiplecodec` can specify whether to allow transmission of all supported codecs or restrict transmission to only one codec. If it is set to `1` (default), more than one codec can be transmitted. If it is set to `0,` only the codec at the top of the negotiated codec list will be transmitted.

4.1.7 Enabling SIP TCP Support

The `sip.transport.X` parameters configures the SIP stack's transport settings. By default `sip.transport.0` has the value `transport0 udp:any:5060.`

To enable TCP transport:

1. Enable `sip.transport.1` and assign its value as `transport1 tcp:any:5060`
2. Enable `sip.route.default.udp` and assign its value as `0`
3. Enable `sip.route.default.tcp` and assign its value as `1`

4.1.8 Burke Draft Support

The VoiceGenie 7.2 Media Platform with the Next-Generation VoiceXML Interpreter supports many features described in the Burke Draft. Please refer to Appendix A for a detailed list of supported Burke Draft features.

4.1.9 Limitations

- Volume Control is not supported for VoIP protocols, including SIP
- Speed Control is not supported for VoIP protocols, including SIP

4.2 H.323

4.2.1 Standards

The Media Platform H.323 implementation of VoIP is compliant to the following standards:

- ITU H323 v4
- ITU H4501
- ITU H4502

Please refer to 4.3 RTP Support to see the list of supported codecs.

4.2.2 Architectures

When both the physical network and necessary call routing parameters have been properly configured, the Media Platform will be able to receive calls using H.323 in the following ways:

- **Direct Connect** – Directly from another H.323 endpoint (i.e. IP phone, software phones), at a H.323 IP address: `<dnis>@machine.voicegenie.com[:1720]`. This assumes that the VoiceGenie software is not configured to run on an alternate endpoint or to restrict incoming calls.
- **H.323 Gatekeeper** – if the VoiceGenie software is configured to register with the H.323 gatekeeper using one or more alias(es) via Admission Reqeuest (ARQ). Usage of ARQ for inbound and outbound calls can be configured independently.

  The alias(es) can be in any ASCII format:

  - PSTN phone number such as `4167360905`
  - A string such as `VGMediaServer`.
- **Softswitch architecture** – The Media Platform can also work with a softswitch controller and SIP/H.323 application server to participate in a fully distributed softswitch architecture
- **Integrated media and signalling gateways** – The Media Platform can receive H.323 and RTP data from PSTN/VoIP gateways such as the Cisco AS5300 gateway. On some gateways it is possible to configure any of the two aforementioned H.323-endpoint identifying mechanisms also as a PSTN phone number.

Once connected, and with the exception of call transfer, there should be no notable differences between the operation of a H.323 based platform and a PSTN-based platform.

4.2.3 Interoperability

VoiceGenie has performed interoperability testing with the following devices:

- Cisco Universal Access Gateways, including the AS5300, AS5350, AS5400 and AS5850
- Cisco 2621/7204 H323 gatekeeper
- Avaya Definity switching software
- Microsoft Netmeeting
- And various other soft-phone clients

VoiceGenie is also deployed within the AT&T V-Plus architecture, and the Voxeo hosted application infrastructure.

4.2.4 Codec Negotiation

H.323 systems can be configured to support a single codec only at a time, using the `h323.codec` parameter accessible via SMC under the `Call Manager Configuration` section.

H.323 media capability exchange can happen in H.245 TCS (Terminal Capability Set) exchange. When faststart and/or tunneling are enabled, the media capability can be included in faststart element or tunneled in the Q.931/H.225.0 call control messages. However, the VoiceGenie Media Platform only supports a single codec simultaneously for H.323 calls.

4.2.5 Limitations

- Volume Control is not supported for VoIP protocols, including H.323
- Speed Control is not supported for VoIP protocols, including H.323

4.3 RTP Support

4.3.1 Standards

The Real Time Protocol (RTP) is used by the VoiceGenie 7.2 Media Platform to send and receive media – which is typically an audio/video conversation – on connections to an end user of the system. RTP streams are carried between a UDP port on the media platform, and a UDP port on a media gateway, IP phone, or other VoIP device. The RTP streams are used in conjunction with SIP and H.323 protocols. Please see 4.1 SIP and 4.2 H.323 regarding SIP and H.323.

Note: The UDP port that the Media Platform uses to receive media is also the port that it uses to send media.

The VoiceGenie Media Platform RTP implementation is compliant to the following standards:

- RFC 3550 Real-time Transport Protocol (RTP)
- RFC 2833 DTMF digit signalling using RTP
- RFC 2190 RTP Payload Format for H.263 Video Streams
- RFC 2429 RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H263+)
- RFC 3267 Real Time Transport Protocol (RTP) Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs

The VoiceGenie Media Platform supports the following codec in the RTP implementation:

- pcmu (G.711 mulaw)
- pcma (G.711 alaw)
- g726 (G.726 32-bit)
- gsm (GSM 6.10)
- g729 (G.729 and G.729A)
- amr (AMR-NB)
- aurora (Aurora)

Note: Aurora is only supported on SIP systems. It is only supported as an input audio format (ie, the VoiceGenie Media Platform does not generate Aurora format output) with ASR engines that can support Aurora audio format.

- tfci (proprietary text based)

Note: Tfci (Telephony Free Client Interface) is a proprietary text-based media format where TTS, audio filenames, and ASR recognition results are transmitted as text. This is currently supported on platforms configured as SIP only.

- h263 (H.263 video)
- h263-1998 (H.263+ video)

Note: The VoiceGenie Media Platform only supports H.263 mode-A transmission as specified in rfc 2190. This is currently supported on platforms configured as SIP only.

4.3.2 General Usage

The UDP ports used to carry RTP traffic are chosen dynamically by the Media Platform on a call-by-call basis. In some network environments, it may be desirable to ensure that UDP ports selected by the media platform for RTP traffic are within a certain range. For example, when the media platform is deployed behind a firewall, it may be desirable to constrain the RTP ports to a particular range that the firewall is configured to pass through, so that VoIP devices that are outside the firewall are able to communicate with the media platform. The configuration parameters mpc.rtp.portlow and mpc.rtp.porthigh can be used to specify a particular range of UDP ports to be used for sending/receiving RTP streams.

4.3.3 DTMF

The VoiceGenie 7.2 Media Platform can support sending of in-band DTMF audio. However, in-band DTMF detections are not supported on the VoiceGenie 7.2 Media Platform under VoIP configuration. Instead, the VoiceGenie 7.2 Media Platform relies on out-of-band events for DTMF detection under VoIP environment. Currently, we support three types of out-of-band DTMF events:

- **RFC2833** – This is the most typical way to transmit DTMF events on VoIP systems. The VoiceGenie Media Platform can detect RFC2833 packets on the RTP stream as incoming DTMF digits.
- **SIP INFO** – When using SIP, the VoiceGenie Media Platform is also capable of detecting incoming DTMF digits through SIP `INFO` messages with content-type `application/dtmf-relay`. The body of the SIP `INFO` messages should be in the `Signal= <digit>` format. The VoiceGenie Media Platform can also generate DTMF events using the same formatted SIP `INFO` messages when `sip.sipinfodtmf` is set to `1` (default is `0`) in Call Manager configuration. The DTMF event will be generated when the VXML application tries to play an audio file that is named `dtmf_<digit>`.vox `or dtmf_<digit>.wav.`
- **H.245 events** – The H.323 line manager is capable of transmitting/receiving DTMFs through the H.245 channel using H.245 user indication events.

4.4 Multiple Line Managers

The VoiceGenie Media Platform has the flexibility to allow multiple line managers to be configured and operate concurrentldy. A Media Platform can be configured to handle both SIP/RTP and H323/RTP calls simultaneously. This flexible configuration allows experimental integration (between VoIP protocols) to be performed easily without re-configurating the VoiceGenie Media Platform.

Chapter

5 Call Control

5.1 Incoming Call

As mentioned in Running Applications on the Media Platform, the Media Platform selects an application to handle an incoming calls based on the DNIS-URL mapping.

H.323 calls have DNIS values embedded in the protocol's messages.

With SIP, calls do not have an explicit DNIS value. Instead, the `To:` field in the incoming `INVITE` message is used to route the call. The SIP URL in the `To:` field is always in the form `sip:user[:password]@address[:port]     [;uri-parameters]`, where the `[:password], [:port] and [;uri-parameters]` are optional and may not be included. The default SIP port of 5060 is used if the `[:port]` is not included. The Media Platform treats the `[:password]` as part of the DNIS. Calls from the PSTN are always redirected to a corresponding SIP URL, so regardless of whether or not a gateway is used, the `To:` field will always be in the format indicated above.

If the user portion of the SIP URL is of the form:

```
dialog.vxml.<value>
```

then `<value>` will be used as the initial VXML URL. Note that `<value>` must encode special characters, such as `:`, that are not permissible in the SIP user field. This usage is based on IETF draft `draft-rosenberg-sip-vxml-00`.

For example, a call to:

```
sip:dialog.vxml.http%3A%2F%2Fvxml.com%2Ftest.vxml@heart.voicegenie.com
```

would result in `http://vxml.com/test.vxml` being used as the initial URL.

Here, `:` is encoded as `%3A` and / is encoded as `%2F`.

Otherwise, the user portion of the SIP URL is treated as the DNIS for the call, and the appropriate entry from DNIS–URL mapping will be used. For instance, a call to `sip:1005@heart.voicegenie.com` would use the DNIS entry for 1005.

Note: If the SIP URL is sent as sip:dialog.vxml.http:...@address or sip:dialog.vxml.file:...@address), only "http" or "file" will actually be used as the <value> (i.e. the initial VoiceXML URL), and everything else up to "@" will be taken as a password. The ':' will probably not be encoded automatically by the calling SIP phone, so the caller should encode it directly in the SIP URL that they enter.

The VXML URL can also be specified using the format based on IETF draft `http://www.ietf.org/rfc/rfc4240.txt`.

For example, a call to:

```
sip:dialog@mediaserver.example.net;voicexml=http://vxmlserver.exa
   mple.net/cgi-bin/script.vxml
```

would result in `http://vxmlserver.example.net/cgi-bin/script.vxml` being used as the initial URL.

When a VXML URL is specified in the SIP INVITE, the SIP Request URI parameter gvp.appmodule can be used to select the VoiceXML interpreter. One can specify gvp.appmodule=VXML for the Legacy Interpreter, and gvp.appmodule-VXML-NG for the Next Generation Intepreter. For instance, the following SIP Request URI will result in Next Generation Intepreter being selected as the interpreter.

```
sip:dialog@mediaserver.example.net;voicexml=http://vxmlserver.exa
   mple.net/cgi-bin/script.vxml;
   gvp.appmodule=VXML-NG
```

If `gvp.appmodule` is not specified, the default VoiceXML Interpreter specified by the Call Manager configuration parameter `sessmgr.default_vxml_interpreter` will be used.

5.2 Outgoing Call

There are two principal mechanisms of initiating outgoing calls with the Media Platform:

- Application-initiated
- Remote Dial

An application, during its interaction with an existing call, can initiate a new outgoing call on a different call leg. For VXML applications, this can be performed using `<call>` tag, or using `<transfer>` tag that performs bridge transfer.

Remote Dial can be viewed as an asynchronous way to initiate outgoing call. Rather than using an existing application to initiate a call, a VXML application can be specified and associated with the new outgoing call. Please see

Chapter

5

Call Control

5.1 Incoming Call Chapter 5 “Incoming Call” section for more information.

When initiating an outbound call using the above mechanisms, sometimes it is desirable to enforce a certain restrictions on the allowable outgoing destination to prevent application misusages. It is also desirable to perform certain automatic address pattern manipulation and translation. The Media Platform supports the use of Dialing Rules, which provides a powerful way to manipulate telephone numbers that are being used for initiating an outbound call. Please see the next section for more information.

Note: For certain switches, dialing `+1` before the local phone number will result in rejected calls.

5.2.1 Dialing Rules

Overview

The Dialing Rules entry can be access under the `Other Configuration` section of the `Configuration` tab on the SMC. It provides a powerful way to manipulate telephone numbers that are being used for initiating an outbound call. This might include the number used for a bridged outbound call (using the `<transfer>` tag), a non-bridged outbound call (for particular call release technologies), or for calls placed with the `<call>` tag, or using the outbound calling interface.

The dialing rules file configures the following capabilities:

- Calls to undesirable numbers can be blocked.
- Calls to certain numbers can be remapped to call other numbers instead.
- Calls to certain numbers can be assigned to a particular line manager.

- Capability to apply rules based on incoming line manager, DNIS, or other properties.

How it Works

The dialing rules configuration provides a mapping of user-specified numbers to the numbers that are actually used for placing the outbound call. The processing transforms.

`{target-number, inbound-line-manager-id, extra-parameters (i.e. DNIS)}`

into

`{destination-address, outbound-line-manager-id, rule-parameters}`

based on the configured rules. The target number can be straight digits (`xxx` will be interpreted as `tel:xxx`), or may have a prefix identifying a type (i.e. `sip:1234`).

Creating Dialing Rules

To create a new Dialing Rules entry, enter the rule details, i.e. rule type, address type, address pattern, etc., and click on the plus sign on the right. The rule will be added in the `Rules` field. The context and priority are optional parameters. To add a context, enter the context attribute and value, then click on the plus sign on the right. The context will be added into the `Context` field. Use the up and down arrows to adjust the order of the rules. To delete a rule from the `Rules` field, select the rule and click on the minus sign. Once all Dialing Rule details have been entered, click on `Create` to create the Dialing Rules entry.

The following table explains the various fields for dialing rules creation:

Field	Meaning
rule-type	`a[ccept]` for accept rules, or `r[eject]` for rejection rules.
addr-type	`tel` to match telephone addresses, or `sip` to match SIP addresses
addr-pattern	Regular expression used for matching against target address. Unlike old rules, a leading ^ is assumed and does not need to be inserted.
xlat-type	`tel` to translate to a phone number, `sip` to translate to a SIP address, or `-` not to translate

Field	Meaning
xlat-pattern	Destination pattern that is parsed and turned into the returned address for the call. This should be `-` if `xlat-type` is `-`. Otherwise, it is a normal string that will substitute `\1`, `\2`, `\3`, etc with the 1st, 2nd, 3rd, etc regular expression matches, as one would expect.
out-lm-id	Is the ID of the line manager on which the call should be made. The possible values are 0, 4 and 8. 0: Use the incoming line ID. (if available) 4: LMSIP2 8: LMH323
rule-params	Specify a list of parameters, in `attribute1=value1,attribute2=value2,...` format. These will be returned for accept rules that are matched against.

Rules may have a context in which they are valid. Rules will only be applied if the context of the call matches the context of the rules. The context for a group of rules is specified through a context line, which identifies the context for all further rules until another context line is hit. The format of a context line is a set of context requirements, enclosed in square brackets. The format of the context line is thus:

```
[attribute1=value1 attribute2=value2 ...]
```

Context requirements will be matched against context information passed by CMAPI when requesting processing of a dial request. The attribute `lm` is a special context attribute which will be matched against the incoming line-id. Extra rules in the context of a dial request that are not explicitly mentioned by the rule context will not affect processing of rules.

For instance

```
[lm=1]
  rule1
[dnis=1234]
  rule2
[lm=1 dnis=1234]
  rule3
```

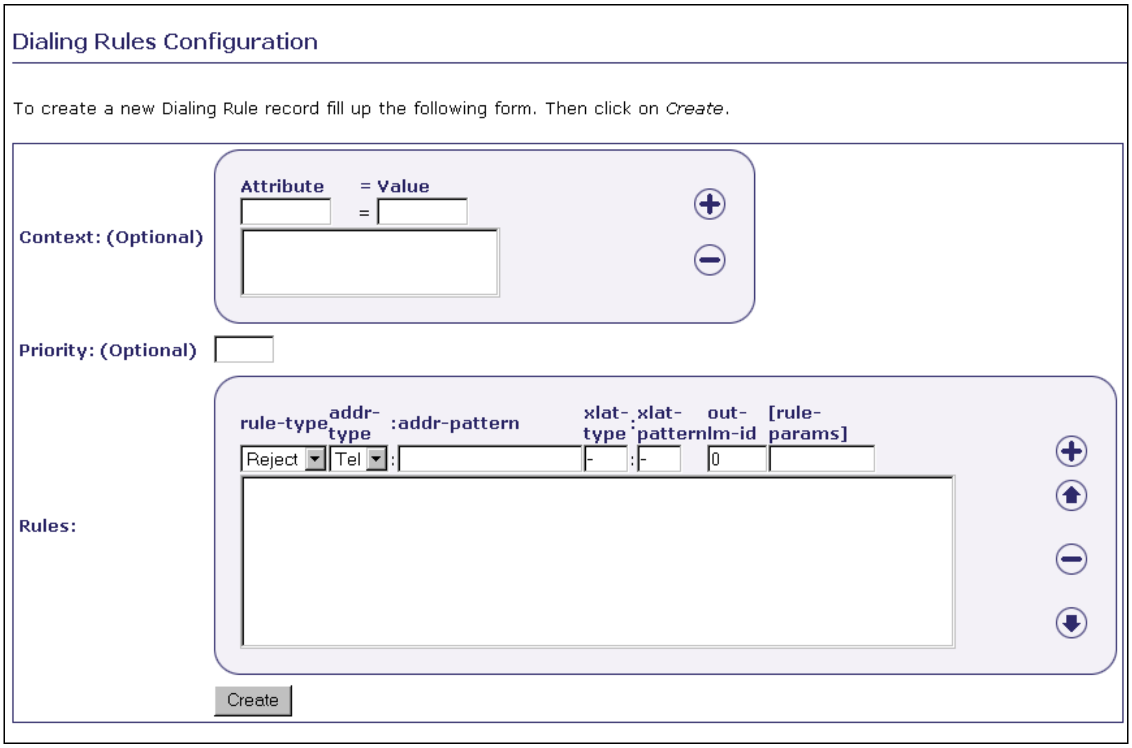
will apply rule 1 if `lm=1` but `dnis <> 1234, rule2 if dnis=1234 but lm <> 1`, and rules 1, 2, 3 if `lm=1` and `dnis=1234`. Any rules that appear before the first context line, or after the null (`[]`) context line will be matched against all calls.

Once the entry has been created, users can click on `Select Target` to target where that Dialing Rules entry is targeted.

To update the contents of an entry, make changes to any of the parameters of the entry and click on `Update`. This will send any changes to the targeted platforms at run time.

To delete an entry simply click on `Delete`.

Note: Reject rules must specify `-:-` for `xlat-type:xlat-pattern` and `0` for `out-lm-id`.



Dialing Rules Configuration

To create a new Dialing Rule record fill up the following form. Then click on *Create*.

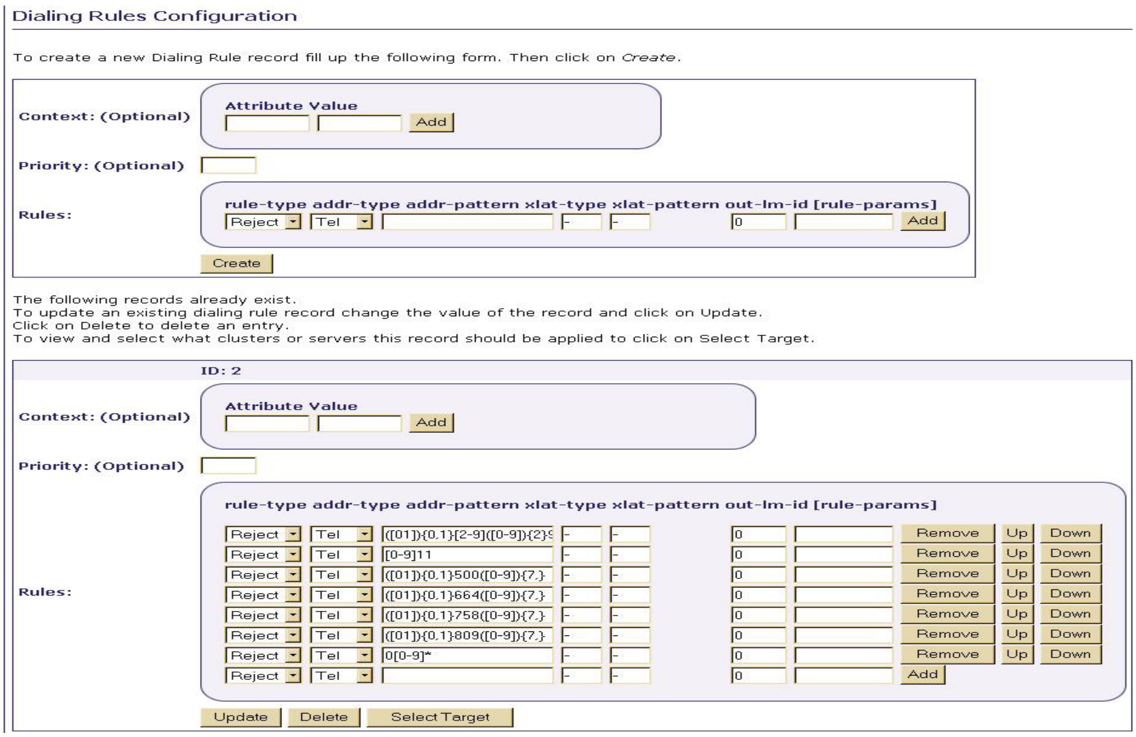
Context: (Optional)

Attribute	=	**Value**
	=	

Priority: (Optional)

Rules:

rule-type	addr-type	:addr-pattern	xlat-type	:xlat-pattern	out-lm-id	[rule-params]
Reject	Tel		-	-	0	



Dialing Rules Configuration

To create a new Dialing Rule record fill up the following form. Then click on *Create*.

Context: (Optional)

Attribute Value

Priority: (Optional)

Rules:

rule-type	addr-type	addr-pattern	xlat-type	xlat-pattern	out-lm-id	[rule-params]
Reject	Tel		-	-	0	

The following records already exist.
To update an existing dialing rule record change the value of the record and click on Update.
Click on Delete to delete an entry.
To view and select what clusters or servers this record should be applied to click on Select Target.

ID: 2

Context: (Optional)

Attribute Value

Priority: (Optional)

Rules:

rule-type	addr-type	addr-pattern	xlat-type	xlat-pattern	out-lm-id	[rule-params]			
Reject	Tel	([01]){0,1}[2-9]([0-9]){2}	-	-	0		Remove	Up	Down
Reject	Tel	[0-9]11	-	-	0		Remove	Up	Down
Reject	Tel	([01]){0,1}500([0-9]){7,}	-	-	0		Remove	Up	Down
Reject	Tel	([01]){0,1}664([0-9]){7,}	-	-	0		Remove	Up	Down
Reject	Tel	([01]){0,1}758([0-9]){7,}	-	-	0		Remove	Up	Down
Reject	Tel	([01]){0,1}809([0-9]){7,}	-	-	0		Remove	Up	Down
Reject	Tel	0[0-9]*	-	-	0		Remove	Up	Down
Reject	Tel		-	-	0		Add		

Rule File Processing

- Rules are processed in the order that they appear in the rules field, without regard to context.
- Rules are processed before `sip.defaultgw` or `h323.defaultgw` settings are applied.
- Only rules for which there is a context match will be processed. Other rules will be ignored.
- If a reject rule matches the destination number being processed, the call will be rejected.
- If an accept rule matches the destination number being processed, the call will be accepted and the translation parameters/outbound-LM/rule-parameters specified in the rule will be returned with the results.
- If no rules match a number, the call will be accepted with no translation, the same incoming & outgoing LM, and no rule parameters.
- If a translation exists, then the translated address will be used as the target address. If not, the original address will be used as the target address.
- If a call is accepted but after applying all of the above rules, the destination LM is equal to zero (possible if a call has no associated incoming line manager, such as the case of remote dial), then the line manager ID will be set to `4` for SIP calls or `8` for H323 calls.

Some operational notes

Phone numbers have the following restrictions at the moment:

- Main phone number: `0–9`
- Extension number: `0–9, a–d, #, *,` and `,`

For details regarding the regular expressions in use, see the standard Linux regex library which supports POSIX 1003.2 extended REs. man 7 regex provides the details under RH7.2.

Dialing Rules and VoIP

The use of dialing rules with VoIP provides two alternative ways of setting up certain behavior. In the case that it is desirable to allow the use of an IP/PSTN gateway to complete calls to PSTN numbers using VoIP, it is possible to enable the use of a default gateway:

```
[Call Manager Configuration]
sip.defaultgw=pstn-gw.voicegenie.com:5060
```

The above line will cause any outbound calls to PSTN numbers directed at the SIP line manager to be routed to the specified default gateway. Alternatively, it is also possible to use the dialing rules module to accomplish this for specific numbers:

```
[Dialing Rules Entry]
a tel:416([2-9][0-9]{6}) sip:1416\1@pstn-gw.voicegenie.com 0
```

The dialing rule shown above will accept any 10-digit phone number starting with `416`, and having a digit from `2–9` as the first digit following the `416`. It will translate the number so that the original number will be used, prefixed by a `1`, with the resulting number being used in a SIP address that directs the call to a PSTN gateway.

Either of the above approaches is acceptable; the choice of which method is used will generally depend on the level of control that is desired. The first approach, using a default PSTN gateway, is simple to configure and easily handles calls to any number. It is also the easiest way to route all PSTN calls to a SIP proxy for further routing decisions to be made. On the other hand, the second approach, using dialing rules, allows finer control over what numbers can and cannot be called, and also allows different numbers to be routed to different gateways:

```
[Dialing Rules Entry]
a tel:416([2-9][0-9]{6}) sip:416\1@gw-416.voicegenie.com 0
a tel:905([2-9][0-9]{6}) sip:905\1@gw-905.voicegenie.com 0
```

5.2.2 Destination Format (VoIP)

In PSTN environments, a telephone number is always used to address possible destinations. A full telephone number consists of a country code, an area code, and a local number, although the country code or area code is not always required, but certain dialing prefixes (such as `1` for national calls or `011` for international calls – these are prefixes for North America only) may be required. By assigning a unique country code to each country/group of countries, and with local authorities in each country assigning area codes, no two telephones have the same phone number, and a fully qualified phone number is capable of reaching any phone worldwide.

In VoIP environments, a single global addressing scheme does not exist. To some degree, an IP address can be used for addressing, but unlike phone numbers IP addresses often change and cannot be migrated. Also, a single IP address might host multiple virtual users. There are many possible solutions for addressing in a VoIP environment, and different protocols use different mechanisms to accomplish this. With SIP in particular, users are addressed using a SIP URL, similar to an HTTP URL:

```
sip:user[:password]@host[:port][;additional-parameters=additional-values]
```

For H.323, the H.323 URL format is as follows:

```
h323:user@host[:port][;additional-parameters=additional-values]
```

The leading `sip:/h323:` tag identifies that the URL is a SIP/H.323 URL, allowing it to be differentiated from other URLs such as HTTP URLs. The `host` (and to some degree, `port`) and `user` fields are the key fields in a SIP/H.323 address; some additional parameters may be present but these are usually for information and do not affect the usage of a SIP address (not for the H.323 case).

The `host` and `port` fields (`5060/1720` is used as the default SIP/H.323 port if it is not specified) provide the IP address (and port number) of a SIP user agent/H.323 endpoint, which is simply a network element that is capable of sending and receiving SIP/H.323 messages – much like an HTTP client or server. SIP user agents may be either clients or servers, and are usually both – allowing them both to initiate requests (such as making an outbound call) as well as to respond to received requests. The same can be applied for H.323 endpoints – that they are capable of making/receiving H.323 calls. In a PSTN environment, this essentially would be the number of your telephone – it is an address unique to each device on a given network that can be used to establish communication with that device.

The `host` field can either specify an IP address, such as `192.168.1.2,` or a hostname, such as `sip.voicegenie.com.` Hostnames will be translated, using DNS, into an IP address to actually communicate with the target system.

Particularly, in SIP, an address can contain additional information beyond what is necessary to reach the network element associated with a transport address. This is the `user` field, and specifies which user on a particular network element should be contacted. Some devices may have only a single user, and may actually ignore the user component of the SIP URL; other devices may support many users (or virtual users), and may use the user component of the SIP URL to select which user participates in a given session.

The above is only a brief introduction to SIP/H.323 URLs, but the underlying principle is that like HTML pages, everything that can participate in a SIP session can be reached by specifying a URL that is unique to that user/entity.

Unlike (sometimes) DNIS and ANI, in SIP, the same format is used for both the calling party address, and the called party address. SIP encodes these in headers called `From` and `To` that are sent in all messages – similar to E-mail. For instance, a SIP request containing:

```
From: sip:jsmith@205.150.90.118:4060
To: sip:voip@sip.voicegenie.com:5060
```

would be interpreted as being from user `jsmith,` whose SIP user agent can be contacted at `205.150.90.118:4060,` and to a user named `VoIP,` whose SIP user agent can be contacted at `sip.voicegenie.com,` on the default SIP port number.

Note: Because of the usage of SIP proxies and the translation of addresses, the actual address contacted to reach `voip@sip.voicegenie.com:5060` could be something other than `sip.voicegenie.com`. 5.3.2 SIP Proxies/Registrars explains this briefly; a full discussion of SIP call routing is beyond the scope of this document.

Finally, it should be noted that only the SIP URL portion of the `From`/`To` headers are important; it is legitimate to format these headers as:

```
From: John Smith <sip:jsmith@205.150.90.118:4060>
To: VoiceXML Gateway <sip:voip@sip.voicegenie.com:5060>
```

5.3 Call Routing in VoIP

5.3.1 IP/PSTN gateway

Voice-over-IP is an extremely flexible technology for building communications networks and network-based media services. However, the vast majority of endpoints that it is desirable to make a call to are still PSTN-based – widespread adoption of VoIP and use of VoIP protocols to make and receive all calls (landline and wireless) is still a long way off. As such, the majority of VoIP networks are interconnected with the PSTN via an IP/PSTN gateway of some sort which bridges between the two networks.

Most IP/PSTN gateways follow the convention of using the phone number in the `user` field of the SIP URL. For instance, to call 4167360905 through an IP/PSTN gateway at `pstn-gw.voicegenie.com`, the destination address should be set to `sip:4167360905@ pstn-gw.voicegenie.com:5060`.

Note: Note that how an IP/PSTN gateway handles the call and actually places the call is also determined by routing rules that specific to the IP/PSTN gateway. Thus, the above destination address would cause the VoiceGenie gateway to deliver a call to the IP/PSTN gateway; however, the IP/PSTN gateway must be appropriately configured to route the call to the PSTN once it is received.

To facilitate existing applications and to move some elements of the routing decision into the platform, VoiceGenie allows the application developer to conveniently use PSTN style URI even at the application layer. Depending on the application context, this can be accomplished using these parameters: `sip.defaultgw`, `sip.outcalluseoriggw = 1`, `h323.defaultgw`, and `h323.usesamegwfortransfer`. When these parameters are used properly, the Media Platform can perform automatic conversion of the PSTN URI to the corresponding VoIP URI and ensure the request is forwarded to the appropriate IP/PSTN gateway.

In general, when making PSTN calls, the `sip.defaultgw/h323.defaultgw` should be set. One exception in H323 is when calls are to be transferred with multiple gateways in the environment, and it is desired to place the outgoing call to the same gateway where the incoming call is from (because of transfer limitations in PSTN gateways). Under this situation, the configuration value `h323.usesamegwfortransfer` should be set to `1`.

Please see the VoiceGenie 7.2 Media Platform System Referenec Guide “Call Manager Configuration” section for more details.

Note that in addition to the forementioned parameters, 5.2.1 Dialing Rules can also play a significant role in destination address manipulation and controlling how a call is routed.

However, extension dialing is supported on PSTN technologies but is *not* supported with VoIP directly by the VoiceGenie gateway. The IP/PSTN gateway may still support the use of extensions that are encoded into the `user` field of the SIP URL; support for this will depend on the IP/PSTN gateway used. Similarly, analysis is not supported with VoIP.

5.3.2 SIP Proxies/Registrars

When making an outbound SIP call, the call destination is specified as a SIP URL, as described in the preceding section. However, how this call will be routed depends on the configuration of the VoiceGenie gateway.

By default, the system will send a SIP `INVITE` message (used to initiate a call) to the IP address and port that are specified via the `host` and `port` parameters of the destination address. Using the example above, if an outbound call or transfer was made to `voip@sip.voicegenie.com:5060`, then the `INVITE` message will be sent to `sip.voicegenie.com`, on port 5060. UDP is used as the default transport for SIP messages.

Note: the actual `INVITE` message will always present the `To:` header exactly as is specified in the destination address field for the outbound call or transfer.

In addition to routing calls based as above, it is also possible to configure an outbound proxy to which calls will be routed. If configured, instead of sending the SIP `INVITE` message directly to the destination, the `INVITE` message will instead be sent to the SIP proxy for further routing and processing. Please see the `Call Manager Configuration` section, `SIP Call Routing` subsection for more details.

Moreover, the Media Platform also supports the use of SIP `REGISTER` message to perform registration with SIP Registrar servers. This allows users to perform look-up with the SIP Registrar server to locate and SIP `INVITE` requests to the Media Platform.

5.3.3 H.323 Gatekeeper

H.323 gatekeeper can be used to translate H.323 aliases. For example, assuming that both the user John Smith has registered his ip phone with an extention 12345 and that the VoiceGenie software is registered to the same H.323 gatekeeper (more details to follow), a call can be placed to John Smith using the H.323 url `phone:12345`. The VoiceGenie software will first consult the gatekeeper by sending an admission request (ARQ), and the gatekeeper will respond an admission confirm containing the ip address of John Smith's soft phone so that the VoiceGenie software knows where to route the call to.

The gatekeeper can be configured using `h.323.gatekeeper.*` and `h323.ras.inarqmode/outarqmode`. The gatekeeper registration information can be configured using `h323.ras.registrationinfo`. More information is available in the `H323 Gatekeeper` configuration section.

Chapter

6 Call Transfer

6.1 General Information

The Media Platform offers many call release and call bridging technologies. The usage of these technologies depends on the protocol and the integration environment. This section provides information on the technologies that the Media Platform is capable of.

From VoiceGenie implementation perspective, the various transfer methods can generally be divided into three main categories:

- **One Leg Transfer** – Transfer that requires only one call leg from VoiceGenie Media Platform's perspective. In other words, the transfer occupies only one channel on the Media Platform. The transfer is performed by sending various signals on the inbound call leg, and the switch supporting the transfer will handle the signal accordingly and perform the transfer; resulting in the original call leg being released from the platform. This is referred as *One-leg-style transfer* in this document.
- **Two Leg Transfer** – Transfer that requires two call legs (ie, occupying two channels), and the two call legs are bridged and released at the network layer. The Media Platform is responsible for making the outbound call request, and then transmits the required signals via the two call legs to the call routing entity (normally a switch). The routing entity, upon receiving the signals, will join the two calls together and release them from the Media Platform. This is referred as *Join-style transfer* in this document (note that this is different from `<join>` where media is joined).
- **Bridged Transfer** – Transfer that requires two call legs, and the Media Platform stays in the signaling path and is responsible for bridging the two call legs. The Media Platform is responsible for making the outbound call request, and then bridging the media path between the caller and the callee. The Media Platform is responsible for maintaining this connection until transfer completes. This is referred as *Bridge-style transfer* in this

document. Note that this style of transfer relies completely on the Media Platform's capabilities (call routing entity, or the switch needs to have the capability)

The following table summarizes the transfer methods under each of the three styles of transfer:

Transfer style	Transfer methods
One-leg	hkf, h450, refer, inband
Join	h450join, referjoin
Bridge	bridge, mediaredirect

It is important to understand these three transfer styles to fully understand the difference between the metrics-billing behaviors among the transfer methods. This also helps understand the relationship between transfer method and transfer type. In addition, for Join-style transfers, when `type=blind,` the transfer request signal(s) will be sent before the outbound call leg is connected. When `type=consultation,` the transfer request signal(s) is only sent after the outbound call leg is connected. Understanding this difference is crucial to understand why some transfer methods (like RLT) cannot be supported with `type=blind.`

One interesting usage of Join-style transfers is `<call>`/`<join type=network>`. Typical `<call>`/`<join>` usages are essentially performing media bridging between the caller and the callee when the `<join>` tag is executed. For `<join>` tag with `type=network,` the outgoing call request is made as usual during the `<call>` tag execution. Just like a normal `<call>` tag usage, the Media Platform can interact with the inbound and outbound call legs in various ways. When a `<join>` tag with `type=network` is executed, transfer request(s) will be made to the call routing entity (or the switch) to perform the actual transfer (or joining at the network layer) and release the call. Hence, different from a typical `<join>` tag, the calls will be released from the Media Platform after execution.

Furthermore, since release 6.4, the Media Platform supports fallback for Join-style transfers. As mentioned, for Join-style transfers, after the outbound call request is made, signals will be sent to either or both inbound and outbound call legs to perform the transfer request. For any reason, if the call routing entity fails the transfer request at this stage, the Media Platform is capable of falling back to a Bridge-style transfer. With this capability, users can assure that transfers can always be made successfully even with a malfunctioning call routing entity or under fail-over scenarios. This behavior can be configured using the `sessmgr.ECS_Fallback` and/or `sessmgr.Join_Fallback` parameter.

6.2 Transfer Framework

Since release 6.4, the Media Platform has updated its internal transfer framework. The major benefits to the users include:

- Support for multiple transfer methods simultaneously
- Allow the VXML application to dynamically select the transfer method when performing a transfer request
- Support VXML 2.1 syntax for `<transfer>` tag
- Separation of transfer behavior at application level and transfer mechanism at telephony level
- Allow fallback transfer method (see "Advanced User" section)

6.2.1 Type and Method

To understand the new transfer framework, it is important to understand the difference between transfer type and transfer method. This can be best understood by first looking into the VoiceXML 2.1 syntax for the transfer tag (VoiceXML 2.0 style of transfer tag is still supported).

Transfer syntax for VoiceXML 2.1

```
<transfer
 name="string"
 expr="ECMAScript_Expression"
 cond="ECMAScript_Expression"
 dest="URI"
 destexpr="ECMAScript_Expression"
 method="string"
 type="blind" | "consultation" | "bridge"
 connecttimeout="time_interval"
 maxtime="time_interval"
 transferaudio="URI"
 analysis="boolean"
 connectwhen="analysis" | "answered" | "immediate"
 aai="string"
 aaiexpr="ECMAScript_Expression"
 detectansweringmachine="boolean"
 signalvar="ECMAScript_Object"
 consultexpr="ECMAScript_Expression">
 child elements
</transfer>
```

One new attribute, `method,` is introduced. Values to the attribute, `type,` also changed.

- `method` signifies the transfer method name (case-insensitive). It defines the actual mechanism to be used to perform the transfer at the telephony layer. In other words, it is one of the methods defined in the previous two sections. In Media Platform 6.4 and above, instead of relying on the old attribute `bridge, type` and the call manager configuration to determine the actual transfer method being performed, the `method` attribute explicitly determines the transfer method. If method is not specified or is an empty string, default method will be chosen depending on the call manager configuration.

- `type` now signifies the transfer behavior viewed by the VoiceXML application.
  - If `blind` is specified, application will get detached from the incoming call (as well as outbound call if it is involved) as soon as transfer is successfully initiated. It will be unable to detect the result of the transfer request once the request can be made to the telephony network. It is because by the time the transfer process fails, application is already detached from the call.
  - If `consultation` is specified, application will get detached from the incoming call once the transfer process finishes successfully. Hence, it is possible to report transfer failure using this type of transfer. If the transfer process fails, application will retain relationship with the call. If the transfer process succeeds, then the VoiceXML application will detach from the call.
  - If `bridge` is specified, application will never get detached from the incoming call unless the incoming call actually disconnects. The control of the call will always return to the application when the transfer ends (regardless of the result).

Different combinations of `type` and `method` create interesting scenarios that were not possible with VoiceXML 2.0. One example is that the transfer method behavior can be such that the transferred call is taking place on the platform, while the `type` mandates that the application be detached from the call. Take for instance a transfer with `method=bridge type=blind`. In this case, from the platform point of view, both outbound and inbound call will exist on the platform while the VoiceXML application already received transfer complete and disconnect event from both inbound and outbound call legs.

On the other hand, it is important to note that some transfer methods may not be supported for all three transfer types, due to the transfer methods' mechanisms. For example, `method=hkf` (Hook Flash) cannot be supported with `type=bridge,` since the call will be disconnected from the Media Platform if the transfer is successful. It is impossible to have the application proceed with the transfer and wait for transfer to end successfully without detaching the call.

The following table summarizes the `method` and `type` attribute values that are allowed for each telephony technology used:

Telephony technology	Description	Method	Blind type	Consult type	Bridge type
All	Bridge transfer	bridge	YES	YES	YES

Telephony technology	Description	Method	Blind type	Consult type	Bridge type
	Inband DTMF transfer	inband	YES	NO	NO
SIP	Hook flash transfer	hkf	YES	NO	NO
	Refer	refer	YES	YES	NO
	Refer with replace header	referjoin	YES	YES	NO
	Media redirect transfer	mediaredirect	YES	YES	YES
H.323	Hook flash transfer	hkf	YES	NO	NO
	H.450.2 transfer with 1 leg	h450	YES	YES	NO
	H.450.2 transfer with 2 legs	h450join	YES	YES	NO
	Media redirect transfer	mediaredirect	YES	YES	NO

For the list of available signalvar, please refer to `http://developer.voicegenie.com/reference.php?ref=variablesdetails#signalvar`.

6.2.2 Backward Compatibility with VoiceXML 2.0

The Media Platform is backward compatible with the use of VoiceXML 2.0 syntax. VoiceXML 2.0 controls the transfer behavior by the combination of `bridge` and `type` attribute, and the Media Platform supports the older syntax by doing the following mapping to the new `method` and `type:`

Old Type Attribute	Local		Network		Supervised		Unsupervised	
Bridge	True	False	True	False	True	False	True	False
New Type Attribute	Bridge	Blind	Bridge	Blind	Bridge	Consultation	Bridge	Blind
Method	empty	empty	empty	empty	empty	empty	empty	empty

Also, when `<join>` tag is used with `type=network,` platform configuration will be used to determine the actual telephony transfer method to use (cannot specify this from the VoiceXML page). This is performed automatically by selecting a transfer method defined on the supported list/bitmap that can be used. In particular, a Join-style transfer must be configured (see next section for more information). If no Join-style transfer method is defined, the transfer request will fail.

6.3 VoIP Transfer

A summary of the Media Platform VoIP supported call transfer methods is shown in the table below:

Call Transfer Method	Protocols	Notes
SIP Refer	SIP	
SIP Refer with Replace header	SIP	
H.450.2 with one call leg	H.323	
H.450.2 with two call legs	H.323	
Hook Flash	H.323 and SIP	Can transmit inband dtmf, or out-of-band RFC 2833 events (or H.245 events for H.323)
Inband	H.323 and SIP	Can transmit inband dtmf, or out-of-band RFC 2833 events (or H.245 events for H.323)
Media Redirect	H.323 and SIP	Media are connected directly between caller and callee (network) while call control events are bridged thru the Media Platform
Bridge	H.323 and SIP	Media and call control events are bridged at the Media Platform

By default, bridge transfer is always supported by the Media Platform. Refer and Referjoin methods are enabled for SIP by default. With the exception of inband transfer, each of the other transfer methods can be configured via SMC under the `Call Manager` configuration using `sip.transfermethods` and `h323.transfermethods.`

Some transfer methods may require further parameters to suit the switch's and the environment's behavior. Please see the corresponding `Call Manager` configuration sections in the VoiceGenie 7.2 Media Platform System Reference Guide for further details.

6.4 Whisper Transfer

Traditionally, when performing transfer with our platform, it is always assumed that the callee always accepts the transfer. This limitation has been relaxed since release 6.2 with the whisper transfer feature (also known as consultative transfer). After the transfer operation is requested and performed, there is an option to delay the connection of the caller and the callee. This allows the platform to continue performing media operations with the callee, and transfer out the call at a later phase. From the application point of view, a VXML application can be written to consult with the callee to determine whether the callee would like to answer the transferred call from the caller. The transfer proceeds as usual if the callee agrees. The callee can also reject the transfer request, in which the callee will be disconnected and the VXML application will return the control to the original caller. This is achieved by using the consultexpr attribute on the `<transfer>` tag.

Currently, this feature is supported with the following transfer methods:

- **SIP** – referjoin, hookflash, bridge, mediaredirect
- **H323** – h450join, bridge

6.5 CTI Call Release

The VoiceGenie Media Platform is often deployed in conjunction with various CTI products. In those deployments, CTI can be used to perform the call transferring capability and release the call from the Media Platform. In addition, the VoiceGenie Call Control Platform provides native integration with the Media Platform to support some of the popular CTI products in the market. Please see the *Call Control Platform* document for more information.

Chapter

7 Video Support

7.1 Overview

The VoiceGenie Media Platform includes rich support for video applications. This allows for the development of applications such as:

- Video voice-mail
- Video conferencing and conferencing management
- Entertainment applications

Video support is not defined as part of VoiceXML 2.0 or VoiceXML 2.1. It is being considered for inclusion in VoiceXML 3.0, currently being developed by the Voice Browser Working Group (VBWG) in the W3C. Video support has been added to the VoiceGenie platform as an extension to VoiceXML, by enhancing the `<audio/>` and `<record/>` tags to support video. The simplest applications will take advantage of this video 'play' and 'record' functionality to build video enabled applications like video voicemail.

Video is supported by the VoiceGenie Media Platform, as well as the VoiceGenie CCXML Platform and SIP Proxy.

7.2 Video Deployment Architectures

The VoiceGenie infrastructure uses SIP call control and RTP media to manage video and audio streams. SIP endpoints can make use of the audio and video capabilities of the VoiceGenie solution directly. Commercial gateways such as those from Dilithium and Radvision can be used to interconnect VoiceGenie infrastructure to the ISDN or SS7 network.

In order to deploy VoiceGenie video, the following components are required:

- VoiceGenie 7.2 Media Platform;
- Video capable SIP softphone;

- Web Application Server;
- Compatible video files;
- VoiceXML pages describing the video application;

Optional components include:

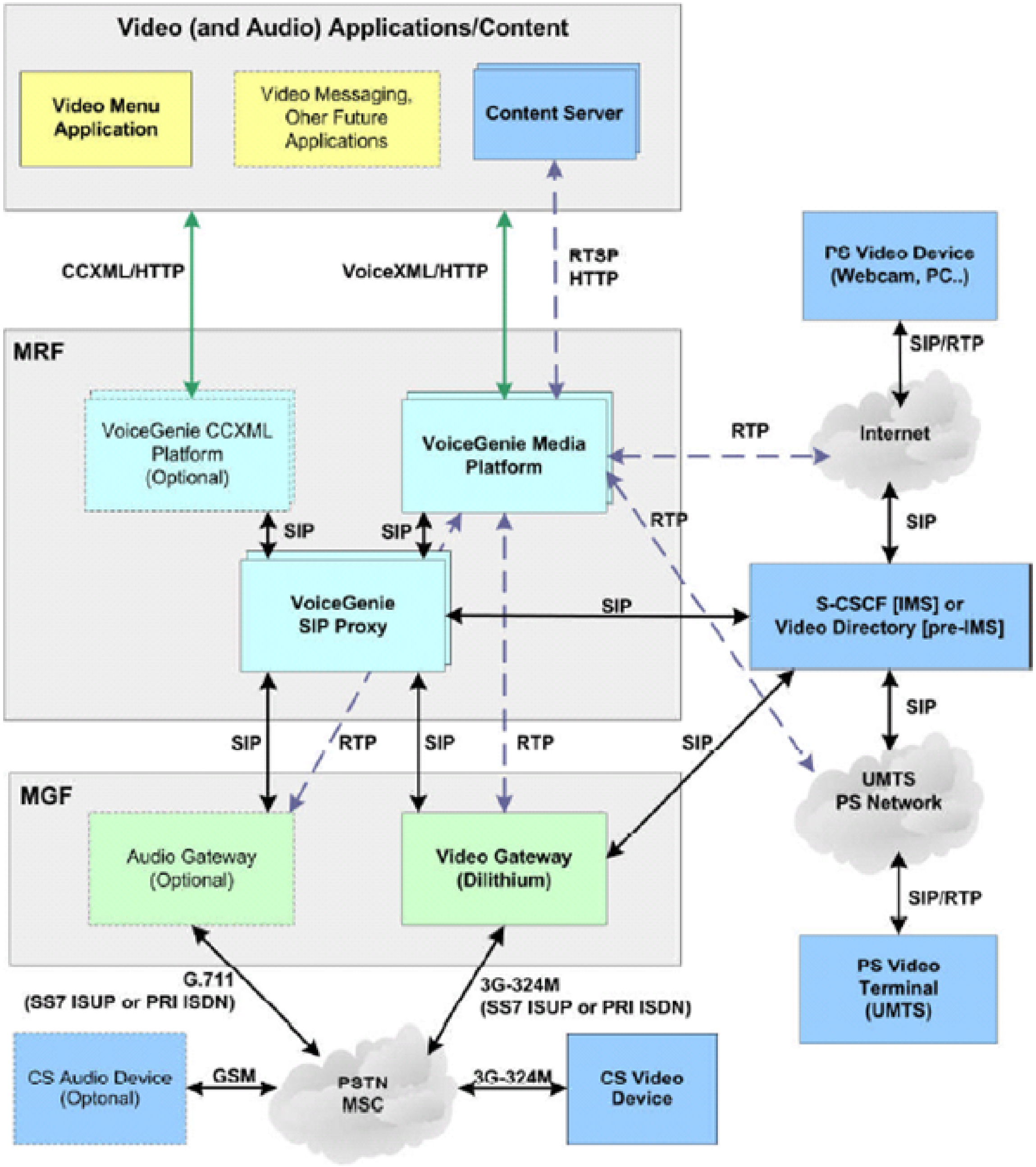
- Video gateway, terminating 3G-324M TDM video, and transcoding to SIP and RTP

When deployed with a video gateway, the conversion between RTP and 3G-324M (including any transcoding) happens at the Dilithium or Radvision gateway.

If AMR-NB audio is being used on the 3G-324M (TDM) side of the gateway, then the video gateway converts the audio from either G.711u or G.711a to AMR-NB. The .AVI container file used by VoiceGenie would therefore contain either G.711u or G.711a. The video gateway does not have to perform transcoding is if AMR-NB is used on the Media Platform.

Similarly, the video gateway converts the H.263 video (being delivered by the VoiceGenie platform) to a format that fits within the 64kbit/s bandwidth available with 3G-324M. Containers and supported protocols will vary depending upon the video gateway. Dilithium supports H.263 + G.711, or H.263 + AMR-NB.

A typical architecture supporting audio-only and video applications is shown in the diagram below.


Video (and Audio) Applications/Content
Video Menu Application
Video Messaging, Oher Future Applications
Content Server
CCXML/HTTP
VoiceXML/HTTP
RTSP
HTTP
PC Video Device (Webcam, PC..)
SIP/RTP
MRF
VoiceGenie CCXML Platform (Optional)
VoiceGenie Media Platform
RTP
Internet
RTP
SIP
SIP
SIP
VoiceGenie SIP Proxy
SIP
S-CSCF [IMS] or Video Directory [pre-IMS]
SIP
SIP
RTP
SIP
RTP
SIP
UMTS PS Network
MGF
Audio Gateway (Optional)
Video Gateway (Dilithium)
SIP/RTP
PS Video Terminal (UMTS)
G.711 (SS7 ISUP or PRI ISDN)
3G-324M (SS7 ISUP or PRI ISDN)
CS Audio Device (Optonal)
GSM
PSTN MSC
3G-324M
CS Video Device

7.3 Supported Protocols and Specifications

The VoiceGenie solution currently supports the following video formats:

- AVI container files with H.263 encoded video and G.711 encoded audio (8kHz)

- 3GPP container files with H.263 encoded video and AMR encoded audio (8kHz).

The following mime types are supported:

<table>
<tr><th>File extension</th><th>Format</th><th>MIME Type</th></tr>
<tr><td>.avi</td><td>AVI</td><td>video/avi;codec=<audio codec>;videocodec=<video codec></td></tr>
<tr><td>.3gp</td><td>3GP</td><td>video/3gpp;codec=<audio codec>;videocodec=<video codec></td></tr>
<tr><td>.263</td><td>RAW</td><td>video/H263 or video/H263-1998 or video/x-h263</td></tr>
<tr><td colspan="3">• audio_codec for AVI can be: ulaw (g.711 mulaw), alaw (g.711 alaw), pcm16 (signed linear PCM 16-bit), or pcm (unsigned linear PCM 8-bit)
• audio_codec for 3GP can be: amr (AMR-NB)
• video_codec for AVI and 3GP can be: h263 (h.263) or h263-1998 (h.263+)</td></tr>
</table>

H.263 media transport over RTP conforms to Mode A transmission as defined in RFC2190.

H.263+ media transport over RTP conforms to RFC2429.

Please contact Genesys sales for additional file containers and encodings and other enhancements planned for future releases.

7.4 VoiceXML Feature Support

Video on the VoiceGenie platform fully supports the following VoiceXML features:

- Video file playback (including embedded audio);
- Video file record (including embedded audio);
- DTMF recognition;
- Speech recognition;
- Speech and DTMF Barge-in;
- Prompt queuing;
- Caching.

Prompt queuing is discussed at:

http://developer.voicegenie.com/tutorials_VoiceGenie.php?tutorial=prompt_queueing

Cache management is described at:

http://developer.voicegenie.com/tutorials_VoiceGenie.php?tutorial=caching_howto2

Here is a sample VoiceXML script providing a menu for video playback:

```
<?xml version="1.0"?>
```

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<meta name="application" content="Video Playback Example"/>
 <form id="Welcome">
 <block name="Hello">
 <audio src="builtin:prompts/sting.vox"/>
Which trailer would you like to watch, Harry Potter or Jurassic
Park?
 </block>
 <field name="movie">
 <option> Harry Potter </option>
 <option> Jurassic Park </option>
 <filled>
      <if cond="movie=='Harry Potter'">
         Here's the trailer for Harry Potter
         <audio src="harrypotter.avi"/>
      <elseif cond="movie='Jurassic Park'"/>
         Here's the trailer for Jurassic Park
         <audio src="jurassic.avi"/>
      </if>
 </filled>
 </field>
 </form>
</vxml>
```

Here is a sample VoiceXML script for recording video:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<meta name="application" content="Video Recording Example"/>
<property name="caching" value="safe"/>
<property name="bargein" value="false"/>
<property name="confidencelevel" value="0.45"/>
<property name="loglevel" value="4"/>
<form>
 <record name="video_message" beep="true" maxtime="30s"
finalsilence="5s" dtmfterm="true"
dest="RecordedFile/" type="video/avi;codec=pcm16;videocodec=h263">
 <prompt> please re cord your message </prompt>
 <filled>
 Here is your video message <value expr="video_message"/>
 </filled>
 </record>
</form>
</vxml>
```

7.5 Advanced VoiceGenie Feature Support

A number of advanced VoiceGenie features work well with video.

7.5.1 VCR Controls

The VoiceGenie platform supports 'VCR Controls', allowing the caller to navigate within an audio or video stream using DTMF keys. The available functions include pause and resume, skip forwards or backwards, and several other features. Please see the tutorial at:

http://developer.voicegenie.com/tutorials_VoiceGenie.php?tutorial=audio_control

for further information.

7.5.2 Advanced Barge-in Features

The VoiceGenie platform supports reporting of barge-in offsets based on time and 'marks' set in the prompt stream. This allows intelligent prompt playback, as well as confirmation of what prompt components have been heard by the caller. There are several VoiceGenie extensions related to this, documented in:

http://developer.voicegenie.com/tutorials_VoiceGenie.php?tutorial=VoiceGenie_audio_offset

There are also related features in VoiceXML 2.1:

http://developer.voicegenie.com/tutorials_VoiceGenie.php?tutorial=vxml21_features#mark

7.5.3 Conferencing

Video conferences can be managed by the CCXML and Media platforms. Features include:

- Full, or half-duplex conference connections, including Listen only, Send-only, or full bidirectional video and audio;
- Video switching;
- Video pre-select;
- Video based on active (loudest) speaker;

The following `<join/>` attributes are added for specifying video conferencing behavior:

- `videoalgorithm = "loudest" | "fixed" | "none" (optional)`

  If this join request allows a participant to connect to a conference, this attribute specifies the video algorithm used for this conference.

  - `loudest` – video from the active (loudest) participant will be selected.
  - `fixed` – pre-select video channel as specified by videosource attribute.
  - `none` – disable video for the conference.

defaults to the value of `<property/>` `com.voicegenie.conference.videoalgorithm.`

- `videosource = "ECMAScript_Expression" (optional)`

  An ECMAScript expression to be evaluated and used as the channel ID of the video used for the conference. Applied only if this join request allows a participant to connect to a conference and `videoalgorithm="fixed".` Default to `0,` which means the creator (first participant) of the conference will be selected.

Information on conferencing with the VoiceGenie CCXML platform is available at:

http://developer.voicegenie.com/tutorials_VoiceGenie.php?tutorial=ccxml_intro

Conferencing on the VoiceGenie Media Platform is documented at:

http://developer.voicegenie.com/tutorials_VoiceGenie.php?tutorial=conferencing

7.5.4 Full Call Recording

Full call recording includes support for recording the video interaction on a call. Control over full call recording is documented at:

http://support.voicegenie.com/tutorials.php?tutorial=wholecallrecording

7.5.5 Media Redirect Transfer

Transfer and Media Redirect Transfer works with video, allowing the VoiceGenie platform to remain in the call control path, while redirecting media to the appropriate endpoints.

7.5.6 SIP/NETANN Access

Access to NETANN services including video behave as expected.

7.6 Known Issues and Limitations

- The H.263 codec is not enabled by default. This can be enabled either during installation, or by manually modifying the platform configuration by:
  - Go to the `Configuration` tab, `Call Manager` section via SMC
  - Edit the configuration profile
  - Add `h263` and/or h263-1998 to `mpc.codec`

  - Make sure the platform is targeted and updated for the configuration change

- Maximum RTP packet size is limited to 20000 bytes by default (configurable using Call Manager configuration parameter `mpc.rtp.maxrtppacketsize`). Since VoiceGenie only supports mode-A transmission for H.263 codec, this imposes a limitation on the maximum size per GOB (Group-Of-Block). Transmission of H263+ codec follows RFC2429 and does not have such limitation.

Note: Each video picture frame can be segmented into multiple GOB. Some advanced video encoder allows GOB size to be adjusted during encoding process.

Chapter

8 Other Features

8.1 Remote Dial

8.1.1 Overview

The VoiceGenie Media Platform provides a complete VoiceXML 2.1, 2.0 and 1.0 implementation, along with many other features that make the platform attractive to those deploying large scale speech applications. One of the most useful features is the ability to initiate outbound calls in an asynchronous manner. This section provides details of how to use the outbound calling features of the VoiceGenie Media Platform. Outbound calling is subject to the restricted calling database maintained by the VoiceGenie Media Platform.

8.1.2 System Requirements

To use the outbound calling functionality, it is required that you have access to a VoiceGenie Media Platform. The Media Platform must be configured with either bi-directional, or outbound channels. You may require additional resources to host the API software, although it can be hosted on the platform itself.

8.1.3 Socket API

The current implementation of the remote dialer includes the Command-line/socket interface. This provides full access to the outbound call functionality. This includes: Outbound call placement, associated with a VoiceXML URL; DNIS delivery; Specification of User to User Information (UUIDATA); detailed status reporting.

Other interfaces are under consideration, including direct Java class access, and an interface supporting XML.

8.1.4 Telnet/Socket Interface

By connecting to port preconfigured for remote dial (default to 6999) on a platform with enabled outbound dialling, the user can make use of a simple interactive interface to place outbound calls. Using telnet for example, the user can request that an outbound call be placed, and provide the URL of a VoiceXML page to be associated with the call. A sample is shown below:

```
pw@galahad 379>
pw@galahad 379> telnet localhost 6999
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
PW RemoteDial>
call 4167360905 4167362012 http://www.voicegenie.com/helloworld.vxml
   0001 Test
!CALL_SENT 1: telno:4167360905 dnis:4167362012
   url:http://www.voicegenie.com/helloworld.vxml uuidata:Test
PW RemoteDial>
!CALL_STATUS 1: CONNECTED: Line is connected.
PW RemoteDial>
!CALL_DROP 1 41: USER_END: User hung up call. (time spent was 41
   secs) (protocol reason: [DlgcChannel] User hangup )
PW RemoteDial>
```

This shows the placement of a call using the command-line interface. This interface is also suitable for use programmatically via a socket connection. Details of the command-line interface follow below.

Telnet Interface Commands

The command line interface provides a number of useful commands. These include:

```
call <telno> <ani> <url> <refno> [uuidata] [defaults]
[parameter_list]
```

The call command initiates an outbound call to the specified telephone number (`<telno>`). `<telno>` can accept up to 1023 characters. When connected, the VoiceXML page referred to by the specified URL (`<url>`) is ‘attached’ to the call, just as if the user had called in themselves. `<platform ANI>` can accept up to 32 characters. The actual number of ANI digits that can be delivered on PSTN depends on the network, e.g. the maximum number on ISDN T1 is 15. The reference number (`<refno>`) is a user-supplied identifier that can be used to associate status replies with this call initiation and should be unique for each active call. The reference number must be an integer between `0` and `2147483647`. There are three other optional parameters that can be specified:

- `[uuidata]` – user-to-user info element, not longer than 254
- `[defaults]` –default voicexml page used by the url
- `[parameter_list]` – the name value pair separated by `|` that can be passed from the interface to the call manager. Please refer to http://developer.voicegenie.com/reference.php?ref=variablesdetails#signalvar for the list of signalvars that can be specified using the parameter list.
  The parameter list can also be used to specifiy the VoiceXML Interpreter to be used to execute the vxml page. One can specify `gvp.appmodule=VXML` to select the Legacy Interpreter, or `gvp.appmodule=VXML-NG` to select the Next Generation Interpreter. If `gvp.appmodule` is not specified, the default VoiceXML Interpreter specified by the Call Manager configuration parameter `sessmgr.default_vxml_interpreter` will be used.

If you wish to specify either of the parameter, you can either specify all the parameters before this parameter or specify `-` for the default value. For e.g., if you wish to specify the parameter list but not the uuidata and the defaults file, you can use `-` to provide default values for uuidata and the defaults in the call command as seen in the example below:

```
PW RemoteDial> call 4167366493 2323
   http://205.150.90.12/developer/main/cgi-bin/index.cgi 1223 - -
   NWNAME=dtiB1T21|NUMBERINGPLAN=0
```

Note: the size of `<uuidata>` cannot exceed 254 characters, and is typically less than 128 characters.

Other valid commands within the interface are:

- `cc` – This command clears all message counters.
- `dump` – This command toggles the display of raw debug information.
- `e / q / x` – Any of these will exit the command-line interface.
- `setto <unit_type> <num_units>` – This command sets the timeout associated with call setup. It can be set to a number of seconds (using `unit_type s`). The `<num_units>` parameter specifies the number of seconds to try to connect the call. The default value is `120` seconds.

- `timelimit <seconds>` – This command sets the maximum number of seconds for the call. The default is `604800` seconds.
- `end <refno>` – This can be used to request to end a call that was initiated from the remdial interface.
- `analysis y|n|a` – This command controls the enabling of Dialogic call analysis with `y` or `a,` which could be used to detect busy, no-answer, fax or answering machine. The default is `y` (yes). The `a` option is for disconnecting the call on answering machine detection.
- `scall` – This command shows the calls in this session.
- `scount` – This command displays counters from this session.
- `show` – This command displays the current settings. Sample output is:

  ```
  !SETTINGS: max_calls:500 timeout_length:120 timeout_unit:s
  call_analysis:enabled time_limit:604800
  ```
- `?` or `h` – This command will get help information for using the interface.

Any other invalid command will be replied with `!UNRECOGNIZED_CMD: <entered command>`

8.1.5 How to Make a Call

After connecting to the service, a call is placed by issuing the call command, as described above. After issuing the call command, you will receive an immediate reply. This message will indicate one of the results shown below.

Notes: The `<refno>` returned in the status message will match the one provided in the call command:

```
!CALL_SENT <refno>: telno:<telno> dnis:<dnis> url:<url>
uuidata:<uuid> defaults_file:<defaults>
parameter_list:<parameter_list>

 !SOCKET_ERROR <refno>: Socket not found!

 !NO_REFNO : No refernce number

 !INVALID_REFNO <refno>: Invalid refernce number

 !TOO_MANY_CALLS <refno>: Too many calls in progress

 !INVALID_TELNO <refno>: Incorrect telephone format

 !INVALID_URL <refno>: Incorrect URL format

 !INVALID_UUIDATA <refno>: Incorrect UUIDATA format

 !INVALID_DEFAULTFILE: Incorrect DEFAULTFILE format.

 !INVALID_PAIRLIST: Incorrect PAIRLIST format.

 !CALL_FAILED <refno>: telno:<telno> dnis:<dnis> url:<url>
uuidata:<uuid> defaults_file:<defaults>
parameter_list:<parameter_list>
```

In all these cases, except for `CALL_SENT,` there will be no further status returned for this call attempt.

Once the call has been placed with `CALL_SENT` notification, there are two possibilities:

1. The call connects successfully, in this case the following status will be returned:

   ```
   !CALL_STATUS <refno>: <follow by one of:>
   ```

   - `CONNECTED:` successfully connected
   - `MACHINE:` answering machine (if `analysis` is `y`)
   - `UNKNOWN_STATUS <status number>:` unknown status

   And then the call is dropped in due course with the following message:

   ```
   !CALL_DROP <refno> <timespent> <network disconnect reason>: <one of
   the disconnect reason listed below:> <protocol reason: protocol
   disconnect string>
   ```

   - `USER_END:` User hung up
   - `APPL_END:` application ended call
   - `TIMELIMIT_END:` timelimit of call reached
   - `UNKNOWN_REASON <internal disconnect reason number>:` unknown reason

2. The call does not connect and is dropped right away. In this case no `!CALL_STATUS` message will be received instead a `!CALL_DROP` message is received:

   ```
   !CALL_DROP <refno> <timespent> <network disconnect reason>:
   <drop_status>. < protocol reason: protocol disconnect string>
   ```

   where `<drop_status>` is one of:

   - `MACHINE:` Answering machine. (Only if `analysis` is `a`)
   - `VXML_DECLINE:` Voicexml Interpreter declined the call.
   - `BUSY:` Line is Busy.
   - `NO_ANSWER:` No answer in <num_units> <unit_type>
   - `NO_RESOURCES:` no free channel or media resource.
   - `CALL_FAILED:` Call failed.
   - `URLTIMEOUT:` Fetch URL timeout.
   - `BADURI:` Invalid URI type.
   - `NOAUTH:` Network denied.
   - `GKREJECT:` H.323 Gatekepper returned ARJ.
   - `SHUTTINGDOWN:` Interpreter is shutting down.
   - `NETWORKTIMEOUT:` Network timeout.
   - `BADDEST:` Invalid number.
   - `NO_LICENSE:` no licenses available.

- `RESTRICTED_TELNO:` Restricted telephone number.
- `UNSUPPORTED_URL:` url is unsupported.
- `INVALID_TELNO:` Invalid telephone number.
- `USER_END:` User hung up
- `UNKNOWN_REASON <internal disconnect reason number>:` unknown reason

The `<network disconnect reason>` and the `<protocol disconnect string>` are returned by the callmanager to give more information about the reason the call was dropped.

8.1.6 Known Issues

- When the commands are sent to the telnet interface programmatically without reading the reply back from the previously sent command, these two commands may be received by the interface as a concatenated single command. This would cause unexpected behavior.
- There is currently no way of terminating a call that was initiated using the OB API.

8.2 Full Call Recording

8.2.1 Overview

A significant component of delivering voice services is the ability to tune and troubleshoot those services. One component of this tuning involves the use of logged information to understand the performance of ASR in a particular application, or the call flow of a particular call.

However, in addition to using information logged about a call, it is also desirable to capture the actual audio associated with a call. Although the `<record>` tag provides some capabilities in this regard, a shortcoming of the `<record>` tag is that only the input from the user can be recorded; output of the platform is not capture by the `<record>` tag. Also, only one `<record>` may be active at a given time, so it is not possible to record two audio files – one for real application use (e.g. recording a voicemail) and the other for tuning/debugging use.

In order to improve capabilities in the above areas, the following improvements are made since 5.8:

- Recording capabilities of the call manager is extended to support a selection of what actually gets recorded – input, or mixed input & output. Using mixed input/output will have performance implications, however, since these capabilities are generally used on either prototype systems or in a limited (spot check) fashion on production systems, this performance impact will likely be acceptable.

- Multiple simultaneous recording operations is supported. This allows a long-running tuning-oriented mixed recording to be performed concurrently with normal application-level recording.

8.2.2 Enabling/disabling Full Call Recording

Existing VoiceGenie extensions to the `<log>` tag allow the destination for a logging message to be controlled via the `dest` attribute. A new value is supported for the `dest` attribute, and the actual text of the `<log>` tag control is used to enable or disable logging of a particular type of information on a particular call. Currently, only full call recording logging is supported.

The following are examples of enabling full call recording logging:

with VoiceGenie Legacy VoiceXML Interpreter:

```
<log dest="calllog">
   directory /usr/local/phoneweb/callrec;
   enable callrec recsrc=mixed;
   keep-files true;
</log>
```

with VoiceGenie NextGen VoiceXML Interpreter:

```
<log vg:dest="calllog">
   directory /usr/local/phoneweb/callrec
   enable callrec recsrc=mixed type=audio/x-wav
   keep-files true
</log>
```

Complete Full Call Recording Commands:

Note: In VoiceGenie NextGen VoiceXML Interpreter, semicolons are no longer used as command delimiters and are simply ignored (or treated as part of the command). Only line breaks are treated as command delimiters. As a result, it is no longer possible to have multiple commands on a single line, delimited by semicolons. Please refer to VG_7_2_Application_Migration_Guide section Full Call Recording for more details.

1. `directory <local-directory-name> [absolute];`

   The `directory` command specifies the directory in which call-specific log files will be collected.

`<local-directory-name>` specifies the directory, such as `/usr/local/phoneweb/callrec,` in which audio/information will be recorded. If this attribute is not provided, the Media Platform will make use of the path as specified in the configuration parameter `calllog.directory` which can be accessed via SMC.

The given directory will be treated as a root directory, and a subdirectory, named based on the call ID of a call (in the format of `<call ID>.<timestamp in YYMMDDhhmmss format>.<file extension>`), will be created and used to store actual files for a particular call. If it is desired to place the files in the directory directly, without a subdirectory, then this can be achieved by specifying `absolute` as the last token on the directory line.

If this command is not given, the default local directory, defined in `callmgr.cfg,` will be used instead.

If specifying the directory in which to save the FCR files, it must be done when the recording starts, along with the `enable callrec` command. It *must* appear in the first instance of a `<log>` tag with the `calllog` destination or it will be ignored. In particular, it is not possible to change the directory later in the application by using only the line `directory <New Directory>` as in the following example (shown using syntax for the Legacy VoiceXML Interpreter):

```
<log dest="calllog">
   directory /usr/local/phoneweb/callrec/newdata;
</log>
```

However, if multiple files are created (by multiple Full Call Recording sessions), it is possible to specify a different directory for each file. For example, if a single VoiceXML application contains the following code snippet (shown with syntax for the Legacy VoiceXML Interpreter), the two resulting Full Call Recording files are saved in two different directories:

```
<!-- start first recording -->
<log dest="calllog">
   directory /usr/local/phoneweb/callrec/dir1;
   enable callrec recsrc=mixed type=audio/basic;
</log>
... record for a while, then stop recording...
...
<!-- start second recording -->
<log dest="calllog">
   directory /usr/local/phoneweb/callrec/dir2;
   enable callrec recsrc=mixed type=audio/basic;
</log>
```

2. `enable callrec [recsrc=<in/mixed>] [type=<MIME-type>];`

   Enables call recording, or restarts call recording (in a new recording file) if it has already been started. All audio data on the call from this point forward will be recorded into the newly opened recording file. This will continue until the call terminates, a subsequent enable callrec command is issued, or until a disable callrec command is issued.

   If `recsrc=<in/mixed>` is specified, it forces call recording to record the inbound (from user), or mixed (combination of from & to user) audio paths. If this attribute is not specified, mixed will be assumed.

   If `type=<MIME-type>` is used, then the given MIME-type will be used to determine the file type used for the recording. The list of MIME-types supported is not specific to full call recording and is the same as defined for the `<audio>` and `<record>` tags. Full rate G.711 raw files can be recorded using `audio/basic` and `audio/x-alaw-basic;` G.726 raw files can be recorded using `audio/x-g726-24, audio/x-g726,` etc. The reference for the `<record>` tag will have up to date information with respect to MIME types supported.

3. `disable callrec;`

   Disables call recording, terminating any recordings in progress. Audio will be recorded up until the point that this command is executed.

4. `keep-files [true/false]`

   Controls whether or not files on the VG server are maintained after the call is terminated; if set to `true,` files will be maintained, otherwise, they will be deleted.
   Note: The default value is `true.` This command may be issued at any time while a call is active. The purpose of this capability is to keep files only if the call ends in an error condition. If a call is disconnected due to excessive `<nomatch>` events, the `<nomatch>` event handler could perform a `<log dest="calllog">keep-files true;</log>` to retain temporary files on the server.

8.2.3 Gain control

Fixed gain control for full call recording is a new feature introduced in release 7. It allows a fixed gain to be applied to all full call recordings when `recsrc=mixed` is used, on a platform-wide basis. The gain level can be configured via SMC using the following call manager parameter:

```
# mediatransport.fcr.gain
#
# Gain on FCR input from call participants (-30 to 30 dB)
#
```

Default is set to `0`.

8.2.4 Known limitations

- Due to prompt queuing feature, the `<log>` tag is sometimes processed before the actual prompt is played. The general rule is that all `<log>` tags within a queued prompt group are processed before the prompts are played.
- It is only possible to enable full call recording by adding `<log>` tag to the VXML application(s). It is currently not possible to turn on full call recording at the platform level without adding `<log>` tag to the VXML application(s).
- In pre-7.0, MIME-type cannot contain the semi-colon character
- The Full Call Recording files destination should not be a network mounted file system (e.g. NFS), as it can block I/O operations and result in unexpected Media Platform behaviors.
- For SIP devices that support multiple codecs, the configuration parameter `sip.transmitmultiplecodec` must be set to `0` for full call recording to work properly.
- For video-enabled Full Call Recording calls with `recsrc` set to `mixed` (note that the mime type must be set to a file container that can record video), the platform will always select platform output as the video source.
- For audio and video enabled FCR, if the input video stream has been paused and resumed during the call, there will be audio video synchronization problem in the recorded file.

8.3 RTSP URI Support

8.3.1 Overview

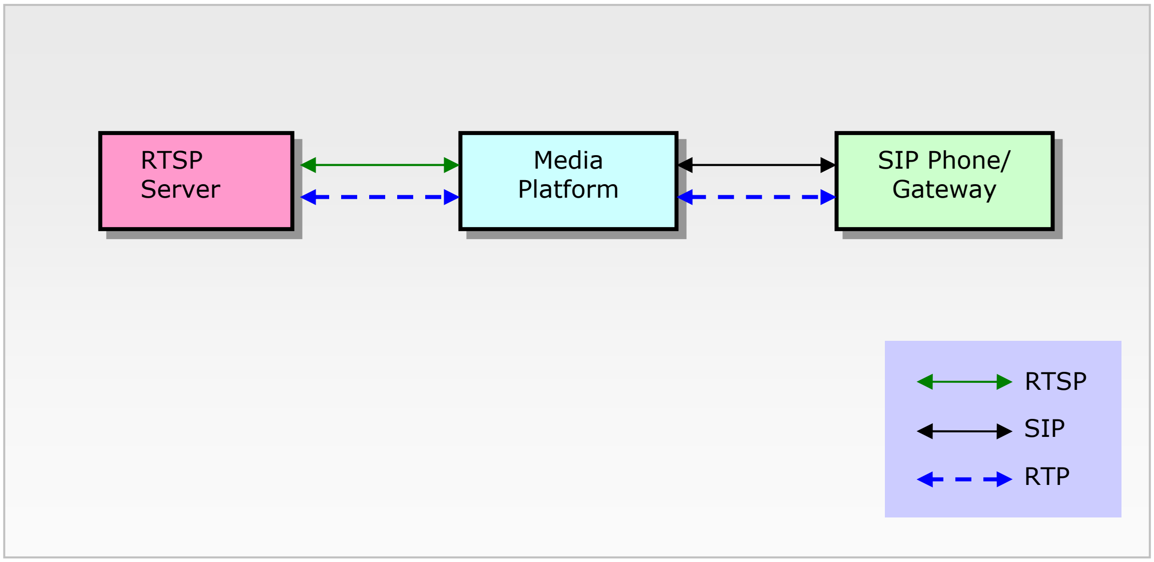
RFC2326 Real Time Streaming Protocol (RTSP) defines a protocol for control over the delivery of data with real-time property. The VoiceGenie 7.2 Media Platform supports RTSP URI that begins with ‘rtsp://’ in the <audio/> tag, which allows development of VoiceXML applications that deliver media content from a RTSP server to the end user. VCR control is also supported with RTSP URI using the VoiceGenie audio control extensions.

The RTSP URI implementation of the VoiceGenie 7.2 Media Platform is compatible with Real Helix Server and Apple Darwin Streaming Server.

8.3.2 RTSP Deployment Architecture

In a typical RTSP server deployment, the RTSP server resides on a separate physical server, although it is also possible to run the RTSP server and the Media Platform on the same physical server. When the Media Platform attempts to play a RTSP prompt, it makes the request to the RTSP server specified in the URI using the RTSP protocol. The request involves confirming the availability of the requested media, ensuring the media codec is supported, and setting up the RTP connection. If the request is successful, the RTSP server will stream the requested media to the Media Platform via the negotiated RTP connection. Note that TCP transport is used for the RTSP connection. A new TCP/RTSP connection is created for each prompt and is torn down when the prompt has completed.

When the Media Platform receives the RTP packets from the RTSP server, it deframes the RTP packets and performs transcoding to the media if required. The media is then re-packetized and transmitted to the SIP phone or SIP gateway. The Media Platform does not directly forward the received RTP packets to the SIP phone or the SIP gateway.



8.3.3 Generate Media Files for RTSP Server

Video files such as 3GP or Quicktime files may have to be encoded with 'Hint Track' in order to work with the RTSP server.

- Software such as QuickTime Pro will be able to generate video files with 'Hint Track'.
- The Media Platfrom can record 3GP files with 'Hint Track' by enabling the Call Manager Configuration parameter `mpc.mediamgr.recordrtphinttrack`. For more information on this parameter, please refer to the *VoiceGenie 7.2 Media Platform System Reference Guide.*

Most Audio files should work with the RTSP server without requiring 'Hint Track'. For an accurate list of supported file containers and media codecs, please refer to the documentation of the respective RTSP server.

Note that the media files must be encoded with audio and video codecs supported by the Media Platform. Please refer to the *VoiceGenie 7.2 Media Platform System Reference Guide* for the list of supported audio and video codecs.

8.3.4 VCR Control

VCR control is supported with RTSP URI using the VoiceGenie audio control extensions. Two type of VCR control is supported depending on the Media Platform's configuration and the RTSP server's capabilities.

VCR native mode is implemented using internally buffered data in the Media Platform. It is used to provide limited VCR control support for RTSP prompts when the RTSP server does not support RTSP PAUSE or the RTSP PLAY range parameter. The limitations of the VCR native mode are mentioned in the *Known Limitations* section below.

VCR enhanced mode provides much improved user experience by making use of the RTSP PAUSE and RTSP PLAY range parameter. When the Media Platform detects a VCR command, it calculates the desired media offset and checks whether the request can be fulfilled by the internal buffer. If the internal buffer does not contain the requested data, it stops the RTSP stream using the RTSP PAUSE command, and issues a new RTSP PLAY command with the range parameter that specifies the desired start offset. As a result, the media data available for VCR control is no longer limited by the internal buffer size. VCR enhanced mode is the default VCR mode.

To use the VCR enhanced mode, the RTSP server must support both RTSP PAUSE and RTSP PLAY range parameter. Otherwise VCR native mode will be used.

The default RTSP server capability can be specified using the Call Manager Configuration parameters `rtsp.mediamgr.pause` and `rtsp.mediamgr.playrange`. The capabilities can be enabled or disabled by setting the parameters to 1 or 0 respectively. For detailed description of the parameters, please refer to the *VoiceGenie 7.2 Media Platform System Reference Guide*.

The RTSP server capability can also be specified on a per-session basis. The URI parameters `vg-rtspserver-pause` and `vg-rtspserver-playrange` can be specified in the RTSP URI. The capabilities can be enabled or disabled by setting the parameters to 1 or 0 respectively

For instance, the following RTSP URI specifies that RTSP PLAY range is not supported, as a result native VCR will be used.

```
rtsp://139.48.28.3/welcome.3gp;vg-rtspserver-pause=1;
vg-rtspserver-playrange=0
```

8.3.5 Known Limitations

- Using VCR native mode, users can only skip back to the beginning of the data buffered in the Media Platform. In addition, skipping back to the previous prompt is not allowed.
- Using VCR native mode, users will experience delay when trying to skip forward.
- When the Media Platform failed to play an RTSP prompt due to error returned from the RTSP server, the Legacy Interpreter will throw an error.internal event, and will not continue with the rest of the prompts. The Next Generation Interpreter will continue to play the rest of the prompts.

Chapter

9 Operations

This section describes how the system does logging, metrics, and tracing, and how data collection can be manipulated. Also discussed is the real-time call monitor, and explanations for the health string entries retrievable via the CLC and SMC.

9.1 Metrics and Logging/Billing

9.1.1 General Metrics

Metrics data is currently written to file(s) on the platform. The file `pw_billingfile` is obsolete and removed from the system since release 6.4. All the billing information is now incorporated into `pw_metricsfile` under `/usr/local/phoneweb/logs` (linux) or `($INSTALLROOT)\mp\logs` (windows) directory.

This data can be migrated off the platform for archival, or processed to generate unified Call Detail Records (CDRs). The current version of the VoiceGenie platform can migrate these records to an offboard MySQL database where it can be accessed later to create CDRs.

Each record has the following fields:

Field	Contents
Timestamp	Timestamp structured as `yyyy-mm-dd/hh:mm:ss.mmm`. Time is 24 hour, based on the platform timezone.
Record Type	Always `METRIC`
Session ID	Globally unique session identifer.

Field	Contents
Operation	incall_initiated, incall_begin, incall_end, incall_reject, bridge_initiated, bridge_begin, bridge_reject, call_initiated, call_begin, call_end, call_reject, outcall_requested, outcall_initiated, outcall_begin, outcall_end, outcall_reject, transfer_initiated, transfer_connected, transfer_result, call_reference
Operation Data	Data specific to the record

The session identifier can be used to assemble all information related to a specific telephone call, or 'session'.

The first phrase of the metrics ID corresponds to the type of the entry. The second phrase of the metrics ID is either `requested` for requested, `initiated` for initiated, `begin` for begin, `reject` for reject, or `end` for end. The exception is the `transfer_result` entry that corresponds to a transfer request for transferring or redirecting a call off the platform. A `begin` or `end` entry corresponds to a call connect or disconnect event.

If you have a `begin`, then you will have an `end` only, not a `reject`; and with a `reject` you will not have a `begin` or `end`. But in all cases, `initiated` will be logged prior to `begin` or `reject`. For outcall entry, `requested` entry exists even before `initiated` to specify a stage where outbound trunk is not selected yet.

For details for each Metrics Entries, please refer to the *VoiceGenie 7.2 Media Platform System Reference Guide*.

Note: the session ID for `transfer_result` is always the parent ID

9.1.2 Metrics for Transfer

Metrics behavior for transfer depends on the transfer style (see Call Transfer for details) to which the transfer method belongs.

Transfer Style	Method
One-leg	hkf, refer, inband
Join	h450join, referjoin
Bridge	Bridge, mediaredirect

The following table shows how metrics is logged for each transfer style:

Style	Accepted	Rejected
One-leg	transfer_initiated (transfer_connected)* transfer_result * Only for whisper transfer	transfer_initiated transfer_result
Join	transfer_initiated bridge_initiated (bridge_begin)* transfer_result bridge_end * May not be logged for type="blind"	transfer_initiated bridge_initiated transfer_result bridge_reject
Bridge	bridge_initiated bridge_begin bridge_end	bridge_initiated bridge_reject

Please see *VoiceGenie 7.2 Media Platform System Reference Guide* to get detailed information about the metric entries.

9.2 Alarms in Media Platform

The VoiceGenie 7.2 Media Platform can be configured to logs alarms and potential problems into the system log, with six levels of severity: CRTI (critical), EROR (error), WARN (warning), NOTE (notice), INFO (information) and DEBUG.

When an unexpected behavior occurs when running the VoiceGenie 7.2 Media Platform, it is recommended to first check if there is any alarm through SMC's Alarm Browser (please see the next 2 sections).

Please see *VoiceGenie 7.2 Media Platform System Reference Guide* to get detailed information about the definitions, impacts, potential causes as well as recommended actions for all possible alarms of various Media Platform components.

9.2.1 Syslog

If the SYSLOG sink is enabled alarms and logs can be sent to the system log. Under Linux and Solaris the logs are sent to Syslog, which is a deamon process that listens for data on port 514. All data received is written to a log file, by default this file is found at /usr/local/phoneweb/logs/pw_logfile.

Under Windows the logs are sent to the Application Log in Event Viewer, which can be accessed under the `Administrative Tools` section of the Control Panel.

Note: On Linux and Solaris Syslog writes logs to the `/usr/local/phoneweb/logs/pw_logfile` file, if this file is deleted the Syslog needs to be restarted to recreate this file. To start, stop or restart Syslog you must be the root user. To become the root user log in to the system and type in `su,` then enter the root password when prompted.

Then, to start the Syslog, issue the following command:

```
/etc/init.d/syslogd start
```

Then, to stop the Syslog, issue the following command:

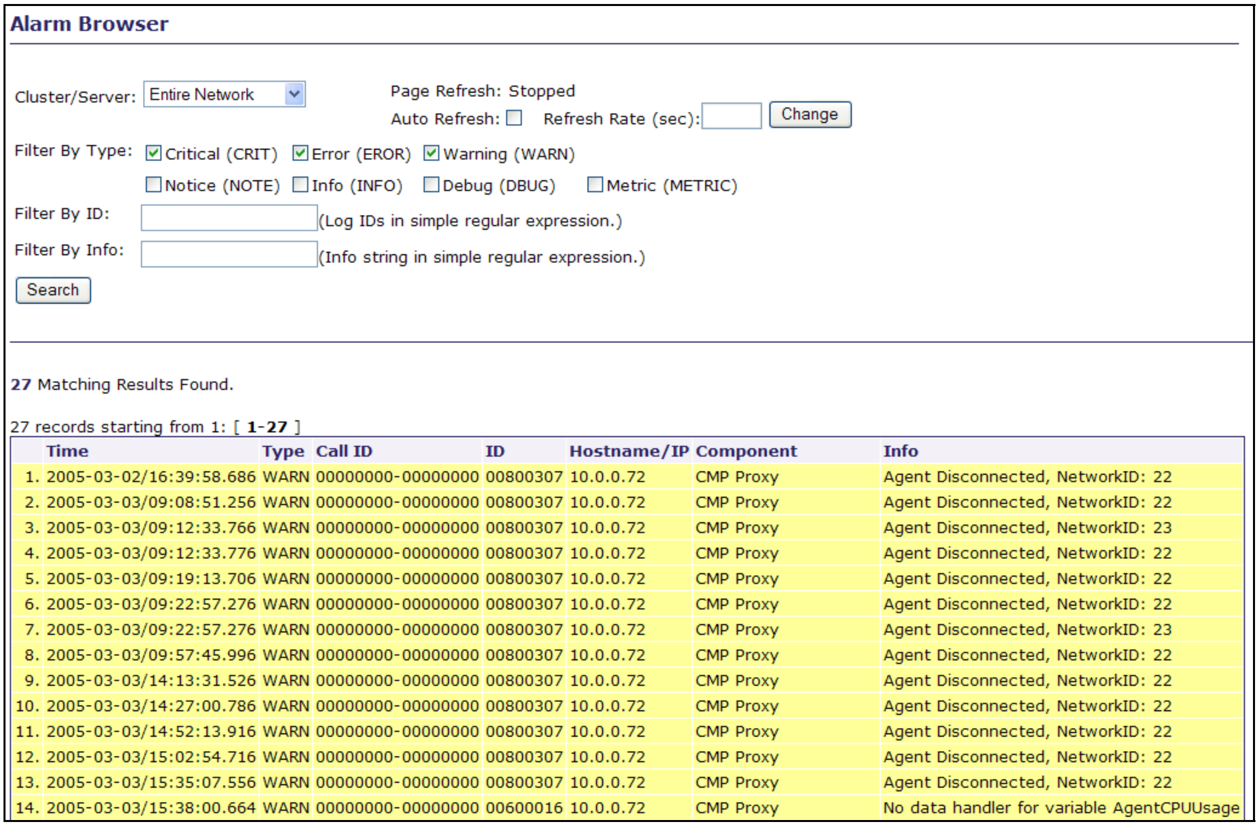
```
/etc/init.d/syslogd stop
```

Then, to restart the Syslog, issue the following command:

```
/etc/init.d/syslogd restart
```

9.2.2 Alarm Browser

The Alarm Browser allows users to view all detailed logging and alarming data that is logged into the database. This includes any alarms (i.e. Critical, Error, Warning), any general logs (Notice, Info, Debug) and call metrics information. The Alarm Browser can be accessed via the SMC interface. The following is a screenshot of the Alarm Browser:



Alarm Browser

Cluster/Server: Entire Network

Page Refresh: Stopped

Auto Refresh: □ Refresh Rate (sec): [ ] Change

Filter By Type: ☑ Critical (CRIT) ☑ Error (EROR) ☑ Warning (WARN)
□ Notice (NOTE) □ Info (INFO) □ Debug (DBUG) □ Metric (METRIC)

Filter By ID: [ ] (Log IDs in simple regular expression.)

Filter By Info: [ ] (Info string in simple regular expression.)

Search

27 Matching Results Found.

27 records starting from 1: [**1-27**]

	Time	Type	Call ID	ID	Hostname/IP	Component	Info
1.	2005-03-02/16:39:58.686	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
2.	2005-03-03/09:08:51.256	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
3.	2005-03-03/09:12:33.766	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 23
4.	2005-03-03/09:12:33.776	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
5.	2005-03-03/09:19:13.706	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
6.	2005-03-03/09:22:57.276	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
7.	2005-03-03/09:22:57.276	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 23
8.	2005-03-03/09:57:45.996	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
9.	2005-03-03/14:13:31.526	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
10.	2005-03-03/14:27:00.786	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
11.	2005-03-03/14:52:13.916	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
12.	2005-03-03/15:02:54.716	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
13.	2005-03-03/15:35:07.556	WARN	00000000-00000000	00800307	10.0.0.72	CMP Proxy	Agent Disconnected, NetworkID: 22
14.	2005-03-03/15:38:00.664	WARN	00000000-00000000	00600016	10.0.0.72	CMP Proxy	No data handler for variable AgentCPUUsage

Users can search the logs using the various search criteria. The search criteria include the cluster or server from where the log was created, the type of log (i.e. Critial, Error, Warning, Notice, Info, Debug or Metric), the Log ID of the log or by text in the info field. Note that this page can be set to be refreshed if desired. In general this is a good place to check for alarms and for system and service impacting conditions.

The color of the row in the results table signifies the severity of the logged event. The events are color coded by severity as follows:

Color	Severity
🟥	Critical
🟧	Error
🟨	Warning
🟦	Notice
▭	Information
□▨	Debug
□▨	Metrics

Also, each event has a timestamp for time at which the event occurred, the type (i.e. severity), the associated callID if one exists, the Log ID of the log (this value is a hexadecimal value), the source IP of the log, the source component of the log and the information text. Please consult Appendix A of the *VoiceGenie 7.2 OA&M Framework User's Guide* for a better understanding of the `Log ID` field.

9.3 Health Status

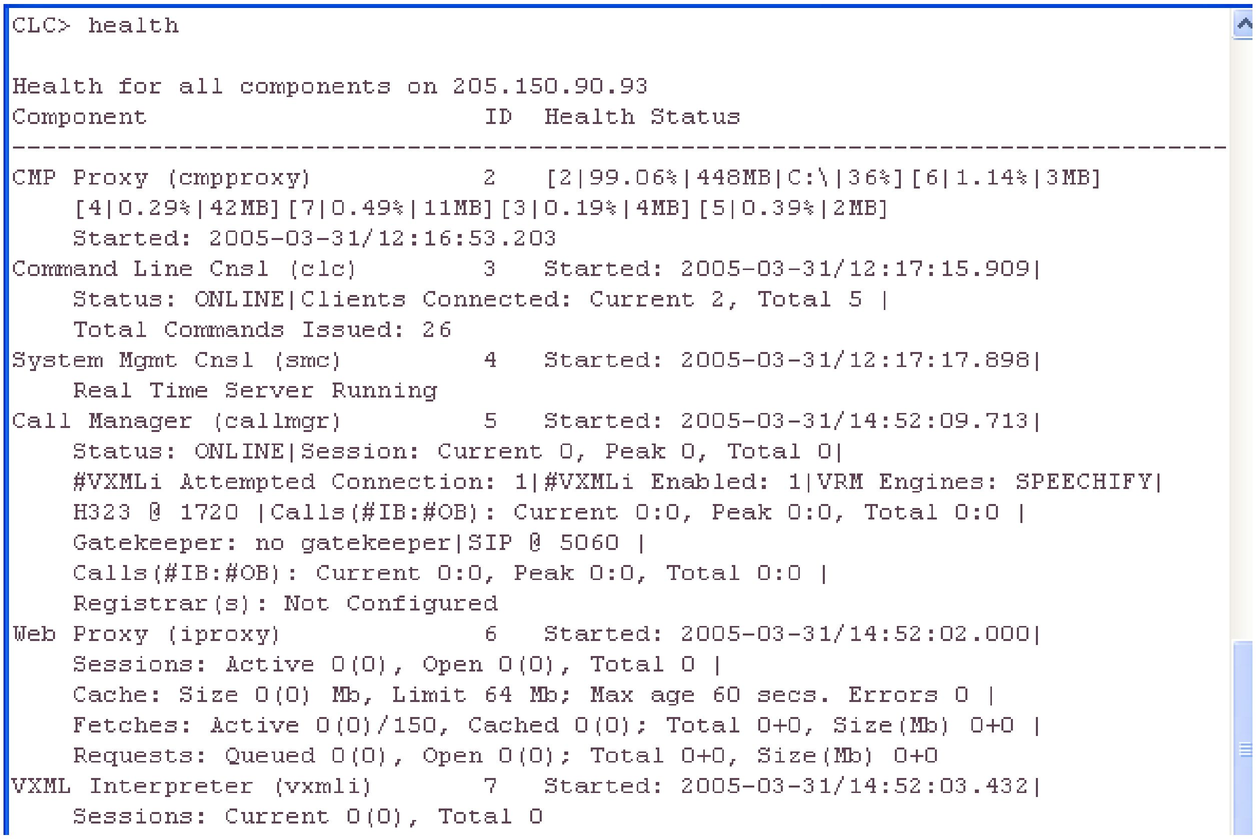
9.3.1 Overview

Users can retrieve real time health status of the Media Platform by:

1. Using the `health` command of CLC
2. Checking the Status Monitor of SMC

CLC health command

By simply issuing a `health` command, the CLC will return a summary of the health status of all Media Platform components. Here is a snapshot:



```
CLC> health

Health for all components on 205.150.90.93
Component                       ID  Health Status
----------------------------------------------------------------------------------
CMP Proxy (cmpproxy)            2   [2|99.06%|448MB|C:\|36%][6|1.14%|3MB]
    [4|0.29%|42MB][7|0.49%|11MB][3|0.19%|4MB][5|0.39%|2MB]
    Started: 2005-03-31/12:16:53.203
Command Line Cnsl (clc)         3   Started: 2005-03-31/12:17:15.909|
    Status: ONLINE|Clients Connected: Current 2, Total 5 |
    Total Commands Issued: 26
System Mgmt Cnsl (smc)          4   Started: 2005-03-31/12:17:17.898|
    Real Time Server Running
Call Manager (callmgr)          5   Started: 2005-03-31/14:52:09.713|
    Status: ONLINE|Session: Current 0, Peak 0, Total 0|
    #VXMLi Attempted Connection: 1|#VXMLi Enabled: 1|VRM Engines: SPEECHIFY|
    H323 @ 1720 |Calls(#IB:#OB): Current 0:0, Peak 0:0, Total 0:0 |
    Gatekeeper: no gatekeeper|SIP @ 5060 |
    Calls(#IB:#OB): Current 0:0, Peak 0:0, Total 0:0 |
    Registrar(s): Not Configured
Web Proxy (iproxy)              6   Started: 2005-03-31/14:52:02.000|
    Sessions: Active 0(0), Open 0(0), Total 0 |
    Cache: Size 0(0) Mb, Limit 64 Mb; Max age 60 secs. Errors 0 |
    Fetches: Active 0(0)/150, Cached 0(0); Total 0+0, Size(Mb) 0+0 |
    Requests: Queued 0(0), Open 0(0); Total 0+0, Size(Mb) 0+0
VXML Interpreter (vxmli)        7   Started: 2005-03-31/14:52:03.432|
    Sessions: Current 0(0), Total 0
```

To view the health information for a particular component, issue the command:

```
health <service>
```

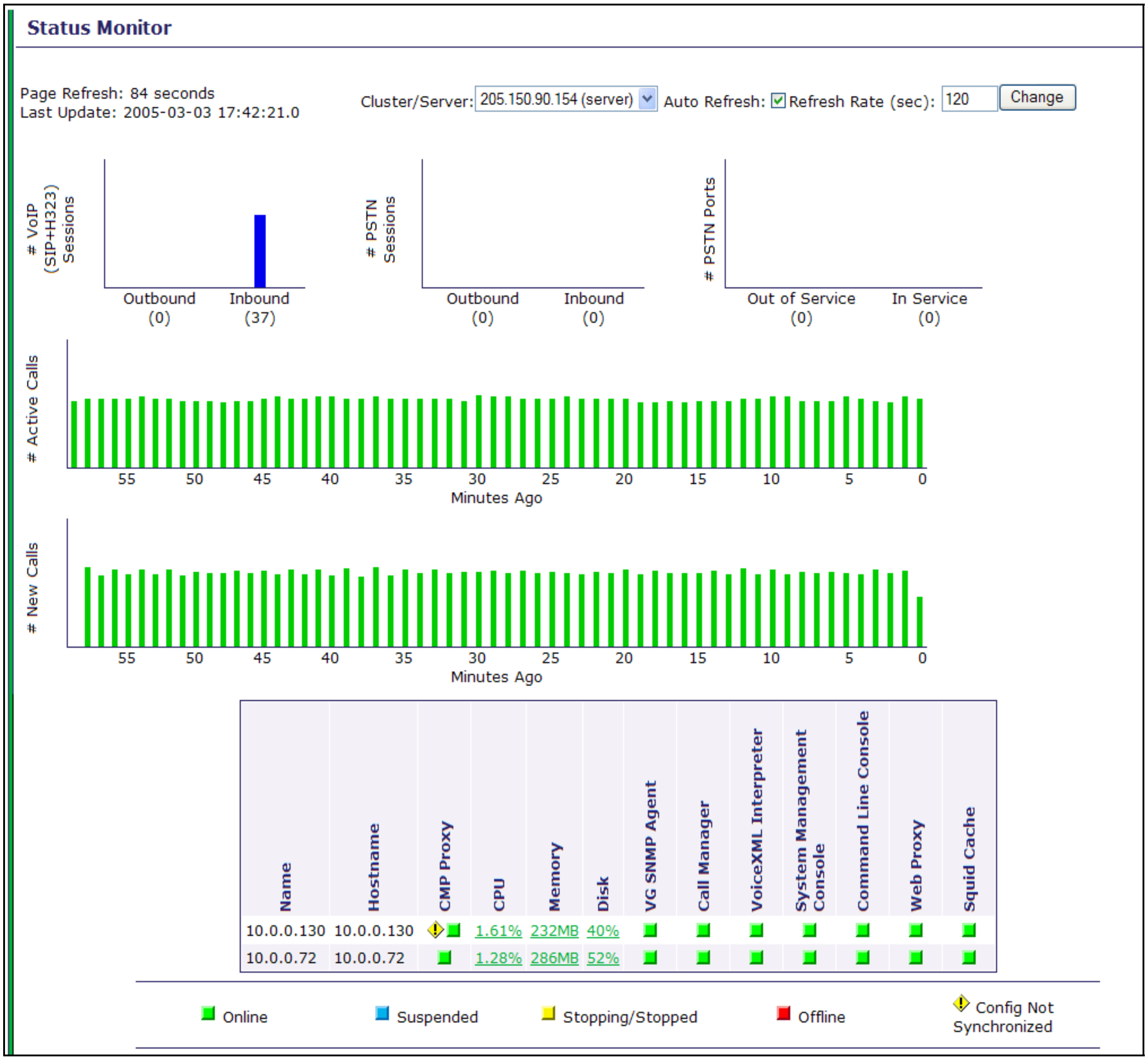
e.g. `health callmgr`

In general most of the components have a time stamp of when they were last started. Details for the health string content for each individual Media Platform components will be discussed in the following sections. For further information regarding CLC, please refer to:

VoiceGenie 7.2 OA&M Framework – CLC User's Guide

SMC Status Monitor

The Status Monitor can be accessed under the `Monitoring` tab of SMC. Here is a snapshot:



Status Monitor

Page Refresh: 84 seconds
Last Update: 2005-03-03 17:42:21.0

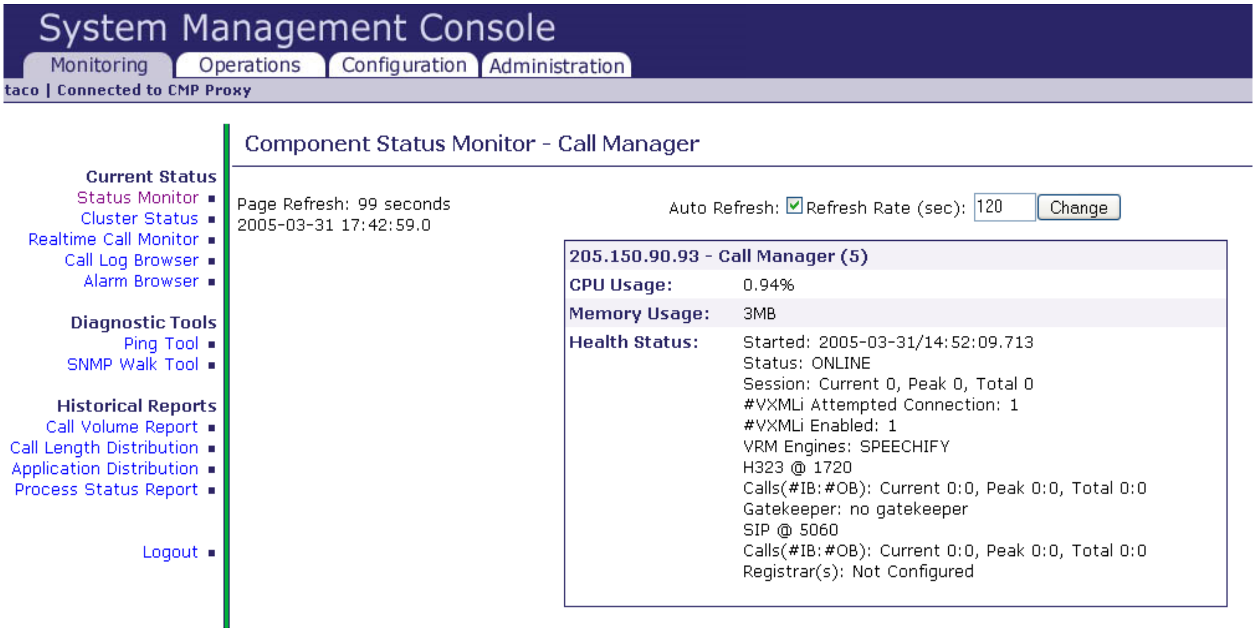
Cluster/Server: 205.150.90.154 (server) Auto Refresh: ☑ Refresh Rate (sec): 120 Change

Name	Hostname	CMP Proxy	CPU	Memory	Disk	VG SNMP Agent	Call Manager	VoiceXML Interpreter	System Management Console	Command Line Console	Web Proxy	Squid Cache
10.0.0.130	10.0.0.130	⚠ 🟩	1.61%	232MB	40%	🟩	🟩	🟩	🟩	🟩	🟩	🟩
10.0.0.72	10.0.0.72	🟩	1.28%	286MB	52%	🟩	🟩	🟩	🟩	🟩	🟩	🟩

🟩 Online 🟦 Suspended 🟨 Stopping/Stopped 🟥 Offline ⚠ Config Not Synchronized

It shows the overall call status of the Media Platform. Further information for each of the components can be accessed by clicking on the respective square of the service, close to the bottom of the page.

For example, this is what a user would get by clicking on the square associated with the Call Manager:



Component Status Monitor - Call Manager

Page Refresh: 99 seconds
2005-03-31 17:42:59.0

Auto Refresh: ☑ Refresh Rate (sec): 120

205.150.90.93 - Call Manager (5)	
CPU Usage:	0.94%
Memory Usage:	3MB
Health Status:	Started: 2005-03-31/14:52:09.713 Status: ONLINE Session: Current 0, Peak 0, Total 0 #VXMLi Attempted Connection: 1 #VXMLi Enabled: 1 VRM Engines: SPEECHIFY H323 @ 1720 Calls(#IB:#OB): Current 0:0, Peak 0:0, Total 0:0 Gatekeeper: no gatekeeper SIP @ 5060 Calls(#IB:#OB): Current 0:0, Peak 0:0, Total 0:0 Registrar(s): Not Configured

The information displayed is the same as what can be achieved via the CLC `health` command, except that the Status Monitor shows additional information such as `CPU Usage` and `Memory Usage`.

For further information regarding SMC, as well as other functions of the Status Monitor, please refer to:

VoiceGenie 7.2 OA&M Framework – SMC User's Guide

9.3.2 Call Manager

Here is a snapshot of the health string output of the call manager:

```
Health for Call Manager (callmgr) on 10.0.0.147
Started: 2005-03-23/16:45:42.657
Status: ONLINE
Session: Current 0, Peak 0, Total 0
#VXMLi Attempted Connection: 1
#VXMLi Enabled: 1
VRM Engines: None
SIP @ 5060
Calls(#IB:#OB): Current 0:0, Peak 0:0, Total 0:0
Registrar(s): Not Configured
```

- `Status` – shows the current operation status of the Call Manager, which can be one of: `ONLINE`, `SUSPENDED` or `OFFLINE`

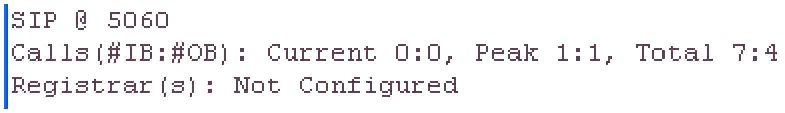
- `Session` – shows the number of call sessions in the Call Manager. `Current` indicates the number of currently active session; `Peak` indicates the maximum number of concurrent sessions thus far since the Call Manager has started; and `Total` indicates the total number of all calls. Note that these figures are the sum of the respective figures from all individual line managers.

Note: The `Current` session count reflects the number of logical call objects currently exist in the system. For efficiency, disconnected call objects are purged periodically. Hence, even if a call is disconnected and the channel is freed for the next call, the call object may still not undestroyed yet until the purge and this may cause slight inaccuracy to the `Current` session count.

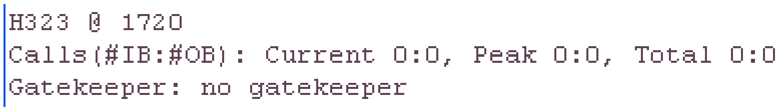
- `VXMLi Attempted Connection` – shows the number of connection attempts the Legacy Interpreter(s) has/have made.
- `VXMLi Enabled` – shows the number of enabled Legacy Interpreter instances. VoiceGenie 7.2 Media Platform is capable of supporting more than one Legacy Interpreter instances at a time.
- `VRM Engines` – shows the TTS/ASR Voice/Speech resources that are accessible by the Call Manager.
- All line managers have a `Calls` entry which shows information call information in terms of inbound and outbound calls. For the meanings of `Current, Peak` and `Total` please refer to the definition of the `Session` entry above.

9.3.2.1 VoIP Line Managers

Here are the snapshots of the health string outputs of the SIP and H.323 line managers:



```
SIP @ 5060
Calls(#IB:#OB): Current 0:0, Peak 1:1, Total 7:4
Registrar(s): Not Configured
```



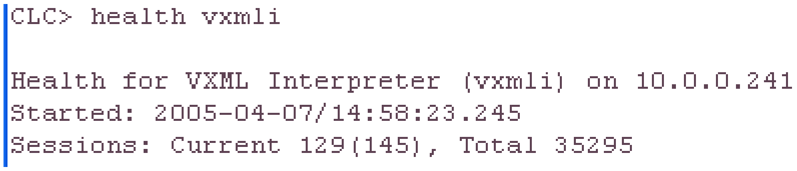
```
H323 @ 1720
Calls(#IB:#OB): Current 0:0, Peak 0:0, Total 0:0
Gatekeeper: no gatekeeper
```

The integer on the right hand side of the VoIP Protocol (SIP or H.323) indicates the port number on which the Call Manager is using for sending/receiving calls.

The SIP line manager can be configured to register with a SIP Registrar, using the parameter `sip.registration;` similarly, the H.323 line manager can be configured to register with a H.323 Gatekeeper using the parameter `h323.ras.registrationinfo.` The health string entries `Registrar(s)` and `Gatekeeper` show such status.

9.3.3 Legacy Interpreter (VXMLi)

Here is a snapshot of the health string output of the Legacy Interpreter:



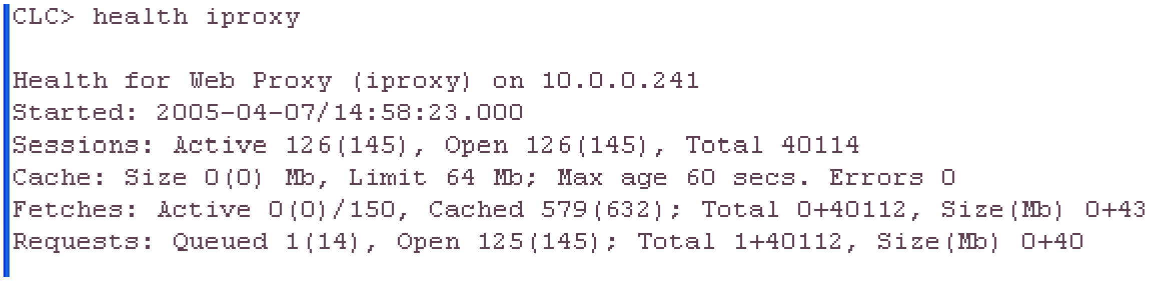
```
CLC> health vxmli

Health for VXML Interpreter (vxmli) on 10.0.0.241
Started: 2005-04-07/14:58:23.245
Sessions: Current 129(145), Total 35295
```

- `Started` – indicates the date and time when the vxmli process was started
- `Sessions` –
  - `Current <number of current running/active sessions> (<peak number of concurrent active sessions>)`
  - `Total <Total number of sessions>`

9.3.4 Fetching Module/Web Proxy (iproxy)

Here is a snapshot of the health string output of the Web Proxy:



```
CLC> health iproxy

Health for Web Proxy (iproxy) on 10.0.0.241
Started: 2005-04-07/14:58:23.000
Sessions: Active 126(145), Open 126(145), Total 40114
Cache: Size 0(0) Mb, Limit 64 Mb; Max age 60 secs. Errors 0
Fetches: Active 0(0)/150, Cached 579(632); Total 0+40112, Size(Mb) 0+43
Requests: Queued 1(14), Open 125(145); Total 1+40112, Size(Mb) 0+40
```

- `Started` – indicates the date and time when the iproxy process was started
- `Sessions` –
  - `Active <Number of currently active sessions that have active clients> ( <Peak number of concurrent active sections>)`
  - `Open <Number of currently open sessions, whether they are active and inactive> (<Peak number of concurrent open sessions)`
  - `Total <Total number of sessions>`
- `Cache` –
  - `Size <Size of the shared memory cache being used in Mbytes> (<Peak size of the shared memory cache concurrently being used in Mbytes>)`
  - `Limit <Limit of the shared memory cache in Mbytes, obtained from configuration parameter iproxy.cache_max_size>`
  - `Max age <Maximum age for data cached in the fetching module in seconds, obtained from configuration parameter iproxy.cache_max_age>`
  - `Errors <Total number of failed fetches>`

- Fetches —
  - Active <Number of currently active fetches initiated by the fetching module to the HTTP proxy/server> (<Peak number of concurrent active fetches initiated by the fetching module to the HTTP proxy/server>) / <Maximum number of concurrent active fetches allowed to be initiated by the fetching module to the HTTP proxy/server, obtained from configuration parameter iproxy.max_connections >
  - Cached <Number of entries currently in the fetching module's cache> (<Peak number of cached entries>)
  - Total <Total number of fetches initiated by the fetching module to the HTTP proxy/server> + <Total number of fetches to retrieve files from local machine or remote machines>
  - Size <Total data size obtained from fetches initiated by the fetching module to the HTTP proxy/server> + <Total data size obtained from fetches to retrieve files from local machine or remote machines>
- Requests —
  - Queued <Number of currently pending requests received from clients> (<Peak size of the pending request queue>)
  - Open <Number of currently opened requests received from clients, including active requests and pending requests> (<Peak number of concurrently opened requests received from cliens, including active requests and pending requests>)
  - Total <Total number of HTTP requests received from clients. Please note that not every request received from a client will initiate a fetch to the HTTP server/proxy. If a fresh response is cached, the cached response will be sent back to the client.> + <Total number of requests received from clients to retrieve file from local machine or remote machines. Please note that if a client requests a file that is in the cache, the Fetching Module will return the cached file to the client>
  - Size <Total data size sent to clients upon HTTP requests> + <Total file size sent to clients>

9.4 Preventive Maintenance

There are a number of performance and stability related recommendations that are relevant when deploying the VoiceGenie platform in various configurations. This section provides a summary of these recommendations. Failure to follow these recommendations may lead to performance or stability issues.

- **Turn all VoiceGenie tracing off**

  VoiceGenie tracing is only intended to be used to resolve platform issues when so instructed by VoiceGenie technical support. Disabling of tracing will reduce the overall load on the system and the system will be less likely to experience problems.

  This can be achieved by setting the parameter `cmp.trace_flag` to `false` via SMC for each and every VoiceGenie components. It can also be done via the CLC `tracelevel` command.

  For details please refer to the “Enabling or Disabling Tracing/Debugging” section of the *VoiceGenie 7.2 OA&M Framework User’s Guide,* and the “General Component Operation Commands” section of the *VoiceGenie 7.2 OA&M Framework – CLC User’s Guide.*

- **Ensure the savetmpfiles property is turned off when not needed for debugging purposes.**

  This property saves all intermediate files related to VoiceXML page processing, and can provide useful information for debugging of a complex application. Note this property can be set in any location in an application. To ensure this is turned off, please check the application root document (`defaults.vxml` for Legacy Interpreter, defaults-ng.vxml for Next Generation Interpreter), as well as each page in the application. If you are using savetmpfiles, be sure to periodically purge the `(VoiceGenie Software install root)/tmp` directory.

- **Turn off redundant logging of audio sent to an ASR engine.**

  It is often possible (depending upon the ASR engine) to capture utterances at more than one place in the system. For example, in the BBN ASR engine, both the VoiceGenie ASR client, and the Hark ASR client component can capture utterances. To turn off utterance captures, set the `com.voicegenie.saveutterance` property to `false,` or simply remove the property from your application.

- **Enable syslog rotation**

  If the syslog rotation is off, the messages file may become large. It then becomes more costly for the OS to seek to the end of the file and there is more load on the system.

- **Do not delete any of the pre-created directories**

  A number of directories are created after the Media Platform installation (such as `logs, tmp, config, audio, utterance,` etc.). These directories are crucial for proper operations of the system. Please do not destroy or rename any of these directories.

Appendix

A Burke Draft Support

This appendix will describe various aspects of the Burke Draft (http://tools.ietf.org/id/draft-burke-vxml-02.txt) and whether or not the feature is supported by the VoiceGenie 7.2 Media Platform with the Next-Generation VoiceXML Interpreter.

The Burke Draft describes a SIP interface to VoiceXML media services. Sections relevant to this appendix include section 2 VoiceXML Session Establishment and Termination, section 3 Media Support, section 3 Returning Data to the Application Server, section 5 Outbound Calling, and section 6 Call Transfer.

A.1 Support

This section of the appendix will describe each section of the Burke Draft and the current support status.

Req ID	Burke Draft Section	Requirement	Current Support	Notes
1	2.1	The parameters voicexml, maxage, maxstale, method, postbody as per [RFC4240].	SUPPORTED	

Req ID	Burke Draft Section	Requirement	Current Support	Notes
2	2.1	The parameter "timeout" for the initial fetch of the URI. The parameter "gvp.defaultsvxml" which allows a user to select which defaults.vxml page to use. Note: Not explicitly defined in Burke Draft, but used in a similar fashion.	SUPPORTED	The "timeout" parameter sets the timeout for requesting the URI, and will override a default value. The "gvp.defaultsvxml" parameter set the defaults.vxml page. It can only be used in conjunction with a "voicexml" parameter.
3	2.1	Incorrectly formed requests rejected with 4xx class response.	NOT SUPPORTED	To continue support of non-Burke Draft formatted requests.
4	2.1	Repeated init-parameters rejected with 400 Bad Request response.	SUPPORTED	
5	2.1	URL-Encoding of parameters.	SUPPORTED	
6	2.2	Upon receipt of INVITE, a provisional response, 100 Trying, will be sent, followed by a 200 OK once the document is fetched. After the ACK is received, the application will begin executing.	SUPPORTED	
7	2.2	Optimization: Execute the application up to point of the first VoiceXML waiting state or prompt flush before sending 200 OK response.	NOT SUPPORTED	This feature is not currently supported.
8	2.2	Request-URI does not conform to the Burke Draft, return 400 Bad Request	NOT SUPPORTED	To continue support of non-Burke Draft formatted requests.
9	2.2	voicexml parameter not provided and default page not configured, return 400 Bad Request and 399 with human readable error message.	NOT SUPPORTED	To continue support of non-Burke Draft formatted requests.
10	2.2	If the VoiceXML document cannot be fetched or parsed, return 500 Internal Error.	SUPPORTED	Requires that vxmli.default.alternate_uri is not set in the call manager configuration.

Req ID	Burke Draft Section	Requirement	Current Support	Notes
11	2.2	Include a Warning header with a 3-digit code of 399 and human readble error message in the message generated in item 2.2.5.	SUPPORTED	
12	2.2	When an INVITE request exceeds the MTU of the underlying network, a transport mechanism appropriate to larger messages (such as TCP) will be used.	SUPPORTED	
13	2.3	Upon starting a media-less Dialog – INVITE without media; 200 OK with offered media; ACK with Media but media port(s) set to 0 OR INVITE with SDP containing no media lines followed by regular INVITE / 200 / ACK flow – the VoiceXML page is not executed until a re-INVITE with port information is sent.	SUPPORTED	
14	2.3	Once a VoiceXML application is running, a re-INVITE that disables media stream (i.e. sets the port to 0) will not affect the executing application (timers still running).	SUPPORTED	
15	2.4	Support for the variable session.connection.local.uri as described in the Burke Draft.	SUPPORTED	
16	2.4	Support for the variable session.connection.remote.uri as described in the Burke Draft.	SUPPORTED	
17	2.4	Support for the session.connection.redirect variable as described in the Burke Draft.	SUPPORTED	
18	2.4	Support for evaluating the variable session.connection.protocol.name to “sip”	SUPPORTED	
19	2.4	Support for evaluating the variable session.connection.protocol.version to “2.0”	SUPPORTED	

Req ID	Burke Draft Section	Requirement	Current Support	Notes
20	2.4	Support for the session.connection.protocol.sip.headers variable as described in the Burke Draft.	SUPPORTED	Set sip.in.invite.headers = *, and add session.connection. protocol.sip.headers|Sip.Invite|6 to vxmli.session_vars in the Call Manager configuration.
21	2.4	Using the variable session.connection.protocol.sip.requesturi as an associative array formed from the URI parameters as described in the Burke Draft.	SUPPORTED	
22	2.4	Support for the session.connection.aai variable as described in the Burke Draft.	SUPPORTED	
23	2.4	Support for the session.connection.ccxml as described in the Burke Draft.	SUPPORTED	
24	2.4	Using session.connection.protocol.sip.media as an array where each array element is an object with the properties outlined in items 2.4.10.1 to 2.4.10.3. Note: This parameter will be updated as the media values involved in the session change.	SUPPORTED	
25	2.4	Array element property: type	SUPPORTED	
26	2.4	Array element property: direction	SUPPORTED	
27	2.4	Array element property: format for each payload type on the m-line	SUPPORTED	
28	2.5	Upon receipt of a BYE, a 200 OK is sent as a response and a 'connection.disconnect.hangup' event is thrown	SUPPORTED	
29	2.5	If a Reason header [RFC3326] is present in the BYE Request, the value of the Reason header is provided verbatim via the '_message' variable.	SUPPORTED	Set `sip.in.bye.headers = Reason` in the Call Manager configuration.

Req ID	Burke Draft Section	Requirement	Current Support	Notes
30	2.5	Termination of a session by issuing a BYE request due to encountering a <disconnect>, <exit>, the application running to completion, or due to unhandled errors within the application.	SUPPORTED	
31	3.1	Offer/answer mechanism of [RFC3264]	SUPPORTED	
32	3.2	Early media support with a 183 Session Progress response	PARTIALLY SUPPORTED	The media connections will be established when a 183 Session Progress message is sent, but early media will not be played. To perform this operation, set sip.sendalert = 2.
33	3.2	Matched SDP payload in final 200 OK response if SDP information sent in 183 Session Progress response	SUPPORTED	
34	3.3	Allow the media session to be modified via a re-INVITE.	SUPPORTED	
35	3.4	Support for G.711 mu-law and A-law support with payload type 0 and 8.	SUPPORTED	Set mpc.codec = pcmu pcma
36	3.4	Support for other audio codecs and payload formats.	SUPPORTED	Set mpc.codec as specified for various codecs.
37	3.4	Support for H.263 Baseline.	SUPPORTED	Set mpc.codec = h263
38	3.4	Support for AMR-NB audio.	SUPPORTED	Set mpc.codec = amr
39	3.4	Support for MPEG-4 video.	NOT SUPPORTED	This format is not currently supported.
40	3.4	Support for MPEG-4 AAC audio.	NOT SUPPORTED	This format is not currently supported..
41	3.4	Support for other video codecs and payload formats.	SUPPORTED	
42	3.5	Support for DTMF events [RFC2833].	SUPPORTED	

Req ID	Burke Draft Section	Requirement	Current Support	Notes
43	3.5	Other means for DTMF detections (ie, inband DTMF detections)	NOT SUPPORTED	Inband DTMF detection is not supported with this release. Future releases may have this feature.
44	4.2	Support encoding of the expr / namelist data in the message body of the BYE request sent from the VoiceXML Media Server as a result of encountering the <exit> or <disconnect> element.	SUPPORTED	
45	4.2	Including the expr / namelist data in response to BYE request.	NOT SUPPORTED	This feature is not currently supported.
46	4.2	Sending a 100 Trying provisional response to the BYE request as per [RFC4320].	NOT SUPPORTED	Since item 4.2.2 is not supported, the response time of the response will not slow enough to warrant this requirement.
47	4.2	__reason=exit in BYE request initiated due to the <exit /> tag.	SUPPORTED	
48	4.2	__exit=<value> when expr paramter is used in <exit /> tag. i.e. <exit expr="<value>">	SUPPORTED	
49	4.2	If namelist parameter is used, support encoding the values as per Requirement 44.	SUPPORTED	Set `sip.bye.contenttype = application/x-www-form-urlencoded;charset=utf-8`.
50	5.1	Support for third party call control mechanisms as displayed in section 2.6.2 of the Burke Draft.	SUPPORTED	
51	5.2	On receipt of the REFER request,a provision 100 Trying response, followed by a 202 Accepted response once the page has been fetched and parsed correctly, following by an output INVITE.	NOT SUPPORTED	

Req ID	Burke Draft Section	Requirement	Current Support	Notes
52	6.1	Blind transfer initiated using the REFER message [RFC3515] on the original SIP dialog, with Refer-To header containing the URI as specified via the ‘dest’ or ‘destexpr’ in the <transfer> tag.	SUPPORTED	Set `sip.defaultblindxfer = REFER`
53	6.1	If event accept, connection.disconnect.transfer event will be thrown.	SUPPORTED	
54	6.1	Use of [RFC4488] to suppress implicit subscription associated with REFER message.	NOT SUPPORTED	Not a requirement of this platform.
55	6.1	REFER response a non-2xx response mapping to the events listed in Requirement 56 to 59.	SUPPORTED	
56	6.1	404 Not Found = error.connection.baddestination	SUPPORTED	
57	6.1	405 Method Not Allowed = error.unsupported.transfer.blind	SUPPORTED	
58	6.1	503 Service Unavailable = error.connection.noresource	SUPPORTED	
59	6.1	(No reponse) = network_busy	SUPPORTED	
60	6.1	(Other 3xx/4xx/5xx/6xx) = unknown	SUPPORTED	
61	6.1	Appending aai / aaiexpr to Refer-To URI as a parameter named “aai”. Reserved characters are URL-encoded.	SUPPORTED	
62	6.2	Appending aai / aaiexpr to Request-URI as a parameter named “aai”. Reserved characters are URL-encoded.	SUPPORTED	
63	6.2	Support for playing the audio specified in the transferaudio attribute.	SUPPORTED	
64	6.2	Early media from Callee to Caller if transferaudio attribute is omitted.	PARTIALLY SUPPORTED	The “connectwhen” attribute will define when the two media streams should be bridged.
65	6.2	Setting the <transfer>’s form attribute to noanswer after issuing a CANCEL when the connecttimeout expires.	SUPPORTED	

Req ID	Burke Draft Section	Requirement	Current Support	Notes
66	6.2	INVITE response a non-2xx response mapping to the events listed in Requirement 67 to 73.	SUPPORTED	
67	6.2	404 Not Found = error.connection.baddestination	SUPPORTED	
68	6.2	405 Method Not Allowed = error.unsupported.transfer.bridge	SUPPORTED	
69	6.2	408 Request Timeout = noanswer	SUPPORTED	
70	6.2	486 Busy Here = busy (480 Temporarily Unavailable)	SUPPORTED	
71	6.2	503 Service Unavailable = error.connection.noresource	SUPPORTED	
72	6.2	(No reponse) = network_busy	SUPPORTED	
73	6.2	(Other 3xx/4xx/5xx/6xx) = unknown	NOT SUPPORTED	Refer to VoiceGenie 7.2 Media Platform System Reference Guide for specific platform behavior.
74	6.2	“Listening” for speech or DTMF hotword results in a near-end disconnect for User Agent 2.	SUPPORTED	
75	6.2	Call duration exeeds maximum duration specified in the maxtime attribute results in a near-end disconnect for User Agent 2.	SUPPORTED	
76	6.2	If User Agent 2 disconnects, the <transfer>’s form item variable receives the value far_end_disconnect and connection.disconnect.transfer is thrown.	SUPPORTED	
77	6.3	Same as Requirement 57 but substitute error.unsupported.transfer.consultation for error.unsupported.transfer.blind.	SUPPORTED	
78	6.3	Support consultation transfer similar to what is illustrated in the diagram in section 6.3 of the Burke Draft.	SUPPORTED	Set sip.defaultconsultxfer = `REFERJOIN`

Req ID	Burke Draft Section	Requirement	Current Support	Notes
79	6.3	Support throwing connection.disconnect.transfer event upon receipt of 200 OK to NOTIFY request.	SUPPORTED	
80	6.3	A non-2xx response to the NOTIFY request sets the associated VoiceXML input item variable to ‘unknown’.	NOT SUPPORTED	Refer to VoiceGenie 7.2 Media Platform System Reference Guide for specific platform behavior.

Revision Theory

Version	Date	Change Summary	Author/Editor
Draft	July 5th, 2004	Initial release	Ates Goral/Anthony Lam
0.9	August 3rd, 2004	Second Draft	David Lee
1.0	Nov 25th, 2004	Updates for 6.4.2	Anthony Lam
1.1	March 10th, 2005	Updates for VoiceGenie 7	Andrew Ho
1.2	April 13th, 2005	Final Revision for VoiceGenie 7	Andrew Ho
1.2.1	July 21st, 2005	PR 14031A	David Lee
1.3	August 23rd, 2005	Updated reference to signalvar	Andrew Ng
1.4	August 16th, 2006	Updated for Release 7.1	Andrew Ho /Anthony Lam
1.5	October 16th, 2007	Updated for Release 7.2	Andrew Ng
1.6	April, 3rd, 2009	Updated for ER **221656841**	Lin Chen
1.7	June 4th, 2009	Updated for ER**211055448,** ER173363157	Lin Chen
1.8	July 22nd, 2009	ER230976089	Lin Chen
1.9	September 16th, 2009	Updated for ER 226426477	Lin Chen
2.0	October 1st, 2009	Updated for ER 236969648	Lin Chen