

# VOICEGENIE®

## VoiceGenie 7 Media Platform Users' Guide

September 8<sup>th</sup>, 2006



VoiceGenie Technologies Inc.  
1120 Finch Ave. W. • Toronto, Ontario • M3J 3H7 • Canada  
T. +1.416.736.4151 • F. +1.416.736.1551 • [support@voicegenie.com](mailto:support@voicegenie.com)  
[www.voicegenie.com](http://www.voicegenie.com)

## VoiceGenie Contacts

**VoiceGenie Technologies Inc.**  
**1120 Finch Avenue West**  
**Toronto, Ontario**  
**Canada**  
**M3J 3H7**

**T. +1.416.736.4151**  
**F. +1.416.736.1551**  
**support@voicegenie.com**

**<http://www.voicegenie.com/index.html>**

## Proprietary / Copyright Information

This material contains proprietary and/or copyright information of VoiceGenie Technologies Inc. and may not be copied, used, or disclosed without the permission of VoiceGenie Technologies Inc.

© COPYRIGHT 2005 VoiceGenie Technologies Inc.

## Notice

Every effort was made to ensure that the information in this document was complete and accurate at the time of printing. However, this information is subject to change without notice.

Please note that VoiceGenie makes no warranties with respect to future releases. The information provided herein is for informational purposes only. VoiceGenie reserves the right to change product release schedules and/or features allocated to a product release at any time.

## Trademarks

All trademarks are the property of their respective owners. Where those designations appear in this document, and VoiceGenie is aware of a trademark claim, the designations have been printed in initial caps or all caps.

## Table of Contents

<b>Proprietary / Copyright Information</b>	<b>3</b>
<b>Notice</b>	<b>3</b>
<b>Trademarks</b>	<b>3</b>
<b>Table of Contents</b>	<b>4</b>
<b>1 Introduction</b>	<b>10</b>
<b>1.1 Terminology</b>	<b>10</b>
<b>1.2 Document Structure</b>	<b>10</b>
<b>1.3 Further Information</b>	<b>11</b>
1.3.1 VoiceGenie Documents	11
1.3.2 Third Party Documents	12
1.3.3 VoiceGenie Web Sites	12
1.3.4 Third Party Web Sites	12
1.3.5 Related Standards and Specifications	13
<b>2 VoiceGenie 7 System Overview</b>	<b>14</b>
<b>2.1 The Media Platform</b>	<b>14</b>
2.1.1 Call Manager	14
2.1.2 The VoiceXML Interpreter (VXMLi)	15
2.1.3 The Fetching Module	15
<b>2.2 Speech Resource Manager (SRM)</b>	<b>16</b>
<b>2.3 Operation, Administration and Management (OA&amp;M)</b>	<b>18</b>
<b>2.4 Other VoiceGenie Components</b>	<b>20</b>
2.4.1 SS7 Connector	20
2.4.2 SIP Proxy	21
2.4.3 CTI Connectors	23
2.4.3.1 Application Server Integration	23
2.4.3.2 Media Platform Integration	23
2.4.3.3 Architectural Overview – VoiceGenie and ICM	23
2.4.4 Call Control XML (CCXML) Platform	25
<b>3 Running Applications on the Media Platform</b>	<b>26</b>
<b>3.1 Application Provisioning (DNIS – URL Mapping)</b>	<b>26</b>
3.1.1 Choosing Defaults for Each Application	28
3.1.2 Multiple Default Settings	28
<b>3.2 VoiceXML Applications and the VoiceXML Interpreter (VXMLi)</b>	<b>29</b>
3.2.1 Communicating with the VoiceXML Interpreter via CLC	30
3.2.2 Limitations	30
<b>3.3 Conferencing</b>	<b>31</b>
3.3.1 Conferencing via SIP Interface	31
3.3.2 Conferencing via VXML Interface	32
<b>3.4 Application Count Service (Partition Definition)</b>	<b>32</b>

3.4.1	Activate PortCount CMAPI application	33
3.4.2	Associate PortCount application with partition name in DNIS-URL mapping	33
3.4.3	Define Partition	33
3.4.4	Alarms and Metrics	34
<b>3.5</b>	<b>HTTP/HTTPS Support</b>	<b>34</b>
<b>3.6</b>	<b>Caching in VoiceGenie 7 Media Platform</b>	<b>36</b>
3.6.1	Caching Architecture	37
3.6.2	Web Proxy caching	37
3.6.3	Caching policies	38
3.6.3.1	VoiceXML 2.1 Caching Algorithms	39
3.6.4	How to use maxage and maxstale attributes	40
3.6.5	Determination of an Expiry Time	41
3.6.6	Squid Caching Proxy	41
3.6.6.1	Squid Configuration	41
3.6.6.2	Using a Second-level Proxy Server	42
3.6.6.3	Squid Expiry Time Generation	42
3.6.6.4	Squid Specific Configuration Elements	43
3.6.6.5	Access.log field definitions	44
3.6.6.6	Refreshing an Object in the Cache	46
3.6.6.7	Purging an Object from the Cache	46
3.6.6.8	Clearing the entire Cache	47
<b>4</b>	<b>Network Interfaces</b>	<b>48</b>
<b>4.1</b>	<b>Voice over IP</b>	<b>48</b>
4.1.1	SIP	48
4.1.1.1	Standards	48
4.1.1.2	SIP Call Connection Mechanisms	49
4.1.1.3	Interoperability	49
4.1.1.4	SIP INFO support	49
4.1.1.4.1	Sending SIP INFO messages	49
4.1.1.4.2	Receiving SIP INFO messages	50
4.1.1.5	SIP customizable headers and parameters	52
4.1.1.6	Codec Negotiation	54
4.1.1.7	Enabling SIP TCP Support	54
4.1.1.8	Limitations	54
4.1.2	H.323	56
4.1.2.1	Standards	56
4.1.2.2	Architectures	56
4.1.2.3	Interoperability	56
4.1.2.4	Codec Negotiation	57
4.1.2.5	Limitations	57
4.1.3	RTP Support	57
4.1.3.1	Standards	57
4.1.3.2	General Usage	58
4.1.3.3	DTMF	58
<b>4.2</b>	<b>PSTN</b>	<b>59</b>
4.2.1	Standards	59
4.2.2	Brooktrout	59
4.2.2.1	Configuring Brooktrout Telephony Cards	59
4.2.2.1.1	ISDN	60

4.2.2.1.2	Robbed-bit	60
4.2.2.1.3	Clear channel	61
4.2.2.1.4	Brooktrout call control configuration file	61
4.2.2.2	Limitations	61
4.2.3	<i>Dialogic</i>	62
4.2.3.1	Configuring Dialogic Telephony Cards	62
4.2.3.1.1	JCT	62
4.2.3.1.2	ISDN	62
4.2.3.1.3	Robbed-bit and CAS	63
4.2.3.1.4	Analog	64
4.2.3.1.5	Clear Channel	64
4.2.3.1.6	DTMF-only ISDN Configuration (linux only)	64
4.2.3.2	Testing Telephone Network Interface Viability (on linux)	65
4.2.3.2.1	ISDN PRI only	65
4.2.3.2.2	Robbed-Bit T1	68
4.2.3.2.3	DMV on Linux	68
4.2.3.2.4	ISDN	69
4.2.3.2.5	Robbed-Bit and CAS	70
4.2.3.2.6	Hookflash Transfer	71
4.2.3.2.7	DMV on Windows	73
4.2.3.3	Configuring Media Platform to reload Dialogic firmware every time during startup (Linux)	73
4.2.3.4	Limitations	74
4.3	<b>VoIP/PSTN Hybrid</b>	<b>75</b>
4.3.1	<i>Overview</i>	75
4.3.2	<i>SIP Clear Channel</i>	76
4.3.2.1	Channel Identification	76
4.3.2.2	Session Description	76
4.3.2.2.1	Representation	77
4.3.2.2.2	Negotiation	77
4.3.2.3	Call Flow	78
4.3.2.3.1	Inbound (to VG Media Platform)	78
4.3.2.3.2	Outbound (from VG Media Platform)	80
4.4	<b>Multiple Line Managers</b>	<b>81</b>
5	<b>Call Control</b>	<b>83</b>
5.1	<b>Incoming Call</b>	<b>83</b>
5.2	<b>Outgoing Call</b>	<b>83</b>
5.2.1	<i>Dialing Rules</i>	84
5.2.1.1	Overview	84
5.2.1.2	How it Works	84
5.2.1.3	Creating Dialing Rules	84
5.2.1.4	Rule File Processing	87
5.2.1.5	Some operational notes	87
5.2.1.6	Dialing Rules and VoIP	88
5.2.2	<i>Hunt Group</i>	88
5.2.3	<i>Destination Format (PSTN)</i>	89
5.2.4	<i>Destination Format (VoIP)</i>	90
5.3	<b>Call Routing in VoIP</b>	<b>92</b>
5.3.1	<i>IP/PSTN gateway</i>	92
5.3.2	<i>SIP Proxies/Registrars</i>	93

5.3.3	H.323 Gatekeeper	93
<b>5.4</b>	<b>Glare Handling</b>	<b>94</b>
<b>6</b>	<b>Call Transfer</b>	<b>95</b>
6.1	General Information	95
6.2	Transfer Framework	96
6.2.1	Type and Method	96
6.2.2	Backward Compatibility with VoiceXML 2.0	98
6.3	PSTN Transfer	99
6.4	VoIP Transfer	100
6.5	Whisper Transfer	101
6.6	CTI Call Release	102
<b>7</b>	<b>Other Features</b>	<b>103</b>
7.1	Remote Dial	103
7.1.1	Overview	103
7.1.2	System Requirements	103
7.1.3	Socket API	103
7.1.4	Telnet/Socket Interface	103
7.1.5	How to Make a Call	105
7.1.6	Known Issues	107
7.2	Call Analysis	107
7.2.1	Limitations	108
7.3	Full Call Recording	108
7.3.1	Overview	108
7.3.2	Enabling/disabling Full Call Recording	108
7.3.3	Gain control	110
7.3.4	Known limitations	110
7.4	UUI Data	110
7.4.1	Platform configurations and application signal variables	111
7.4.2	Notes	111
<b>8</b>	<b>Operations</b>	<b>113</b>
8.1	Metrics and Logging/Billing	113
8.1.1	General Metrics	113
8.1.2	Metrics for Transfer	115
8.2	Alarms in Media Platform	116
8.2.1	Syslog	116
8.2.2	Alarm Browser	116
8.3	Health Status	119
8.3.1	Overview	119
8.3.1.1	CLC "health" command	119
8.3.1.2	SMC Status Monitor	120
8.3.2	Call Manager	121
8.3.2.1	PSTN Line Managers	122

8.3.2.2	VoIP Line Managers	123
8.3.3	VoiceXML Interpreter (VXMLi)	123
8.3.4	Fetching Module/Web Proxy (iproxy)	124
<b>8.4</b>	<b>Preventive Maintenance</b>	<b>126</b>



## Revision History

Version	Date	Change Summary	Author/Editor
Draft	July 5 <sup>th</sup> , 2004	Initial release	Ates Goral/Anthony Lam
0.9	August 3 <sup>rd</sup> , 2004	Second Draft	David Lee
1.0	Nov 25 <sup>th</sup> , 2004	Updates for 6.4.2	Anthony Lam
1.1	March 10 <sup>th</sup> , 2005	Updates for VoiceGenie 7	Andrew Ho
1.2	April 13 <sup>th</sup> , 2005	Final Revision for VoiceGenie 7	Andrew Ho
1.2.1	July 21 <sup>st</sup> , 2005	PR 14031A	David Lee
1.3	August 23 <sup>rd</sup> , 2005	Updated reference to signalvar	Andrew Ng
1.3.1	September 8 <sup>th</sup> , 2006	Updates for 7.0.11	Anthony Lam

## 1 Introduction

This is the Users' Guide for the VoiceGenie 7 Media Platform product. It provides an overview of the VoiceGenie 7 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 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 platform. The remainder of this document is structured as follows:

- **Introduction** – A brief introduction to the VoiceGenie Media Platform

- **VoiceGenie 7 System Overview** – Explains the overall VoiceGenie 7 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 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 Installation Guide](#) – Information regarding the installation and deployment of the VoiceGenie platform.
- [VoiceGenie 7 Media Platform System Reference Guide](#) – A guide that provides in-depth references to configuration information, metrics and alarm entries of the VoiceGenie 7 Media Platform.
- [VoiceGenie 7 OA&M Framework Users' 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 OA&M Framework – SMC Guide](#)  
[VoiceGenie 7 OA&M Framework – CLC Guide](#)  
[VoiceGenie 7 OA&M Framework – SNMP Guide](#)

- [VoiceGenie 7 SRM Users' Guide](#) and [VoiceGenie 7 SRM System Reference Guide](#) – The guides that covers all the information for the VoiceGenie 7 Speech Resource Management components.
- **VoiceGenie 7 Media Platform Release Notes** – Contains information regarding known issues, fixed problems, and behavioral information regarding the VoiceXML Gateway.
- [VoiceGenie 7 VoiceXML Language Reference Guide](#) -- provides a detailed overview of VoiceGenie's VoiceXML support, which includes VoiceXML 2.1, 2.0 and 1.0.

- [VoiceGenie 7 Application Migration Guide](#) – A guide that outlines the new features and changes in the VoiceGenie 7 release that may require changes to existing VoiceXML applications.

Additional documentation may be provided with particular system releases.

### 1.3.2 Third Party Documents

A number of third party documents provide information useful for operation of the VoiceGenie platform. These documents are available from VoiceGenie.

- **Squid Configuration Manual** – A detailed description of all information relevant to the configuration of the squid caching proxy.
- **Core JavaScript Reference** – The complete reference for the JavaScript (ECMAScript) engine embedded within the VoiceGenie platform.
- **Core JavaScript Users Guide** – The users guide for the JavaScript (ECMAScript) engine embedded within the VoiceGenie platform.
- **Dialogic System Release Bookshelf** – A set of documentation for the Dialogic telephony product.

In addition to these, each supported speech engine may include vendor provided documentation. These documentation sets are available from VoiceGenie.

### 1.3.3 VoiceGenie Web Sites

VoiceGenie 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 VoiceGenie customers, including documentation, tutorials, and other information.
- <http://www.voicegenie.com> - VoiceGenie's corporate web site, providing case studies, white papers, and general information regarding VoiceGenie.

### 1.3.4 Third Party Web Sites

There are a number of third party web resources that provide useful information regarding VoiceGenie 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 2388 Returning Values from Forms: multipart/form-data
- RFC 2396 Uniform Resource Identifiers (URI): Generic Syntax
- 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 3515 The SIP REFER Method
- RFC 3261 SIP: Session Initiation Protocol
- RFC 3264 An Offer/Answer Model with the Session Description Protocol (SDP)
- Some extensions from proposed IETF draft "A SIP Interface to VoiceXML Dialog Servers"
- Some extensions from proposed IETF draft "Basic Network Media Services with SIP" (<http://www.ietf.org/internet-drafts/draft-burger-sipping-netann-11.txt>)

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

## **2 VoiceGenie 7 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. This includes complete call control capability in the case of ISDN and CAS. In other call control environments, i.e. SS7, the Call Manager may interact with external entities to perform overall call management. In either case, 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, H.323 and PSTN) 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 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 Dialogic (providing access to the GlobalCall family of protocols), Brooktrout, SIP (using either RTP or Dialogic/Brooktrout clear channel for media transport), and H323.
- Media Transports -- multiple media transport mechanisms are available, capable of using Dialogic voice channels, Brooktrout voice channels, or RTP streams. A number of custom line managers have also been produced by VoiceGenie or third parties, to allow integration with proprietary call control mechanisms.

### **2.1.2 The VoiceXML Interpreter (VXMLi)**

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 Interpreter implements 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 specification
- 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)**

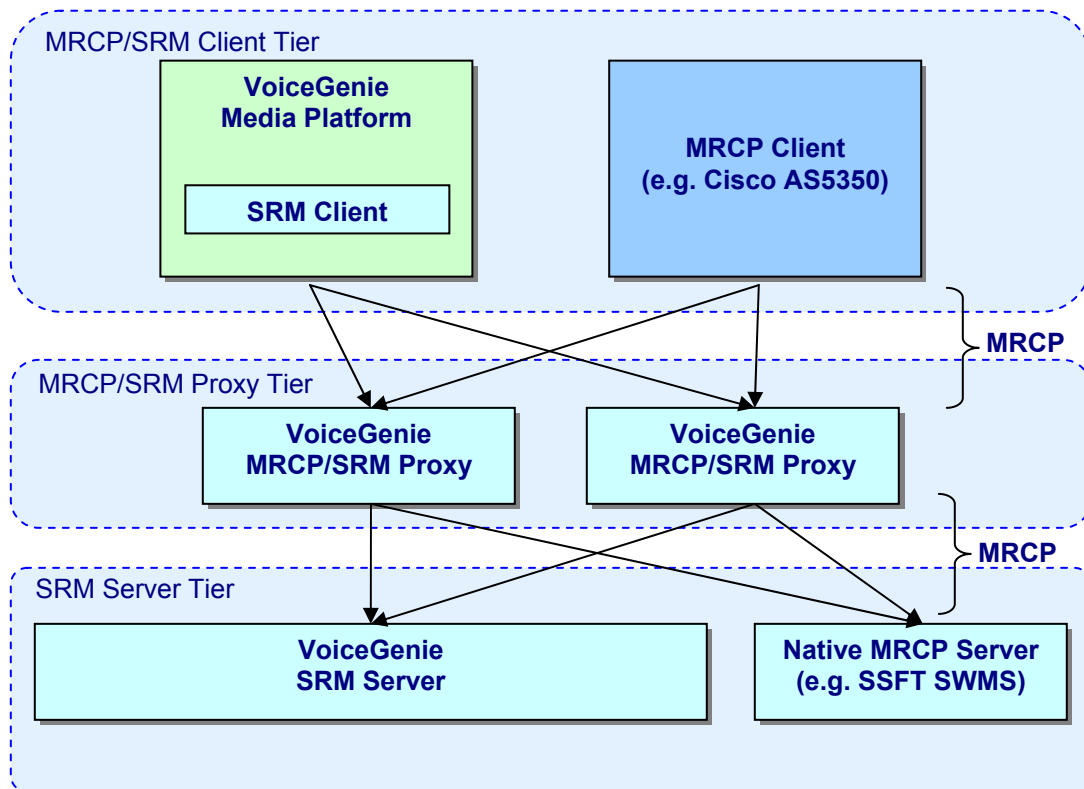
With modern day telephony applications, it is no longer sufficient to provide services to customers using only touch-tones input and pre-recorded audio. VoiceGenie Media Platform can accept speech as user input and can provide dynamically generated prompts by interacting with the Speech Resource Management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

The SRM is used to manage 3<sup>rd</sup> 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:



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

[VoiceGenie 7 Speech Resource Management Users' Guide](#)  
[VoiceGenie 7 Speech Resource Management System Reference Guide](#)  
[VoiceGenie 7 MRCP Proxy Users' Guide](#)  
[VoiceGenie 7 MRCP Proxy System Reference Guide](#)

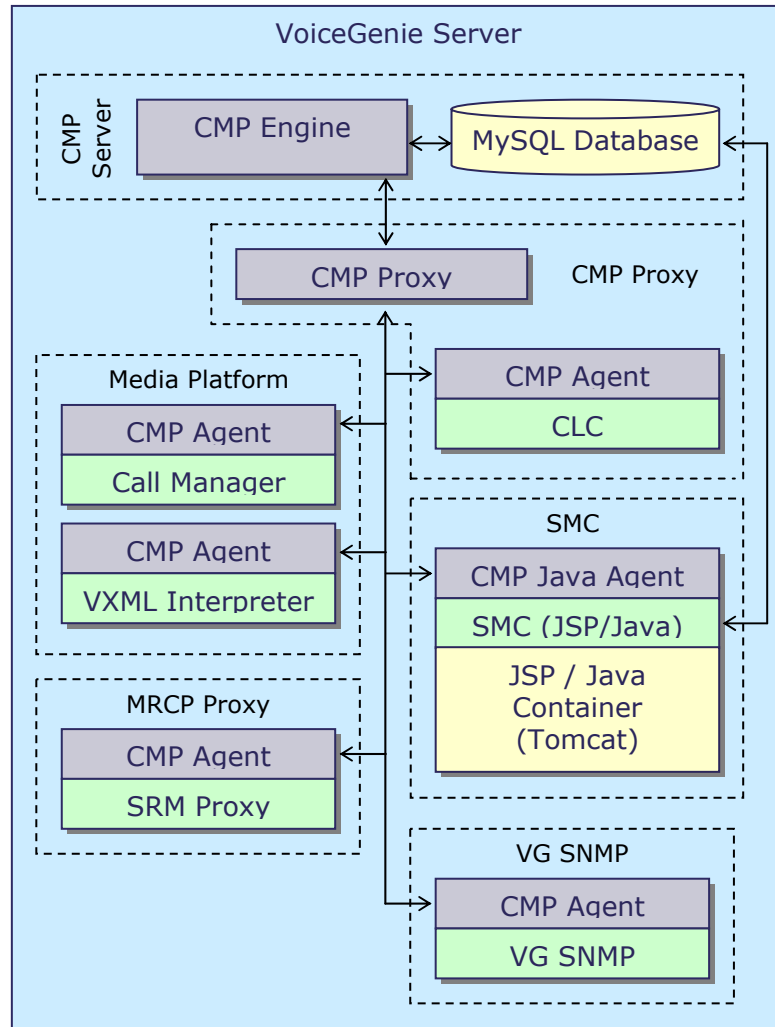
## 2.3 Operation, Administration and Management (OA&M)

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 OA&M Framework User Guide](#)  
[VoiceGenie 7 OA&M Framework – SMC Guide](#)  
[VoiceGenie 7 OA&M Framework – CLC Guide](#)  
[VoiceGenie 7 OA&M Framework – SNMP Guide](#)

## 2.4 Other VoiceGenie Components

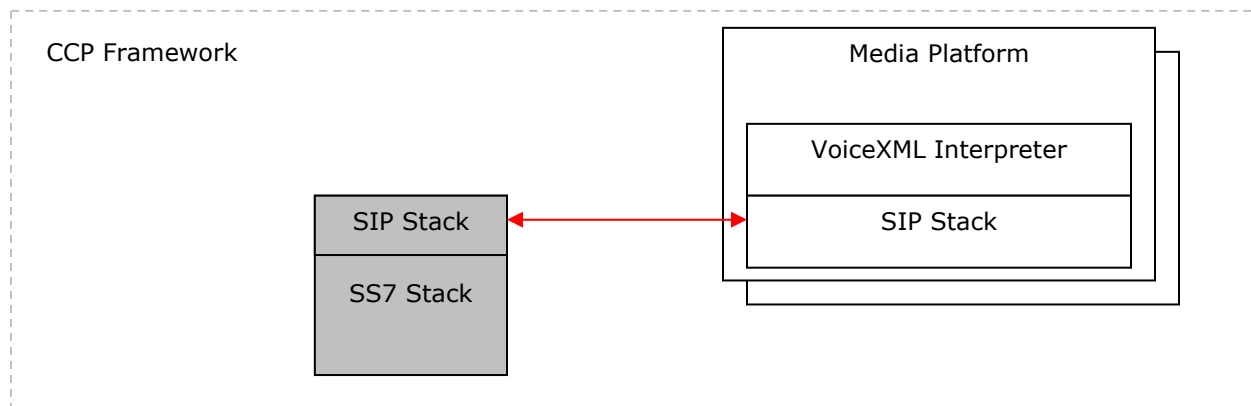
### 2.4.1 SS7 Connector

The SS7 signaling protocol is a widely deployed protocol that is used within service provider networks for the purposes of providing call control and enhanced services within the telecommunications network. Unlike other channel-associated signaling, SS7 signaling is independent of the actual voice path by which voice traverses the network.

VoiceGenie SS7 capabilities will be delivered through the VoiceGenie SS7 Connector framework, which supports ISUP call control, and allows different types of external signaling to be integrated with the media resources within the framework (Media Platform and other media resources). Specifically, VoiceGenie supports SS7/ISUP, and CCP also supports CTI integration and external SIP media gateway.

The Call Control Platform manages the presence of the media platforms and directs SS7 calls into the media platform. The CCP framework allows call control applications such as CCXML to manage the interaction of the call if it needs to modify the route of the call or have the call to access multiple media resources for the duration of the call.

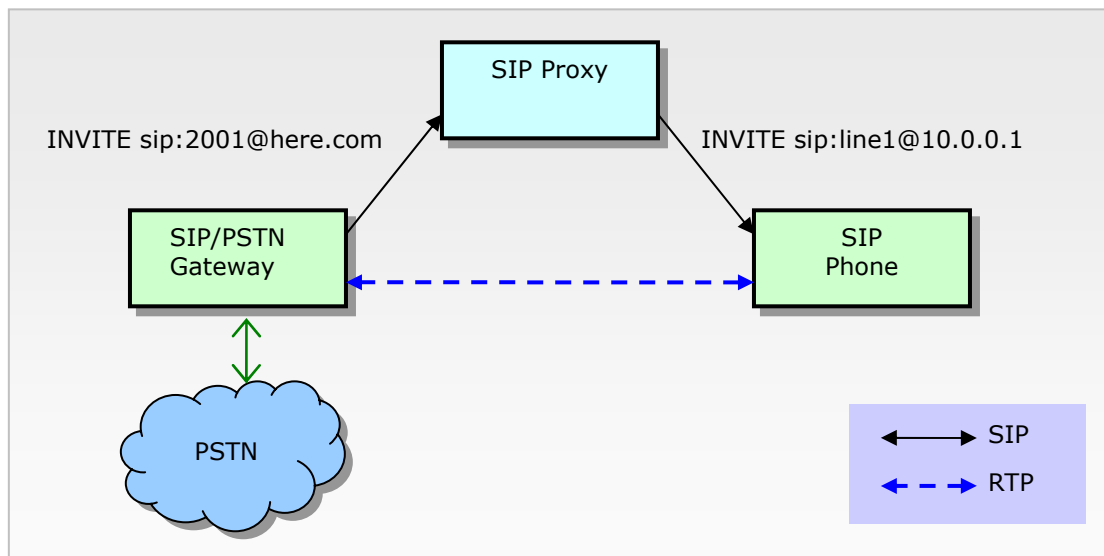
There are multiple components that are managed by the Call Control Platform; a simplified diagram below provides the components for a SS7 configuration:



The line in red above donates SIP communication path, and the gray components are Call Control Platform components. All of the components above contains a SIP stack and uses SIP as the standard communication protocol between all components within the SS7 Connector framework. Media Platform participates in the SS7 Connector framework as a media resource to receive incoming ISUP calls and execute VoiceXML or other applications.

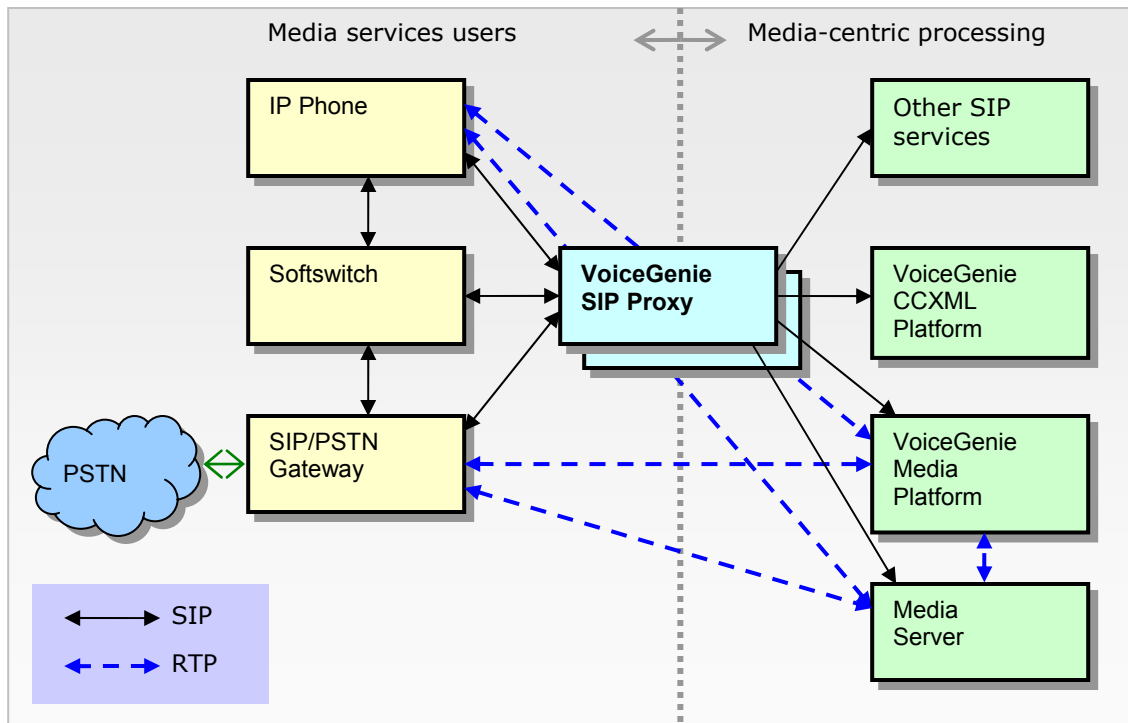
## 2.4.2 SIP Proxy

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 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.

### 2.4.3 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

#### 2.4.3.1 Application Server Integration

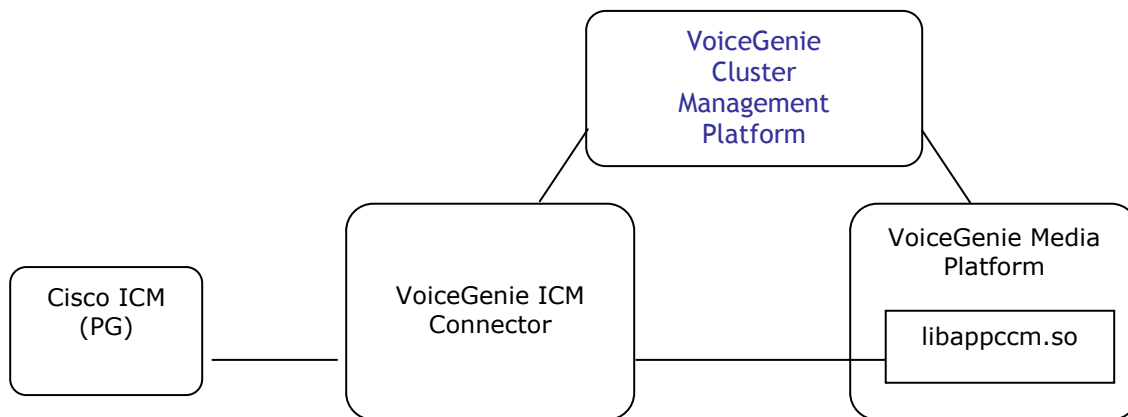
The VoiceGenie platform delivers call-related data to the application server as part of the initial page fetch related to a VoiceXML call. This data includes the information required to interface to an external CTI server – information such as port number, session identifier, and so on. This allows the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

#### 2.4.3.2 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.

#### 2.4.3.3 Architectural Overview – VoiceGenie and ICM

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



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).

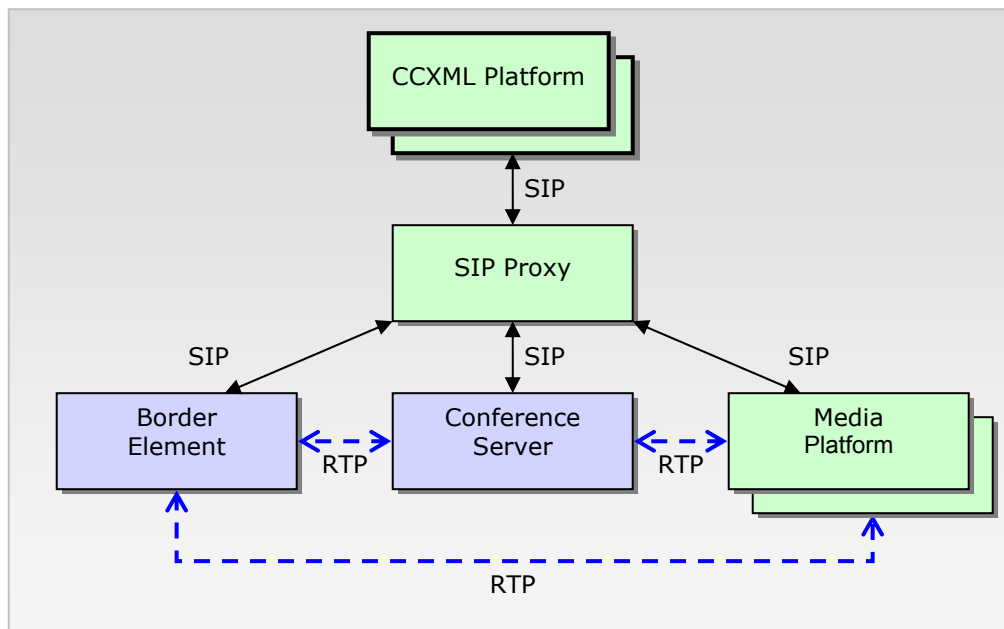


#### 2.4.4 Call Control XML (CCXML) Platform

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.

### 3 Running Applications on the Media Platform

The VoiceGenie 7 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 (VXML)** – This application module interfaces with VXML interpreter for the required VXML-defined application logic for a call. See the Section 3.2 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 ICM Connector Users' Guide](#)

[VoiceGenie 7 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.

**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 the Section 3.3 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 the Section 3.4 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:

One can specify Parameter Name = "FOO" and Value = "BAR", and add to session\_var parameters "session.connection.foo|FOO|0" under the VoiceXML Interpreter section via SMC. Then in the vxml page, a check such as the following can be performed:  
<value expr="session.connection.foo"/> == 'BAR'

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>
```

## System Management Console

Monitoring
Operations
Configuration
Administration

taco | Connected to CMP Proxy

**Concise Config View** ■

**OA&M Framework**

- CMP Server
- CMP Proxy
- Command Line Console
- System Mgmt Console

**Media Platform**

- Call Manager
- VoiceXML Interpreter
- Web Proxy

**DNIS - URL Mapping** ■

- Dialing Rules
- Hunt Groups
- Partition Definition
- Speech Resources

**Installation**

- Product Manager
- Config Profile Manager
- Deployment Manager
- Deployment History

■ Logout

### 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:**

**DNIS:**

**URL:**

**VoiceXML Defaults:**

VoiceXML

VoiceXML

ICM Call Control

Policy Client

Continuity Check

Conferencing

VoiceXML w/PortCount

**Parameter Name:**  **Value:**  Add

**Comment:**

Create

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:**

**DNIS:**

**URL:**

**VoiceXML Defaults:**

VoiceXML

XXXX

file:///voicegenie/mp/samples/helloaudio.xml

defaults.xml

**Parameter Name:**  **Value:**  Add

**Comment:**

Update
Delete
Select Target

ID: 1

### 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 Default 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 Interpreter (VXMLi)

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 the HTTP/HTTPS Support and Caching in VoiceGenie 7 Media Platform.

If the page can be successfully fetched, the VXMLi 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 VXMLi will first try to fetch and compile the page as specified by the parameter "ALTER\_URL" (which can be added in the "DNIS-URL Mapping" Configuration via SMC, please see the Application Provisioning (DNIS - URL Mapping) Section of this document for details), and if that fails also it will use the application page as specified by ALTERNATE\_INITIAL\_PAGE (configurable via SMC in the VoiceXML Interpreter Configuration section).

For details regarding the VoiceXML language, please visit:

<http://support.voicegenie.com>  
<http://developer.voicegenie.com>

And refer to the

[VoiceGenie 7 VoiceXML Language Reference Guide](#)

### 3.2.1 Communicating with the VoiceXML 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 VXMLI 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 VXMLI 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 a VXMLI 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] [instance] [recipient_address] [sender_address] [message]	Failed, invalid format command.
Can not deliver message	Failed.
Failed to send message	Failed.

### 3.2.2 Limitations

- When the available disk space has reached a point so low that the VXML 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

### 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 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/internet-drafts/draft-burger-sipping-netann-11.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 'WWW'. There is no need to set any URL; the SMC will set it automatically. (See [Application Provisioning](#) section for details on how to add DNIS-URL Mapping entries through SMC.)

Make a SIP call to the VoiceGenie platform with WWW 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:WWW@<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 'WWW\*'. There is no need to set any URL; the SMC will set it automatically. (See Application Provisioning (DNIS – URL Mapping) Section 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

**WWWconfid<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.

## 3.4 Application Count Service (Partition Definition)

Partition 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.

### 3.4.2 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.

### 3.4.3 Define Partition

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

Partition Name:	<input type="text"/>
Description:	<input type="text"/>
Port Count - Soft Limit:	<input type="text"/>
Port Count - Hard Limit:	<input type="text"/>
Treatment:	<input checked="" type="radio"/> Disconnect with Cause Code: <input type="text"/> <input type="radio"/> Run VoiceXML page at URI: <input type="text" value="http://"/>
Port Count - Minimum Allocation:	<input type="text"/>
Treatment:	<input checked="" type="radio"/> Disconnect with Cause Code: <input type="text"/> <input type="radio"/> Run VoiceXML page at URI: <input type="text" value="http://"/>
<input type="button" value="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. "Cause Code" is the ISDN cause code.

- "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 call affected are the calls coming into other partitions.

#### 3.4.4 Alarms and Metrics

Whenever a limit is exceeded, an alarm will be raised. Please refer to the [VoiceGenie 7 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|20051202564290013|N/A|N/A|N/A|i
nterrupt
```

### 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.

The platform 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. The following are the parameters relevant for SSL configuration:

SSL Configuration Settings	
<b>iproxy.ssl_cert</b>	The file name of your certificate. The default format is "PEM" and can be changed with the configuration parameter iproxy.ssl_cert_type
<b>iproxy.ssl_cert_type</b>	The format of the certificate.  Possible values: PEM, DER  Default: PEM
<b>iproxy.ssl_key</b>	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.
<b>iproxy.ssl_key_type</b>	The format of the private key.  Possible values: PEM, DER, ENG  Default: PEM
<b>iproxy.ssl_key_passwd</b>	The password required to use the iproxy.ssl_key.
<b>iproxy.ssl_engine</b>	The identifier for the crypto engine you want to use for your private key.
<b>iproxy.ssl_engine_default</b>	Sets the actual crypto engine as the default for (asymmetric) crypto operations.
<b>iproxy.ssl_version</b>	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
<b>iproxy.ssl_verify_peer</b>	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.

	<p>Possible values: 0, 1</p> <p>Default: 0</p>
<b>iproxy.ssl_ca_info</b>	The file name holding one or more certificates to verify the peer with.
<b>iproxy.ssl_ca_path</b>	The path holding multiple CA certificates to verify the peer with. The certificate directory must be prepared using the openssl c_rehash utility.
<b>iproxy.ssl_random_file</b>	The path to a file which is read from to seed the random engine for SSL.
<b>iproxy.ssl_verify_host</b>	<p>Should the Common name from the peer certificate in the SSL handshake be verified?</p> <p>Possible values: 0, 1, 2</p> <p>Default: 0</p>
<b>iproxy.ssl_cipher_list</b>	<p>The list of ciphers to use for the SSL connection. The list must be syntactically correct, it consists of one or more cipher strings separated 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: <a href="http://www.openssl.org/docs/apps/ciphers.html">http://www.openssl.org/docs/apps/ciphers.html</a>.</p> <p>Default: 0</p>

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
```

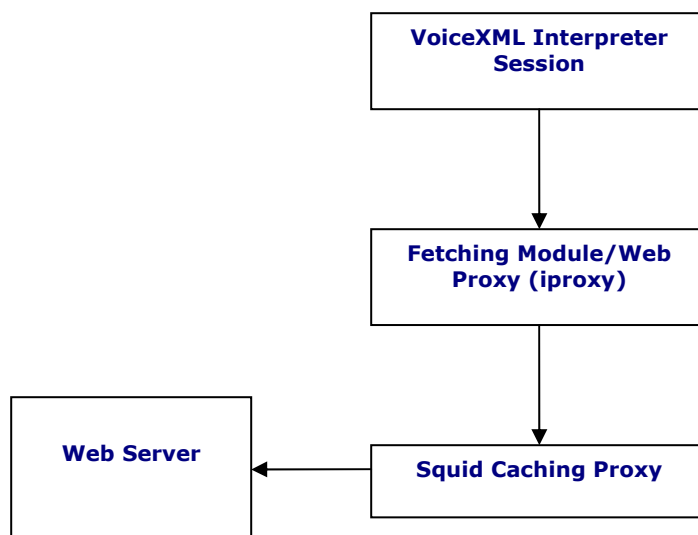
Additional caching proxy configuration is described in the following sections.

### 3.6 Caching in VoiceGenie 7 Media Platform

The VoiceXML interpreter context, like visual browsers, can use caching to improve performance in fetching documents and other resources; audio 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 the [Maxage Maxstale Table](#).

### 3.6.1 Caching Architecture

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



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 sections 9.3 to 9.6 below. 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 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

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.

### 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.

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.

### 3.6.3.1 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

#### 3.6.6.1 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 "=" 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>).

### 3.6.6.2 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.yourdomain.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).

### 3.6.6.3 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.

#### 3.6.6.4 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

#### 3.6.6.5 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.

<b>TCP_HIT</b>	A valid copy of the requested object was in the cache.
<b>TCP_MISS</b>	The requested object was not in the cache
<b>TCP_REFRESH_HIT</b>	The object was in the cache, but STALE. An If-Modified-Since request was made and a "304 Not Modified" reply was received.
<b>TCP_REF_FAIL_HIT</b>	The object was in the cache, but STALE. The request to validate the object failed, so the old (stale) object was returned.
<b>TCP_REFRESH_MISS</b>	The object was in the cache, but STALE. An If-Modified-Since request was made and the reply contained new content.
<b>TCP_CLIENT_REFRESH</b>	The client issued a request with the "no-cache" pragma.
<b>TCP_CLIENT_REFRESH_MISS</b>	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 forcing the 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
<b>TCP_IMS_HIT</b>	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.
<b>TCP_IMS_MISS</b>	The client issued an If-Modified-Since request for a stale object.
<b>TCP_NEGATIVE_HIT</b>	A previously failed request is satisfied from the cache, as the proxy believes it still be a problem.

<b>TCP_SWAPFAIL</b>	The object was believed to be in the cache, but could not be accessed.
<b>TCP_DENIED</b>	Access was denied for this request

#### **Inter-Cache Protocol Entries**

"UDP\_" refers to requests on the ICP port

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

#### **3.6.6.6 Refreshing an Object in 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 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>
```

#### **3.6.6.7 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>

#### **3.6.6.8 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

## 4 Network Interfaces

The physical telephony interconnect depends upon the selected telephony support.

E1/T1 can be connected using RJ-45 connectors and UTP cable. In the case of UTP cable, pin outs will depend upon the terminating device. When connected to the PSTN demarcation point, a straight-through cable will typically be used, while in the case of particular CSU/DSU or echo-cancellation connections, a crossover cable may be required.

In the case of VoIP a separate telephony network connection is not required. Call control and media traffic transit one of the LAN interfaces.

The Media Platform can be deployed in various telephony configurations, including:

**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.

**PSTN Only:** PSTN interface card(s) installed; both call control and media operations are performed with the PSTN interface card(s). Currently, Media Platform supports telephony cards manufactured by Brooktrout or Dialogic.

**VoIP/PSTN Hybrid:** PSTN interface card(s) installed; both VoIP and PSTN capabilities are partially or fully enabled.

### 4.1 Voice over IP

#### 4.1.1 SIP

##### 4.1.1.1 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)
- Some extensions from proposed IETF draft "A SIP Interface to VoiceXML Dialog Servers"
- Some extensions from proposed IETF draft "Basic Network Media Services with SIP"  
(<http://www.ietf.org/internet-drafts/draft-burger-sipping-netann-11.txt>)

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



#### 4.1.1.2 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.

#### 4.1.1.3 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

#### 4.1.1.4 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 Media Platform is capable of sending and receiving SIP INFO messages.

##### 4.1.1.4.1 Sending SIP INFO messages

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
```

#### 4.1.1.4.2 Receiving SIP INFO messages

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 vxmli (VXML Interpreter) session receives a message sent from another vxmli 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).

For more details about <receive>, please refer to  
[http://developer.voicegenie.com/voicexml2tagref.php?tag=cc\\_receive&display=calltags](http://developer.voicegenie.com/voicexml2tagref.php?tag=cc_receive&display=calltags)

#### 4.1.1.5 SIP customizable headers and parameters

The SIP version (using SIP2 line manager) of VoiceGenie 7 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 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.Invite.<headername>.<paramname> formats respectively.

Detailed information for the above two parameters, along with examples, can be found in the: [VoiceGenie 7 Media Platform System Reference Guide](#), under the "Customizable Headers and Parameters" section of the SIP line manager.

The VXML Interpreter configuration parameter **SESSION\_VARS** can be defined to provide the session variables to the VXML applications. For details please see the VoiceGenie 7 Media Platform System Reference Guide, under the VXML 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.paramA RequestURI.paramB To.paramC
```

VXML Interpreter:

```
session_vars=...|session.connection.protocol.sip.invite.from
|Sip.Invite.From|0|session.connection.protocol.sip.invite.requesturi.paramA|Sip.Invite.RequestURI.param
A|0|session.connection.protocol.sip.invite.requesturi.paramB|Sip.Invite.RequestURI.paramB|0|session.co
nnection.protocol.sip.invite.to.paramC|Sip.Invite.To.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.i nvite.from	<sip:VoiceGenie@205.150.90.207:5060> ;tag=9FB30200-B96C-01D0-5052- C114EBCA0416
session.connection.protocol.sip.i nvite.requesturi.paramA	valueA
session.connection.protocol.sip.i nvite.requesturi.paramB	valueB
session.connection.protocol.sip.i nvite.to.paramC	valueC

The VoiceGenie 7 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>, <call> 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 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 <call>, <transfer> involving two call legs and remdial calls):

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

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
```

#### 4.1.1.6 Codec Negotiation

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

`Sip.codec` allows a space-delimited list that specifies the list of codec that can be supported. For SIP calls, the VoiceGenie 7 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 `sip.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. Note that for SIP devices that support multiple codecs, this parameter must be set to 0 for full call recording to work properly.

#### 4.1.1.7 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

#### 4.1.1.8 Limitations

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

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

## **4.1.2 H.323**

### **4.1.2.1 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 the RTP Support Section to see the list of supported codecs.

### **4.1.2.2 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 Request (ARQ). Usage of ARQ for inbound and outbound calls can be configured independently.  
The alias(es) can be in any ASCII format:
  - i) PSTN phone number such as 4167360905
  - ii) 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.

### **4.1.2.3 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.



#### 4.1.2.4 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.

#### 4.1.2.5 Limitations

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

### 4.1.3 RTP Support

#### 4.1.3.1 Standards

The Real Time Protocol (RTP) is used by the VoiceGenie 7 Media Platform to send and receive media – which is typically an audio 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 SIP and H.323 regarding SIP and H.323.

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

- RFC 1889 Real-time Transport Protocol (RTP)
- RFC 2833 DTMF digit signalling using RTP

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)\*
- g729d (G.729D)\*
- g729e (G.729E)\*
- aurora (Aurora)\*\*
- tfci (proprietary text based)\*\*\*

\* Note 1: The current version of the VoiceGenie Media Platform does not support transcoding to/from any G.729 formats. G.729B packets are also not handled. Hence, for conferencing, full-call recording, and silence detection, G.729 formats are not supported. When G.729 is used, VoiceGenie Media Platform can only be configured as a DTMF-only system that plays raw G.729 format audio files (wave files are not supported). Note also that G.729D and G.729E are not supported for H.323 systems.

\*\* Note 2: 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.

\*\*\* Note 3: 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 support on platforms configured as SIP only.

#### 4.1.3.2 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 `rtp.portlow` and `rtp.porthigh` can be used to specify a particular range of UDP ports to be used for sending/receiving RTP streams.

#### 4.1.3.3 DTMF

The VoiceGenie 7 Media Platform can support sending of in-band DTMF audio. However, in-band DTMF detections are not supported on the VoiceGenie 7 Media Platform under VoIP configuration. Instead, the VoiceGenie 7 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.

## 4.2 PSTN

### 4.2.1 Standards

The Media Platform supports ISDN, robbed-bit and analog signaling by integrating with our third-party telephony board vendors. The type of signaling that can be performed depends on the capability of the telephony card used.

In general, supported robbed-bit protocols include:

- Trunk side E1/T1 (including Feature Group D, Feature Group B, Feature Group A, etc.)
- Line side E1/T1
- FXS/FXO T1
- Custom variants of all of these

Supported ISDN protocols include:

- AT&T 4ESS custom switch
- Lucent 5ESS custom switch
- National ISDN-2
- Northern Telecom custom switch (DMS series)
- EURO-ISDN (CTR4, ETSI)
- DASS2

Our ISDN signaling support is compliant with:

- Layer-2: Q.921 (ITU-T)
- Layer-3: Q.931 (ITU-T)
- Layer-2: ETS 300 402-2 (ETSI)
- Layer-3: ETS 300 403-1 (ETSI)

### 4.2.2 Brooktrout

The Media Platform currently has the following Brooktrout voice card offering:

- TR1000 family (1 or 2 spans, T-1 ISDN/E-1 ISDN/T-1 RB)

The current Brooktrout package distribution is SDK v3.2.

#### 4.2.2.1 Configuring Brooktrout Telephony Cards

Via SMC, please make sure following parameters are set correctly under Call Manager Configuration section:

```
brooktrout.minplaybuffer = 1500  
brooktrout.endpromptbuffer = 250
```

These parameters should already be set properly automatically during the installation/deployment process.

For different protocols Voicegenie has included working Brooktrout configuration files for several specific configurations. To obtain a list of these file please refer to `/usr/local/phoneweb/config/brooktrout_configs/ Index.txt`.

The user can set the protocol for the platform by setting brooktrout.board00.sigprotocol in callmgr.cfg file to one of T1-RB, E1-RB, T1-ISDN, E1-ISDN, T1-CC and E1-CC - where T1-RB and E1-RB define the T1 Robbed Bit and E1 Robbed Bit protocols respectively; T1-ISDN and E1-ISDN define the T1 ISDN and E1 ISDN protocols respectively; T1-CC and E1-CC define T1 Clear Channel and E1 Clear Channel protocol respectively.

This parameter must be set for each board.

#### 4.2.2.1.1 ISDN

The Media Platform provides support for different ISDN protocols, including:

- AT&T 4ESS custom switches
- AT&T 5ESS custom switches
- Ericsson MD-110 switches ( US & international )
- Nortel DMS-series switches
- Siemens EWSD switches (North American).

The default switch-type is 4ESS. This can be changed by modifying the brooktrout.board00.span0.switchtype parameter via SMC. The platform also supports different switch variants which indicates the variation of Q.931 used on this span. The switch variant can be set by using brooktrout.board00.span0.switchvariant parameter via SMC. This setting works in combination with the aforementioned switchtype parameter. Please refer to the table below for the supported combinations of these fields. The values to use for the brooktrout.board00.span0.switchtype and for brooktrout.board00.span0.switchvariant are shown in parenthesis.

The aforementioned parameters must be set for each board.

Although users may be able to set a different switch type and switch variant for each board, only configurations with the same switch type and switch variant on each board has been quality assured in VoiceGenie testing.

Table Switch Type & Variant Matrix\*

Switch Type	Supported Switch Variant
AT&T 4ESS{4ESS}	AT&T Custom{ATT}
AT&T 5ESS{5ESS}	AT&T Custom{ATT}
NT DMS-100{DMS100}	NT Custom{NT}
NT DMS-250{DMS250}	NT Custom{NT}
MD-110 (T1){MD110T1}	General ITU-T{GEN-ITU}
MD-110 (E1){MD110E1}	General ITU-T{GEN-ITU}, NET-5{NET5}
Siemens EWSD{EWSD}	NET-5{NET5}, GEN-ITU}, TS014{PRI}{TS014}
Unknown switch that conforms to ITU-T standards.{UNKNOWN}	NET-5{NET5}, General ITU-T{GEN-ITU}, TS014(PRI){TS014}

**\*For more details please refer to TR1000 manual on brooktrout support website (<http://www.brooktrout.com/support/tr1000>) for more details.**

#### 4.2.2.1.2 Robbed-bit

The Brooktrout Media Platform supports robbed-bit T1. Supported protocols are:

- E&M (AT&T PUB 43801) Wink.

- E&M (AT&T PUB 43801) Immediate

For Robbed Bit the brooktrout.board00.span0.switchtype and brooktrout.board00.span0.switchvariant parameters are ignored.

#### 4.2.2.1.3 Clear channel

From SMC, ensure that call manager configuration has the following settings:

- 1) callmgr.modules = Brkt SIP2
- 2) callmgr.devices = BrktDevice
- 3) callmgr.mediatransports = Brkt
- 4) callmgr.linemanager = LMBrkt LMSIP2
- 5) sip.transport=PSTN
- 6) brooktrout.board00.sigprotocol=T1-CC or E1-CC  
(Depending on if you want to configure for E1 clear channel with a board that supports 30 channels per span or a T1 clear channel that supports 24 channels per span. You have to do this for every board in the system.)

#### 4.2.2.1.4 Brooktrout call control configuration file

The call control configuration file (/usr/local/phoneweb/config/callctrl.cfg on linux and C:\VoiceGenie\mp\config\callctrl.cfg on Windows) contains general PCM configuration parameters for all telephony hardware units and static telephony connections to be formed for all modules. There are some parameters such as **clock\_mode**, **clock\_source**, **line\_coding**, **line\_type** etc. that may need to be changed. For more details, please refer to **Bfv API Reference Manual Volume 5 Appendices – Appendix A Configuration Files**. Please contact VoiceGenie or Brooktrout support for a copy of this document.

#### 4.2.2.2 Limitations

- If the machine running the Media Platform crashes because of critical errors, power failure or any other reasons, the existing calls may be interrupted by continuous noise, and may be dropped after the machine is restarted.

### 4.2.3 Dialogic

The Media Platform supports the Dialogic JCT and DMV series voice cards, including:

- D/120JCT-LS-U (12 port analog loop start)
- D/120JCT-LS-U-Euro (12 port analog loop start)
- D/480JCT-1T1-U (1 span ISDN, 1 span T-1 RB)
- D/480JCT-2T1-U (1 span T-1 ISDN or 2 spans T-1 RB)\*
- D/600JCT-1E1-120-U (1 span E-1 ISDN, 1 span E-1 CAS)
- DM/V480A-2T1-PCI (2 spans T-1 ISDN/RB)
- DM/V960A-4T1-PCI (4 spans T-1 ISDN/RB)
- DM/V1200A-4E1-PCI (4 spans E-1 ISDN/RB)
- DM/V1200BTEB (4 spans T-1/E-1 ISDN/RB)\*\*

Note \*: ISDN configuration has a resource limitation and hence only one span is allowed. A 2-span ISDN configuration is possible for 'playonly' configuration where no ASR and recording operations are required.

Note \*\*: B-series DM/V cards are supported on w2k configuration only.

The current Linux Dialogic package distribution is SR5.1SP1. The current W2K Dialogic package distribution is SR6.0. In both cases, GlobalCall Protocol library version 4.1 is used.

#### 4.2.3.1 Configuring Dialogic Telephony Cards

Media Platform supports two series of Dialogic card: JCT and DMV. This section describes the specific configuration steps necessary to be performed with these series of Dialogic cards.

##### 4.2.3.1.1 JCT

Via SMC, please make sure following parameters are set correctly:

```
dlgc.board = JCT
dlgc.READCALLBACK_BUFFER_SIZE = 8
```

These parameters should already be set properly during the installation/deployment process.

##### 4.2.3.1.2 ISDN

The Media Platform provides rich support for ISDN protocols, both European (ETSI) and North American, including:

- AT&T 4ESS custom switches
- Lucent 5ESS custom switches
- National ISDN-2
- Nortel DMS-series switches
- EURO-ISDN (CTR4, ETSI)
- DASS2

The default switch-type is 4ESS. This can be changed by modifying the file `/usr/dialogic/cfg/dialogic.cfg` to use the appropriate FirmwareFile and ParameterFile settings. Refer to the [VoiceGenie 7 System Reference Guide](#) for a detailed list of the valid entries for these parameters.

The ParameterFiles can be found in:

/usr/dialogic/data/<ISDNProtocolName>.prm

In general, the settings of these files do not need to be changed, and the default configuration should work accordingly with the switch.

#### 4.2.3.1.3 Robbed-bit and CAS

The Media Platform has deployed using robbed-bit and CAS signaling in many locations. Supported protocols include:

- Trunk side E1/T1 (including Feature Group D, Feature Group B, Feature Group A, etc.)
- Line side E1/T1
- FXS/FXO T1
- Custom variants of all of these

Support is provided for tone detection and generation to support various networks, including international connectivity.

When a robbed-bit configuration is selected, a default robbed bit configuration will be used. Some configuration parameters may need to be changed depending on your robbed-bit environment. The parameters below are some common parameters that may need to be changed, however the listed files do have many other parameters that advanced users may wish to modify:

##### **/usr/dialogic/cfg/us\_mf\_io.cdp**

\$4 : 0 (to change to the default 150ms wink duration)  
 \$6 : 1 (to enable faster connection timeout – in some environments, a higher number may be required)  
 \$9 : 0 (or higher, 0 will attempt to answer the call right away)  
 \$47 : 0 or 64 (0 for wink start, 64 for immediate start)

##### **/usr/dialogic/cfg/icapi.cfg**

\$14 : 1 (REQUIRED if using robbed-bit T1)

##### **/usr/dialogic/data/spandti.prm**

000C : 30 or 03 (30 if the switch requires that you wink by raising BOTH the A & B bits;  
 03 if the switch requires that you wink by lowering BOTH the A & B bits  
 – used commonly in line-side FXS environments)  
 000D : 23 (to define the bit transitions that will be used when utilizing hook-flash transfers)  
 0014 : 01 (add this line when you need to use ESF)\*  
 0020 : 01 (change to 01 when you need to use B8ZS signalling)\*

\* When changing parameters 0014 and 0020, BOTH should either be commented out or set to 01

##### **/usr/dialogic/cfg/pdk\_us\_ls\_fxs\_io.cdp**

Some switches may require customized bit-transition patterns. Contact VoiceGenie if the default set-up does not work.

##### **/usr/dialogic/cfg/pdk\_us\_mf\_io.cdp**

Some switches may require customized bit-transition patterns. Contact VoiceGenie if the default set-up does not work.

The above list of files addresses most standard configurations for T1 Robbed-Bit. Other files may exist and require changes depending on the country of installation and the connected switch.

#### 4.2.3.1.4 Analog

Analog configurations are generally used in lower density deployments and some development environments. Analog connections can come directly from a telephony company (CO) or from an analog extension of a PBX.

Normally, if the analog line is supplied by the CO, the default configurations will work. If the analog line is supplied by a PBX, there may be some parameter changes required as follows (generally referring to disconnection):

##### **/usr/dialogic/cfg/na\_an\_io.cdp**

```
$9 : 1          (to reduce the number of rings before the system will accept the call)
$30 : 1         (change to 0 if loop-drop is not available as a disconnection signal)
$31 : 0         (change to 1 if Dial-Tone is considered to be a disconnect)
$32 : 1         (change to 0 if Busy-Signal is not considered a disconnect)
```

#### 4.2.3.1.5 Clear Channel

The Media Platform can be configured to run in clear channel mode where the telephony cards are used for media signaling and processing only, and call control signaling is performed using SIP instead. This is currently supported for JCT cards only (not for DMV cards).

The system can have a robbed-bit or analog based configuration.

From SMC, ensure that call manager configuration has the following settings:

- 1) callmgr.modules = Dlgc SIP2
- 2) callmgr.devices = DlgcDevice
- 3) callmgr.mediatransports = MTDlgc
- 4) callmgr.linemanager = LMSIP2  
(Do not include LMDlgc as it would interfere with proper functioning of SIP/Clear channel)
- 5) sip.transport=PSTN
- 6) dlgc.MT\_FREECHANNEL\_MANAGE= FALSE
- 7) dlgc.protocol = null

#### 4.2.3.1.6 DTMF-only ISDN Configuration (linux only)

Due to CSP resource limitation, Dialogic D/480JCT-2T1 cards cannot support two speech-enabled ISDN T1 spans. The second span's vox (i.e. voice) devices are usually used as first span's recording processing resources with the following configuration (instead of dxxxB1C1 dxxxB1C2 ... dxxxB6C3):

```
dlgcv.voxdevlist.span01=dxxxB7C1 dxxxB7C2 dxxxB7C3 dxxxB7C4 dxxxB8C1 dxxxB8C2 dxxxB8C3
dxxxB8C4 dxxxB9C1 dxxxB9C2 dxxxB9C3 dxxxB9C4 dxxxB10C1 dxxxB10C2 dxxxB10C3 dxxxB10C4
dxxxB11C1 dxxxB11C2 dxxxB11C3 dxxxB11C4 dxxxB12C1 dxxxB12C2 dxxxB12C3
```



For non-speech-enabled applications, call manager can be provisioned as a DTMF-only system so that separate recording processing resources are not required. This allows full utilization of both T1 spans for dtmf only applications.

In order to deploy a DTMF-only system, in call manager configuration, the following parameter must be set properly via SMC:

```
dlgc.playonlymode=true
```

As well, the voxdevlist parameter should be configured to utilize both spans:

```
dlgc.voxdevlist.span01=dx B1C1 dx B1C2 dx B1C3 dx B1C4 dx B2C1 dx B2C2 dx B2C3  
dx B2C4 dx B3C1 dx B3C2 dx B3C3 dx B3C4 dx B4C1 dx B4C2 dx B4C3 dx B4C4  
dx B5C1 dx B5C2 dx B5C3 dx B5C4 dx B6C1 dx B6C2 dx B6C3
```

```
dlgc.voxdevlist.span02=dx B7C1 dx B7C2 dx B7C3 dx B7C4 dx B8C1 dx B8C2 dx B8C3  
dx B8C4 dx B9C1 dx B9C2 dx B9C3 dx B9C4 dx B10C1 dx B10C2 dx B10C3 dx B10C4  
dx B11C1 dx B11C2 dx B11C3 dx B11C4 dx B12C1 dx B12C2 dx B12C3
```

Finally, the board configuration section in /usr/dialogic/cfg/dialogic.cfg should be configured to use the appropriate protocol's firmware and parameter file. An example is shown below for 4ess ISDN configuration:

```
[Genload - PCI ID 1] /* T1/E1 PCI HD */  
FirmwareFile=spis4ess.fwl  
FirmwareFile2=spis4ess.fwl  
ParameterFile=4ess.prm  
ParameterFile2=4ess.prm
```

#### 4.2.3.2 Testing Telephone Network Interface Viability (on linux)

Currently, only ISDN PRI can be easily tested.

##### 4.2.3.2.1 ISDN PRI only

After the Dialogic Telephony Firmware is loaded to the installed Dialogic card, the card will try to synchronize with the network (the Central Office Switch or PBX). Once this synchronization happens, the LED on the back of the Dialogic card will change from red to green. Refer to Appendix G for troubleshooting information.

Once the cards are synchronized, use Dialogic's **isdiag** utility to confirm that the card is functioning properly:

Stop all VoiceGenie software (if running) as described in section 3.2, and reboot the system. You can check to see if any VoiceGenie software is running by typing:

```
ps -eaf | grep pwcallsmgr
```

If anything is returned, besides the **grep pwcallsmgr** then the VoiceGenie software is running and must be stopped.



**CAUTION:** If the isdiag utility is used while the VoiceGenie software is running, isdiag will not work properly. The VoiceGenie software must be stopped and the system rebooted prior to running isdiag.

Log onto the VoiceGenie server as 'root', and issue the following command:

**cd /usr/dialogic/bin**

Type the following to initialize the isdiag utility (for ISDN T1):

**./isdiag 1 1 t s v**



**NOTE:** The first 1 refers to board number 1, and the second 1 refers to channel number 1. Replace the 't' with an 'e' if you're testing E1. You can type **./isdiag** for a detailed description on how to use the utility.

The D channel status is displayed at the main menu screen which looks like:

```

xterm
MAIN MENU                                Version 5.06
LOS  OOF  RAI  AIS  CRC  R/E  DCH
0    0    0    0    0    0    0

1. set call parameter
2. request calling party number(ANI)
3. send maintenance request
4. display information
5. drop call
6. make call
7. play
8. set info
9. send message
10. start/stop/browse trace
11. restart and waitcall
12. shell to UNIX
13. change current channel
14. Hold/Retrieve Call
15. dnss supp service
Please enter choice: █
ESC exit
  
```

If any of the values at the top right portion of this display are non-zero, then it means there is some type of problem with the T1 connection. Contact VoiceGenie Technologies or contact your telephony service provider for more information. For more information about **isdiag**, refer to the Dialogic web site: <http://resource.intel.com/telecom/support/appnotes/isdn03.htm>

The program is menu-based and is able to make calls, wait for incoming calls, as well as set/browse various parameters for B channels and D channels. In particular, you can display all the B channel status by selecting option 4 (display information) and then option 3 (Display B channel information). For a system configured for T1, the screen should look like:

```
xterm
BOARD: 1
CHAN: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
STATUS: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
CHAN: 16 17 18 19 20 21 22 23
STATUS: 0 0 0 0 0 0 0 0
IN Service=0, Maintenance=1, OUT Service=2
```

The status 0 indicates the B channel is "in-service". A value of 1 means "maintenance" and 2 means "out-of-service".

Once you have confirmed that all the B channels and the D channel are in service, you are ready to place a call into the server. To do this, you should dial one of the telephone numbers given to you by your telephone company.

When the call is connected, you will not hear anything at the phone, but you will see the following information appear at the bottom of the screen on the main menu page of the isdiag utility.

RX(B1C1): Call Incoming, ANI = 4167360905, DNIS = 4167369731

RX(B1C1): received CONNECT ACK message from NETWORK

For the information above:

RX(B1C1) - The call was received on Board 1, Channel 1. ISDIAG will receive a call on any channel, but only on the first Board. Each Board consists of 23 channels.

ANI - This represents the telephone number from which the call was placed (Caller ID). ANI may not always be available.

DNIS - This normally represents the telephone number that is dialed. In some installations, the DNIS may be part of the dialed number, or it may even have no relation to the number dialed. Some carriers send 10 digits, while others send only 7 or fewer.

Place a few calls into the platform and try to identify a pattern for which channels are used when receiving calls. Some patterns are as follows:

Bottom-Up - always on channel 1, unless channel 1 is busy, then on channel 2

Top-Down - always on channel 23 (or channel 30 for E1), unless channel 23 is busy, then channel 22  
w/ Memory - either Bottom-Up or Top-Down, except the calls will rotate through all the channels before starting at the beginning

Least Used - calls received on the channel that has been idle for the longest time

Hang up the telephone, and call into the system with each telephone number that the carrier has provided. Make sure to check the DNIS for each of the numbers called. This information will be used when configuring the VoiceGenie server to accept calls.

To confirm the functionality of outbound dialing, choose option 1 (set call parameters) from the main menu, then choose option 11- change called party number. Enter the number to which you would like to dial.



**NOTE:** The number should be a direct number that does not require an extension. Once the call is initiated, the only way to confirm that the call is successfully placed is to hear the telephone ring, and see the CONNECT message in isdiag when the call is answered. You may need to try different combinations of the called number, depending on the configuration of the ISDN PRI T1/E1. For example, you may need to put a 9 in front of the number, or add the area code for local calls, or add a 1 for long-distance calls.

Return to the main menu, and choose option 6 (make call). You should see a SEND SETUP MESSAGE TO NETWORK message, and the telephone should ring. When the call is picked up, you should see a CONNECT message on the screen. If this fails, then you may need to change the dialed number as described in the NOTE above. If you see the connect message on the screen, then you have confirmed that outbound dialing is functioning properly. You will not hear any response when you take the call.

#### 4.2.3.2.2 Robbed-Bit T1

After the operating system is loaded, the LED on the back of the Dialogic card should change from red to green (refer to Appendix G or contact VoiceGenie if the LED never turns green). This means that the card is correctly identifying the D-channel, and the system should be able to accept calls. To further test the system, Dialogic's **isdiag** utility can be used as follows:

Stop all VoiceGenie software (if running) as described in section 3.2, and reboot the system. You can check to see if any VoiceGenie software is running by typing:

```
ps -eaf | grep pwcallegr
```

If anything is returned, besides the **grep pwcallegr** then the VoiceGenie software is running and must be stopped.

**CAUTION:** If the isdiag utility is used while the VoiceGenie software is running, isdiag will not work properly. The VoiceGenie software must be stopped and the system rebooted prior to running isdiag.

#### 4.2.3.2.3 DMV on Linux

Under the Call Manager Configuration section via SMC, following entries must be set properly:

```
dlgc.board = DM3  
dlgc.READCALLBACK_BUFFER_SIZE = 4
```

These parameters should be set properly by the installation rpm. Please consult with VoiceGenie support if a manual update is required.

The main Dialogic configuration file for DMV card(s) is /usr/dialogic/cfg/pyramid.scd. It configures the types of protocol and switch support.

Note that a parameter must be manually changed in /usr/dialogic/data/ml2\_dsa\_4ess.config and ml2\_qsa\_4ess.config, to enable outbound call analysis.

Variant CallProgress y ! y=Allow call progress, n=disallow

Should be replaced with:

Variant CallProgress n ! y=Allow call progress, n=disallow

#### 4.2.3.2.4 ISDN

Similar to JCT offering, the Media Platform provides support for the following ISDN protocols:

- AT&T 4ESS custom switches
- Lucent 5ESS custom switches
- National ISDN-2
- Nortel DMS-series switches
- EURO-ISDN (CTR4, ETSI)
- DASS2

The default rpm installation configures the 4ESS switch by default. If another switch is required, the following modifications to the Dialogic configuration will be required:

- Under /usr/dialogic/cfg directory, edit pyramid.scd file to use the FCD and PCD files for the switch
- Under /usr/dialogic/data directory, update the corresponding .config file for the switch configuration
- Under /usr/dialogic/data directory, run "fcdgen <corresponding switch>.config" to generate fcd file

Here is an example of pyramid.scd file:

```
; SCD File
; replaced by VoiceGenie DM3 installation script
NumStreams      : 4000
NumBindHandles  : 4000
[Board 1] {
PCDName         : ml2_qsa_4ess.pcd
FCDName         : ml2_qsa_4ess.fcd
PCMEncoding     : MULAW
SlotNumber      : 0
BusType         : PCI
LogFile         : board1.log
DisplayConfig   : YES
TimeToSendMsg   : 30
MasterStatus    : PRIMARY
TDMBusType      : H100
BusCR           : 8
Group1CR        : 8
```

```

Group2CR      : 8
Group3CR      : 8
Group4CR      : 8
PrimaryLines   : CT_A
DeriveClockFrom : NETREF_1
NetRef1       : YES
NetRef1From    : 1
NetRef1CR      : 1
}

```

In above example, 4ESS ISDN is used. If the switch type or protocol type is changed, the PCDName and FCDName should be replaced with other values accordingly. Please refer to the following document for details:

[http://resource.intel.com/telecom/support/releases/unix51/linux51/SR5.1\\_Linux/Onldoc/html\\_files/dm3conrf/1272-003.htm](http://resource.intel.com/telecom/support/releases/unix51/linux51/SR5.1_Linux/Onldoc/html_files/dm3conrf/1272-003.htm)

For ISDN, there are two parameters necessary to be changed in \*.config file. For example, for a ISDN 4ESS switch, the following entries should be configured in /usr/dialogic/data/ml2\_qsa\_4ess.config file:

```

Setparm=0x1311,2    ! InitialChanState. 1=InService; 2=OutOfService
                    ! Default is 1
SetParm=0x1312,0    ! Parm DisableBlock set to (1), the Maintenance Message is
                    ! NOT sent out on a B-channel status change.
                    ! Parm DisableBlock set to (0), the Maintenance Message is
                    ! Sent out on a B-channel status change.

```

The installation rpm normally takes care of these settings unless a non-4ESS switch type is provisioned.

#### 4.2.3.2.5 Robbed-Bit and CAS

For DMV RB/CAS, Dialogic Global Call Protocol 4.1 is required. Only a PDK library can be used for any DMV RB or CAS protocols (JCT supports PDK or ICAPI library).

Similarly for DMV ISDN, under /usr/dialogic/data directory, edit the file pyramid.scd to use the appropriate FCD and PCD files for the switch and protocols:

- For T1 RB, use ml2\_qsa\_cas.pcd and ml2\_qsa\_cas.fcd
- For E1 CAS, use ml2\_qsa\_r2mf.pcd and ml2\_qsa\_r2mf.fcd

Next, a robbed-bit protocol and its corresponding cdp configuration file under /usr/dialogic/cfg directory must be selected. pdk\_us\_mf\_io is a common RB protocol. It is possible to configure wink-start or immediate-start configuration with pdk\_us\_mf\_io.cdp:

```

/* Generate a Seizure Acknowledgement CAS_WINK after */
/* receiving a CAS_SEIZE. Set this to 1 for WINK start */
/* or to 0 for immediate start protocols */
All BOOLEAN_t CDP_IN_WinkStart      = 1
All BOOLEAN_t CDP_OUT_WinkStart     = 1

```

Depending on the switch configuration, the following parameters should be set accordingly:

```

/* Remove the DNIS prefix digit before returning digit string */
/* to application. */

```

```

All BOOLEAN_t CDP_IN_DNIS_KP_Needed      = 0
All BOOLEAN_t CDP_OUT_DNIS_KP_Needed     = 0

/* If this is set then it generate a CAS_WINK immediately after */
/* receiving the ANI digits. */
All BOOLEAN_t CDP_IN_ANI_WINK_Needed     = 0
All BOOLEAN_t CDP_OUT_ANI_WINK_Needed    = 0

/* Remove the ANI prefix digit before returning digit string */
/* to application. */
All BOOLEAN_t CDP_IN_ANI_KP_Needed      = 0
All BOOLEAN_t CDP_OUT_ANI_KP_Needed     = 0

/* enable features */
/* For DM3, one must remove "feature_ANI" from SYS_FEATURES */
All CHARSTRING_t SYS_FEATURES =
"feature_outbound,feature_inbound,feature_DNIS,feature_transfer,feature_hold"

```

For other protocols (including E1 CAS), please use default \*.cdp files. Please refer to Dialogic documentation and consult your PSTN service provider before making any adjustments. It is not recommended that inexperienced users modify these parameters.

As well, the file /usr/dialogic/cfg/pdk.cfg has to be created manually. This file should include one line:  
board <board-id> pcdfile <pcd-file> fcdfile <fcd-file> variant <cdp-file>

Where:

- the board id must be consistent to the real board ID. If required, please check the ID by typing the "listboards" command.
- pcdfile name and fcdfile name must be consistent to the file names in pyramid.scd.
- variant specifies which cdp (Country Dependence Protocol) file is used (different country and different switch need different cdp files). Please check Dialogic documentation for details of using non-default protocols.

For example, the file should contain the following line for a typical RB T-1 configuration:  
board 1 pcdfile ml2\_qsa\_cas.pcd fcdfile ml2\_qsa\_cas.fcd variant pdk\_us\_mf\_io.cdp

Finally, note that in callmgr.cfg, the following parameters should be set as:  
pstn.use\_set\_chan\_state=false  
pstn.hfflashdur = 0

Otherwise the Call Manager cannot be started up properly.

#### 4.2.3.2.6 Hookflash Transfer

With DMV, the hookflash transfer implementation is based on the use of the gc\_BlindTransfer API function provided by Dialogic. Hence, there are several parameters that should be configured at the Dialogic level inside the corresponding cdp file for customizing the hookflash transfer behavior:

**/\* If set to 0, no hookflash is sent by the protocol when supervised and blind  
\* transfer is requested \*/**

**All INTEGER\_t CDP\_HOOKFLASH\_ON\_XFER = 1**

If this parameter is 1, hookflash transfer will be attempted when gc\_BlindTransfer is called. Set this to 1.

**/\* Time between blind transfer hookflash and dialing \*/**

**ALL INTEGER\_t CDP\_BLIND\_XFER\_PRE\_TIME = 2000**

This is the amount of time between the flash character and the digit dialing. For existing JCT users, this maps to the time duration specified in **pstn.hfprefix** parameter in callmgr.cfg. The time duration in callmgr.cfg file used to be specified by the number of ',' characters in the pstn.hfprefix parameter where one ',' implies one second pause time. This parameter in the cdp file allows more convenient configuration of a time period.

**/\* Time between blind transfer dialing and hangup \*/**

**ALL INTEGER\_t CDP\_BLIND\_XFER\_POST\_TIME = 0**

This is the amount of time the protocol waits after performing the dial until it drops both calls. This corresponds to the **pstn.hfdisctimer** in callmgr.cfg. The default value is 0.

**/\* Transfer hookflash CAS definition \*/**

**ALL CAS\_SIGNAL\_PULSE\_t CAS\_XFER\_HOOKFLASH = 11xx,00xx,50,50,0,80,450,500,550**

This defines the hookflash as well as the duration of the hookflash for detecting and performing hookflash.

The diagram below corresponds to the hookflash definition defined above.

Bits	A	B	C	D
Initial State	1	1	x	x
Before HKF				

500 ms

After HKF	0	0	x	x

According to the definition of the parameter the signalling bits will be transitioned from 11xx to 00xx for a duration of 500 ms. 'x' signifies that a bit will not be affected. So A and B bits will change from 1 to 0 for 500 ms for a hookflash.

For existing JCT users, this maps to the configuration parameter **pstn.hfflashdur** in callmgr.cfg. One should set the value of this to 500 ms (which is the default for pstn.hfflashdur).

**/\* Set to 0 to disable waiting for dialtone during blind transfer \*/**

**ALL INTEGER\_t CDP\_BLIND\_XFER\_DIALTONE\_TIMEOUT = 5000**

This defines the maximum time-out to wait for dial tone during a blind transfer. The default value is 5000 (5 seconds).

**/\* Set to 0 to disable waiting for dialtone during supervised transfer \*/**

**ALL INTEGER\_t CDP\_SETUP\_XFER\_DIALTONE\_TIMEOUT = 5000**

This defines the maximum time-out to wait for dial tone during a supervised transfer. The default value is 5000 (5 seconds).



#### 4.2.3.2.7 DMV on Windows

On the Windows platform, a GUI interface, DCM (Intel Dialogic Product Configuration Manager) is available for Dialogic configuration. The DCM provides a GUI for choosing the protocol. Once the appropriate protocol is selected, the fcd and pcd files will be generated by the tool automatically.

For A series cards, the configuration is straightforward. Select the corresponding fcd and pcd files based on the protocol (or the switch type). For A series cards, usually media load 2 is used, so the fcd file name looks like:

```
ml2_qsa_<protocol>.fcd -- For quad span A-series DMV card
ml2_dsa_<protocol>.fcd -- For dual span A-series DMV card
```

For B series cards, the board can be configured through firmware to handle only T1. The protocols running on the each T1 trunk may also be individually specified. Hence, the trunk type is required to be selected accordingly from "Trunk Configuration" field. For B series cards, usually universal media load is used, so the fcd file name looks like:

```
gul1_qsb_<protocol>.fcd -- For quad span B-series DMV card
gul1_dsb_<protocol>.fcd -- For dual span B-series DMV card
```

However, DCM does not take care of all parameters specifically for the VoiceGenie platform, so some manual changes are still required. Please see the details in the 4.2.3.2.3. In addition, the following parameter is required to be updated on Windows:

If any change is made in \*.config file, then it would be required to re-run the fcdgen tool to generate the new fcd file (see DMV Linux ISDN for fcdgen tool execution).

#### 4.2.3.3 Configuring Media Platform to reload Dialogic firmware every time during startup (Linux)

By default, the Media Platform does not reload the dialogic firmware except for the pdk\_us\_ls\_fxs\_io protocol. The purpose for reloading is to ensure the connection with the switch is reset properly. However, this will significantly increases the startup time for the Media Platform (for at least 3 minutes).

The starting script located in /usr/local/phoneweb/bin/cmgr\_start\_script has the following section:

```
# -----
grep "pdk_us_ls_fxs_io" /usr/local/phoneweb/config/callmgr.cfg >> /dev/null 2>&1
if [ $? -eq 0 ]
then
  echo "This system is using Dialogic pdk_us_ls_fxs_io configuration,"
  echo "the Dialogic boards must be restarted before Voicegenie. This will"
  echo "take about 3 minute, please do not interrupt this process."
  echo ""
  echo "Stopping Dialogic boards (This takes about 30 seconds) ..."
  /usr/bin/sudo /etc/init.d/ct_intel stop >> /tmp/vg-dlgc-startstop 2>&1 || /bin/true
  echo "Re-starting Dialogic boards (This takes about 120 seconds) ..."
  /usr/bin/sudo /etc/init.d/ct_intel start >> /tmp/vg-dlgc-startstop 2>&1 || /bin/true
  echo "Proceeding to start VoiceGenie software ..."
fi
```

To make the Media Platform reload dialogic firmware automatically, comment out the lines:

```
grep "pdk_us_ls_fxs_io" /usr/local/phoneweb/config/callmgr.cfg >> /dev/null 2>&1
if [ $? -eq 0 ]
then
and
fi
```

#### 4.2.3.4 Limitations

Here is a list of known limitations associated with the Dialogic line manager

- Audio control may not function on the last 2 seconds of a prompt for VoiceGenie Media Platforms using Dialogic JCT cards
- Audio control may not function on the last 3 seconds of a prompt for VoiceGenie Media Platforms using Dialogic DMV cards
- After a power failure, the Dialogic firmware may need to be manually reloaded for the Media Platform to restart properly. Here are the steps to reload the firmware:
  - Stop the platform via CLC or SMC
  - Switch to root/administrator access
  - On Linux:  
Run the script `"/etc/init.d/ct_intel stop"`. Wait for a minute or so.  
Run the script `"/etc/init.d/ct_intel start"`

On Windows:

Click on the "Start" button, then "Intel Dialogic System Release 6.0 PCI", then Launch the "Configuration Manger – DCM".

Click on the stop button (i.e. The red square). Wait for a minute or so.

Click on the start button (i.e. The triangle on the furthest left).

- Start the platform via CLC or SMC

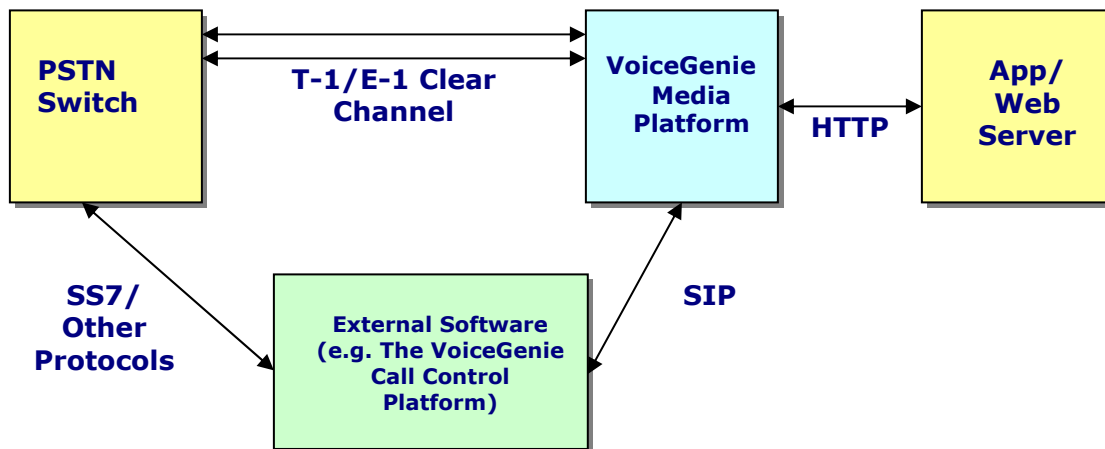
## 4.3 VoIP/PSTN Hybrid

### 4.3.1 Overview

In PSTN environments, call signaling which controls the starting and termination of a call may be governed by a variety of means. With simple technologies, such as analog, robbed-bit T-1, call signaling is delivered essentially in-band with the actual media, whereas with ISDN and SS7, separate control channels are used to deliver call signaling information.

In some situations, including multiple environments that ultimately support SS7 for call signaling, it is desirable to separate call signaling from the VoiceGenie Media Platform, which is normally the entity that exchanges call signaling information with the network. In such cases, VoiceGenie proposes that where possible, a SIP-based integration be used to exchange signaling, while retaining the use of PSTN technologies for media channels. Thus, instead of negotiating the use of RTP for media transfer to/from the VoiceGenie software, SIP is used to negotiate the use of particular PSTN clear channel DS0s for media transfer, in which the PSTN channel is completely devoid of any kind of call signaling information.

The typical simplified architecture of a system deployed based on the signaling method described in this document is as follows:



In fact, the VoiceGenie Call Control Platform (CCP) can act as the external software to integrate with SS7 network. In those deployments, the Media Platform is configured to execute under SIP Clear Channel mode. For a complete end-to-end SS7 solution from VoiceGenie, please also refer to the Call Control Platform documentation.

A typical SIP Clear Channel Media Platform would require both SIP and Dialogic (or Brooktrout) shared libraries be loaded. Usually SIP line manager would be configured to work with the PSTN (Dialogic or Brooktrout) media transport module. Currently, SIP Clear Channel operation is supported for:

- Brooktrout using T-1 or E-1 (for Configuration details, see Brooktrout Clear channel Section)
- Dialogic with JCT using T-1, E-1 or analog (for Configuration details, see Dialogic Clear Channel Section)

### **4.3.2 SIP Clear Channel**

As introduced earlier, SIP clear channel operation is a configuration where a SIP-based integration is used to exchange call control signaling, while retaining the use of PSTN technologies for media channels. Thus, instead of negotiating the use of RTP for media transfer to/from VoiceGenie Media Platform, SIP will be used to negotiate the use of particular PSTN clear channel DS0s for media transfer, in which the PSTN channel is completely devoid of any kind of call signaling information.

This section provides further technical information that would be required to integrate with the Media Platform for this configuration.

#### **4.3.2.1 Channel Identification**

In order for a PSTN channel to be negotiated between VoiceGenie Media Platform and the external software that interfaces using SIP, it is necessary that a common scheme for channel identification be used, such that both sides can agree on a particular PSTN channel to be used for a call. In SS7 situations, a separate circuit identification code (CIC) might be used for each trunk, but in the general case, there is no well-defined scheme for identifying a particular DS0, and the PSTN switch will typically use one scheme while the VoiceGenie Media Platform uses another. It is the external software that is deemed to be responsible for performing any required translation between the scheme used on the PSTN switch side, and on the VoiceGenie side. The channel identification scheme described here refers to the scheme used on the SIP leg between the VG Media Platform and the external software. This approach is considered to be particularly appropriate because the external software might talk to multiple VoiceGenie Media Platforms, thus requiring a level of translation that is not possible within a single VoiceGenie Media Platform.

The channel identification scheme used by VoiceGenie Media Platform is as follows:

1. All physical T-1/E-1 trunks will be assigned an integer identifier. This integer identifier will be between 1 and N, where N is the number of physical trunks that are connected to that server.
2. Each logical PSTN channel (DS0) within a trunk will be assigned an integer identifier, which is unique only within that trunk. This integer identifier will be in the range 1:M, where M is the total number of logical channels carried by a particular physical trunk. For T-1, M=24, and for E-1, M=30.
3. The combination of the physical trunk identifier and logical PSTN channel identifier are required to uniquely identify a PSTN channel within the system. In text form, this will be represented as "X:Y" where X is the identifier of the physical trunk, and Y is the identifier of the logical PSTN channel within that trunk. For instance, "2:17" represents trunk # 2, channel # 17.
4. From a physical standpoint, the trunk identifiers will correspond directly to board numbers. That is, the VoiceGenie software will not be capable of designating a particular physical trunk as trunk # 1 and another as trunk # 2; rather, the hardware will dictate which physical trunk is # 1, # 2, etc.

#### **4.3.2.2 Session Description**

Currently, Session Description Protocol (SDP, as defined in RFC 2327) is used to describe a proposed session. SDP is used to describe a set of capabilities that is proposed for a given session. SDP does not define negotiation; for instance, if party A in a session proposes capabilities {X, Y}, and party B in the same session proposes capabilities {Y, Z}, SDP does not specify that capability {Y} will be used. More importantly, if both offer {X, Y}, it does not specify which to use – indeed it is possible to use both. In PSTN environments, it is necessary to define not only how SDP will represent a particular session, but also how negotiation will occur, since both sides must reach an exact agreement on which PSTN channel to use for a call.

#### 4.3.2.2.1 Representation

For a SIP/RTP call, the VoiceGenie software currently sends the following SDP content in SIP messages:

```
v=0
o=VoiceGenie 67108865 1 IN IP4 205.150.90.208
s=phone-call
c=IN IP4 205.150.90.208
t=3249665715 0
m=audio 7000 RTP/AVP 0 96
a=rtpmap:0 pcmu/8000
a=rtpmap:96 telephone-event/8000
a=fmtp:96 0-15
```

For a SIP/PSTN call, the following modification is proposed to the above sample:

```
v=0
o=VoiceGenie 67108865 1 IN IP4 205.150.90.208
s=phone-call
c=IN IP4 205.150.90.208
t=3249665715 0
m=audio XXXXY VG/PSTN 0 96
a=rtpmap:0 pcmu/8000
a=rtpmap:96 telephone-event/8000
a=fmtp:96 0-15
```

In the above, "VG/PSTN" replaces "RTP/AVP" as the transport protocol in the "m=" media line, specifying that the media transport will occur over PSTN instead of over RTP. Also, although the port number will remain a single integer, its format will consist of a one or two digit physical trunk identifier (no leading zeroes), followed by a three digit logical PSTN channel identifier (leading zeroes). For instance, the media line for trunk # 2, channel # 17 (2:17) is:

```
m=audio 2017 VG/PSTN 0 96
```

In some cases, it may be necessary to specify the use of a particular trunk, but not a specific channel, or even any trunk and any channel. For the former case, the channel number should be set to 0; thus, trunk # 2, any channel would be represented as "2000" in the "m=" line. For the any trunk/channel case, this will be represented simply as '0'.

#### 4.3.2.2.2 Negotiation

When negotiating the use of a PSTN channel using SDP, it is necessary that both sides agree on precisely one PSTN channel to use. The convention will be as follows:

1. Once advertised, SDP capabilities describing PSTN channels cannot change. Thus, if party A advertises trunk # 2, any channel, it cannot change this at a later point in time, and must accept the selection of any channel on trunk # 2 by the remote party.
2. Any SDP capability descriptor sent must be compatible with previously received SDP descriptors. Thus, if an SDP descriptor is received specifying trunk # 2, any channel, the valid responses are 'trunk # 2, any channel', or 'trunk # 2, channel # Y'. Therefore, if trunk # 2, channel # 17 is advertised, the only valid response is essentially a confirmation that trunk # 2, channel # 17 will be used.
3. The SDP sent in a final '200 OK' response to a SIP INVITE must always contain an SDP descriptor that completely identifies the PSTN channel to be used, unless such a descriptor was sent in an earlier message in which case no SDP descriptor needs to be included, with the prior version used unchanged.
4. Since the platform needs to perform discrimination between PSTN and RTP calls early, an SDP descriptor specifying the use of PSTN must exist in the initial SIP INVITE message.

Note that the above describes only how SDP will be used; the inclusion of SDP in SIP messages is consistent with the way that SIP is traditionally used with SDP.

Although the above is the general way that negotiation will occur without respect to which side is which, the VoiceGenie box will ALWAYS leave selection of the trunk/channel for a call to the external software, and always expects that.

#### 4.3.2.3 Call Flow

The following scenarios show the SIP messages associated with several typical call flows. Note that the actual SIP message flow is identical regardless of whether the underlying channel used for media transport is based on RTP, or on SIP.

##### 4.3.2.3.1 Inbound (to VG Media Platform)

Caller EXT	Message → INVITE →	Called VG
	<p>Signals setup request to VoiceGenie system, for destination '1234', on trunk # 2, channel # 17.</p> <pre> INVITE sip:1234@192.168.0.2 SIP/2.0 Via: SIP/2.0/UDP 192.168.0.1:5060 From: &lt;sip:4167360905@192.168.0.1:5060&gt; To: &lt;sip:1234@192.168.0.2&gt; Date: Mon, 23 Dec 2002 21:07:44 GMT Call-ID: call-1040677664-11@192.168.0.1 CSeq: 1 INVITE Contact: &lt;sip: 4167360905@192.168.0.1&gt; User-Agent: External SIP agent Content-Type: application/sdp Content-Length: (msg size)  v=0 o=ExtSw 5 5 IN IP4 192.168.0.1 </pre>	

	s=phone-call c=IN IP4 192.168.0.1 t=0 0 m=audio 2017 VG/PSTN 0	
<b>EXT</b>	<p><b>← 100 TRYING ←</b></p> <p>Signals receipt of the INVITE message by the VoiceGenie system, indicates that the request is being processed.</p> <p>SIP/2.0 100 Trying  Via: SIP/2.0/UDP 192.168.0.1:5060  From: &lt;sip:4167360905@192.168.0.1:5060&gt;  To: &lt;sip:1234@192.168.0.2&gt;  Call-ID: call-1040677664-11@192.168.0.1  CSeq: 1 INVITE  Contact:  &lt;sip:VoiceGenie@192.168.0.2:5060&gt;  Content-Length: 0</p>	<b>VG</b>
<b>EXT</b>	<p><b>← 200 OK ←</b></p> <p>Indicates that the VoiceGenie system has accepted the call. Note that the SDP confirms the trunk/port selection</p> <p>SIP/2.0 200 OK  Via: SIP/2.0/UDP 192.168.0.1:5060  From: &lt;sip:4167360905@192.168.0.1:5060&gt;  To: &lt;sip:1234@192.168.0.2&gt;  Call-ID: call-1040677664-11@192.168.0.1  CSeq: 1 INVITE  Contact:  &lt;sip:VoiceGenie@192.168.0.2:5060&gt;  Content-Type: application/sdp  Content-Length: (msg size)</p> <p>v=0  o=VoiceGenie 67108665 1 IN IP4  192.168.0.2  s=phone-call  c=IN IP4 192.168.0.2  t=0 0  m=audio 2017 VG/PSTN 0</p>	<b>VG</b>
<b>EXT</b>	<p><b>→ ACK →</b></p> <p>Acknowledges the receipt of a 200 OK response, formally completing the call.</p> <p>ACK sip:1234@192.168.0.2 SIP/2.0  Via: SIP/2.0/UDP 192.168.0.1:5060  From: &lt;sip:4167360905@192.168.0.1:5060&gt;  To: &lt;sip:1234@192.168.0.2&gt;  Call-ID: call-1040677664-11@192.168.0.1  CSeq: 1 ACK  dp  Content-Length: 0</p>	

<p>At this point, the call is setup, and media is exchanged normally.</p> <p>Call termination can be initiated by either side, it is shown below being initiated from the EXT side</p>		
<b>EXT</b>	<p><b>→ BYE →</b></p> <p>Requests the termination of a session; equivalent to a PSTN hangup</p> <pre> BYE sip:1234@192.168.0.2 SIP/2.0 Via: SIP/2.0/UDP 192.168.0.1:5060 From: &lt;sip:4167360905@192.168.0.1:5060&gt; To: &lt;sip:1234@192.168.0.2&gt; Call-ID: call-1040677664-11@192.168.0.1 CSeq: 1 BYE Content-Length: 0 </pre>	<b>VG</b>
<b>EXT</b>	<p><b>← 200 OK ←</b></p> <p>Indicates that the VoiceGenie system has confirmed the termination.</p> <pre> SIP/2.0 200 OK Via: SIP/2.0/UDP 192.168.0.1:5060 From: &lt;sip:4167360905@192.168.0.1:5060&gt; To: &lt;sip:1234@192.168.0.2&gt; Call-ID: call-1040677664-11@192.168.0.1 CSeq: 1 BYE Content-Length: 0 </pre>	<b>VG</b>

#### 4.3.2.3.2 Outbound (from VG Media Platform)

Caller	Message	Called
<b>VG</b>	<p><b>→ INVITE →</b></p> <p>VoiceGenie system signals request for an outbound call to destination number 4167360905. Note that the VoiceGenie system will always leave trunk/channel selection to the remote system, unless configured otherwise</p> <pre> INVITE sip:4167360905@192.168.0.1 SIP/2.0 Via: SIP/2.0/UDP 192.168.0.2:5060 From: &lt;sip:VoiceGenie@192.168.0.2:5060&gt; To: &lt;sip: 4167360905@192.168.0.1&gt; Call-ID: 0030324@192.168.0.2 CSeq: 1 INVITE Contact: &lt;sip: VoiceGenie@192.168.0.2&gt; User-Agent: VoiceGenie Content-Type: application/sdp Content-Length: (msg size)  v=0 o=VoiceGenie 67108665 1 IN IP4 192.168.0.2 </pre>	<b>EXT</b>



	s=phone-call c=IN IP4 192.168.0.2 t=0 0 m=audio 0 VG/PSTN 0	
<b>VG</b>	<p><b>← 100 TRYING ←</b></p> <p>(Optional) signals receipt of the INVITE by the external system</p> <p>SIP/2.0 100 Trying  Via: SIP/2.0/UDP 192.168.0.2:5060  From: &lt;sip:VoiceGenie@192.168.0.2:5060&gt;  To: &lt;sip: 4167360905@192.168.0.1&gt;  Call-ID: 0030324@192.168.0.2  CSeq: 1 INVITE  Contact:  &lt;sip:4167360905@192.168.0.1:5060&gt;  Content-Length: 0</p>	<b>EXT</b>
<b>VG</b>	<p><b>← 200 OK ←</b></p> <p>Indicates that the external system has accepted the call. Note that the SDP specifies the trunk/port selection.</p> <p>SIP/2.0 200 OK  Via: SIP/2.0/UDP 192.168.0.2:5060  From: &lt;sip:VoiceGenie@192.168.0.2:5060&gt;  To: &lt;sip: 4167360905@192.168.0.1&gt;  Call-ID: 0030324@192.168.0.2  CSeq: 1 INVITE  Contact:  &lt;sip:4167360905@192.168.0.1:5060&gt;  Content-Type: application/sdp  Content-Length: (msg size)</p> <p>v=0  o=ExtSw 5 5 IN IP4 192.168.0.1  s=phone-call  c=IN IP4 192.168.0.1  t=0 0  m=audio 2017 VG/PSTN 0</p>	<b>EXT</b>
<p>At this point, the call is setup, and media is exchanged normally.</p> <p>Call termination occurs identically to the inbound scenario.</p>		

## 4.4 Multiple Line Managers

The VoiceGenie Media Platform has the flexibility to allow multiple line managers to be configured and operate concurrently. A Media Platform can be configured to handle both SIP/RTP calls simultaneously with PSTN calls from either Dialogic or Brooktrout telephony cards. It also works in the same fashion with H.323/RTP. In fact, it is possible to configure SIP/RTP, H.323/RTP and PSTN to work in parallel. This

flexible configuration allows experimental integration (between VoIP and PSTN) to be performed easily without re-configuring the VoiceGenie Media Platform. Note that however, it is currently not possible to configure both Brooktrout and Dialogic cards to run on the same platform.

## 5 Call Control

### 5.1 Incoming Call

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

With PSTN case, the DNIS information is well-defined in the telephony protocol (for example, a field in the setup message for ISDN).

Similarly, H.323 calls also 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@address[:port], where the [:port] is optional and may not be included if the default SIP port of 5060 is used. 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.

The VXML URL can also be specified using the format based on IETF draft draft-burger-sipping-netann-11.txt.

For example, a call to:

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

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

### 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 section 5.1 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 section 15.2 for more information.



**NOTE:** For certain switches, dialing '+1' before the local phone number will result in rejected calls.

## 5.2.1 Dialing Rules

### 5.2.1.1 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.

### 5.2.1.2 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").

### 5.2.1.3 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[cccept] 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
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. If can be set to zero to simply use the incoming line ID (if available).
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 rule1 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*.

Attribute = Value  
 =   
Context: (Optional)

Priority: (Optional)

rule-type  
Reject

addr-type  
Tel

:addr-pattern

xlat-type  
-

xlat-pattern  
-

out-lm-id  
0

[rule-params]

Create

#### Dialing Rules Configuration

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

Context: (Optional) Attribute Value   **Add**

Priority: (Optional)

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

**Create**

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   **Add**

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		

**Update Delete Select Target**

#### 5.2.1.4 Rule File Processing

- Rules are processed in the order that they appear in the rules field, without regard to context.
- 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 '1' for telephony calls or '2' for SIP calls.

#### 5.2.1.5 Some operational notes

Phone numbers have the following restrictions at the moment:

- Main phone number: 0-9
- Extension number: 0-9, a-d, #, \*, ', '.

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.

#### **5.2.1.6 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 Hunt Group**

The Hunt Group Configuration controls the channel and trunk hunting logic for making outgoing calls. It can be accessed via the SMC, under the "Configuration" tab in the "Media Platform" subsection. There are four hunt algorithms available for selection. When an outbound call is requested, the platform uses the hunt algorithm to first select the trunk and then selecting the channel to make the outbound call. The outbound call request can specify a hunt group to dial, and a hunt group consists a hunt algorithm and a priority matrix.

The description of the hunt algorithms are below:

- Round-robin ascending with memory – Trunks are selected in round robin fashion. When the trunk is selected, the channel is selected in ascending order with memory.
- Round-robin descending with memory – Trunks are selected in round robin fashion. When the trunk is selected, the channel is selected in descending order with memory.



- Least-available – The trunk with the least number of channels available is selected. When the trunk is selected, the channel is selected in ascending order.
- Most-available - The trunk with the most number of channels available is selected. When the trunk is selected, the channel is selected in ascending order.

An example of Hunt group configuration is below:

Hunt Group	Algorithm	Priority Matrix	
<b>0</b>	<b>Round-robin ascending with memory</b>	Priority	<b>Trunks</b>
		<b>1</b>	<b>1,2,3,4</b>
<b>1</b>	<b>Least-Available</b>	<b>Priority</b>	<b>Trunks</b>
		<b>1</b>	<b>4</b>
		<b>2</b>	<b>1,2,3</b>

The default hunt group is 0.

When hunt group 1 is selected, it will select the trunks by ascending priority. In this case, trunk 4 is considered by the Least-available algorithm first. If trunk 4 is full, it will select priority 2's trunks (1,2, and 3) for consideration.

### 5.2.3 Destination Format (PSTN)

A Phone URI (Uniform Resource Identifier) defines a scheme for specifying the location of a terminal in the phone network and the connection types (modes of operation) that can be used to connect to that entity. Starting in 6.4, VoiceGenie Media Platform will add support for IETF RFC2806 Phone URI syntax, in addition to the existing VoiceGenie proprietary Phone URI syntax.

The following summarizes the existing VoiceGenie proprietary Phone URI syntax:

- phone-URI = ( "tel://" / "phone://" / "tel:" / "phone:" ) optional
- optional = ( "+" / phone-uri ) / phone-uri
- phone-uri = (phone-number "x" phone-number) / phone-number
- phone-number = 1\*( DIGIT / vg-alpha / separator / "," )
- vg-alpha = "A" / "a" / "B" / "b" / "C" / "c" / "D" / "d" / "\*" / "#"
- separator = "-" / "." / "(" / ")"

Examples:

- tel://+1.800.555.1212
- phone:18005551212x314
- phone://1800555(12)12x(3.1.4)

The RFC2806 Phone URI syntax can be found at <http://www.ietf.org/rfc/rfc2806.txt>.

However, there are some minor adjustments made to the RFC2806 syntax that should be noted. These enhancements make VoiceGenie's support more lenient than the RFC2806 syntax. This is mainly because Media Platform needs to remain compatible with the existing proprietary syntax as mentioned. This is also because some of the parameters specified in the RFC2806 may not be applicable or fully supported by VoiceGenie software.

The following list all the adjustments made to the RFC2806 support and all the special assumptions made:

2. When coming across a “//” in the URI type (eg, phone: and phone://), the VoiceGenie Media Platform will ignore it.
3. VoiceGenie has extended to support the existing usage of ‘x’ to specify extension dialing. The extension portion is treated similarly as a postd parameter. Hence, the Media Platform has implicitly extended to support custom parameters even with the old URI format (eg, tel:4161234567x890;param=value is now supported). However, it will be treated as an error if both postd and ‘x’ is used to specify extension.
4. The VoiceGenie Media Platform will treat the “p” (pause) wait-character the same way as the “,”. Both will function as an approximately one-second pause.
5. The VoiceGenie Media Platform does not have support for wait-for-dialtone. Hence, it will ignore the “w” (wait-for-dialtone) wait-character and throw a warning message.
6. The VoiceGenie Media Platform will not support the <isdn-subaddress> parameter and will ignore the value in the “isub” parameter without checking its syntax.
7. The VoiceGenie Media Platform will not support the <area-specifier> parameter and will ignore the value in the “phone-context” parameter without checking its syntax. Customers requiring this support should use the dialing rules feature.
8. The VoiceGenie Media Platform will not support the <service-provider> parameter and will ignore the value in the “tsp” parameter without checking its syntax.
9. The VoiceGenie Media Platform will not fail the parsing if there is a “;;” in the parameters list.
10. The VoiceGenie Media Platform does not have the capability to check for global phone number usage. The leading “+” character is treated as a separator character and is ignored.

The VoiceGenie Media Platform places a generic length limit on the phone number portion of the phone URI (60 characters) and places a generic length limit for each custom parameter (196 characters). The extension portion is treated as a custom parameter and hence has a 196 character length limit. Some telephony board/protocol may have even stricter restrictions on the length limit.

11. The VoiceGenie Media Platform will not support the <future-extension> syntax and will ignore the value in the <future-extension> syntax without check.

#### 5.2.4 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@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](mailto:voip@sip.voicegenie.com:5060) could be something other than sip.voicegenie.com. The section routing and proxies below 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/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 Call Manager Configuration section for more details.

Note that in addition to the forementioned parameters, 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](mailto: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 extension 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.

## **5.4 Glare Handling**

With PSTN, when making an outbound call, the channel selected may have an incoming call arriving at the same time. This situation is known as glare. Glare detected during an outbound dial attempt results in the failure of the outbound call – the inbound call is accepted. The glare event results in the logging of an 'outbound reject' metrics record, with a status of 'glare'.

There are a number of techniques that can be used to avoid glare conditions. These include:

- Proper sizing of the platform
- Appropriate designation of channels as inbound, outbound or bi-directional
- Appropriate switch configuration (hunt algorithms, etc.)

## 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, refer, tc1, tc2, tc3, inband
Join	rlt, tbct, cmg, ect, qsig, 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.



**NOTE:** On analog systems using dialogic telephony cards, it is not recommended to set the "**connectwhen**" attribute to "**immediate**". This is due to the fact that the dialed DTMF tones may echo in an analog environment and cause unintended barge-in events to the VoiceXML application.

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
	Inband DTMF transfer	inband	YES	NO	NO
SIP	Hook flash transfer	hkf	YES	YES	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
Dialogic	RLT transfer	rlt	NO	YES	NO
	TBCT transfer	tbct	YES	YES	NO
	ECT transfer	ect	YES	YES	NO
	AT&T Call Merge	cmg	YES	YES	NO
	Hook flash transfer	hkf	YES	YES (NO for DMV)	NO
	TC1 transfer	tc1	YES	NO	NO
	TC2 transfer	tc2	NO	YES	NO
	TC3 transfer	tc3	YES	YES	NO
	Q.SIG	qsig	YES	YES	NO
Brooktrout	RLT transfer	rlt	NO	YES	NO
	TBCT transfer	tbct	YES	YES	NO
	Hook flash transfer	hkf	YES	YES	NO

For the list of available signalvar, please refer to  
["http://developer.voicegenie.com/reference.php?ref=variablesdetails#signalvar"](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 PSTN Transfer

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

Call Transfer Method	Protocols	Switches	Notes
Release Link Trunk (RLT)	ISDN	DMS	
Two B-channel Transfer (TBCT)	ISDN	DMS, 5ESS	NI-2 protocol
Explicit Call Transfer	ISDN	DMS and others	ETSI ISDN; No Brooktrout support currently
AT&T Call Merge	ISDN	4ESS	No Brooktrout support currently
Transfer Connect (TC) - Courtesy	ISDN	4ESS	No Brooktrout support currently
Transfer Connect (TC) - Consult	ISDN	4ESS	No Brooktrout support currently
Transfer Connect (TC) - Conference	ISDN	4ESS	No Brooktrout support currently
Q.SIG	ISDN	Definity G3	No Brooktrout support currently
Hook Flash	Robbed bit, CAS, Analog	Definity G3, Meridian M100, Rockwell, others	
Inband	Any	Any supporting switch	Expects switch to initiate transfer based on the inband DTMF played
Bridge	Any	Any	Media are bridged at the Media Platform

Bridge transfer is always supported by the Media Platform by default. With the exception of inband transfer, each of the other transfer methods can be configured under the Call Manager configuration (callmgr.cfg) using the BLIND\_TRANSFER\_TYPE parameter (see section 12.2). Inband transfer can be enabled under the Call Manager configuration (callmgr.cfg) using the sessmgr.inbandxferenable parameter.

Some transfer methods may require further parameters to suit the switch's and the environment's behavior. The following table summarizes the relevant parameters for transfer methods that require platform-level configuration changes. Please see the corresponding configuration sections for further details:

Transfer Method	Related Parameters
RLT	Call Manager configuration: - INFO_ELEMENT1 - INFO_ELEMENT2
TBCT	Call Manager configuration: - TBCT_DATA
ECT	Call Manager configuration: - ECT_TRANSFER_ON_ALERT - CALLING_PRESENTATION
Transfer Connect (Courtesy)	Call Manager configuration: - TC_DATA - TC_HOLD - TC_RETRIEVE - TC_DROP
Transfer Connect (Consult)	
Transfer Connect (Conference)	N/A
Q.SIG	Call Manager configuration: - dlgc.qsigputcalleronhold - dlgc.qsigdisctimer - dlgc.qsigfaildiscnopr - dlgc.qsigfailontimeout - dlgc.qsigsendctactive - dlgc.qsigdropotherleg - dlgc.qsigscreeningindicator1 - dlgc.qsigscreeningindicator2 - dlgc.qsigsendredirnum - dlgc.qsigactionrecvinvokresult - dlgc.qsigpathreplacecode - dlgc.qsiginvokeforwardlist - dlgc.qsiginvokefaillist - dlgc.qsigcallstatusalert
Hook Flash	Call Manager configuration: - INFO_HOOK_FLASH - HOOK_FLASH_TIME - HOOK_FLASH_DISCONNECT_TIME Dialogic Configuration (for DMV RB users): (see Section 11.2 on Hook Flash)
Inband	Call Manager Configuration: - sessmgr.inbandxferenable - sessmgr.inbandxferprefix - sessmgr.inbandxfertimeout

## 6.4 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. Media Redirect transfer is enabled for H.323 protocol 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. Inband transfer can be enabled using the sessmgr.inbandxferenable parameter.

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 for further details.

## 6.5 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
- Dialogic – rlt, tbct, ect, cmg, qsig, hookflash, bridge
- Brooktrout – rlt, tbct, bridge



**NOTE:** Whisper transfer does not work with multiple interpreter instances.

## **6.6 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.

## 7 Other Features

### 7.1 Remote Dial

#### 7.1.1 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.

#### 7.1.2 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.

#### 7.1.3 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.

#### 7.1.4 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 4167360905x4424 4167362012 http://www.voicegenie.com/helloworld.vxml 0001 Test
```

```
!CALL_SENT 1: telno:4167360905x4424 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. There are three other optional parameters that can be specified:

[uuidata]	-	user-to-user info element, not longer than 254
[defaults]	-	path of the defaults voicexml file for this call
[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. For ISUP/SS7 configurations using the VoiceGenie SS7 Connector, "ISUP=TRUE" should be set as one of the parameters on this list.

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 - - NWNNAME=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>



**Note:** On Solaris, telnet input line is limited to 256 characters.

### 7.1.5 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.



**Note:** 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 reference number
!INVALID_REFNO <refno>: Invalid reference number
```

!TOO\_MANY\_CALLS <refno>: Too many calls in progress  
 !INVALID\_TELNO <refno>: Incorrect telephone format  
 !INVALID\_DNIS <refno>: Incorrect DNIS 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>



*Note:* 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)  
 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.  
 REORDER: Reorder.  
 VACANTCODE: Vacant Code.  
 NOCIRCUIT: No Circuit.  
 INTERCEPT: Intercept.  
 INEFFECTIVE\_OTHER: Ineffective other.  
 FAX: FAX machine.  
 NO\_LICENSE: no licenses available.  
 RESTRICTED\_TELNO: Restricted telephone number.  
 UNSUPPORTED\_URL: url is unsupported.  
 INVALID\_TELNO: Invalid telephone number.  
 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.



**Note:** When analysis is n, MACHINE, REORDER, VACANTCODE, NOCIRCUIT, INTERCEPT, INEFFECTIVE\_OTHER, FAX won't be returned as a call drop status.

For dialogic boards with ISDN, BUSY will not be detected for analysis=n while it will be detected with Robbed Bit.

For Brooktrout BUSY will not be detected for analysis=n with either ISDN or Robbed Bit.

For SIP no call analysis is supported and hence MACHINE, REORDER, VACANTCODE, NOCIRCUIT, INTERCEPT, INEFFECTIVE\_OTHER, FAX will not be detected.

### 7.1.6 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.
- For Dialogic configuration, when analysis=n, the behavior of busy detection depends on connect timeout value. If platform connect timeout < switch timeout then, ISDN version may return NO\_ANSWER for busy signal and platform connect timeout > switch timeout then, ISDN version may return CALL\_FAILED. RB may detect busy signal properly.  
/usr/dialogic/cfg/us\_mf\_io.cdp should have \$1 set to somewhere between 20 and 50. This will make the CALL\_FAILED rate go down.

## 7.2 Call Analysis

Call analysis is used in a number of situations to provide information regarding call progress, or identification of the entity that has answered a call. This is supported in a PSTN environment by the telephony board(s). The VoiceGenie Media Platform currently does not perform tone analysis at software level and hence call analysis is not available for VoIP configurations.

In general, there are three main call analysis functions:

- Call Progress Analysis (ring, busy, etc.)
- Detection of System Information Tones (SIT), indicating network status information
- Detection of answering entity – fax, answering machine, human, etc.

Detection of each of these takes place with varying levels of reliability. Control over these features is provided both in the line manager configuration (see the call manager configuration section) and at the application level, where each outbound call or transfer request can enable or disable call analysis. In addition, the following two parameters control whether to enable call analysis or not:

Parameter	Default	Description
-----------	---------	-------------

DLGC_CALL_ANALYSIS	yes	Can be set to 'yes' or 'no', and controls whether Dialogic call analysis is enabled.
BT_CALL_ANALYSIS	yes	Can be set to 'yes' or 'no', and controls whether Brooktrout call analysis is enabled.

For further details for utilizing the Call Analysis feature in a VoiceXML application, please refer to the <call> and <transfer> tag sections in the **VoiceGenie 7 VoiceXML User Guide**.

### 7.2.1 Limitations

1)

Note that there is an implication for performing call analysis using Brooktrout hardware, as it takes at least 5 seconds:

Thus the connecttimeout attribute must be set to a value larger than 5 seconds when call analysis is turned on using <call> or <transfer> tags, Otherwise an outbound call may fail and detected as "noanswer", when the physical call may have successfully connected.

## 7.3 Full Call Recording

### 7.3.1 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 captured 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.

### 7.3.2 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 that 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 is an example of enabling full call recording logging:

```
<log dest="callog">
    enable callrec recsrc=mixed;
</log>
```

The following commands are supported:

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

The directory command specifies the directory in which call-specific log files will be collected. If used, this command **MUST** appear in the first instance of a <log> tag with the "callog" destination; otherwise, it will be ignored, since files cannot be moved after they have already been created.

<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 **callog.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.

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-g726-32", 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.

### 7.3.3 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.
```

### 7.3.4 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.

## 7.4 UUI Data

UUI data is supported for Dialogic ISDN configurations. In release 7 and above, the VoiceGenie Media Platform supports:

- Allowing user to specify UUI protocol to be used for outgoing UUI IE with either configuration parameter or signal variable (with the latter taking precedence)
- Extracting and removing protocol field from incoming UUI IE and pass it to an application as a parameter
- Allowing user to turn off support for the protocol field. In this case no protocol field will be set in outgoing UUI or extracted from incoming UUI IEs.
- Allowing user to turn off encoding of non-printable characters for incoming and outgoing UUI

- Adding an optional codeset parameter and signal variable with only acceptable value of '7'. With this parameter set outgoing UUI will be placed after locking shift to Codeset 7.

When using user-defined protocol (0x00) it is user's responsibility to format UUI content according to application needs. UUI content will be placed into a UUI IE starting from octet 4, unless support for protocol field is disabled. In this case user-provided UUI content will start from octet 3.

#### 7.4.1 Platform configurations and application signal variables

The following configuration parameters and signal variables can be accessed via the System Management Console (SMC), "Call Manager" section under the "Configuration" tab:

Feature	Configuration parameter	Signal variable
<b>UUI Protocol.</b> Select protocol for outgoing UUI IE. Allowed values: 0x00 .. 0xff Default: 0x04	pstn.uuiprotocol	uuiprotocol
<b>UUI Codeset</b> Select Codeset for outgoing UUI IE. Allowed values: 0 and 7 Default: 0	pstn.uuicodeset	uuicodeset
<b>Enable support for UUI protocol</b> Extract and remove UUI protocol field from incoming UUI and set protocol field in outgoing UUI Default: true	pstn.enableuuiprotocol	enableuuiprotocol
<b>Enable encoding of non-printable characters</b> Encode non-printable characters before passing them to an application and decode before populating UUI IE Default: true	pstn.enableuuiencode	enableuuiencode

Variable **session.connection.uuiprotocol** will contain protocol field from incoming UUI (if not disabled with **pst.enableuuiprotocol = false**)

#### 7.4.2 Notes

- TC\_UUI\_PROTOCOL configuration parameter is no longer supported since 7.0. It is with pstn.uuiprotocol.

- If user-defined protocol (0x00) is selected, current implementation prepends user-supplied UUI data with data forwarding information (text tag). [TR50075] (AT&T Toll Free Transfer Connect Service, Issue 3.0, September 1999) APPENDIX A defines this format as a “suggested encoding” and not a requirement. This feature will be removed; an application must format UUI data as required.



## 8 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.

### 8.1 Metrics and Logging/Billing

#### 8.1.1 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 identifier.
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 Media Platform System Reference Guide](#).



*Note:* the session ID for transfer\_result is always the parent ID

### 8.1.2 Metrics for Transfer

Metrics behavior for transfer depends on the transfer style (see "Transfer for Advanced User" section for details) to which the transfer method belongs.

Transfer Style	Method
One-leg	hkf, refer, tc1, tc2, tc3, inband
Join	cmg, ect, h450join, referjoin, rlt, tbct
Bridge	Bridge, mediatedirect

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 transfer_result
Bridge	bridge_initiated bridge_begin bridge_end	bridge_initiated bridge_reject

## 8.2 Alarms in Media Platform

The VoiceGenie 7 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 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 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.

### 8.2.1 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 daemon 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

### 8.2.2 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 (DEBUG) ☐ Metric (METRIC)

Filter By ID:  (Log IDs in simple regular expression.)  
 Filter By Info:  (Info string in simple regular expression.)

Search



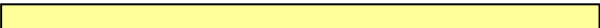

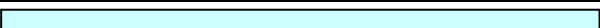


27 Matching Results Found.

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. Critical, 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 OA&M Framework User Guide for a better understanding of the Log ID field.

## 8.3 Health Status

### 8.3.1 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

#### 8.3.1.1 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:

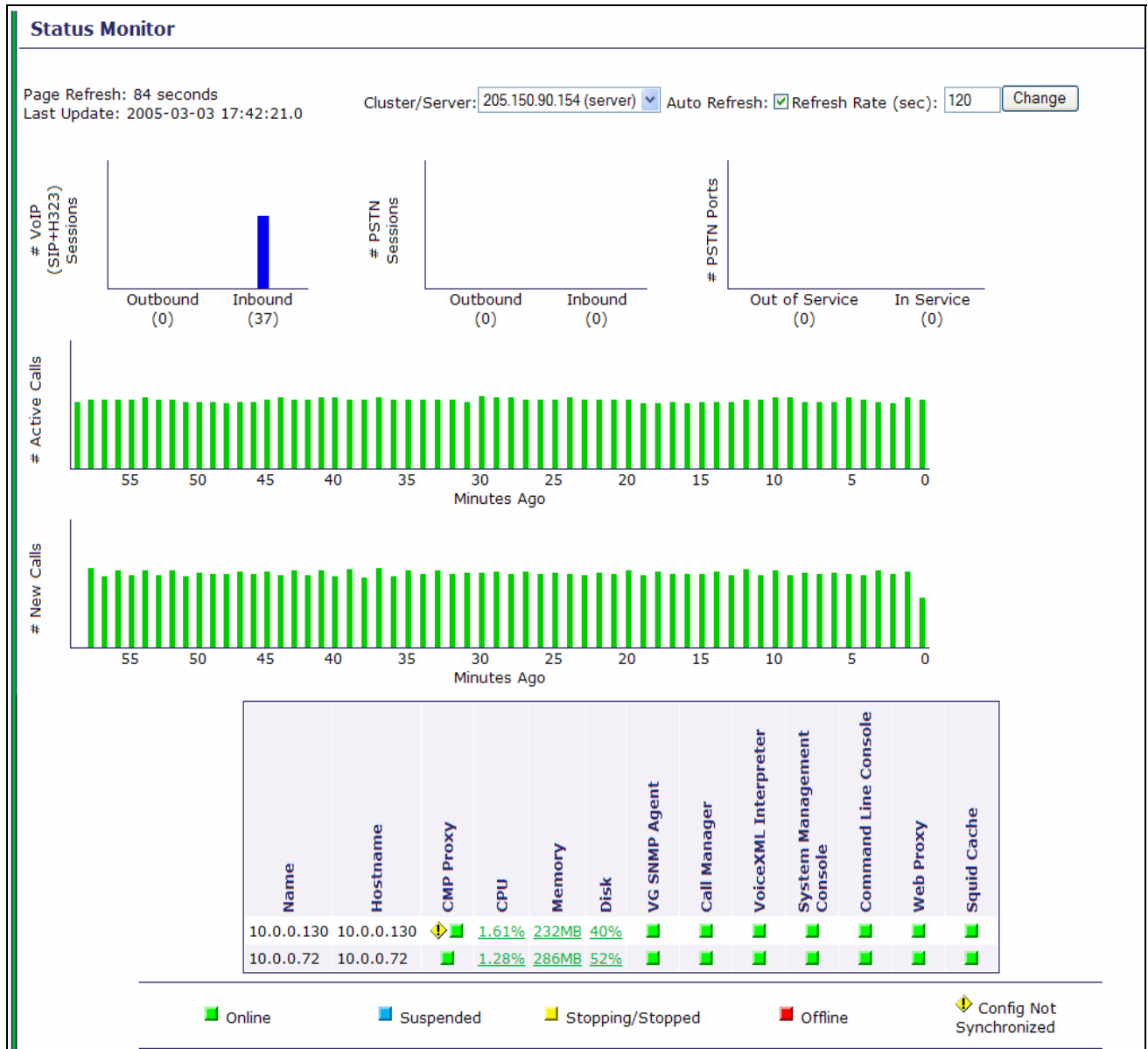
health <service>

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 OA&M Framework – CLC Guide

### 8.3.1.2 SMC Status Monitor

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





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:

**System Management Console**  
Monitoring Operations Configuration Administration  
taco | Connected to CMP Proxy

**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)**

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

**Current Status**

- Status Monitor
- Cluster Status
- Realtime Call Monitor
- Call Log Browser
- Alarm Browser

**Diagnostic Tools**

- Ping Tool
- SNMP Walk Tool

**Historical Reports**

- Call Volume Report
- Call Length Distribution
- Application Distribution
- Process Status Report

[Logout](#)

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 OA&M Framework – SMC Guide

### 8.3.2 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 12, Peak 17, Total 4944895
#VXMLi Attempted Connection: 1
#VXMLi Enabled: 1
VRM Engines: None
BRKT
Calls (#IB:#OB): Current 7:10, Peak 10:16, Total 2472448:2472445
Channels (#D:#I:#O): Enabled 46:0:0, Total 46:0:0
Board Status:   BOO[ISDN] ( SOO[UP] SOI[UP] )
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 VXML Interpreter(s) has/have made.
- "VXMLi Enabled" -- shows the number of enabled VXML Interpreter instances. VoiceGenie 7 Media Platform is capable of supporting more than one VXML 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.

#### 8.3.2.1 PSTN Line Managers

Here are the snapshots of the health string outputs of the Brooktrout and Dialogic line managers:

```
BRKT
Calls (#IB:#OB): Current 7:10, Peak 10:16, Total 2472448:2472445
Channels (#D:#I:#O): Enabled 46:0:0, Total 46:0:0
Board Status:   BOO[ISDN] ( SOO[UP] SO1[UP] )
```

```
DLGC
Calls (#IB:#OB): Current 0:0, Peak 46:0, Total 618:0
Channels (#D:#I:#O): Enabled 0:0:0, Total 46:0:0
Board Status:   :N_dtiB1:P_isdn(down) :N_dtiB2:P_isdn(down)
```

Since all PSTN line managers are associated with one or more hardware telephony cards, their health string outputs have entries which are different from other line managers:

"Channels" -- shows the number of enabled and available channels. Each channel can either be set as duplex mode (#D -- i.e. takes and makes both inbound/outbound calls), outbound mode (#O -- i.e. makes only outbound calls) or inbound mode (#I -- i.e. takes only inbound calls). The numbers shown in "Enabled" and "Total" can be different, as the Media Platform allows users to disable/suspend individual channels.

"Board Status" -- shows the status of the boards as well as the span(s) information of each board. From the example snapshots, both spans of the Media Platform operating with the Brooktrout telephony card (using ISDN protocol) are up, whereas the Media Platform operating with the Dialogic telephony card have both spans down.

### 8.3.2.2 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.

### 8.3.3 VoiceXML Interpreter (VXMLi)

Here is a snapshot of the health string output of the VoiceXML 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>

### 8.3.4 Fetching Module/Web Proxy (iproxy)

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

```
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 clients, 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>

## 8.4 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 OA&M Framework – Users' Guide](#), and the "General Component Operation Commands" Section of the [VoiceGenie 7 OA&M Framework – CLC 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 defaults.vxml file, the application root document, 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.
- **Ensure that the relevant Dialogic/Brooktrout driver patches have been installed.**  
For systems using the Dialogic/Brooktrout telephony cards, contact VoiceGenie support ([support@voicegenie.com](mailto:support@voicegenie.com)) for information regarding which patches you require for your particular release. Please ensure that you include the output of the 'vginfo\_linux.sh' command with your request. These patches correct a number of driver issues with ISDN and robbed bit configurations.