GENESYS
AN ALCATEL-LUCENT COMPANY

VoiceGenie 7.2.2

Speech Resource Management

User's Guide

About Genesys

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

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

Notice

Your Responsibility for Your System's Security

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

Trademarks

Technical Support from VARs

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

Technical Support from Genesys

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

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

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

Ordering and Licensing Information

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

Released By

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

Document Version: 06-2009

Table of Contents

Chapter

1 Introduction

This document provides detailed information about the various components of the Speech Resource Management, notably the SRM Server and the SRM Client. Also, it contains information about how the SRM components are integrated into other VoiceGenie components.

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)

1.1 Functional and Architecture Overview

With modern day telephony applications, it is no longer sufficient to provide services to customers using touch-tones and pre-recorded audio. The Service Providers require dynamic prompts to be generated and play back to the Callers; while the Callers demand a better and more natural way of interacting with the application.

The Speech Resource Management (SRM) system is used to integrate with 3rd party Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) engines. This component is integrated into the VoiceGenie Media Platform to provide Speech Recognition and Synthesis functionalities to the application developers. Using this component, products from many 3rd party vendors has been integrated to work with the VoiceGenie Media Platform.

The Speech Resource Management system consists of three components: the SRM Client component, the SRM Server component and the SRM Proxy component (also referred to as the MRCP Proxy). The three components are integrated with the VoiceGenie Media Platform in two different architectural models, the three-tiered (client/proxy/server) model and the two-tiered (client/server) model. The *VoiceGenie 7.2 MRCP Proxy User's Guide* and the *VoiceGenie 7.2 MRCP Proxy System Reference Guide* contain information for the MRCP Proxy and the three-tiered model. This document will focus on the SRM Client and Server components.

The client component is a Dynamic Link Library (either a `.so` library on Linux or a `.dll` library in Windows) used by the Media Platform to access ASR/TTS functionality. The server component accepts and handles requests from the client component. The server component contains integration software written specifically to work with each of the 3rd party ASR/TTS products.

With the emergence of the Media Resource Control Protocol (MRCP) as a widely-supported standard, VoiceGenie has adopted MRCP as the protocol of choice for ASR and TTS integration. Many ASR/TTS vendors have also adopted MRCP as the method of control for their products. VoiceGenie SRM client, server and proxy all use the Media Resource Control Protocol (MRCP) v1 specification version: Internet Engineering Task Force Internet-Draft draft-shanmugham-mrcp-04. Since VG7.2.1, the SRM client implemented the MRCPv2 specification version: draft-ietf-speechsc-mrcpv2-12.

The following diagram offers an architectural view of the client/proxy/server model, note that all communication between client, proxy and server components is via MRCP:

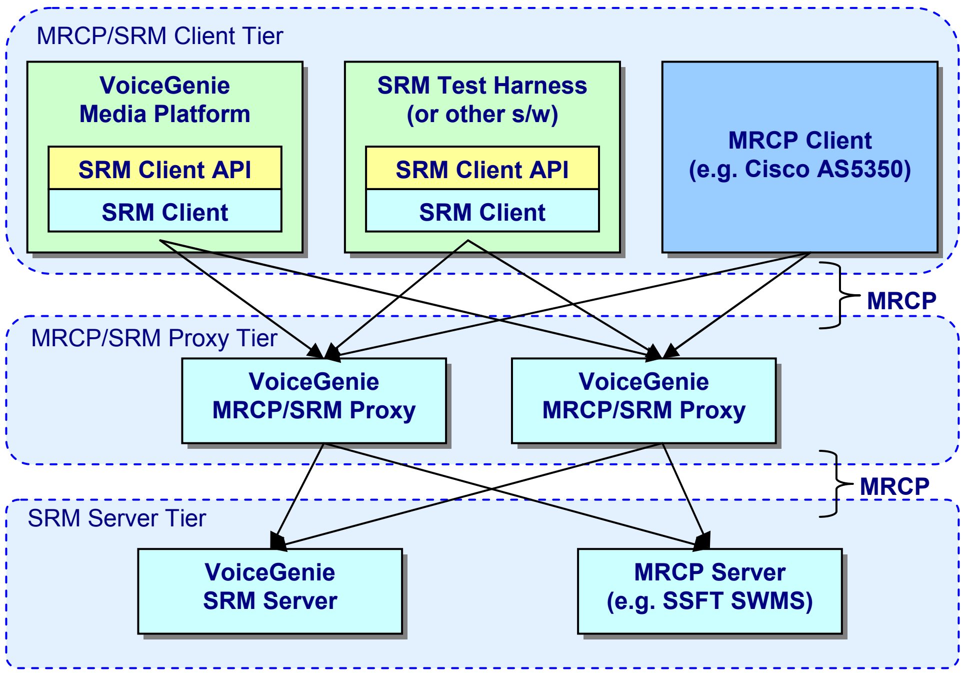

MRCP/SRM Client Tier
VoiceGenie Media Platform
SRM Client API
SRM Client
SRM Test Harness (or other s/w)
SRM Client API
SRM Client
MRCP Client (e.g. Cisco AS5350)
MRCP
MRCP/SRM Proxy Tier
VoiceGenie MRCP/SRM Proxy
VoiceGenie MRCP/SRM Proxy
MRCP
SRM Server Tier
VoiceGenie SRM Server
MRCP Server (e.g. SSFT SWMS)

Figure 1: SRM Client/Server/Proxy Architecture Overview

From the VoiceGenie Media Platform's perspective all communication to Speech Resources is done via MRCP. However, not all Speech Resources are fully MRCP compliant, as a result, two approaches for communication via MRCP exist, *MRCP Native* and *MRCP(v1 and v2) Direct.*

Figure 2 illustrates the MRCP Direct integration architecture. This is the architecture used when the 3rd party Speech Resource supports MRCP. Most integrations fall under this case. Examples of MRCP Direct integrations include OSR via SWMS, Realspeak via SWMS, Nuance MRCP 1.0 ASR Server, IBM ASR/TTS, Nuance Recognizer via NSS and RealSpeak via NSS. In this architecture all communication between the SRM Client and MRCP Servers is via MRCP.

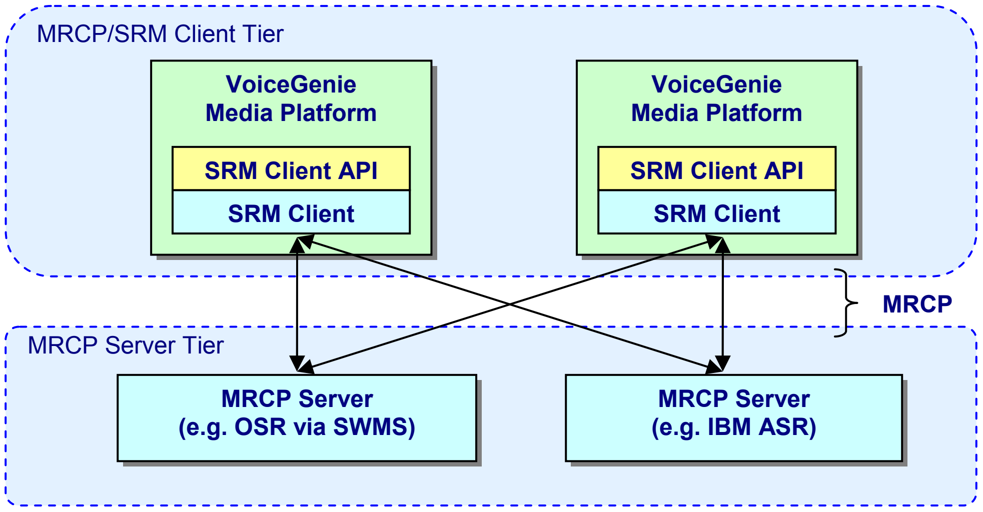

MRCP/SRM Client Tier
VoiceGenie Media Platform
SRM Client API
SRM Client
VoiceGenie Media Platform
SRM Client API
SRM Client
MRCP
MRCP Server Tier
MRCP Server (e.g. OSR via SWMS)
MRCP Server (e.g. IBM ASR)

Figure 2: MRCP Direct Integration Architecture

Figure 3 illustrates the MRCP Native integration architecture. In this architecture the SRM Client still communicates via MRCP to the SRM Server, but the SRM Server then interacts with the 3rd party vendor software via a native API. In this scenario the VoiceGenie SRM Server contains software from both VoiceGenie and the 3rd party ASR/TTS vendor.

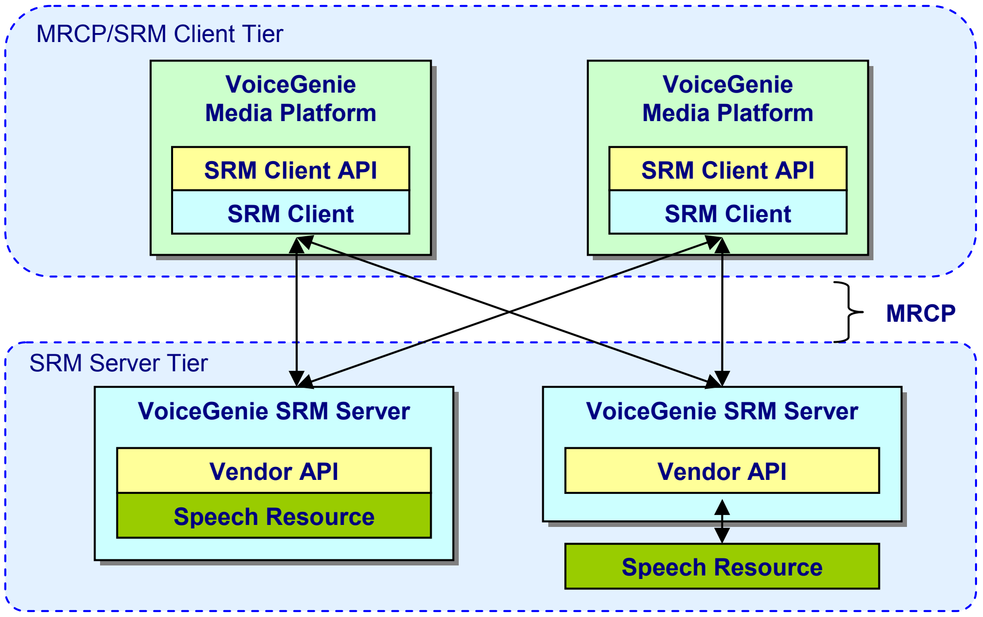


Figure 3: MRCP Native Integration Architecture

1.2 SRM Component Interfaces

This section details other VoiceGenie components that the SRM components interface with.

1.2.1 VoiceGenie Media Platform

The VoiceGenie Media Platform contains a number of components including the Call Manager component. This component manages the various resources of a call. It handles incoming/outgoing calls, accepts instructions from the VoiceXML Interpreter component, and makes requests to use ASR and TTS accordingly. The Call Manager is the component that loads and uses the SRM Client. It uses the SRM Client to communicate via MRCP to speech resources for the purposes of accessing ASR and TTS resources.

1.2.2 VoiceGenie OA&M Framework

The VoiceGenie OA&M Framework is the management layer for all VoiceGenie software, it is used for operations, administration, and maintenance. It consists of the following four components:

1. Monitoring and management agent: An OA&M Framework agent is embedded into all VoiceGenie software components so that the component can be managed and monitored by the OA&M Framework. The OA&M Framework agent allows monitoring/management information to be exposed via SNMP. In addition, the OA&M Framework agent provides up to date health status about the component.
2. Monitoring and management proxy: This is a process that runs on every VoiceGenie server. It is responsible for starting and monitoring the VoiceGenie processes as well as reporting system health information.
3. Monitoring and management engine: This part of the OA&M Framework provides the centralized data collection and statistics gathering and allows multiple systems in a deployment to be managed as a cluster.
4. System Management Console (SMC): This web-based interface allows users to view the current status and configuration of VoiceGenie components. Also, this interface can be used to view alarms, create graphs and perform installations.

1.3 Application Servers

Along with the SRM Client, an application server is required to serve grammars generated by the Media Platform to off-board ASR servers. An Apache server is used on Linux for this purpose, while an IIS server is used on the Windows Platforms.

The application server must be configured to server three types of grammars generated by the Media Platform.

1.3.1 Hotkey Grammars

Hotkey Grammars are used to match the UNIVERSALS properties, for the “Help”, “Cancel”, and “Exit” hotwords. These are installed in the directories `/usr/local/phoneweb/grammar/<engine name>/hotkey/` and `C:\VoiceGenie\MP\grammar\<engine name>\hotkey\` respectively for Linux and Windows, for each of the engines to be used by the Media Platform. The default hotkey grammars may not contain the correct string for cancel, help or exit depending on the language needed. Correct strings should be added for languages not provided by default.

1.3.2 Builtin Grammars

Some engines do not support VoiceXML builtin grammars internally on the engine side. For these engines, the builtin grammars also reside on the Media Platform, in the directories `/usr/local/phoneweb/grammar/<engine name>/` and `C:\VoiceGenie\MP\gramamr\<engine name>\` for Linux and Windows respectively.

1.3.3 Inline and Implied Grammars

Inline grammars and Implied grammars (menu/option grammar) are generated by Media Platform dynamically. These grammars are temporarily written in `/usr/local/phoneweb/tmp` and in `C:\VoiceGenie\MP\tmp` for Linux and Windows respectively.

1.3.4 Saved Utterances

In the MRCP Native integration, the SRM Server could be located on a machine that is separate from the Media Platform. In such a case an application server is required to serve saved utterance data created by the SRM Server or ASR engine. An Apache server is used on Linux for this purpose, while an IIS server is used on Windows Platforms.

Chapter

2 SRM Client

The SRM Client is a component that interacts with speech resource servers using the MRCP protocol to access the speech resources required by the Media Platform. It conveys data from the Media Platform to the speech resource servers. It also provides fail-over features by monitoring speech resources.

2.1 Provisioning Speech Resources

Speech Resources that are accessed by the SRM Client are provisioned via the SMC. During installation and deployment Speech Resources are automatically created. The actual provisioning data is stored in the centralized database of the OA&M Framework, however, it is also stored locally on the machine in the Call Manager provisioning file, which is located at `/usr/local/phoneweb/config/cm_provision.dat` under Linux and `C:\VoiceGenie\mp\config\ cm_provision.dat` under Windows.

A typical speech resource will have the following format in the `cm_provision.dat` file.

```
<entry id="302" type="6" name="Speech Resources" >
<param name="vrm.client.resource.name" value="SPEECHWORKS"/>
<param name="vrm.client.resource.address" value="10.0.0.78"/>
<param name="vrm.client.resource.port" value="22000"/>
<param name="vrm.client.resource.uri"
   value="rtsp://10.0.0.78/recognizer"/>
<param name="vrm.client.resource.type" value="ASR"/>
<param name="vrm.client.SendVGParams" value="true"/>
<param name="vrm.client.HotKeyBasePath"
   value="/vggrammarbase/speechworks/hotkey"/>
<param name="vrm.client.HotKeyLocalPath"
   value="$VGROOT/grammar/speechworks/hotkey"/>
<param name="vrm.client.TransportProtocol" value="MRCPv1"/>
</entry>
```

The provisioning data of a speech resource describes the specific configurations that the resource requires. User can change the default value for the provision data to meet special requirement. User can also provision speech resource to take advantages of the fail-over feature that SRM client offers.

2.1.1 Provisioning Data

The following screen capture illustrates a typical Speech Resource Provisioning.

Each speech resource has its identity data, they are the `Speech Resource Name`, `Hostname/IP` and `Port`. The identity data cannot be modified for each deployed resource. The `Speech Resource Name` is the name of the TTS/ASR engine which can be specified by property `TTSENGINE/ASRENGINE` by an application. The `Speech Resource Name` only allows upper case alphanumeric, dash, dot, and underscore. It is recommended that the application append the query string to distinguish an MRCPv1 and an MRCPv2 engines. For example:

<property name=”TTSENGINE” value = “REALSPEAK?MRCPv1/>

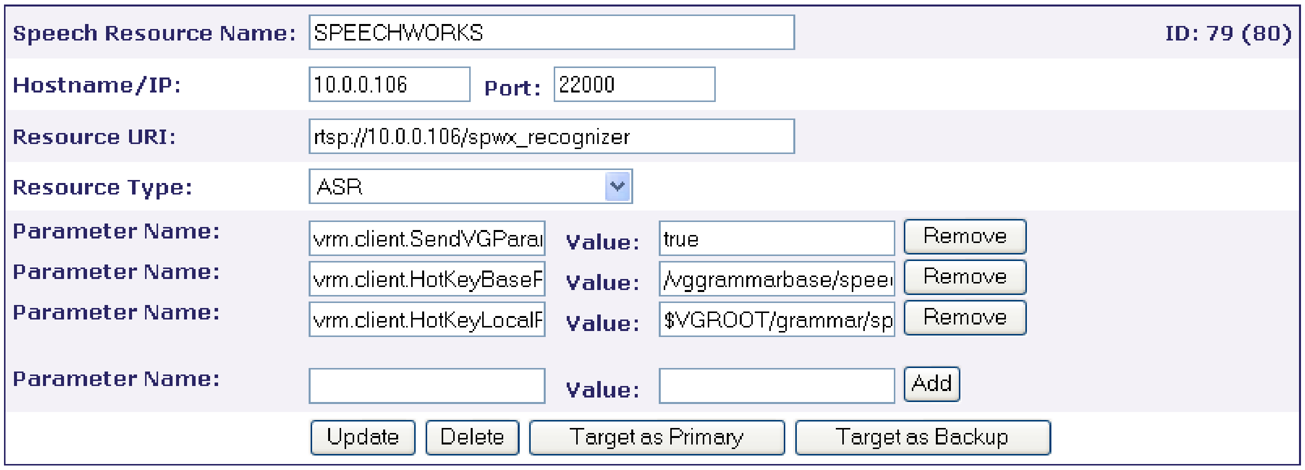
<property name=”ASRENGINE” value = “SPEECHWORKS?MRCPv2/>

If a VoiceXML application does not use the query string to specify the preferred MRCP protocol, the platform will find the required ASR/TTS engine in all the MRCP protocol modules that are defined by vrm.client.modules in the Call Manager configuration file (callmgr.cfg). By default, the value of vrm.client.modules is "MRCPv1 MRCPV2". The platform will always first try to allocate the engine in the MRCPv1 module. If the engine is not provisioned as an MRCPv1 engine, it will try to find the engine in the MRCPv2 module.

If the platform is only provisioned with MRCPv2 ASR/TTS engines, it is recommended that the user configure vrm.client.modules to "MRCPV2" only. In this way, the platform can find the ASR/TTS engine in MRCPv2 module only without first trying on the MRCPv1 module.

`Resource URI` defines how this resource should be accessed from a MRCP client. A resource URI is normally defined in the resource configuration file on the server side. The value of this data must match with the value defined by the server.

`Resource Type` defines if the server is a TTS or an ASR.



Speech Resource Name:	SPEECHWORKS			ID: 79 (80)
Hostname/IP:	10.0.0.106	Port:	22000	
Resource URI:	rtsp://10.0.0.106/spwx_recognizer			
Resource Type:	ASR			
Parameter Name:	vrm.client.SendVGPara	Value:	true	Remove
Parameter Name:	vrm.client.HotKeyBaseF	Value:	/vggrammarbase/spee	Remove
Parameter Name:	vrm.client.HotKeyLocalF	Value:	$VGROOT/grammar/sp	Remove
Parameter Name:		Value:		Add

Users can freely add the Per-EngineType parameters listed under the `Speech Resources Configuration Section` in each table, by typing the name (note always with prefix `vrm.client.`) in the `Parameter Name` box and its value in the `Value` box and clicking the `Add` button to confirm.

Users can remove an existing parameter by clicking the `Remove` button beside the parameter.

Clicking the `Update` button will update the database as well as the `cm_provision.dat` for all the changes. The user must re-target the resource to a Media Platform to ensure the change takes effect on that Media Platform.

Clicking the `Delete` button will un-target from all the already targeted Media Platform and will delete the Speech Resource data from database and `cm_provision.dat`.

By clicking the `Target as Primary` button, users can target/un-target the Speech Resource as a primary engine to a Media Platform.

By clicking the `Target as Backup` button, users can target/un-target the Speech Resource as a backup engine to a Media Platform.

For a full list of all provisioning parameters, please refer to the *SRM System Reference Guide,* “SRM Client Speech Resource Configuration Parameters” Section.

2.1.2 Provisioning and Fail-over

The SRM client provides fail-over ability by using the speech resources provisioning data.

From the provision data, the SRM Client maintains two lists of configured servers for each ASR/TTS engine: a list of primary servers and a list of backup servers. Primary and backup speech resources with the same `Speech Resource Name` will be grouped as a resource pool.

The SRM client uses the DECRIBE method to check heart beat of all the servers.

From SRM client point of view, a speech resource is marked as failed if:

1. Socket disconnection is detected. This is not applicable to connect-per-setup server.
2. The resource does not reply to the ping message within `vrm.client.ping.timeout` value.
3. Failed in a session SETUP or INVITE request.

When a request is made to the SRM Client to start a new recognition/synthesis session with an engine, the SRM Client will round robin through the list of primary servers which are consider “active” to select a server to send set up a new session with.

If request a Speech Resource failed in one of the following situations, the SRM client will try the next available engine.

1. The selected Speech Resource responded with any code other than the 200 (OK) for a SETUP or an INVITE request

2. Sending SETUP or INVITE message failed (at socket level)

3. Connecting to a connect-per-setup server failed

As a result a failed resource will be quarantined (marked as “non-active”) for 10 sec.

If none of the primary servers is active, the backup server list will be used by the SRM Client to allocate a server to set up a new session. In other words, the backup servers are not used until all the servers in the primary list are considered inactive. The backup resources have the same fail-over capability as the primary ones.

The number of times that SRM Client retries with the available resources is controlled by two aspects, whichever one happens first:

1.All the resources are exhausted (disconnected, ping failed, or quarantined).

2.OpenSession timeout goes off.

2.1.3 Access Provisioned Speech Resources

VoiceXML application can use property TTSENGINE and ASRENGINE to access provisioned TTS or ASR engines. By appending the query string, the application can distinguish an MRCPv1 and an MRCPv2 engines. For example:

<property name=”TTSENGINE” value = “REALSPEAK?MRCPv1/>

<property name=”ASRENGINE” value = “SPEECHWORKS?MRCPv2/>

The TTSENGINE property allows the application to access the REALSPEAK TTS engine by using MRCPv1 protocol; and the ASRENGINE property allows the application to access the SPEECHWORKS ASR engine by using MRCPv2 protocol.

2.2 HTTP Access to Grammars

Off-board ASR servers need to fetch grammars (such as the hotkey and built-in grammars described above, as well as inline and implicit grammars dynamically generated by the applications) from the Media Platform. As a result, a web server is installed and configured on the Media Platforms. An Apache server is used on Linux for this purpose, while an IIS server is used on Windows Platforms.

2.3 Directory structure

This section describes the directory layout of SRM client component on the VoiceGenie platform.

2.3.1 The software

Starting from the VoiceGenie 7 release, this shared library is located in `/usr/local/phoneweb/lib/libvrmclient.so` (Linux) and `C:\VoiceGenie\mp\lib\libvrmclient.dll` (Windows).

Starting from VoiceGenie 7.2.1, in Linux the SRM client share libraries are located in :

```
/usr/local/phoneweb/lib/libmrcpclient.so

/usr/local/phoneweb/lib/libmrcpv1client.so

/usr/local/phoneweb/lib/libmrcpv2client.so

In windows:

C:\VoiceGenie\mp\bin\libmrcpclient.dll

C:\VoiceGenie\mp\bin\libmrcpv1client.dll

C:\VoiceGenie\mp\bin\libmrcpv2client.dll
```

2.3.2 Configuration

The SRM client obtains its configuration information from its respective module (i.e. the Call Manager, VoiceXML Interpreter). The configuration parameters for the SRM client are embedded in each of their configuration files, which can be accessed through the SMC interface.

2.3.3 Hotkey and Builtin Grammars

Hotkey Grammars are used to match the UNIVERSALS properties for the “Help”, “Cancel”, and “Exit” hotwords. These are installed in the directories `/usr/local/phoneweb/grammar/<engine name>/hotkey/` and `C:\VoiceGenie\MP\grammar\<engine name>\hotkey\` respectively for Linux and Windows, for each of the engines to be used by the Media Platform.

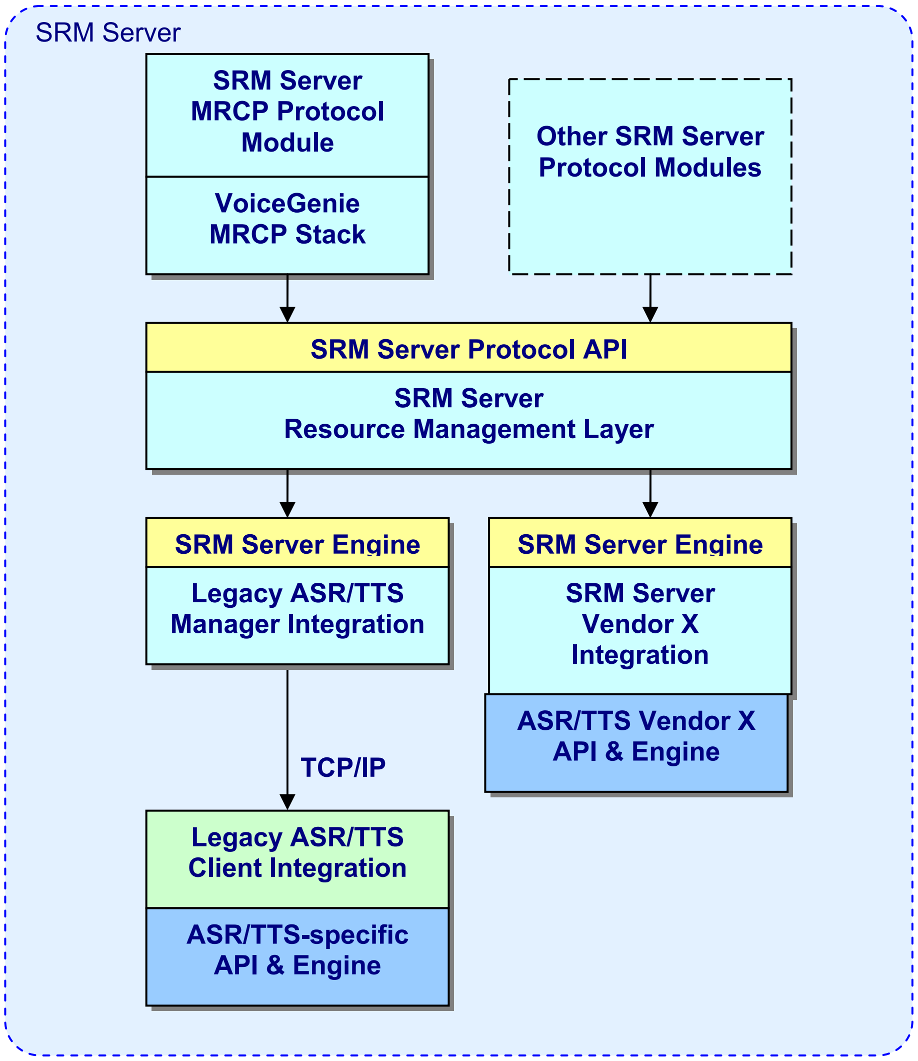
For some engines, they do not support built-in grammars internally on the engine side. For these engines, the built-in grammars also reside on the Media Platform, in the directories `/usr/local/phoneweb/grammar/<engine name>/` and `C:\VoiceGenie\MP\grammar\<engine name>\`.

Chapter

3 SRM Server

3.1 Architectural Overview

The SRM server is an independent process used to handle Speech Recognition and Synthesis requests in MRCP Native configuration as illustrated in Fig.3 in 1.1 Functional and Architecture Overview. The SRM server creates and executes other processes to perform the recognition and synthesis, which can be illustrated by the following diagram:


SRM Server
SRM Server MRCP Protocol Module
VoiceGenie MRCP Stack
Other SRM Server Protocol Modules
SRM Server Protocol API
SRM Server Resource Management Layer
SRM Server Engine
Legacy ASR/TTS Manager Integration
SRM Server Engine
SRM Server Vendor X Integration
ASR/TTS Vendor X API & Engine
TCP/IP
Legacy ASR/TTS Client Integration
ASR/TTS-specific API & Engine

For each ASR/TTS product VoiceGenie has integrated with, there is a client process that works with the SRM server that acts as a bridge between the SRM server and the ASR/TTS product itself. The advantage of this approach is that, whenever changes are needed in the integration of one of the partners, or a support for ASR/TTS products are added/deleted, it should have minimal effects on the integration of other ASR/TTS products.

As was mentioned at the beginning of this document, the SRM Server communicates with the SRM Client using MRCP. In addition to starting, monitoring and managing these ASR/TTS clients, the SRM server accepts MRCP requests for ASR/TTS resources, and forwards these requests to the appropriate ASR/TTS clients. The communication between SRM Server and the ASR/TTS client uses a proprietary VoiceGenie messaging format. However, in order to continue support for functionalities currently not available by the MRCP protocol, the SRM Server can accept some vendor-specific parameters so such functionalities are maintained.

3.2 Directory Structure

As a SRM server can be deployed in a separated system from Media platform, we first describe the directory structure to give users an overview where the SRM server software and configuration files are located.

This section describes the directory structure for the SRM server (Linux version). For the Windows version, simply replace the root install directory of the Linux version (`/usr/local/srm-server/`) with the root install directory (`C:\VoiceGenie\srm-server\`).

The following directories are used in for the SRM Server:

Directory	Description
/usr/local/srm-server/bin	This directory contains the executable binaries for the SRM server, as well as all the ASR/TTS clients to be started by the SRM Server.
/usr/local/srm-server/lib	This directory contains the shared libraries to be loaded by the SRM Server.
/usr/local/srm-server/config	This directory contains the configuration files to be used for the SRM server as well as all the ASR/TTS clients.
/usr/local/srm-server/logs	This directory is where the logs from the SRM server as well as the ASR/TTS clients are written.
/usr/local/srm-server/tmp	This directory is where the SRM Server will write its temporary files.

3.3 Application Server

Using the convention established by the MRCP standard, the SRM server provides the utterance from a recognition session by way of an HTTP URL. As a result, a web server is installed and configured on the Media Platforms. An Apache server is used on Linux for this purpose, while an IIS server is used on Windows Platforms.

3.4 Provisioning

Via the OA&M Framework SMC, users can provision the SRM server to make use of the various ASR and TTS engines by managing the corresponding client process. The provisioning information resides in the `/usr/local/srm-server/config/srmserver.cfg` file. This file should not be modified directly.

The provisioning parameters are organized in a tree-like structure, where the top-level parameter(s) are used to define the other parameters to be used and read.

Please refer to the *SRM System Reference Guide* for a full list of all SRM Server, ASR/TTS clients and OSR 3.0 configuration parameters.

3.5 Metrics and Logging

Metrics data is currently written to file(s) on the platform. All Metrics/logging information can be found in the `pw_metricsfile` under `/usr/local/phoneweb/logs` (linux) or `($INSTALLROOT)\mp\logs` (windows) directory.

Each record has the following fields:

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

Note: An ASR metrics has a session id suffixed with `-asr` to VG call ID.

A TTS metrics has a session id suffixed with `-tts` to VG call ID.

A state in an ASR metrics can be one of `UNAVAILABLE`, `INIT`, `INIT_READY`, `INIT_LOADED`, `IDLE`, `USED`, `LOADING`, `LOADED`, `STARTING`, `RECOGNIZING`, `STOPPING` and `UNLOAD`.

A state in a TTS metrics can be one of `UNAVAILABLE`, `IDLE`, `USED`, `SPEAK`, `SPEAKING` and `STOPPING`.

Please refer to the *SRM System Reference Guide* for the list of all SRM Server entries, their causes and recommended actions.

3.6 CLC and SMC Operations

3.6.1 Starting/Stoping the SRM Server

The following command can be carried out in the CLC to stop the local SRM Server:

```
CLC> stop srmserver - -
```

The following command can be carried out in the CLC to start the local SRM Server:

```
CLC> start srmserver - -
```

Note: An SRM Server should ideally be explicitly stopped only if there is no active ASR or TTS session in progress, otherwise the Call Manager process may restart.

3.6.2 Change Operational State of the SRM Server

The srmserver has the following three operation states:

`ACTIVE:`

In this state srmserver accepts client request, allocates right resource and maintains a media (TTS/ASR) session.

`SUSPENDED:`

In this state the srmserver continues to service the existing sessions, until the sessions complete. At the same time it will not accept new requests from client.

`DISABLED:`

In the state srmserver will terminate all the on going sessions and not accept any new requests.

```
CLC> suspend srmserver <host> <instance> <false>
```

to transfer vermserver from either `DISABLED` or `ACTIVE` to `SUSPENDED`

```
CLC> suspend srmserver <host> <instance> true
```

to transfer sermserver from either `SUSPENDED` or `ACTIVE` to `DISABLED`

```
CLC> resume srmserver <host> <instance>
```

to transfer sermserver from either `SUSPENDED` or `DISABLED` to `ACTIVE`

The SMC has the similar operations as the above CLC commands in the `Operation > Start/Stop Software` page. The only difference is that SMC cannot transfer the srmserver state to `DISABLED`.

3.6.3 Other CLC Commands

`clearstats srmserver <localhost> <instance>`

to reset all the statistic data of each host resource and module.

`clienttrace [service] <host> <instance> [asr/tts engine name] <lev>:`

to enable and disable trace of a hosted TTS/ASR client process.

where:

`[asr/tts engine name]` can be any of the engines displayed in the srmserver health information.

`<lev>` must be `enable` or `disable`.

Health information about the SRM Server can be retrieved using the CLC `health srmserver` command or by clicking on the colored light for the SRM Server in the SMC’s `Status Monitor` page. The following information is provided:

```
Health for SRM Server (srmserver) on <IP of the SRM Server>
Started: <timestamp when it started>
Status: <ONLINE or OFFLINE>
 Engine: <ASR Engine Name>
 URI: <RTSP URI for the ASR Resource>
 Available: <Number of the ASR ports available>
 Total: <Total Number of the ASR ports available>
 Peak: <Peak/Maximum ASR ports utilized>
 Died: <Number of ASR clients died>
 Recognition Failures: <Number of failed Recognition requests>
 Recognized: <Number of successful Recognition requests>


 Engine: <TTS Engine Name>
```

```
URI: <RTSP URI for the TTS Resource>
Available: <Number of the TTS ports available>
Total: <Total Number of the TTS ports available>
Peak: <Peak/Maximum TTS ports utilized>
Died: <Number of TTS clients died>
Synthesis Failures: <Number of failed Synthesizer requests>
Synthesis Success: <Number of successful Synthesizer requests>
```

Chapter

4 SRM RTSP-TTS Engine

The Real Time Streaming Protocol (RTSP, described in `http://www.ietf.org/rfc/rfc2326.txt`) is a popular protocol for streaming real-time media from a content server to listeners. VoiceGenie Media Platform will communicate with the SRM Server to make MRCP TTS requests. The RTSP-TTS Server then translates the incoming MRCP TTS requests into RTSP PLAY requests and forwards them to the RTSP Stream Server.

For details regarding the SRM RTSP-TTS Engine, please refer to the *VoiceGenie 7 RTSP User's Guide.*

Chapter

5 SRM TTY/TDD Engine

TTY, also known as TDD, enables people with hard hearing capability to communicate over telephone lines. These instruments use a typewriter style with a display to allow users to communicate over text messages. TDD/TTY is based on frequency shift keying (FSK), in which data is transmitted character-at-a-time, with a 5-bit character set, using a pair of frequencies (1800Hz and 1400Hz) to represent zero and one respectively. TDD/TTY operates at two rates; one at 20ms per bit (50 baud), and the other at 22ms per bit (45.45 baud).

Note: Only the mulaw audio format is supported for TTY/TDD

For details regarding the SRM TTY/TDD Engine, please refer to the *VoiceGenie 7 TDD User's Guide.*

Appendix

A ASR/TTS Specific Information

This section contains information that is specific to ASR and TTS engines supported by VoiceGenie.

A.1 Nuance MRCP 1.0

Nuance MRCP 1.0 requires the use of the Nuance 8.5 ASR Engine or greater. As a result, customers who are using Nuance 8 will need to migrate to the newer version when upgrading to VoiceGenie 7 or greater. During this migration phase all existing Nuance precompiled or Nuance Grammar Objects (NGO) will need to be recompiled for Nuance 8.5.

A.2 AT&T Watson ASR Server

Starting from Watson 6.1, AT&T Watson ASR server supports the MRCP v1 protocol with the following limitations:

1. Slot is not supported by Watson 6.1 MRCP direct server.

   The following grammar defines a slot “correct” in Watson MRCP native engine. The slot is NOT supported by Watson 6.1 MRCP direct server:

   <default> = _{ correct paper }_;

   In order to get an interpretation of the slot “correct” Watson 6.1 MRCP direct server requires the grammar in the following format:

   <default> = _{ "$='correct'" paper }_;

2. Application shadow variable .lastresult$.info is not supported by Watson 6.1 MRCP direct server.
3. Grammar fetch timeout is not supported by Watson 6.1 MRCP direct server.

   Watson 6.1 MRCP direct server has a hardcoded 3 second time for fetching a grammar.

4. Neither Watson MRCP direct server nor MRCP native server supports hot-word barge-in.

Revision History

Version	Date	Change Summary
1	August 31st, 2004	Initial release
2	Dec 1st, 2004	Added VRMProxy Section
3	Feburary 16th, 2005	Added Moncton Provisioning Section
4	March 11th, 2005	Additional Updates for VoiceGenie 7
4.1	March 29th, 2005	Revised version
4.2	April 13th, 2005	Final Revision for VoiceGenie 7
5	March 2nd, 2006	Updates for VoiceGenie 7.1
5.1	September 5th,2006	Updates for VoiceGenie 7.1
6	September 21th,2007	Updates for voicegenie 7.2
6.1	March 5, 2008	Updates for voicegenie 7.2.1
6.2	Dec 12, 2008	Updated to fix ER**207720168**
6.3	June 12, 2009	Updated to fix ER218167652