

SPOT SIP Engine



System Overview



Document Number: B00002011020909
For SPOT Release: V7.0.0
Date Created/Updated: 1 April 2011

1225 L Street, Lincoln, NE 68508
Ph:402.476.8786 Fax:402.476.7473
iisupport@iivip.com www.iivip.com



**Interact
Incorporated**
Software Systems

Table of Contents

1. Introduction	3
1.1. Executive Overview	3
1.1.1. Interact's Voice Products	3
1.1.2. Targeted Voice Applications	3
1.1.3. Design Objectives	4
1.2. Scope	4
1.3. Terminology and Acronyms	5
1.4. SPOT SIP Engine Reference Set	6
2. SPOT SIP Engines and IVRs	7
2.1. Architecture of an Interact IVR	7
2.2. Functional Components and Options	9
2.2.1. Interpreter Layer	9
2.2.2. Audio Call Switch Layer	9
2.2.3. Console	12
2.2.4. Middleware (ARCADE)	12
2.2.5. SNMP (An Optional Component)	13
2.3. Multi-server Environments: Flexible, Configurable, Scalable, Redundant	13
2.3.1. Scalability	13
2.3.2. Voice Application Redundancy	13
2.3.3. Speech Engine Integration	14
2.3.4. Media Gateways - Interfacing with the TDM World	14
2.3.5. "Intelligent Peripheral" in a Carrier Infrastructure	15
2.3.6. Realize - Centralized Statusing and Monitoring Application	16
3. Hardware, Capacity and Performance Considerations	17
3.1. Linux Distribution and Server Requirements	17
3.2. SPOT SIP Engine - IVR Functionality	17
3.3. Capacity Planning / Hardware Requirements	18
4. Your Voice Application - Using the SPOT SIP Engine	20
4.1. VoiceXML Ecosystem	20
4.2. Application Portability	21
4.3. Interact as Your Technology Partner	22
4.4. Take our SPOT SIP Engine Out for a Run!	23
Appendix A. SPOT SIP Engine Specifications	24

1 Introduction

1.1 Executive Overview

The SPOT SIP Engine, with its VoiceXML and Call Control XML (CCXML) Interpreters, is a high performance fully compliant "state-of-the-standards" Voice over IP telephony software system using SIP Call Control and host media processing for all audio management. This voice engine is the basis for high performance production voice applications and systems such as:

- Interactive Voice Response (IVR) systems (server or cluster of servers) in fixed/mobile carrier infrastructures.
- Combined Application Server/Media Server for IP Multimedia Subsystems (IMS) per the 3GPP (3rd Generation Partnership Project) architecture.
- IVRs in Call Centers and other enterprise environments.
- IVRs used with (traditional or IP) PBXes as automated attendants and/or voicemail servers, or as an IP-PBX itself.

This document is a high level overview of Interact's SPOT SIP Engine, providing:

- An introduction to the engine's capabilities for the new SPOT User
- An easy technical introduction to the SPOT SIP Engine's features and capabilities for both the Communications Industry and the general Enterprise marketplace
- Specifications of the SPOT SIP Engine.

It is intended to be a high level standalone and readable reference - an introduction to the more specialized References and User Guides listed at the end of this Introduction section.

1.1.1 Interact's Voice Products

Interact uses the following terminology to distinguish its voice products:

- A *SPOT SIP Engine* is software used to create an IVR on a Linux server. It provides the interpreters that coupled with VoiceXML/CCXML application documents creates a voice application, along with the software interfacing CCXML with the external telephony world (VoIP or TDM, the latter via a media gateway), and software for managing all the audio. It is essentially a "software toolbox" for creating IVRs and other interactive voice systems.
- A *VIP (Voice and Information Processing) Media Platform or VIP System* is a complete voice application solution provided by Interact; it consists of: server hardware, Linux operating system, an earlier generation SPOT Voice Engine, along with all the necessary VoiceXML/CCXML application documents (scripts) for a specific application (generic or customized). It may optionally include telephony hardware.

1.1.2 Targeted Voice Applications

The SPOT SIP Engine is targeted at these types of Voice Applications:

- In-bound: typically self-service applications where the user initiates a call, the system solicits information from the caller, and then either provides the requested information or transfers the call to a call center agent - examples are reordering prescriptions, obtaining bank balances, calling your health care provider ...

- Intelligent Peripheral: a specialized inbound application used in carrier infrastructures, where the IVR is a "specialized resource function" in an "Intelligent Network" architecture, or as a combined application server/media server in the IMS portion of the 3GPP architecture
- Out-bound: the system initiates calls to users - examples are notifications, infomercials, polling ...
- Bridging: where an inbound call results in an outbound call being placed, and when answered the called individual is bridged to the inbound caller - the classical examples are the automated attendant, voicemail system, or an IP-PBX
- Conferencing: several callers are placed into a teleconference.

The intended users and deployments are:

- System Integrators providing voice applications for Fixed/Mobile carrier infrastructures, whether in IN or IMS infrastructures
- Value Added Resellers (VARs) creating/providing voice applications/systems for vertical markets
- Hosted IVR and IP-PBX servicers, organizations providing hosted voice applications for their clients
- Enterprise organizations (IT, Call Center, other) providing facilities for voice applications in their enterprise, whether they or another department are responsible for writing the application scripts (VoiceXML/CCXML documents)
- Web sites where voice applications play a role in content delivery
- Casual users wishing to explore VoiceXML (we recommend they use the SPOT Test Portal for that purpose)
- Interact itself uses SPOT-based voice technology as the foundation of its VIP systems.

1.1.3 Design Objectives

Scanning the targeted users and applications above, one can infer the underlying design objectives for a SPOT SIP Engine:

- Support creation of full featured voice applications via VoiceXML and CCXML documents
- Provide compliance with the World Wide Web Consortium's standards and the IETF, and ECMA specifications used within those standards
- Support a highly scalable, distributed server environment, with standby server redundancy
- Provide the highest performance possible within the constraints outlined above

1.2 Scope

This document is part of the SPOT SIP Engine reference set, intended to provide not only a high level introduction to the SPOT SIP Engine's architecture, functionality and capabilities, but also a lead in to the more technical references and guides in that set. It defines and describes:

- The VoiceXML and Call Control XML (CCXML) Interpreters
- The SPOT SIP Engine components and optional extensions
- Utilization of third party Speech Engine's via MRCP V2.0 for Speech Recognition and Text-to-Speech
- Combining systems to create voice-applications ranging from a single server to large complex server farms or clusters with application redundancy for high density (TDM or VoIP) environments
- Performance and specifications.

1.3 Terminology and Acronyms

We use the following terms and acronyms throughout:

- SPOT - an acronym meaning "Speech Processing of Tomorrow", which Interact uses as the name or brand associated with its voice engine technology
- SPOT SIP Engine - a pure software voice application engine used to create an VoIP IVR on a Linux server. This is the 7th incarnation of Interact's SPOT voice engine technology
- IVR - an Interactive Voice Response system
- Console - the name of SPOT SIP Engine's web browser based system management application for status, comand and control of the IVR
- Channel - The unit of capacity of an IVR, and unless specified, always assume it means a full duplex voice channel enabling a dialog between IVR and user. (Note that other IVR vendors may use different terms such as "time slot" or "port", terms that arose from various contexts in the historical evolution of IVRs, but generalize to the channel concept)
- CPS - calls per second, a metric for the performance of an IVR, measuring call rate for a standardized voice application. An application may take anywhere from a few seconds ("Thank you for calling Interact Inc.") to several minutes for a call, so for a given channel count, cps and the application characteristics are the only measure of comparative performance
- W3C - the World Wide Web Consortium which sets the standards for all things web related, including XML based markup languages
- XML - Extensible Markup Language, a W3C recommendation for a general-purpose markup language used to share structured data across different information systems, particularly via the Internet
- VBWG - The Voice Browser Working Group, the folks in W3C creating the VoiceXML and CCXML standards, and the ancillary standards for grammars, semantic interpretation and speech synthesis incorporated within them
- CCXML - Call Control Extensible Markup Language, created by the VBWG
- VoiceXML (or VXML) - Voice Extensible Markup Language, created by the VBWG
- ECMA - The international industry association founded in 1961 (as the European Computer Manufacturers Association) now dedicated to the standardization of Information and Communication Technology (ICT) and Consumer Electronics (CE). It specifies the ECMAScript (Javascript) standard used as the computational component of the W3C standards
- IETF - Internet Engineering Task Force is the body in charge of all things technical regarding the Internet
- VoIP – Voice over Internet Protocol, descriptor for packet switched telephony over the internet
- SIP - Session Initiation Protocol, an IETF protocol for call control over the Internet
- RTP - Real Time Protocol, an IETF protocol for carrying packetized voice over the Internet
- TDM – Time Division Multiplex, descriptor for circuit switched telephony
- PSTN – Public Switched Telephone Network
- TTS - Text to Speech, typically provided by a 3rd party engine
- ASR - Automatic Speech Recognition, typically provided by a 3rd party engine
- MRCP - Media Resource Control Protocol, an IETF protocol for interfacing with ASR and/or TTS engines

- RHEL - Red Hat Enterprise Linux, a Linux distribution distributed by the Red Hat corporation
- CentOS - Community Enterprise Operating System, an open source Linux distribution based on RHEL systems
- RPM - i) Red Hat Package Manager, or ii) a file formatted for use by the Red Hat Package Manager
- SNMP - Simple Network Management Protocol, an IETF protocol for managing network entities
- Realize - Interact's web based network statusing and monitoring application

1.4 SPOT SIP Engine Reference Set

SPOT SIP Engine Quick Start Guide - B00002011020901

SPOT SIP Engine Installation, Configuration and Administration Guide - B00002011020902

SPOT SIP Engine Application Developer Reference - B00002011020903

SPOT SIP Engine Console User Guide - B00002011020904

SPOT SIP Engine Advisory Message Reference - B00002011020905

SPOT SIP Engine SNMP User Guide - B00002011020906

SPOT SIP Engine Realize User Guide - B00002011020908

SPOT SIP Engine System Overview - B00002011020909

2 SPOT SIP Engines and IVRs

This section describes the actual architecture of a IVR implementing VoiceXML and CCXML.

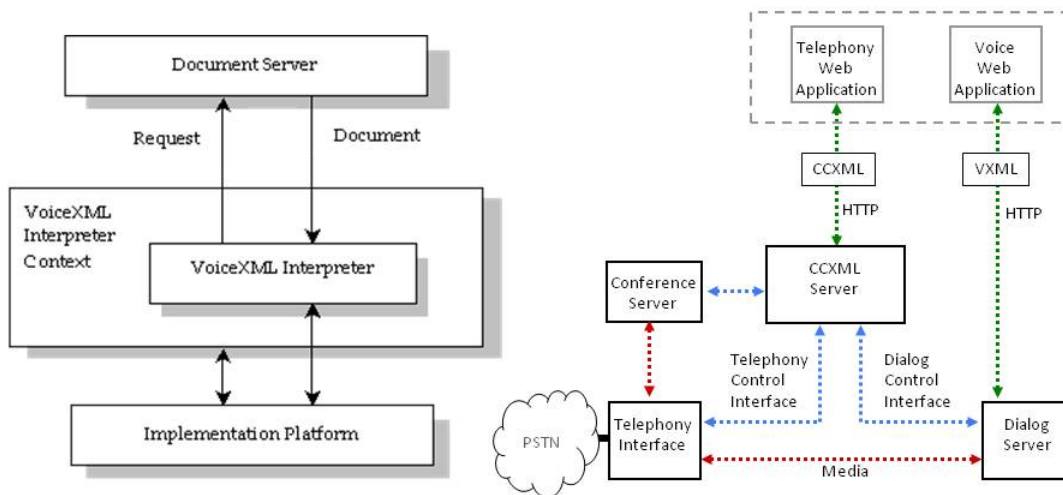
Additionally it discusses the functionality and capabilities of the SPOT SIP Engine, whether configured to work in a single-host system or in a complex multi-host systems environment – systems where voice and information processing are equally important.

Further, since many of Interact’s customers are telecommunications carriers, it discusses many of the “carrier-grade” requirements: performance, capacity, resilience, redundancy, scalability, reliability, configurability and flexibility.

2.1 Architecture of an Interact IVR

The Voice Browser Work Group (VBWG) in the W3C, the organization responsible for both the VoiceXML and CCXML standards, provided in each of those standards an abstract architectural diagram. These are meant to serve as a conceptual framework within which to discuss the language concepts of the standard:

VoiceXML and CCXML Architectures



Interact Inc. Software Systems has making, selling and deploying IVRs since the mid 80’s, providing turnkey solutions, professional services and IVR solutions and platforms, and in the VoiceXML based IVR space since the mid 2000s.

This section then puts flesh on the VBWG’s abstract architecture, from Interact’s perspective.

A typical Interact VoiceXML-based IVR consists of:

- a server in a distributed Ethernet based communications infrastructure
- running a Linux operating system, e.g. CentOS 5
- hosting a SPOT voice application engine
- and the VoiceXML/CCXML scripts or documents specifying the user application as files on that server.

Perhaps the easiest way to understand the architecture of a SPOT IVR is to view it as a "solution stack" - the set of subsystems or components needed to deliver a fully functional solution, e.g. a product or service.

A "Stack" View of an Interact IVR



These layers and/or components provide the following functionality

- User Application Layer - All the VoiceXML and CCXML documents/script files and pre-existing audio files or canned prompts that define/make up the voice application(s) being carried out by the SPOT IVR. The files themselves may be local to the server or fetched via HTTP, but in either case represent the "document server" in the abstract architectures
- XML Interpreters - in addition to the VoiceXML and CCXML interpreters, this layer provides the "VoiceXML context" and the equivalent parts of the "CCXML server"
- Audio Call Switch Management - A layer that handles 1) all telephony (call control, switching, conferencing and bridging) and 2) audio management (play, record, mix) including access to ASR and TTS Servers. This and all the lower layers make up the "implementation platform" for the VoiceXML, and the remainder of the CCXML architecture components including conference server, telephony interface, and all media management

- ARCADE Middleware - Interact's proprietary middleware (a layer between voice subsystems and the Linux operating system) that facilitates distributed processing, process management and provides common services used across many Interact products
- Linux OS - The Linux operating system and any vendor telephony software, toolkits or device drivers
- Hardware Layer - The X86 server or host for the IVR, along with Ethernet access and any actual telephony hardware (cards, blades, or switches) that may be required.

This generalized stack notion is applicable whether the IVR has internal telephony cards, is controlling an external switch, or has no hardware telephony components at all.

The SPOT SIP Engine is an example of the last, using SIP for call control (set up and tear down) and sending/receiving packetized audio using RTP.

2.2 Functional Components and Options

Lets look at these layers, and some other optional components, in a bit more depth.

2.2.1 Interpreter Layer

This layer provides:

- The VoiceXML and CCXML interpreters implementing our SPOT voice application technology
- Tools, DTDs, and syntax checkers to assist SPOT application developers in the creation of VoiceXML and CCXML applications (script documents)
- Console - a web based status and control application for status, command and control of the system, isolating the voice application developer from having to use the Linux command line interface
- Default and reference VoiceXML and CCXML applications used for testing, including validating a working initial system state when first installed
- Example Pack, a set of heavily documented VoiceXML and CCXML scripts that introduce new users of SPOT to script examples for common tasks
- An Apache extension to provide an HTTP event interface (supports appendices K and L of the CCXML specification)
- Files that assist Interact proprietary test tools

In the same sense that VoiceXML and CCXML are abstractions and independent of the specifics of the underlying telephony and audio transmission mechanisms, this layer maintains that independence.

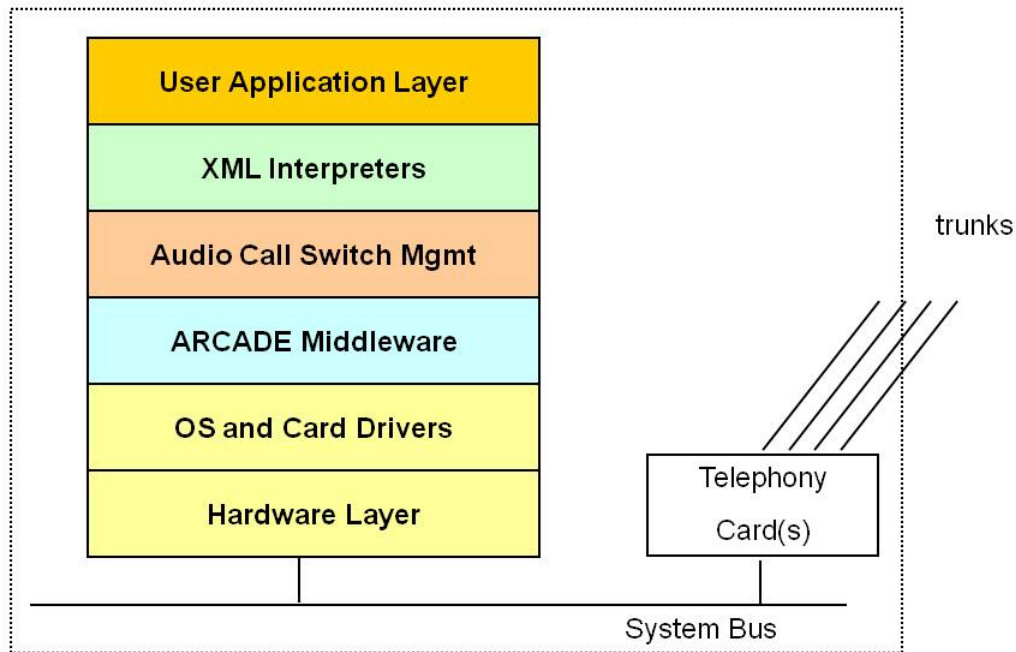
2.2.2 Audio Call Switch Layer

If the Interpreter layer is an abstraction, then the Audio, Call, Switch layer is where the real world of audio and telephony management come into play. This layer has responsibility for:

- Managing hardware and/or software that interfaces with the TDM/PSTN and/or VoIP worlds for both signaling and digital audio/bearer channels
- Answering/placing calls, routing digital audio streams, detecting DTMF tones, playing prompts, recording audio and playing it back, moving recorded audio to/from servers/file systems
- Managing the MRCP interface with speech engines for TTS (Text to Speech) and ASR (Automatic Speech Recognition), if third party speech engines are used.

Before VoIP telephony became prevalent, a typical standalone IVR was an x86 server containing telephony cards with one or more T1/E1 trunks.

Traditional IVR Model

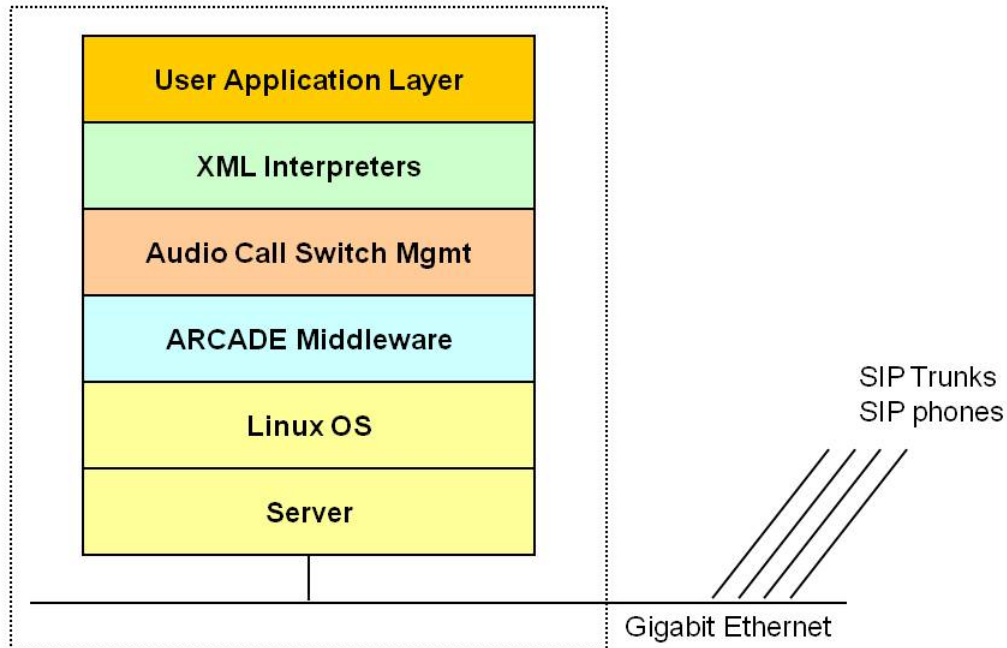


For example, if you have two or three telephony cards in a host server, an IVR could support about 240 - 360 channels. One would aggregate multiple hosts to increase the channel density, leading to a cluster or server farm of such systems - all doing the same voice application and all connected to some telephone switch for both signaling and bearer(audio) channels.

The SPOT SIP Engine creates an IVR that looks nearly identical to this traditional IVR, but instead is a pure software implementation harnessing the power of a multi-core multi-CPU server, without any additional add-in hardware. That is, it uses the "host media processing" capability of a modern processor to do all the audio work that in the past required DSPs (Digital Signal Processors) on cards inserted into the host server.

This IVR uses SIP for call control or signaling (call set up, tear-down, bridging), and packet-switched audio (packetized digital audio transmitted over IP) in place of circuit-switched audio (digitized audio transmitted on T1 or E1 trunks).

A SPOT SIP Engine IVR



The ACS layer then manages full duplex audio channels using RTP, and all audio mixing, conferencing and switching is done in software. A typical SPOT SIP Engine server can support up to 1400 channels depending on processor capability and application characteristics. As more capacity (channel count) is required, one simply adds additional servers.

Bridging calls simply means incoming audio on one channel is routed out on another. If an incoming call is to be bridged to a call leg on another server, its a simple matter with SIP to move a call from one server to another. One is not encumbered by fixed circuits as in the TDM case.

Interact's SPOT SIP Engine was re-designed from the ground up, with a brand new set of interpreters but on a very established software base.

The genesis for the redesign was two-fold:

- first, be conformant with all the relevant standards (VoiceXML, CCXML, and the W3C Speech Interface Framework),
- then obtain the highest performance within the above constraint (all SPOT-based products reflect a concern for performance).

2.2.3 Console

Console is the name of SPOT SIP Engine's web browser based system management (status and control) application. Using a browser, system management personnel or voice application developers can:

- Stop, start and restart the SPOT SIP Engine
- Obtain status and activity counts of the Engine's various components (Interpreters, active CCXML sessions, active VoiceXML dialogs, MRCP sessions, conferences ...)
- Obtain status of the host server (memory, CPU utilization, Network utilization)
- View, edit and update the system's configuration file)
- View/manage component, application and exception logs
- Create/manage system diagnostic files (normally forwarded to support)
- View the current Interact license file
- View online documentation (as HTML or as pdfs).

Note that a server hosting a voice application built on the SPOT SIP Engine, once set up and installed, can be managed without resorting to Linux commands. This enables:

- Deployment in enterprise environments without requiring high skill levels in Linux
- Voice application developers to test and debug an application remotely and/or without having to have Linux skills
- Service organizations to provide access to virtual machines running the SPOT SIP Engine where the voice applications are created and managed by totally different organizations/companies.

2.2.4 Middleware (ARCADE)

The proprietary ARCADE middleware is a layer between software systems Interact builds on a Linux server, and the Linux operating system itself. It facilitates seamless distributed applications – both in their development and their operation. It provides an API that enables:

- process control
- inter-process communications
- resource sharing
- logging
- application redundancy

A key functionality of the middleware layer is its responsibility for all cross server communications. A process on a server that wants to send a message to another process need not be aware whether the target process is on its local host, or resides on another host. This is exploited in the SPOT SIP Engine's application redundancy support.

The middleware layer also provides:

- Standalone installation, configuration, upgrade and rollback functionality
- an HTTP interface for access to status information and logs (and used by the Console application)

- SNMP support – an optional component for the SPOT SIP Engine

A more detailed description of processes and tools is provided in the Installation, Configuration and Administration Guide, the Application Developer Reference, and the SNMP User Guide

2.2.5 SNMP (An Optional Component)

SNMP support is an optionally available component of the SPOT SIP Engine. With any of the many available SNMP applications, an IT organization can monitor specific values available in the SPOT SIP Engine, and set traps to trigger on specific events.

Interact uses the AgentX protocol to extend the snmpd daemon that comes with the Linux distribution. See the SNMP User Guide for details on the SPOT SIP Engine's MIB

The capability exists also to provide extensibility for user provided applications and subsystems.

2.3 Multi-server Environments: Flexible, Configurable, Scalable, Redundant

IVRs built using SPOT voice technology are designed to be deployable in distributed multi-host high-performance carrier-grade voice application production environments. (If they support that high stress environment, they can be used anywhere.) SPOT systems then must be flexible, configurable, highly scalable, and support redundancy/failover.

2.3.1 Scalability

Assume you have a voice application running on a server with the SPOT SIP Engine, and you decide to add additional capacity to your installation. The first step is to see if there is additional capacity in your existing server. If there is, its a simple matter to license additional channels and modify the configuration file to run those channels.

If not, then simply deploy a new server with the SPOT SIP Engine and license it.

All audio mixing, bridging, conferencing and switching is all done in software on the SPOT SIP Engine. If an incoming call is to be bridged to a call leg on another server, or placed into a conference that is active in another server, its a simple matter with SIP to move a call from one server to another. (Remember, one is not encumbered by fixed circuits as in the TDM case.)

2.3.2 Voice Application Redundancy

There are two types of voice application redundancy supported by IVRs running the SPOT SIP Engine:

- Total System Bandwidth – based on the concept of over-provisioning resources. If 10 systems are required for the minimum acceptable call application load, 11 systems are deployed as active call processors. If one of the 11 systems fails, then a minimum of 10 systems are still operating. In this scenario, all activities taking place on the failed system and any established calls are lost
- N+1 – based on the concept of having a spare system operating in standby mode (an extension of the Worker/Standby model). If 10 systems are required for the planned application load, then 10 systems are deployed as active (or Worker) call processors and an extra system is configured and deployed as a "Standby". If one of the 10 Worker systems fails, the Standby system steps in and recovers the established calls. This model is clearly superior from a caller experience standpoint.

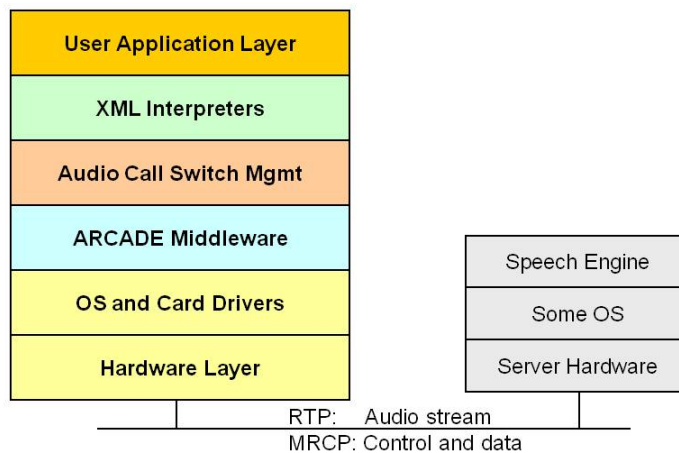
A redundancy management component intrinsic to the SPOT SIP Engine on each active IVR is constantly updating the standby server with call state information. If an active system should experience a failure, the calls in process are recovered by the standby server. All calls in a connected state are kept alive via the built in redundancy management. It is up to the CCXML application to rebuild conferences, bridged connections, and/or re-launch any VoiceXML dialogs when the *failover.complete* event is received. Depending on the sophistication of the voice application, one can restart the dialog, or simply replay the last prompt.

The N+1 redundancy model does require voice applications designed for call failover and recovery, to accompany the intrinsic redundancy support in the Engine.

2.3.3 Speech Engine Integration

The SPOT SIP Engine provides for integration of Speech Engines (third party software providing Voice Recognition and Speech Synthesis services) using the MRCP V2 protocol, an HTTP like request/response protocol between an IVR client and a Speech Engine server.

Adding a Speech Engine



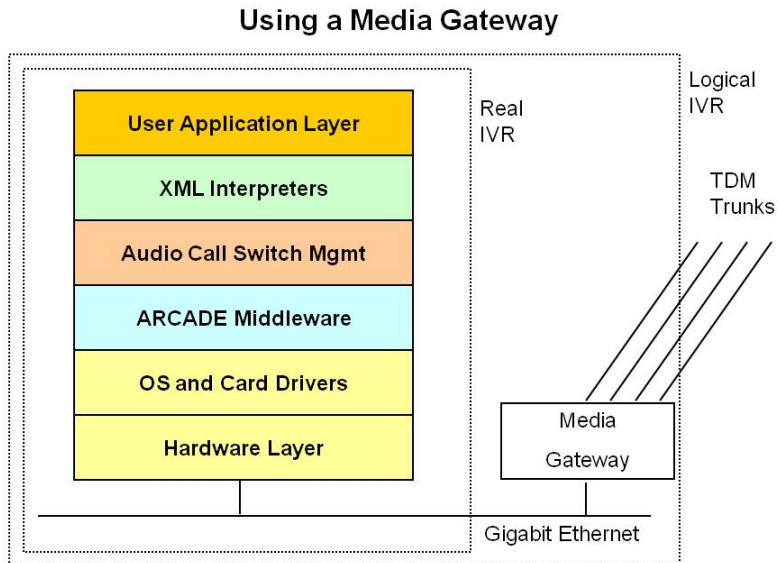
Speech Engines simply reside on the Ethernet along with the IVR.

A cluster of IVRs can use a cluster of Speech Engines, with available licensed channels in each cluster appropriate to the workload of the voice application desired.

2.3.4 Media Gateways - Interfacing with the TDM World

An IVR built on the SPOT SIP Engine, can serve as an IVR in the TDM world when coupled with a Media Gateway, a device available from a number of manufacturers which converts SS7 or ISDN signaling and the 64 kbps bearer channels, into SIP call control and RTP audio streams, and visa versa. In essence, the Gigabit Ethernet connection between the Media Gateway and the server hosting the SPOT SIP Engine is part of the internal

infrastructure of a "logical IVR" interfacing with the traditional telephony world (SS7 or ISDN).



Interact has performed interoperability testing with the TelcoBridges high density family of TMG Media Gateway products, and with the Dialogics IMG1010.

2.3.5 "Intelligent Peripheral" in a Carrier Infrastructure

(This functionality, available in earlier generation SPOT Engines, is planned for availability in late 2011.)

For carriers (fixed or mobile) that use an SS7 Intelligent Network infrastructure, a SPOT SIP Engine can be used as an SRF (Special Resource Function) or IP (Intelligent Peripheral), implementing the 4 key INAP CS1/CS2 or CAMEL phase 2 transactions:

- AssistRequestInstructions
- PlayAnnouncement
- SpecializedResourceReport
- PromptAndCollectUserInformation

as specified in ETSI TS 101 441 V7.0.1 – Release 1998.

Communication with the SCP in the IN infrastructure is provided by a component in the ACS layer that maps TCAP messages to/from CCXML events. The TCAP layer itself can be provided by:

- a licensed SIGTRAN stack integrated into the ACS layer (to be available as a priced option)
- the TCAP I/F of a TelcoBridges TMG Media Gateway (using either the SS7 links riding in the trunks, or SIGTRAN)
- the custom integration of the TCAP layer of a 3rd party SS7 stack (under a professional services contract)

2.3.6 Realize - Centralized Statusing and Monitoring Application

Realize is an optional Interact web application providing centralized status information capability on a dedicated web server for IVRs and other Interact Systems like Invigorate X , Interact's rating engine.

The Console application previously mentioned provides status for the IVR/server on which it resides. Realize provides similar and augmented capability for a collection of IVRs, whether concentrated at a single site, or scattered geographically in multiple locations. Although redundant with Console, it can be used with a single IVR server. The Realize web application itself runs on Apache and Tomcat, and requires network connectivity with all the systems it monitors

Its principal capabilities are:

- Obtain/display IVR statistics
- Notification Services - alerts triggered by occurrence of specified events.

Please see the Realize User Guide for a more detailed description of this tool.

3 Hardware, Capacity and Performance Considerations

In this section we discuss:

- Hardware and Operating System requirements
- Available IVR functionality
- Capacity Planning and Hardware Requirements
- Representative Performance

3.1 Linux Distribution and Server Requirements

The SPOT SIP Engine is designed to operate on the following Linux distributions (either 32 or 64 bit) running on x86 servers:

- Red Hat Enterprise Linux 5
- CentOS 5

Its implementation is designed to take advantage of modern microprocessors, with multiple cores per CPU, multiple CPUs per server.

There are interesting implications of this design.

Historically one spoke of IVR or server "farms" where either telephony hardware or VoiceXML performance limited the number of ports or channels an individual server could contain. Today we can talk instead about an "urban high-rise" analogy, where we have a large number of processing cores and lots of system memory, and add capacity by adding additional processor and memory capability. We are adding "height" or additional compute capability into server requirements, rather than "area" or an enlarging footprint by adding additional servers to meet the computational requirements.

Further, in a N+1 redundancy model, because we are using VoIP telephony rather than physical trunks, when a server fails and the standby takes over, we are able to preserve all established calls, and do not lose circuits/trunks tied to hardware on the failing server.

This makes for a lower total cost of deployment and cost of ownership.

3.2 SPOT SIP Engine - IVR Functionality

IVR functionality provided by the SPOT SIP Engine can be broken down into constituent tasks and ranked from basic DTMF functionality, enables Conferencing (a more complex telephony task), enables speech (requires a speech engine), or provides special purpose capabilities:

Basic Functionality

- Answer - Respond to an incoming call on a channel
- Play - Route pre-recorded audio to a client on an incoming call
- Detect - Determine DTMF tones pressed by client, determine if a caller has hung up
- Record - Capture spoken audio on inbound line, and save recorded audio as a file for later playback
- Call - Place an outbound call
- Bridge/Switch - Place an outbound call and switch the calls together.

Conferencing

- Setup - Create conference with designated host and multiple participants (ad hoc or prescheduled)
- Add - Place participant to existing Conference,
- Signal - Audio prompt to Conference as participants arrive/depart
- Mix - Combine audio of all but one participant, route to omitted participant
- Record - Capture conference audio output, and save as a file for later playback
- Manage - Participants are receive only, or receive/talk
- Tear Down - Remove the conference.

Advanced Functionality

- MRCP - Media Resource Control Protocol. Interface with a MRCP compliant speech engine on a separate speech engine server (the protocol is speech engine vendor agnostic)
- TTS - Text To Speech. Interface to a speech engine to convert a block of text to an audio stream played out over an audio channel
- ASR - Automatic Speech Recognition. Interface to a speech engine to convert spoken audio input on a channel to text data.

Special Purpose capabilities:

- Fax tone detection
- T.38 fax transmission and relay
- Live speaker or answering machine discrimination
- Video streaming

Planned future capability:

- Intelligent Peripheral - ability to function as a "special resource function" in a carrier's Intelligent Network (*late 2011*)
- Combined Application Server/Media Server in an IMS Infrastructure (*date tbd*)

3.3 Capacity Planning / Hardware Requirements

Capacity in an IVR context is the number of voice or bearer channels the system can support.

Capacity used to be gated by the trunk capacity of the telephony hardware used.

Today, the SPOT SIP Engine simply has a configuration parameter specifying the maximum number of VoIP full duplex channels to be supported on the server, although it is constrained by the physical memory and compute capacity on the host.

We recommend the following hardware minimums for general VoiceXML applications:

- 3 MB RAM per bearer channel, with a 512 MB RAM minimum
- For 32 bit systems, 13 MHz per channel, 2.5 GHz minimum, 1400 channels maximum, 200 CPS maximum
- For 64 bit systems, 19 MHz per channel, 2.5 GHz minimum, 1000 channels maximum, 200 CPS maximum

The total computational capacity of a server with two dual-core 2.4GHz CPUs is: 2 (CPUs) x 2 (cores each) x 2.4GHz (clock speed) = 9.6 GHz. To calculate the number of channels, 9600 divide by 13 is about 738, so as long as you had 738 x 3 MB of RAM (about 2.5GB) you could expect to run close to 738 channels of VoIP.

The above "rule-of-thumb" calculations are based on an internal "VoIP Ultimate Load Test" voice application. This application is characterized as an inbound/outbound dialer with playback, sending/detecting DTMF (RFC 2833), grammar matching, and a handful of web requests. Also built in to the above recommendations is maintaining about 40% head room on total compute capacity - that is, maintaining about 40% CPU idle time.

It also holds true for a "whisper" application that simply answers a call, plays a 4 second prompt, and hangs up. This application is typical of a carrier infrastructure application used to thank a customer for using their service, and is a stress test on call set up/tear down.

More or less hardware resource may be required depending on the actual application size and complexity and whatever other processing may be going on with the data solicited from a caller. Other factors that will impact calls per second are:

- Number of documents (CCXML/VoiceXML) in the call flow
- Fetching documents from a web server rather than from the host file system
- Using HTTPS instead of HTTP
- Fetching prompts/audio files from a server rather than from a local file system
- Using dynamically generated documents, rather than fixed documents with parameters
- Size of the VoiceXML/CCMXL documents
- Use of the <subdialog> tag in documents, and depth of nested sub-dialog calls.

Now, given that "the average number of channels in use" = "calls per second" x "average call duration" , we can use the above to infer:

- For given voice application characteristics (call rate, average duration), one determines the number of channels needed, and using the capacity recommendations above, one can compute hardware requirements (how big a server, or how many servers of a given size)
- Given the average call duration for an application and a server's channel capacity, one can compute the call rate a given server can handle,

For evaluating two IVRs doing the same application, CPS provides an excellent performance comparison metric:

- assuming they are each driven to a maximum call rate while holding comparable processor busy, and maximum channel capacity hasn't been reached on either, the one with the higher CPS is clearly yielding higher performance.
- holding both at the same call rate, the one with lower hardware requirement (processing capacity) while maintaining equal processor idle status is yielding higher performance

Note that doubling the number of cores in a server does not really double its capacity. Rather, experience says that 2 times cores = 1.85 x capacity, which can be neglected for up 4 cores as there's sufficient slop in the general recommendations above.

Finally, these recommendations assume we have reached equilibrium in the application; when an application is first begun on an IVR, there is a start-up cost until document and web page caching have reached a steady state.

4 Your Voice Application - Using the SPOT SIP Engine

If you already have existing voice applications in VoiceXML or if you are converting applications to VoiceXML from a proprietary implementation, the obvious question is how difficult is it to port them over to the SPOT SIP Engine, and is it worth it?

If you are implementing from scratch, what do you gain by implementing them with the SPOT SIP Engine rather than another platform?

Choosing to use the SPOT SIP Engine, you will have all the benefits that open standards bring, but you will also have:

- A pure software solution, not restricted by limitations of telephony cards in servers, or switches beside servers
- A solution that can exploit the latest generation of multi-core multi-CPU servers, whether 32 or 64 bit
- Future proofing - the SPOT SIP Engine is based on the signaling technology used in soft-switches, Next Generation Networks, IMS
- Usable in the TDM world simply by using a Media Gateway
- The performance usually associated with proprietary IVRs
- Be able to do more (calls per second) with substantially less resources (server hardware, operating systems licenses, floor space, power, operations personnel to manage a server farm, ...)
- Have a choice of Speech Engines (via MRCP V2), if your application is speech enabled
- Enjoy a reduced Total Cost of Ownership.

To help in that decision, the SPOT SIP Engine is available under Demo and Developer arrangements, in addition to Production licenses. And Interact's Professional Services is ready to implement your applications if desired.

In this final section, we will look at:

- The VoiceXML Ecosystem - a brief look at the ever growing set of standards used by and surrounding VoiceXML
- A look at the realities of porting applications to/between VoiceXML platforms
- The SPOT SIP Engine's implementation of the VoiceXML ecosystem
- SPOT and Interact as your Technology Partner

4.1 VoiceXML Ecosystem

The W3C Voice Browser Working Group's "Speech Interface Framework" consists of a growing number of inter-related specifications (all at various stages in the process, from first draft, late draft, last call, and recommendation). This section briefly describes the various roles each has in creating Voice Applications. A list of the various specifications, and their current status in the W3C standards process, is available at <http://www.w3.org/Voice/>.

The two key specifications, VoiceXML and CCXML are intended to operate together, but each is independent of the other. That is, VoiceXML can be used with some other mechanism for initiating dialogs, and has some minimalist call control contained in the specification. The CCXML specification can be used with other dialog management mechanisms besides VoiceXML.

VoiceXML 2.1 provides a small set of additional features, with complete backwards compatibility with the VoiceXML 2.0 specification. VoiceXML 2.1 achieved recommendation status in June of 2007.

These, and all the other W3C specifications in the Speech Framework, are XML-based application languages. And since such languages are scripts in ASCII text, software processes that execute them are generally called Interpreters, (although a VoiceXML interpreter is also called a Voice Browser, especially by the Voice Browser Working Group).

VoiceXML is designed to work with digitized audio streams - Automatic Speech Recognition (ASR) for input, and Text To Speech (TTS) for output - as well as with DTMF tones and pre-recorded prompts. Thus there are two additional W3C standards for the ASR side of voice processing, and one for TTS.

There is one other standard the VoiceXML Forum and the W3C's VBWG consider to be a key enabling technology, MRCP. It is not a markup language, but rather is a specification built on SIP and RTP, which is under the Internet Engineering Task Force (IETF) Speech Services Control (SpeechSC) working group, and not the W3C VBWG. The MRCP v2 specification defines a client/server interface to be used by Speech Engines for ASR and TTS, and thereby allows client devices, such as a voice browser, to interact with these resources in a standards-based, vendor-agnostic manner.

References for these standards follow (note that the references cited in the VoiceXML 2.0 specification not only point to earlier drafts of the actual recommendation, but modifies those specifications for its own use):

- Voice Extensible Markup Language (VoiceXML) Version 2.0 W3C Recommendation 16 March 2004 <http://www.w3.org/TR/voicexml20/>
- Voice Extensible Markup Language (VoiceXML) 2.1 W3C Recommendation 19 June 2007 <http://www.w3.org/TR/voicexml21/>
- Voice Browser Call Control: CCXML Version 1.0, W3C Candidate Recommendation 1 April 2010 <http://www.w3.org/TR/2010/CR-ccxml-20100401/>
- Speech Recognition Grammar Specification (SRGS) Version 1.0 W3C Recommendation 16 March 2004 <http://www.w3.org/TR/speech-grammar/>
- Semantic Interpretation for Speech Recognition (SISR) Version 1.0 W3C Recommendation 5 April 2007 <http://www.w3.org/TR/semantic-interpretation/>
- Speech Synthesis Markup Language (SSML) Version 1.0 W3C Recommendation 7 September 2004 <http://www.w3.org/TR/speech-synthesis/>
- Media Resource Control Protocol Version 2 (MRCPv2) draft-ietf-speechsc-mrcpv2 <http://datatracker.ietf.org/doc/draft-ietf-speechsc-mrcpv2/>

4.2 Application Portability

Application portability and vendor independence are really two sides of the same coin, and one of the benefits supposed to be attendant with open standards.

However, application portability isn't truly "real" in the VoiceXML specification:

- Several VoiceXML features are optional, their inclusion or exclusion is a platform dependent choice
- Proprietary extensions are necessarily built into the specification, again via platform-dependent:
 - session variables
 - properties
 - <object> tags.

So a more appropriate analog to "VoiceXML Portability" might be that entailed in porting an application from one Linux distribution to another.

The Forum itself recognizes this problem. From the VoiceXML Review, Volume 7, Issue 1 - April/May 2007 article titled "The 2006 VoiceXML Forum Survey":

"Platform portability remains a weakness. The standard is not simple, and some features can be interpreted in various ways. Thus moving to a new voice platform can require a non-trivial amount of porting. Proprietary extensions are also an issue."

Now set aside any incompatibility issues due to custom objects, properties and session variables.

The SPOT SIP Engine's VoiceXML is conformant to both V2.0 and V2.1 VoiceXML specifications. In house testing with the latest compliance suite shows we pass all mandatory tests for the required elements and all optional elements we have implemented. Obtaining the Forum's Certification is simply a matter of logistics and scheduling, and is targeted for second half of 2011. Therefore:

- Any voice application that runs on the SPOT SIP Engine will run the same on a VoiceXML 2.1 Forum Certified Platform that implements minimally the same optional elements
- A voice application that runs on the SPOT SIP Engine, restricted to use only VoiceXML 2.0 syntax, will run the same on any other VoiceXML 2.0 Forum Certified Platform, same restriction on optional elements.

The converse is also true. Applications that run on a Forum Certified platform:

- restricted to use only the SPOT SIP Engine's supported optional components
- will run the same on a SPOT SIP Engine.

When porting an existing VoiceXML application to the SPOT SIP Engine, it is very likely that the application will already fit within its VoiceXML , or any necessary change to do so will be straightforward. For specific details, please see the SPOT SIP Engine Application Developers Reference.

4.3 Interact as Your Technology Partner

Our interpreters in the SPOT SIP Engine are designed to function in a distributed (multi-host) and scalable environment - providing a pure software implementation for SIP based VoIP telephony.

As such, the SPOT SIP Engine differs significantly in certain design assumptions from those used by the designers of VoiceXML, so our interpreters' internals are very different from other interpreters' internals. It is those differences that yield our superior performance. These differences normally do not manifest themselves in VoiceXML applications or documents used in the typical way many VoiceXML documents are employed in the IVR world.

As stated in the Introduction (section 1), our core design objectives are:

- Support creation of full featured voice applications via VoiceXML and CCXML documents
- Provide compliance with the World Wide Web Consortium's standards and and the IETF, and ECMA specifications used within those standards
- Support a highly scalable, distributed server environment, with standby server redundancy
- Provide the highest performance possible within the constraints outlined above

To these, we can add additional design objectives and features oriented to the IVR and/or voice application developers:

- Bias for production environments and applications in large scale deployments,

- Deploy on the preferred IT operating system environment (Linux), with optionally available (and extensible) SNMP support
- Provide a web based status and management interface, Console, isolating the VoiceXML developer from having to be Linux specialist
- Support failover at the CCXML application layer, allowing for preserving established calls
- Provide application development support and related specifics in a reference guide suitable for professional IVR voice application developers, value added resellers and vertical market application specialists, and in-house enterprise specialists with responsibility for creating and maintaining their voice applications and their IVR's "voice user interface".
- Provide "carrier grade" support capability for integrators selling into the carrier market, with future-proofed SIP/VoIP capability mapped to CCXML events for answering, bridging, transferring, out-bound dialing, teleconferencing, with easy integration of user-provided processes
- Provide hosted service providers easy tools to implement and deploy IVRs for their customers.

When we look at our capacity and performance in terms of channels and calls per second, not only have we met/exceeded our objectives, but are confident we run circles around any other commercially available voice engine.

The impact this has on TCO (Total Cost of Ownership) is significant. We can support more capacity and higher call rates on a given host than any other interpreter. Rather than throwing server hardware, operating system licenses, floor space, power, and people to manage a large server farm for a specific IVR application, we accomplish the same application with substantially less hardware, floor space, and money.

4.4 Take our SPOT SIP Engine Out for a Run!

You can do this in several ways:

- It is available on the www.iivip.com web site for free download and trial
- A SPOT Test Portal is available for playing with, and/or for creating/testing a voice application, available using a web browser and a SIP phone
- Our Professional Services team can give you a quote for porting your application, or porting a document and performance testing it for you. Call 1.800.242.8649 or +1.402.476.8786.
- A SPOT SIP Engine Development Kit ("SDK") is available allowing Developers to create and test commercial applications (but excludes running them in a production environment).
- Interact also has a portfolio of complete solutions available on the VIP Media Platform (using an earlier version of Interact's SPOT technology). Call 1.800.242.8649 or +1.402.476.8786.

Appendix A - SPOT SIP Engine Specifications

SIP Call Control Features

- Inbound call acceptance
- Outbound call placement
- Call leg bridging
- Call rejection
- Call routing
- Call transfer
- REINVITE audio re-routing
- REFER support
- Registrar server support
- Proxy server support
- Authentication support
- Configurable timers
- Address restriction
- Early media support
- ISUP/SIP interworking (per SIP-I/SIP-T) for terminations/endpoints
- Configurable RTP port range

SIP Compatability/Interoperability

- Interoperable with the following SIP services:
 - Axvoice
 - Bandwidth.com
 - BroadVoice
 - BroadVox
 - SER/OpenSER
 - Asterisk
 - Metaswitch
- Interoperable with the following SIP Devices:
 - SJ Phone
 - Cisco/Linksys/Sipura
 - Polycom
 - Grandstream
 - X-Lite
- Interoperable with the following Gateways:
 - TelcoBridges Media Gateway

- Dialogics IMG1010
- Quintum Tenor VoIP Gateways
- Successfully tested against SIP Forum SIPConnect V1.0 specification (PBX)

IVR Functionality

- Answers Inbound Calls
- Plays prerecorded audio prompts/announcements
- Detects DTMF tones (via RFC 2833) in RTP stream
- Records incoming audio
- Plays back recorded audio
- Places outbound calls
- Generate DTMF(RFC 2833) in RTP stream
- Bridges audio between call legs
- Transfers calls (blind, bridge, consultation)
- Conference call management (setup, add/delete participants, announce tones, listen only or active participants, teardown)
- Conference audio mixing, record conference for later playback
- Speech Recognition (ASR)
- Speech Synthesis (TTS)
- Vendor independant ASR/TTS via MRCP V2
- Interoperable with following Speech Vendors:
 - Loquendo Speech Server (ASR/TTS)
 - LumenVox Speech Engine (ASR/TTS)
 - Nuance Recognizer/Vocalizer (ASR/TTS)
- Supports simultaneous usage of different vendor TTS and/or ASR engines
- Supports VoiceXML 2.0/2.1 IVR functionality
- Supports CCXML 1.0 IVR functionality
- Scales from one to thousands of ports/channels
- Configurable RTP port range

Audio/Media Processing

- Packet loss concealment
- Fixed gain control (per channel)
- Configurable jitter buffer
- Codec transcoding
- Voice activity detection
- Live speaker or answering machine discrimination
- Fax tones detection

- T.38 Fax send/receive
- H.263 Video Streaming to/from 3gpp files along with bridged calls

Codecs Supported

- G.711 uLaw (ITU-T unless stated otherwise)
- G.711 aLaw
- G.722 ("HD audio")
- G.726 ADPCM (32 kbps only)
- G.729a (requires license)
- GSM (RPE-LTP Full Rate - 13kbps)
- AMR-NB (3gpp codec, requires license)
- iLBC (RFC 3952 Internet Low Bit Rate)
- H.263 Video

Audio/Video Mime Types

- audio/x-wav
- audio/x-adpcm
- audio/x-alaw-basic
- audio/x-basic
- audio/x-gsm
- audio/mpeg
- video/3gpp

System Features

- Runs on x86 architecture
- Runs on RHEL5 CentOS5
- Delivered as a tgz file of RPMs, installs using RPM Manager
- Install utility has checkpoint/rollback capability
- "Installation verification" voice application included in initial install
- Premise or hosted solution
- Web browser based management console
- Documented HTTP management interface
- Online documentation on SPOT SIP Engine server
- Heavily commented ExamplePack (online)
- Worker/standby and N + 1 redundancy server configurations
- Keeps active calls alive during failover
- SNMP Support (Option)
- Open platform, user can add own applications/processes

- SNMP support extensible to user's add-ins

W3C/ECMA/ITU-T Standards Supported

- VoiceXML 2.0/2.1 Compliant (W3C unless stated otherwise)
- CCXML 1.0 Compliant (Implementation Report participant)
- SRGS 1.0 Speech Grammars
- SSML 1.0 Speech Markup
- SISR 1.0 Speech Semantic Interpretation
- Extensible Markup Language (XML) 1.1
- XML Document Object Model (DOM)
- XML Namespaces
- Standard ECMA-262 ECMAScript Language Specification (ECMA)
- Q.1912.5 Interworking between SIP and BICC or ISUP (ITU-T Recommendation)

IETF Standards

- RFC 1889 - Transport Protocol for Real-Time Applications (Obsoleted by RFC 3550)
- RFC 1890 - RTP Profile for Audio and Video Conferences with Minimal Control (Obsoleted by RFC 3551)
- RFC 2246 - The TLS Protocol Version 1.0 (Obsoleted by RFC 4346)
- RFC 2326 - Real Time Streaming Protocol (RTSP)
- RFC 2327 - SDP: Session Description Protocol (Obsoleted by RFC 4566)
- RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1
- RFC 2833 - RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals (Obsoleted by RFC 4733, RFC 4734)
- RFC 2976 - The SIP INFO Method (Obsoleted by RFC 6086)
- RFC 3164 - The BSD Syslog Protocol (Obsoleted by RFC5424)
- RFC 3195 - Reliable Delivery for syslog
- RFC 3204 - MIME media types for ISUP and QSIG Objects
- RFC 3261 - SIP: Session Initiation Protocol
- RFC 3162 - Reliability of Provisional Responses in SIP
- RFC 3263 - SIP: Locating SIP Servers
- RFC 3264 - An Offer/Answer Model with SDP
- RFC 3265 - SIP: Specific Event Notification
- RFC 3266 - SDP: Session Description Protocol (Obsoleted by RFC 4566, Updates RFC 2327)
- RFC 3310 - HTTP Digest Authentication Using Authentication and Key Agreement (AKA)
- RFC 3311 - SIP UPDATE Method
- RFC 3323 - A Privacy Mechanism for SIP
- RFC 3324 - Short Term Requirements for Network Asserted Identity
- RFC 3325 - Private Extensions to SIP for Asserted Identity within Trusted Networks

- RFC 3372 - SIP-T: Context and Architectures
- RFC 3515 - SIP REFER Method
- RFC 3550/STD 0064 - RTP: A Transport Protocol for Real-Time Applications (Obsoletes RFC 1889)
- RFC 2551/STD 0065 - RTP Profile for Audio and Video Conferences with Minimal Control (Obsoletes RFC 1890)
- RFC 3581 - An Extension to SIP for Symmetric Response Routing
- RFC 3725/BCP 0085 - Best Current Practices for Third Party Call Control SIP
- RFC 3842 - Message Summary and Message Waiting Indication Event Package for SIP
- RFC 3891 - SIP Replaces Header
- RFC 3952 - RTP Payload Format for iLBC
- RFC 3960 - Early Media and Ringing Tone Generation in SIP
- RFC 4028 - Session Timers in SIP
- RFC 4346 - TLS Protocol Version 1.1 (Obsoletes RFC 2246, Obsoleted by RFC 5246)
- RFC 4733 - RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals (Obsoletes RFC 2833)
- RFC 4734 - Definition of Events for Modem, Fax, and Text Telephony Signals (Obsoletes RFC 2833, Updates RFC 4733)
- RFC 4475 - SIP Torture Test Messages
- RFC 4566 - SDP: Session Description Protocol (Obsoletes RFC 3266, RFC 2327)
- RFC 5424 - The Syslog Protocol (Obsoletes RFC 3164)
- draft-ietf-speechsc-mrcpv2 - Media Resource Control Protocol Version 2