Voice over IP using Session Initiation Protocol

Version 6.1
Note

Before using this information and the product it supports, read the general information under "Notices" on page 181.
Contents

Figures ........................................... vii

About this information ......................... ix
Where to find more information ................ ix
Useful Web sites ................................... ix
Typographic conventions ....................... x
Accessibility ....................................... x
Notes on terminology ............................. xi
Information on VoIP and SIP ................... xi
Useful Web sites ................................... xii
Making comments on this book .................. xii

Chapter 1. Introducing WebSphere Voice
Response Voice over IP ............................ 1
What is Voice over IP? ............................ 1
How does Voice over IP work? ................... 2
Components of a VoIP network ................... 5
Using VoIP with WebSphere Voice Response .... 6
Overview of Voice over IP support .............. 7
What is needed to support VoIP with
WebSphere Voice Response ..................... 8
How many voice channels can a VoIP system
support? ............................................ 10
Increasing the channel capacity of a
WebSphere Voice Response T1 VoIP system by
25% .................................................. 12
Compression ....................................... 13
SIP signaling ....................................... 13
Application support .............................. 14
What happens when WebSphere Voice
Response receives a call? ......................... 14
How WebSphere Voice Response processes
incoming SIP Invites .............................. 15
Outgoing SIP Invites .............................. 17
Outbound arbitrary SIP headers ................. 18
Using outbound arbitrary SIP headers in
state tables ....................................... 18
Using outbound arbitrary SIP headers in
Voice XML and Call Control XML ................ 20

Chapter 2. Installing WebSphere Voice
Response Voice over IP ............................ 23
Installing the DTNA adapter ..................... 23
Supported machines ......................... 23
System unit connection ..................... 24
Installing the DTNA device drivers ............. 24
Setting ownership of the DTNAs .............. 25
Software installation ............................ 27
Installing the WebSphere Voice Response
VoIP software ..................................... 27
Setting the dtuser file permissions ............. 28
Importing the VoIP custom server .............. 29

Chapter 3. Configuring WebSphere Voice
Response Voice over IP ............................ 31
Using an allowed host list ....................... 31
Using a SIP Registrar ............................. 32
Setting the country/region ...................... 36
Configuring WebSphere Voice Response for
Voice over IP ..................................... 37
Setting the SIP transport IP address .......... 44
Shutting down and restarting WebSphere
Voice Response ................................. 45
Activating your channels ...................... 46

Chapter 4. Problem determination ................... 47
Diagnosing DTNA problems .................... 47
Analysing errors .................................. 48
Summary of useful tools for debugging
VoIP/SIP ......................................... 48
RTP and RTCP port allocation ................. 50
Codecs .......................................... 50
Performance implications ..................... 52
DTMF handling .................................. 52
NAT/Firewall considerations ................. 53
Tromboning with DTNA ......................... 53
DTNA packet size interval ..................... 54
Setting Voice over IP trace levels .......... 54
Using VOIP_MONITOR ......................... 55
Starting VOIP_MONITOR ....................... 55
Stopping VOIP_MONITOR ....................... 56
Logging VOIP_MONITOR trace
information to a file ........................... 56
Analyzing the VOIP_MONITOR output ......... 56
Frequently asked questions ................. 56
Installation ....................................... 56
Configuration ..................................... 56
State table applications ....................... 57
Miscellaneous .................................... 57
<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A simple two-way call using SIP</td>
<td>5</td>
</tr>
<tr>
<td>2. Pack Configuration window for DTNA configuration</td>
<td>37</td>
</tr>
<tr>
<td>3. Pack Configuration window for DTNA configuration showing NPACKS</td>
<td>38</td>
</tr>
<tr>
<td>4. Trunk Interface Parameters window for DTNA configuration</td>
<td>39</td>
</tr>
<tr>
<td>5. Line Signaling window</td>
<td>41</td>
</tr>
<tr>
<td>6. Channel Identification window</td>
<td>42</td>
</tr>
<tr>
<td>7. Pack Configuration window for DTNA configuration</td>
<td>43</td>
</tr>
<tr>
<td>8. VoIP SIP Signaling window</td>
<td>44</td>
</tr>
<tr>
<td>9. VoIP SIP Signaling/Override SIP Transport IP Address window</td>
<td>45</td>
</tr>
<tr>
<td>10. Symmetric key cryptography</td>
<td>61</td>
</tr>
<tr>
<td>11. Asymmetric key cryptography</td>
<td>62</td>
</tr>
<tr>
<td>12. Obtaining a digital certificate</td>
<td>67</td>
</tr>
<tr>
<td>13. Chain of trust</td>
<td>68</td>
</tr>
<tr>
<td>14. Overview of the SSL or TLS handshake</td>
<td>72</td>
</tr>
<tr>
<td>15. The digital signature process</td>
<td>76</td>
</tr>
</tbody>
</table>
About this information

This information provides an overview of the IBM® WebSphere® Voice Response for AIX® with DirectTalk® Technology Voice over IP capabilities. It explains how telephone conversations can be conducted over an IP network and how WebSphere Voice Response can be set up to operate in an IP network environment rather than a dedicated telephony environment. This information also provides detailed instructions for installing and configuring WebSphere Voice Response Voice over IP.

Throughout this book, the IBM WebSphere Voice Response for AIX voice processing system is referred to as WebSphere Voice Response.

Where to find more information

The information provided in the WebSphere Voice Response library will help you complete WebSphere Voice Response tasks more quickly. A complete list of the available publications and where you can obtain them is shown in “List of WebSphere Voice Response and associated documentation” on page 191.

Useful Web sites

The following Web sites are useful sources of information about WebSphere Voice Response and related products:

WebSphere Voice Response


IBM WebSphere developerWorks resources (including WebSphere Voice products)


VoiceXML Version 2.0 and 2.1 specifications

http://www.w3.org/TR/voicexml21/
http://www.w3.org/TR/voicexml20/

CCXML Version 1.0 specification

http://www.w3.org/TR/2011/PR-ccxml-20110510/

Genesys

For more information on Genesys products go to the Genesys Web site at http://www.genesyslab.com

Typographic conventions

This book uses the following typographic conventions:
**boldface**

Identifies an item that is in a WebSphere Voice Response window. The item might be a keyword, an action, a field label, or a pushbutton. Whenever one of the steps in a procedure includes a word in boldface, look in the window for an item that is labeled with that word.

**boldface italics**

Are used for emphasis. Take extra care wherever you see bold italics.

**italics**

Identify one of the following:

- New terms that describe WebSphere Voice Response components or concepts. A term that is printed in italics is usually followed by its definition.
- Parameters for which you supply the actual names or values.
- References to other books.

**monospace**

Identifies one of the following:

- Text that you type in an AIX window. Because AIX is case sensitive, ensure that you type the uppercase and lowercase characters exactly as shown.
- Names of files and directories (path names).

---

**Accessibility**

WebSphere Voice Response for AIX is a voice application enabler. The applications that are developed to run on WebSphere Voice Response provide telephone access to business data and services. In this way, WebSphere Voice Response provides accessibility for people who cannot access the data and services by using regular Web pages or traditional graphic interfaces. These telephone user interfaces are fully accessible to people who are blind or have low vision and, if speech recognition is used, to people with mobility impairments or limited hand use. Speech recognition capability can be provided by IBM WebSphere Voice Server, or other MRCP-V1.0-compliant speech recognition products. In addition, support for users of Telephony Devices for the Deaf (TDD) is provided as part of the WebSphere Voice Response product.

With WebSphere Voice Response you can perform many application development and system administration tasks with a text editor or line commands—these are accessible if you use a screen reader product to interface with them. Also, the default settings of the WebSphere Voice Response graphical user interface can be changed to produce large fonts and high contrast colors. Details of how to use these accessibility features can be found in the *WebSphere Voice Response for AIX: User Interface Guide* book.
Alternatively, application development can be done with Java™ or VoiceXML development tools that are supplied by IBM and third parties.

You can also use a screen-reader product to access the WebSphere Voice Response publications in HTML format (for details of their availability see “List of WebSphere Voice Response and associated documentation” on page 19).

**Notes on terminology**
- A glossary of commonly-used terms is at the end of this book.
- The full product name of WebSphere Voice Response for AIX with DirectTalk Technology is generally abbreviated in this book to WebSphere Voice Response.
- The term pSeries® is generically used in this book to refer both to PCI-based RS/6000® computers and to appropriate models of the System p5® and pSeries ranges. (Consult your IBM representative for details of models that are supported for use with WebSphere Voice Response.) RS/6000 computers with an MCA bus are not supported.
- The IBM Quad Digital Trunk Telephony PCI Adapter is generally referred to in this book by its abbreviation DTTA. This adapter is a replacement for the IBM ARTIC960RxQ Quad Digital Trunk PCI Adapter, which is generally referred to by the abbreviation DTXA. The DTXA is not supported with WebSphere Voice Response Version 6.1.
- References made to the VoiceXML 2.1 specification are intended to include VoiceXML 2.0 unless otherwise specified.

**Information on VoIP and SIP**

The following are useful sources of information about Voice over Internet Protocol and Session Initiation Protocol:
- RFC 1889 Transport Protocol for Real-Time Applications
- RFC 2327 SDP: Session Description Protocol
- RFC 2782 A DNS RR for specifying location of services (DNS SRV)
- RFC 2833 IETF RTP Payload for DTMF Digits
- RFC 3261 SIP: Session Initiation Protocol (Current RFC June 2002)
- RFC 3264 Offer/Answer Model with SDP
- RFC 3265 SIP Specific Event Notification
- RFC 3515 SIP REFER Method
- RFC 3551 RTP Profile for Audio and Video Conferences with Minimal Control
- Draft-ietf-sip-cc-transfer-05 SIP Call Control Transfer
- Draft-ietf-sip-replaces-05.txt
- Draft-levy-sip-diversion-06.txt
Useful Web sites

The following Web sites are useful sources of information about Voice over Internet Protocol and Session Initiation Protocol:

http://www.sipcenter.com

SIP RFC
http://www.ietf.org/rfc/rfc3261

Making comments on this book

If you especially like or dislike anything about this book, feel free to send us your comments.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information that is in this book and to the way in which the information is presented. Speak to your IBM representative if you have suggestions about the product itself.

When you send us comments, you grant to IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

You can get your comments to us quickly by sending an e-mail to idrcf@hursley.ibm.com. Alternatively, you can mail your comments to:

User Technologies,
IBM United Kingdom Laboratories,
Mail Point 095, Hursley Park,
Winchester, Hampshire,
SO21 2JN, United Kingdom

Please ensure that you include the book title, order number, and edition date.
Chapter 1. Introducing WebSphere Voice Response Voice over IP

This chapter introduces Voice over Internet Protocol (VoIP), and explains how VoIP works with WebSphere Voice Response for AIX.

This chapter contains the following sections:

- “What is Voice over IP?”
- “How does Voice over IP work?” on page 2
- “Components of a VoIP network” on page 5
- “Using VoIP with WebSphere Voice Response” on page 6
- “Overview of Voice over IP support” on page 7
- “What is needed to support VoIP with WebSphere Voice Response” on page 8
- “How many voice channels can a VoIP system support?” on page 10
- “Compression” on page 13
- “SIP signaling” on page 13
- “Application support” on page 14
- “What happens when WebSphere Voice Response receives a call?” on page 14

What is Voice over IP?

Voice over IP is a set of protocols that allows you to perform telephony voice communications over a standard Ethernet IP network instead of over dedicated switched telephony voice channels and trunk circuits using ISDN or channel associated signaling.

The advantages of using an IP network to carry telephone conversations are:

- Standard computer equipment and networking hardware can be used for the network and endpoints. This is usually much cheaper than conventional telephony hardware and occupies less physical space.
- A single network can be used for both data and voice transfer.
- The network capacity for handling voice can be increased at peak call times by controlling the volume of data transfer. When call volume is low, the excess bandwidth can be used for data.
- It is easier to create applications that integrate data with voice when the same network is used for both.
• New telephony features can be quickly deployed as they are implemented in the network endpoints rather than within the network itself.
• Techniques can be used to compress the voice data and suppress the sending of voice data during periods of silence; this enables much more efficient usage of the network and lower call costs.

The disadvantages are:
• Network bandwidth must be carefully managed to avoid voice quality problems.
• Voice packets may take longer to get across the network than if the same data were to be sent over a circuit switched connection, resulting in longer delays for your voice applications.

How does Voice over IP work?

Voice and signaling are sent using standard TCP/IP protocols over a physical link such as an Ethernet network. This exchange of signaling and voice information takes place in both directions at the same time with each endpoint sending and receiving information over the IP network.

In any telephony system, two things are carried by the network: voice data and signaling information. Voice is the sound information detected by the microphone in the telephone and transmitted to the receiver over a communication channel. Signaling is the information exchanged between stations participating in the call when a call is started or ended, or when an action (for example, call transfer) is requested.

Traditionally, both voice and signaling information have been sent together through dedicated circuit switched telephony channels (used, for example, with channel associated signaling and ISDN). However, with VoIP, voice and signaling are sent using standard TCP/IP protocols over a physical link such as an Ethernet network. This exchange of signaling and voice information takes place in both directions at the same time with each endpoint sending and receiving information over the IP network.

How is voice data sent over an IP network?

With VoIP, voice data is digitally encoded using μ-law or A-law Pulse Code Modulation (PCM). The voice data can then be compressed if necessary and sent over the network in User Datagram Protocol (UDP) packets. Standard TDM telephony sends voice data at a low constant data rate. With VoIP, relatively small packets are sent at a constant rate. The total overall rate of sending data is the same for each kind of telephony.
The advantage of VoIP is that one high-speed network can carry the packets for many voice channels and possibly share with other types of data at the same time (for example, FTP, HTTP, and data sockets). A single high-speed network is much easier to set up and maintain than a large number of circuit switched connections (for example, T1 circuits).

The User Datagram Protocol is used to transmit voice data over a VoIP network. UDP is a ‘send and forget’ protocol with no requirement for the transmitter to retain sent packets should there be a transmission or reception error. If the transmitter did retain sent packets, the flow of real-time voice would be adversely affected by a request for retransmission or by the retransmission itself; especially if there is a long path between transmitter and receiver.

The main problems with using UDP are that:

- There is no guarantee that a packet may actually be delivered.
- Packets can take different paths through the network and arrive out of order.

To overcome these problems, the Real-time Transport (RTP) is used with VoIP. RTP provides a method of handling disordered and missing packets and makes the best possible attempt to recreate the original voice data stream (comfort noise is intelligently substituted for missing packets).

**Signaling**

The Signaling Invite message is used by the VoIP phone that initiates a call (the calling party) to inform the called party that a connection is required. The called party can then accept the call or reject the call (for example, if the called party is already busy). Other signaling exchanges will be initiated by actions like near or far end hangup, and call transfer.

For VoIP, several signaling protocols are in general use:

- Session Initiation Protocol (SIP) is a modern protocol that is becoming increasingly popular.
- Media Gateway Control Protocol (MGCP) is used internally within telephone networks.
- H.323 is an older VoIP protocol, the elements of which are very similar to ISDN telephony protocols. (Unlike SIP, which uses internet based URIs for addressing.)

WebSphere Voice Response supports SIP as the only Voice over IP signaling protocol. The WebSphere Voice Response version of SIP fully conforms to RFC 3261 which is the standard definition for SIP in the industry.
SIP is based on URI messages which are exchanged between endpoints whenever any signaling is required. These message exchanges are mapped by WebSphere Voice Response SIP support to standard telephony actions within the WebSphere Voice Response product. Standard telephony actions include:

- Incoming calls
- Outgoing calls
- Near end hangup
- Far end hangup
- Transfers (several types are supported including 'blind' and 'attended')

SIP signaling messages can use either TCP (a reliable, guaranteed message exchange) or UDP (a non-guaranteed datagram protocol).

SIP is becoming established as the industry standard for multi-media session control over IP networks and is defined in the IETF standard RFC 3261 Session Initiation Protocol. The following diagram shows the exchanges which take place between two SIP endpoints in a simple two-way call with far-end hang-up.
Components of a VoIP network

There are three main components of a VoIP network: user agents, gateways, and proxy servers.

User Agent

In a VoIP network, any device that can make or receive telephone calls is called a User Agent (UA). Each User Agent contains a User Agent Server (UAS) responsible for handling requests from another endpoint, (for example, inbound calls) and a User Agent Client (UAC).
which generates requests, (for example, outbound calls) for other endpoints. Examples of User Agent Clients and User Agent Servers are:

- A SIP hard phone.
- A SIP soft phone.
- WebSphere Voice Response (which simulates a number of phones) for incoming or outgoing calls.

**Gateways**
A gateway is a device which acts as a bridge between VoIP and the PSTN network. A gateway can take an incoming call from a T1 interface and convert the signaling into SIP message exchanges, and convert the voice from TDM into RTP packets.

**Proxy servers**
In a SIP system, a proxy server (used with a registrar and a location server), can provide the following services:

- Call Routing including URI translation.
- Registration.
- Access (authentication) to a SIP network.

A Proxy server is the means by which calls are routed within a SIP VoIP network. For example, a telephony gateway might be configured to send all incoming calls to the SIP proxy server which will then route the calls to specific endpoints (this can include load balancing or skills-based routing).

---

**Using VoIP with WebSphere Voice Response**

The Voice over IP (SIP) feature for WebSphere Voice Response allows a WebSphere Voice Response application to act as an IP telephony User Agent (UA).

With the ability to run up to 480 channels at the same time using the DTNA VoIP device driver ‘adapter’, WebSphere Voice Response can provide up to that number of SIP UAs. As with standard telephony, each UA behaves like a real telephone.

WebSphere Voice Response applications that interact with the SIP network can be written using State Tables, Java (with the Java and VoiceXML environment), VXML, or CCXML. As with traditional telephony, these applications can do things like:

- Receive incoming calls.
- Play voice to the caller (prerecorded or synthesized).
- Wait for the caller to speak or enter DTMF keys.
Perform voice recognition on the caller’s spoken data.
Hangup.
Perform call transfer (blind or attended).
Make outgoing calls.
Trombone two calls together.
Accept external refer requests.

As long as some simple ‘numbering plan’ rules are met within the SIP network, existing WebSphere Voice Response applications should run unmodified (as they operate in terms of standard E.164 telephone numbers). If required, WebSphere Voice Response applications can be written specially for VoIP to exploit fully the additional functionality provided by SIP (for example, textual addresses - URIs).

Overview of Voice over IP support

Voice over IP (specifically using the SIP signalling protocol) is implemented in WebSphere Voice Response using a SIP signalling stack, and one or more ‘adapters’ that transfer the voice data associated with a SIP call.

These two components are:

- A SIP signalling stack (the ‘SIP sigproc’ which runs as part of WebSphere Voice Response in the system unit)
- One or more ‘adapters’ which are responsible for transferring the voice data over the IP network using Real Time Protocol (RTP) once a call has been established using the SIP sigproc.

Although WebSphere Voice Response uses two types of adapter available for voice data transfer, the SIP Signalling Process is the same for both. It communicates on one side with the ‘core’ of WVR and on the other, with the AIX TCP/IP stack, implementing SIP sequences to provide signalling functions such as call setup, teardown, transmission and so on.

The only currently available ‘adapter’ available for WebSphere Voice Response is a ‘software’ adapter known as DTNA (Digital Trunk No Adapter). DTNA uses the system unit’s built-in Ethernet interface. DTNA is software function which is provided (at no extra charge) in WebSphere Voice Response.

Note: The DTEA hardware adapter in now no longer available but is still supported by WebSphere Voice Response Version 6.1. For information on using the DTEA, please refer to the WebSphere Voice Response Version 4.2 edition of this book, GC34-6383-04.

Although many applications will run unchanged using VoIP, there are some application considerations in migrating an application from DTTA (standard...
What is needed to support VoIP with WebSphere Voice Response

To assemble a WebSphere Voice Response system that supports VoIP, you will need one of the three following configurations:

**BladeCenter using DTNA**
- An IBM BladeCenter computer with AIX PowerPC® blades (JS20, JS21, JS22, PS700, PS701, or PS702).
- AIX 6.1 or 7.1 installed.

**System p using DTNA**
- An IBM System p with Power 5, Power 6, or Power 7 processors.
- AIX 6.1 or 7.1 installed on the system unit.

To send and receive voice data over an IP network your WebSphere Voice Response system requires at least one DTNA adapter. The DTNA is installed using a single AIX command and then configured using WebSphere Voice Response Pack and/or System configuration.

Capacity planning for VoIP systems using DTNA involves considerations about the use of DTMF detection and voice compression:

**DTMF detection**
- With DTNA, the system unit’s processors are used for DTMF detection. This cost can be avoided if the VoIP network supplies DTMF send/receive keys using RTP payload packets - RFC2833 (Telephone Event).

**Compression**
- As previously described, DTNA only supports uncompressed voice over a Voice over IP link. However, the use of compressed voice within the voice application (the 5:1 ‘GSM’ algorithm) can have a big effect on the processor requirements for DTNA.

For compressed voice (playing and/or recording)
- The DTNA adapter software needs to do a decompression on play and a compression on record. This can take a significant processing effort.
- iLBC compression is significantly more complex than G.711 and so, understandably, requires greater processing power. A system that can currently support 480 channels of G.711 can...
typically support up to 120 iLBC channels. Systems with a higher/lower rPerf will support more/less concurrent channels, respectively, up to the maximum 480 per system/partition. See [http://www.ibm.com/systems/power/hardware/notices/rperf.html](http://www.ibm.com/systems/power/hardware/notices/rperf.html) for further information on rPerf.

**For uncompressed voice (playing and/or recording)**

No decompression and subsequent compression is required, so the CPU load for DTNA processing is reduced.

DTNA adapters provide the following functionality:

- Implement RTP for transmission and reception of voice packets on up to 120 channels at the same time.
- Provide the interface for all channels to the Ethernet in the system unit for DTNA.
- Perform ‘back-end processing’ for the WebSphere Voice Response Device Driver in the same way as the DTTA (for more information see [WebSphere Voice Response for AIX: Installation](#)).

DTNAs (software adapters for VoIP) can be installed in the same system as DTTAs. However, calls cannot be tromboned between DTTAs and DTNAs, although they can be tromboned within each group of similar adapters, as follows: DTNA to DTNA, DTTA to DTTA, but not DTNA to DTTA or DTTA to DTNA.

Some example combinations of adapters are:

- One to four DTNAs - For a VoIP-only system for four, eight, 12 or 16 ‘pure VoIP’ trunks.
- One DTNA, and one, two, or three DTTAs for a mixed VoIP/PSTN configuration.
- Two DTNAs, and two DTTAs for a mixed VoIP/PSTN configuration.

The DTNA can be mixed with DTTAs, with a maximum of four instances of either type.

Adapterless configurations have the following double-trunking capabilities:

- Calls can be double-trunked between adapterless connections using the Trombone Custom Server.
- Adapterless connections can coexist with DTTA in the same system unit, allowing hybrid PSTN/VoIP configurations in, for example, IP Call Centers. However, calls cannot be double-trunked between adapter and adapterless connections.
- Adapterless DTNA VoIP/SIP connections cannot co-exist in the same system with DTEA (hardware VoIP adapter) connections.
How many voice channels can a VoIP system support?

As with a PSTN WebSphere Voice Response telephony configuration, the number of channels available on each pSeries computer or BladeCenter computer depends on the number of adapters installed in the system unit.

A single DTNA adapter supports 96 channels of VoIP streaming when configured in T1 mode, and 120 channels when configured to work in E1 mode.

In WebSphere Voice Response, the internal coding of voice is always μ-law for T1 trunks and A-law for E1 trunks. If one or more DTTA PSTN adapters, are present in the same system as one or more DTNA adapters, the DTNAs must operate in the same mode (either T1 or E1) as the PSTN adapters (whose operating mode is determined by the network attachment). For VoIP-only configurations, with no PSTN adapters, it is possible to use all 120 channels of the DTNA in a T1 country/region by configuring the system to be E1 with a default μ-law VoIP encoding (however, in this case, voice segments must be stored as A-law).

Note: When you configure WebSphere Voice Response for VoIP, you still need to select either T1 or E1 mode even though the DTNAs are not actually connected to PSTN trunks.

WebSphere Voice Response supports up to a total of four DTEAs and DTTAs, so for a VoIP-only configuration, the maximum number of TDM or mixed VoIP channels available on each pSeries computer (or LPAR on such a computer) is 384 (4×96) for T1 and 480 (4×120) for E1.

Capacity planning for DTNA

The use of DTNA imposes some additional load on your system processor over and above that of the DTTA and the DTEA adapters (both of which are now unavailable). This is because work which was previously done on hardware adapter cards is now being done in the main system processor.

Individual core processing speed is no longer relevant for WebSphere Voice Response. The overall system relative performance, or rPerf, is the defining factor in how many channels can be supported. As a guideline, an rPerf of 20 can support a 480-channel WebSphere Voice Response LPAR running speech-enabled VoiceXML applications. See [http://www.ibm.com/systems/power/hardware/notices/rperf.html](http://www.ibm.com/systems/power/hardware/notices/rperf.html) for further information on rPerf.

The use of secure SIP and secure RTP, increases the system performance requirements of WebSphere Voice Response by approximately 30%. As a
guideline, a system performance rPerf of 26 can support a 480-channel secure SIP and secure RTP WebSphere Voice Response LPAR running speech-enabled VoiceXML applications.

Up to four WebSphere Voice Response LPARs are supported per pSeries computer or BladeCenter blade.

To minimize CPU impact, consider the following:

- Ensure that RFC2833 RTP Payloads are used for DTMF transmission rather than DTMF tones. Most modern SIP devices support RFC2833, usually by default. The VOIP_MONITOR utility, provided with WebSphere Voice Response to support SIP, can be used to ensure that Telephone Event is being offered in the SDP (body) part of the SIP Invite message, and then accepted as the DTMF method to be used, as indicated in the SDP (body) part of the 200 OK response to the SIP Invite. If this is not found to be the case, either WebSphere Voice Response (DTEA and DTNA parameters) or the remote SIP endpoint (or both) need to be reconfigured to support RFC2833.

- Ensure that your application uses uncompressed play and recording. This is because the DTNA processing overhead of converting between the WebSphere Voice Response internal 5:1 algorithm voice format and uncompressed G711 A and µ law as sent/received over the line, is significant.
  - For Java and VoiceXML applications, uncompressed play and recording is always used, and no further action need be taken.
  - Most State Table applications will already have been written to use uncompressed voice (on account of the higher quality) and therefore no further actions are required.
  - However, if your State Table application uses compressed prompts (or messages, audio names and/or greetings) but does no recording in the compressed format, you need to take this into account in capacity sizings. If your application also does compressed recording, this factor also needs to be taken into account when planning channel capacity. Alternatively, consider changing your application to use uncompressed voice throughout.
  - Unified Messaging for WebSphere Voice Response is a voice messaging application. The WebSphere Voice Response and Unified Messaging supports the use of compressed or uncompressed voice in messages, audio names and greetings through setting options in WebSphere Voice Response System Configuration. If you are planning to operate Unified Messaging for WebSphere Voice Response in uncompressed mode for all voice objects from the outset, or to migrate your existing Unified Messaging for WebSphere Voice Response system to uncompressed voice, you need take no further action.
For the most recent information on suitable processors and capacity planning for DTNA, contact your IBM representative.

**Increasing the channel capacity of a WebSphere Voice Response T1 VoIP system by 25%**

The default maximum channel capacity for a T1 VoIP system is 384 (24 channels per trunk, with a maximum of 16 trunks). The limit only applies when connecting WebSphere Voice Response to a traditional telephone network.

When configured to use a VoIP connection with DTNAs, there is no physical reason for this limit. It is therefore possible to configure WebSphere Voice Response as an E1 system when in a traditional T1 environment to enable an increase in capacity to 480 channels (30 channels per trunk, with a maximum of 16 trunks).

**Restrictions:**

1. This change will only work if the WebSphere Voice Response system is a VoIP only configuration. If there are any physical adapters, DTXA or DTTA, the change cannot be made.
2. If any state table applications use uncompressed voice segments, this change will not work as the audio is stored in mu-law for T1, and so will play in the wrong format for an E1 system.

**Configuration process**

To configure WebSphere Voice Response in this way, the **Country / Region** section of Pack Configuration needs to be set to **Other E1**. To do this, complete the following steps:

1. Use the `dt_setowner` command to reconfigure the DTNAs as E1 as described in the section “Setting ownership of the DTNAs” in *WebSphere Voice Response for AIX: Voice over IP using Session Initiation Protocol*.
2. Start WebSphere Voice Response.
3. In the WebSphere Voice Response Welcome window, click **Configuration** -> **Pack Configuration** -> **Change**.
4. Click **Country / Region**, and choose **Other E1** as the country, press **OK** and then **OK** again to confirm the choice.
5. Configure the packs and channels as usual, as described in the section “Configuring WebSphere Voice Response for Voice over IP” in *WebSphere Voice Response for AIX: Voice over IP using Session Initiation Protocol*, save the changes, and restart WebSphere Voice Response.
If the WebSphere Voice Response system has been used as a T1 system previously, and is running VoiceXML applications, it will also be necessary to delete the VXML2 audio cache before taking new calls. To do this, perform the following steps:

1. Stop VRBE if it is running (`dtjstop` followed by `dtjshost -exit`).
2. As dtuser, `cd` into `$CUR_DIR/voice/ext/v2c` and delete all directories and files within that directory.
3. Restart VRBE (`dtjshost` followed by `dtjstart`).

**Compression**

WebSphere Voice Response VoIP supports uncompressed data only over RTP streams.

The WebSphere Voice Response VoIP DTNA adapter supports G.711 (A-law) and G.711 (µ-law) uncompressed voice data formats over RTP streams.

**SIP signaling**

In a WebSphere Voice Response Voice over IP system, signaling information is sent and received through the AIX system unit LAN connection (usually Ethernet). This should be connected to an IP network such that all other SIP UAs (such as gateways, and phones) can contact each other over the network.

The DTNA adapter only handles the RTP voice data (not SIP signaling) and could be on a separate, and possibly dedicated, network interface to ensure that the voice real time data transfer is not adversely affected by other traffic.

For greater security, WebSphere Voice Response supports IPSEC over the system Ethernet that carries the SIP signaling (this is a feature of AIX). For more information on IPSEC see "SIP and IP support" on page 173.

To provide SIP signaling support, WebSphere Voice Response uses a SIP signaling stack that allows WebSphere Voice Response to communicate over the system unit IP network adapter and a WebSphere Voice Response signaling process that links the SIP signaling stack to WebSphere Voice Response using the standard signaling library API. These are integrated within a single WebSphere Voice Response signaling process (the VoIP-SIP custom server).

When you configure WebSphere Voice Response for Voice over IP, WebSphere Voice Response installs the VoIP signaling process for the channels you require. Inbound and outbound signaling for those channels then flows through the signaling process and the SIP stack to the IP network (rather than over T1 or E1 trunks as would happen in a system with DTTAs installed).
Application support

With Voice over IP, most inbound or outbound only applications will work unmodified. However, SIP capability can be further exploited if required (see “Voice over IP tags” on page 104).

If your application requires call transfer capability, refer to “Call transfer” on page 114 for more information.

What happens when WebSphere Voice Response receives a call?

When WebSphere Voice Response receives a call over Voice over IP, a specific sequence of events is followed:

1. WebSphere Voice Response receives a SIP signaling sequence (beginning with an INVITE message) which is processed by the SIP stack and passed on to the SIP signaling process within the VoIP-SIP signaling process.
2. The SIP signaling process informs WebSphere Voice Response that an incoming call is in progress.
3. WebSphere Voice Response parses the URI and looks to see if the ‘user’ part contains a valid E.164 telephone number; if it does, this number is used as the calling number and WebSphere Voice Response tries to match the number to an application (State Table, Java, CCXML or VoiceXML). If a match is found WebSphere Voice Response launches the application on the channel receiving the call. If no number is available, one of the following is used as the calling number:
   - The default number of WebSphere Voice Response
   - The default application number for all SIP channels is configured in the Channel Identification window which is displayed from the Pack Configuration window by clicking on the Channel IDs button for a Pack. Refer to the WebSphere Voice Response for AIX: Configuring the System book for details.
4. When WebSphere Voice Response connects the call it sends a command to the DTNA to start streaming voice (RTP) in both directions.
5. The DTNA converts the continuous voice stream into RTP packets to be sent across the network. Packets of voice data that arrive from the network are received by the DTNA, put into order and converted into a continuous voice stream, which is passed to the ‘core’ of WebSphere Voice Response.
6. When the call is ended the signaling process sends or receives signaling information (depending on whether the call was ended by near-end or far-end hang up) and the voice streaming is stopped.

Outgoing calls operate in a very similar way to inbound except that the initial signaling is sent from the WebSphere Voice Response signaling process to a User Agent Server (UAS) (usually through a SIP proxy server).
How WebSphere Voice Response processes incoming SIP Invites

To determine the Called Party number, the To and Request headers are extracted from the message and processed as follows:

1. Depending on the setting of the ‘Use Request header’ system parameter, the content of the To or Request header is analyzed to look for a valid numeric ‘User Part’. If found, this becomes the numeric Called Number which is passed to WebSphere Voice Response using SV185.
2. If set, the Request header must have a ‘sip’ URI for the number for example, <sip:123456@anyhost.com>.
3. If set, the To header can be comprised of one ‘sip’ or ‘tel’ URI.
4. Any numeric user part is processed to remove leading E164 international dialling prefix characters as described in step 1 of the next section.
5. If the set header (To or Request) does not meet the above criteria, the Called Number (SV185) is set to the configured SIP default calling line id. If this has not been set, then the SV185 will be set to NULL.

To determine the Calling Party, Original Called and Last Redirecting numbers, the following additional headers are extracted from the incoming Invite:

- The From header (FROM_HDR)
- Up to two P-Asserted-Identity headers (PASSID_HDR)
- Remote-Party-ID (RPARTYID_HDR)
- First and last Diversion Headers (DIV_HDR)
- Privacy Header (PRIVACY_HDR)

These are processed as follows:

1. If present, P-Asserted-Identity headers (one or two) are processed looking for a ‘sip’ or ‘tel’ URI with a valid numeric user part (priority is given to ‘sip’ over ‘tel’). Leading E164 characters are removed and the resulting number is passed to the WebSphere Voice Response application in SV186.
2. If a valid P-Asserted-Identity header has not been found and a Remote-Party-ID header is present, it is examined for a valid numeric user part (with leading E164 digits removed) and is passed to the WebSphere Voice Response application in SV186. If a valid calling number has been found, the privacy parameter is extracted from the message for later use.
3. If a valid numeric calling number cannot be found in either P-Asserted-Identity or Remote-Party-ID headers, the From header is examined for a valid numeric user part. If one can be found, any E164 leading digits are removed and the resulting number is passed to the WebSphere Voice Response application in SV186.
4. If no valid Calling Party number can be found in the P-asserted-ID, Remote-Party-ID or From headers (in that order of priority), the WebSphere Voice Response application receives NULL in SV186 as a Calling Number.

5. If the Override Privacy system parameter is set, the privacy string extracted from either the Privacy header or the privacy parameter of the Remote-Party-ID header is set to NULL.

6. If one or more Diversion Headers are present, the topmost (first) and bottommost (last) are inspected for ‘sip’ or ‘tel’ URLs with valid numeric user parts. If found, the numeric user part of the topmost Diversion header is passed to WebSphere Voice Response in SV187 (Original Called Number) and the bottommost Diversion header user part is passed in SV188 (Last Redirecting Number).

7. The following parameters are extracted from the first (or only) Diversion header: reason, counter, limit, privacy, screen, and extension. These are made available to WebSphere Voice Response as tagged string attributes.

8. The Accept-Language header is extracted, and if present, is passed to the WebSphere Voice Response application using a tagged string.

E164 processing on any number is as follows:

1. If a number does not begin with a ‘+’, it is assumed to NOT be E164 formatted and no further action is taken.

2. For E164 numbers, the + is always removed in addition to any further digits as defined in the ‘E164 Strip’ system parameter. If the ‘E164 Strip’ system parameter is set to 1,44, 393, for example, +1, +44, or +393 is removed from the number.

If present in the SIP header and defined in the configuration file /usr/lpp/dirTalk/db/sys_dir/voip/siphdrtags.cfg, the following information is then passed to the WebSphere Voice Response application using VoiceXML and CCXML variables and the SV542 tagged string (SV542 must be saved in another System Variable prior to the AnswerCall action being executed):

- PROTOCOL — Always set to SIP (for SIP calls)
- TO_HDR — the full contents of the SIP Invite To header
- REQ_HDR — SIP Request header
- FROM_HDR — SIP From header
- DIV_HDR — Topmost (or only) Diversion header
- A_LANG_HDR — Accept Language header
- CALLID_HDR — SIP Call-ID header which contains a unique call identifier
- CALLINFO_HDR — SIP Call-Info header which provides additional information about the caller.
• PRIVACY_HDR — value of the SIP Privacy header if a P-Asserted-Identity header was used for the Calling Party number or the privacy parameter of the Remote-Party-ID if that header was used for the Calling Party Number.
• Attributes for the Diversion Header: reason, counter, limit, privacy, screen and extension
• Up to 10 other (non-standard) valid, unique SIP headers can be added to the configuration file /usr/lpp/dirTalk/db/sys_dir/voip/siphdrtags.cfg. These headers are processed in the order in which they are defined in the file, subject to sufficient space being available in the buffer for the header data. The total limit for all headers is 512 bytes. Any subsequent non-standard SIP headers found in siphdrtags.cfg are ignored. If the same header is specified multiple times within a single INVITE message, the data from each header is appended to the first instance, separated by commas. Headers definitions cannot be more than 31 characters long, and must not include any of the following special characters:
  * & $ ( ) / ; : = _ ? @ ^ ` { | } [ ]

The SIP Invite To and From headers are always processed, but the processing of the other SIP headers above is controlled by the configuration file /usr/lpp/dirTalk/db/sys_dir/voip/siphdrtags.cfg. As supplied, the configuration file entries for the SIP Call-ID and Call-Info headers are commented out. To use the information in these headers in state table, VoiceXML, or CCXML applications, you must edit the configuration file manually and remove the # character in front of the header or headers that you want processed.

In addition, the following system variables are set:
• SV185: Called Party number: extracted from the user part of Request or To header (selected by system parameter).
• SV186: Calling Party number: extracted from the user part of P-Asserted-Identity, Remote-Party-ID or From headers (in that order)
• SV187: Original Called Number: extracted from the user part of the topmost (first) Diversion header (if present)
• SV188: Last Redirecting Number: extracted from the user part of the bottommost (last) Diversion header (if present)

Outgoing SIP Invites

For outgoing SIP Invites, the process is as follows:
• State Table applications must set any desired SIP headers as tagged strings in SV541 prior to the call being made.
• A header is generated only if that header is set in SV541.
• The TO_HDR must always be present. (This is the address of the sip endpoint to which the Invite will be sent.)
• Optionally, the following headers can be added:
  – FROM_HDR (default is supplied if not set)
  – REQ_HDR (default is supplied)
  – DIV_HDR: Diversion header. Also DIV_HDR attributes: reason, counter, limit, privacy, screen and ext
  – A_LANG_HDR: Accept Language
  – PRIVACY_HDR: Privacy
  – RPARTYID_HDR: Remote Party ID
  – PASSID_HDR: P-Asserted_ID

Outbound arbitrary SIP headers

Headers and associated values in the following situations on the identified outbound SIP messages:
• Outbound call (initial INVITE message)
• Blind transfer (REFER message)
• Attended transfer (INVITE message used to establish second leg of transfer)
• Near end disconnect (BYE message)

As with inbound headers, up to 10 outbound headers can be defined in file /usr/lpp/dirTalk/db/sys_dir/voip/siphdrtags.cfg

Duplicates, and any exceeding 10 are ignored. Header entries are treated as defining both inbound and outbound headers. The following standard tags supported by state tables are exempt from this limit of 10 headers, and can always be sent:

TO_HDR, FROM_HDR, REQ_HDR, DIV_HDR, A_LANG_HDR,
PRIVACY_HDR, PASSID_HDR, RPARTYID_HDR

Using outbound arbitrary SIP headers in state tables

The values specified in siphdrtags.cfg are used in association with state table API special variable 541 (SV541) tags to set header and header values in outgoing SIP messages.

Any header specified in an SV541 tag, but not specified in siphdrtags.cfg is ignored. Headers need to be entered as tags in SV541 in the same case that they are defined in siphdrtags.cfg. For example, if siphdrtag1 is defined in siphdrtags.cfg, SIPhdrtag1 specified as a tag in SV541 is ignored.

Note: These headers will not go out on an INVITE message used during transfer processing to place a call on or off hold.
There is no checking of the effects of adding a header. It is the application developer’s responsibility to ensure that adding a header will have no adverse affects on the SIP message exchange.

Any processing errors associated with adding a SIP header to an outbound message are recorded in the standard WebSphere Voice Response errorlog. In this error case, the SIP message is sent, but without the header in error.

If a header is specified that WebSphere Voice Response would add by default to a message, the SV541 tag-supplied value is appended (preceded by a comma) to the value supplied by WebSphere Voice Response. An example of where this may prove useful is with supported or allow headers.

Up to 10 other (non-standard) valid, unique SIP headers can be added to the configuration file /usr/lpp/dirTalk/db/sys_dir/voip/siphdrtags.cfg. These headers are processed in the order in which they are defined in the file, subject to sufficient space being available in the buffer for the header data. The total limit for all headers is 512 bytes. Any subsequent non-standard SIP headers found in siphdrtags.cfg are ignored. If the same header is specified multiple times within a single INVITE message, the data from each header is appended to the first instance, separated by commas.

Multiple values of same header are not supported, so an application must combine values using a semi-colon in a single header. If multiple values of same header are supplied, the final value is used.

Headers definitions cannot be more than 31 characters long, and must not include any of the following special characters:
*#$&(),./;'<>@\^`{|}[]

**Outbound call example**

1. siphdrcfg.tag has the following line added.
   
   ```
   Test-Tag
   ```

   WebSphere Voice Response is then restarted to make the change effective.

2. Make an outbound SIP call using a state table with the following action specified prior to the MakeCall action:
   
   ```
   AssignData(SV541, "PUT_TAG", "Test-Tag", "test data");
   ```

3. The resultant outgoing INVITE message contains the following header and data:
   
   ```
   Test-Tag: test data
   ```
Blind Transfer example
1. siphdrcfg.tag has the following line added.
   Test-Tag
   WebSphere Voice Response is then restarted to make the change effective.
2. Blind transfer a SIP call using a state table with the following action specified prior to the TransferCall action:
   AssignData(SV541, "PUT_TAG", "Test-Tag", "test data");
3. The resultant outgoing REFER message contains the following header and data:
   Test-Tag: test data

Blind Transfer example
1. siphdrcfg.tag has the following line added.
   Test-Tag
   WebSphere Voice Response is then restarted to make the change effective.
2. Attended transfer a SIP call using a state table with the following action specified prior to the initial TransferCall action:
   AssignData(SV541, "PUT_TAG", "Test-Tag", "test data");
3. The resultant outgoing INVITE message contains the following header and data:
   Test-Tag: test data

Near End Disconnect example
1. siphdrcfg.tag has the following line added.
   Test-Tag
   WebSphere Voice Response is then restarted to make the change effective.
2. Terminate a SIP call using a state table with the following action specified prior to the CloseEverything action:
   AssignData(SV541, "PUT_TAG", "Test-Tag", "test data");
3. The resultant outgoing BYE message contains the following header and data:
   Test-Tag: test data

Using outbound arbitrary SIP headers in Voice XML and Call Control XML

Outbound headers configured in siphdrtags.cfg can be sent as headers on outbound SIP messages. There are two mechanisms for accomplishing this depending on whether you are using CCXML and VoiceXML, or just VoiceXML. If you are using VoiceXML and not CCXML, you can specify the
values of the outbound headers using the following functions that are automatically available within a VoiceXML application:

**getInboundHeader(name, category)**

Returns null if the header does not exist or if any part of the `session.connection.protocol.category.name` chain is null, otherwise returns the header object. `name` is the header name requested. `category` can be omitted and defaults to `sip` if not specified. You then need to specify the attribute (usually `.value`) of the returned object. For example:

```javascript
getInboundHeader('user-to-user').value
```

**printObject(toPrint)**

Prints out the attributes and associated values of a JavaScript object into a string and returns it. Useful for visualizing what is set on an object, or for seeing all available inbound headers with one log statement. For example:

```xml
<log>Inbound headers are
<value expr="printObject(session.connection.protocol.sip)"/>
</log>
```

**clearOutboundHeaders()**

Removes all values from the outbound array. This will stop them being used on subsequent VXML transfers and disconnects for the same call.

**setOutboundHeader(name, value)**

Multi-functional method for adding outbound headers either from the inbound headers or by the user supplying a value. `name` is the header name to add to the outbound headers in `session.connection.protocol.sip.outbound` if specified is the value to assign to that header’s `.value` attribute. If not specified, the whole header object is copied from the inbound array.

Returns the outbound header object to which you can then assign additional attributes if you need to (this would be uncommon), or null if the outbound header could not be created due to invalid name or a null value coupled with no such named header being in the inbound array.

The outbound headers values can be specified manually by manipulating the `session.connection.protocol.sip.outbound` object itself.

The headers are not cleared between actions, so if a call is transferred and as a result hung up. WebSphere Voice Response sends whatever headers have been configured on both the TRANSFER and BYE messages unless you clear them.
Conversely, if you are using VoiceXML started from CCXML, the headers are instead handled by creating an object that is passed as the hints object on the relevant CCXML tag that initiates the message (such as createcall and send when sending ibmwvr.transfer). For example:

```xml
<ccxml version="1.0">
<script>
  function makeOutboundHeader(value) {
    var ret = new Object();
    ret.value = value;
    return ret;
  }
</script>
<eventprocessor>
  ...
  <transition event="dialog.transfer">
    <var name="target" expr="event$.URI"/>
    <var name="outbound" expr="new Object()"/>
    <var name="outbound['x-user-to-user']" expr="makeOutboundHeader('Example header value')"/>
    <var name="outbound['x-checksum']" expr="makeOutboundHeader('1234567890')"/>
    <log expr="Outbound header x-user-to-user set as ' + outbound['x-user-to-user'].value'"/>
    <send target="event$.connectionid" targettype="connection" name="ibmwvr.transfer" namelist="target" hints="outbound"/>
  </transition>
  ...
</eventprocessor>
</ccxml>
```

In the above example, x-user-to-user and x-checksum must be defined in siphdrtags.cfg or the headers will not be sent out on the messages.

Please note that values set as described for VoiceXML-only transfers will appear on the dialog.transfer event but will not automatically be added to the headers, as that responsibility is taken by CCXML.
Chapter 2. Installing WebSphere Voice Response Voice over IP

This chapter gives instructions for installing hardware and software specific to WebSphere Voice Response Voice over IP. This information assumes that you have already installed WebSphere Voice Response. If you have not yet installed WebSphere Voice Response, see WebSphere Voice Response for AIX: Installation for additional instructions.

This chapter includes the following sections:

- “Installing the DTNA adapter”
- “Software installation” on page 27

Installing the DTNA adapter

DTNA is a software device driver supplied with WebSphere Voice Response Version 6.1. This device driver provides the Voice-over-IP (SIP) streaming interface between WebSphere Voice Response and the system unit TCP/IP (probably Ethernet) adapter and interface.

It is possible for DTNA software adapters to be installed in the same system unit as the DTTA adapters with the restriction that a maximum of FOUR adapters of any type can be installed in the system unit.

Each installed DTNA adapter supports FOUR trunks with each trunk providing VoIP telephony support for either 24 or 30 channels if configured as T1 or E1 respectively. If you have DTTA PSTN adapters in the same system unit for which you are configuring DTNAs, you must set the mode (T1 or E1) of the DTNAs to be the same as DTTA. If your system is ‘pure VoIP’ (no PSTN adapters), it is not mandatory for you to set the mode appropriate to your country but it is highly recommended that you do this.

Note: Although VoIP does not imply the concept of ‘trunks’, WebSphere Voice Response is still organized in 24 or 30 Channels per Trunk. This is done to maintain compatibility with the PSTN version of the product and also to allow coexistence of PSTN and VoIP in the same system to be easily supported.

Supported machines

WebSphere Voice Response Version 6.1 with DTNA can be used in any IBM system unit that uses the Power 4, Power 5, Power 6, or Power 7 processors.
and the AIX 7.1 or AIX 6.1 operating system, for example, IBM Bladecenter (with JS processors) or IBM System p system units.

Using DTNA, up to four WebSphere Voice Response LPARs are supported. An appropriate allocation of CPU, memory, storage and Ethernet resources must be assigned to each WebSphere Voice Response LPAR. As a guideline, a system performance rPerf of 20 can support a 480-channel WebSphere Voice Response LPAR running speech-enabled VoiceXML applications. See http://www.ibm.com/systems/power/hardware/notices/rperf.html for further information on rPerf.

The use of secure SIP and secure RTP, increases the system performance requirements of WebSphere Voice Response by approximately 30%. As a guideline, a system performance rPerf of 26 can support a 480-channel secure SIP and secure RTP WebSphere Voice Response LPAR running speech-enabled VoiceXML applications.

See also “Capacity planning for DTNA” on page 10.

System unit connection
Unless your system has built-in Ethernet, to use SIP signaling, you will need an Ethernet adapter installed and configured for use with AIX. See your AIX manuals for instructions. You can then connect to your IP network to use it for SIP signaling using the Ethernet port on the installed card or at the back of the pSeries computer. See WebSphere Voice Response for AIX: Installation for more information on connecting WebSphere Voice Response to the network.

Note that you can choose to use separate Ethernet adapters for SIP and voice (RTP) communication, or a separate Ethernet adapter for both SIP and voice (RTP) communication.

Installing the DTNA device drivers
In order to use DTNA and SIP for VoIP, you must install the following WebSphere Voice Response filesets as a minimum:

- dirTalk.DT
- devices.dirTalk.artic960
- dirTalk.VOIP_SIP

The filesets dirTalk.DT and devices.dirTalk.artic960 are mandatory for WebSphere Voice Response without Voice over IP but dirTalk.VOIP_SIP is optional for WebSphere Voice Response. If you have already installed WebSphere Voice Response but are unsure whether or not dirTalk.VOIP_SIP is installed, follow the instructions in step 2 on page 27 of the procedure in “Installing the WebSphere Voice Response VoIP software” on page 27.
Installing the device drivers

The DTNA device driver and associated files (methods and so on) are supplied in the fileset `devices.artic960.rte`. The device driver for DTNA is named `vnaio` and is placed by the install process in the `/usr/lib/drivers` directory.

1. Log on to AIX as **root**, if you are not logged on already.
2. To determine if the `devices.artic960.rte` fileset is installed, run the command `lslpp -l devices.artic960.rte` and ensure that the file name is shown as being installed.
3. To ensure that the DTNA device driver is present in its operating directory, run the command:
   ```bash
   ls -al /usr/lib/drivers/vnaio
   ```
4. If either of these checks shows that the DTNA device driver is not installed, locate the `devices.artic960.rte` fileset on your WebSphere Voice Response installation media and use `smitty` to install the fileset.
   a. To activate the new device driver, shut down and restart AIX, by running the command `shutdown -Fr` and pressing Enter.
   b. Repeat steps 2 and 3.

DTNA devices are not fully installed until the ownership of the devices is specified. See “Setting ownership of the DTNAs.”

Setting ownership of the DTNAs

This section assumes that you have four DTNAs installed in your system unit. DTTAs can be installed alongside DTNAs (up to a maximum of four adapters). For more information on setting ownership of DTTAs, and for example output for these adapter types, see *WebSphere Voice Response for AIX: Installation*.

If you are using DTNAs with WebSphere Voice Response VoIP, you need to set WebSphere Voice Response as the owner of these adapters.

1. Log on to AIX as **root**, if you are not logged on already.
2. Ensure that WebSphere Voice Response is not running (for example, by typing `ps -ef | grep NODEM`)
3. Type the following command and press ENTER:
   ```bash
   . /usr/lpp/dirTalk/tools/vae.setenv
   ```
   Make sure that you leave a space between the . (period) and `/usr`.
4. You need to create a device for each of the DTNAs you want installed in the system unit and register WebSphere Voice Response as the owner of
each of these devices. Up to four DTNAs can be installed in a system unit where each DTNA adapter provides support for four trunks, giving a total of 16 trunks.

The `dt_setowner` command is used to create and register these devices. When installing a DTNA, you need to supply three options to the `dt_setowner` command: the adapter type `-n` for DTNA, the adapter number `-s`, and the trunk type `-t`, which can be either T1 or E1.

For DTNA, the `dt_setowner` command is in the format:

```
dt_setowner -n -s adapter_number -t T1 | E1
```

where `adapter_number` is the last digit of the device identifier, starting at 0 for the first adapter and increasing by 1 for each subsequent adapter up to a maximum value of 3.

For example, to create four T1 DTNA adapters and register WebSphere Voice Response as the owner, type the following commands, pressing Enter after each one:

```
dt_setowner -n -s 0 -t T1
dt_setowner -n -s 1 -t T1
dt_setowner -n -s 2 -t T1
dt_setowner -n -s 3 -t T1
```

**Note:** If you have DTTA PSTN adapters in the same machine, DTNAs must be configured as the same type (T1 or E1).

If you only have DTNA in the machine, the preference is to use T1 in US or Japan and E1 in the rest of the world. The advantage of E1 is that it allows the full 120 channels of DTNA to be used. However there may be complications which must be carefully considered as the internal mode of operation of WebSphere Voice Response will now be A-law rather than the natural mode of µ-law for T1.

If you have DTTA PSTN adapters in the same machine, ensure that the DTTA and DTNA adapters are configured to use different addresses.

For more information on using `dt_setowner`, see *WebSphere Voice Response for AIX: Installation*.

Each `dt_setowner` command creates three AIX devices:

- A DTNA 'adapter' `vnaio<n>`
- A DTNA device driver `ddvnaio<n>`
- A WebSphere Voice Response line device `dtline<n>`

(Check using `lsdev -C | egrep "vna|dtline"`).

The following is an example of a session which sets up two DTNA adapters:
Software installation

Installing WebSphere Voice Response Voice over IP related software consists of the following tasks:

- "Installing the WebSphere Voice Response VoIP software"
- "Importing the VoIP custom server" on page 29

Information on installing and configuring device drivers for the DTNA adapter is included in "Installing the DTNA device drivers" on page 24

Installing the WebSphere Voice Response VoIP software

You can install the VoIP software fileset at the same time as you install WebSphere Voice Response or later. For more information on how to install WebSphere Voice Response, see WebSphere Voice Response for AIX: Installation.

Procedure

1. Log on to AIX as root, if you are not logged on already.
2. To check if the WebSphere Voice Response VoIP software is already installed, type the following command and press Enter:
lslpp -l 'dirTalk.VOIP_SIP'

An output similar to the following (showing a Level of 6.1.0.0 or higher) should be displayed:

<table>
<thead>
<tr>
<th>Fileset</th>
<th>Level</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Path: /usr/lib/objrepos</td>
</tr>
<tr>
<td>dirTalk.VOIP_SIP</td>
<td>6.1.0.0</td>
<td>COMMITTED</td>
<td>Voice Response VOIP feature for SIP</td>
</tr>
</tbody>
</table>

This means that the VoIP software is already installed and you can skip the rest of this procedure.

3. To mount the CD containing the device drivers, type the following command and press Enter:

   smitty mountfs

4. Press F4 in the **FILE SYSTEM name** entry field to display a list of file systems.

5. Select the CD-ROM file system (for example /dev/cd0).

6. Press Enter.

   The system mounts the CD. Review the output when it has finished.

7. Press F10 to exit from SMIT.

8. Use the cd command to move to the file location of your install directory (if you are not already in the correct directory).

9. Type smitty install_latest, then press Enter.

10. In the **INPUT device/directory for software** field, type . (period) for the current directory.

11. In the **SOFTWARE to install** field, press F4 to list the filesets in this directory.

12. Select the dirTalk.VOIP_SIP fileset by pressing F7, then pressing Enter.

13. The system installs the software.

14. Review the output.

15. Press F10 to exit from the install menu.

16. To activate the new software, shut down and restart AIX, by typing the shutdown -Fr and pressing Enter.

**Setting the dtuser file permissions**

You need to use the **vae.setuser** utility to ensure that the file permissions and ownerships are correct. You need to run this utility whether you are using the dtuser account provided by the WebSphere Voice Response installation process or you have set up your own AIX account.

1. Log on to AIX as root, if you are not logged on already.

2. Type the following command and press Enter:

   /usr/lpp/dirTalk/tools/vae.setuser
3. The prompt Enter user ID for running WebSphere Voice Response (Must begin with alpha character) is displayed. Type dtuser (unless you have set up an AIX account with a different name).

4. Reboot the system.

**Importing the VoIP custom server**

1. Become dtuser.

2. Start WebSphere Voice Response by typing the following command, then pressing Enter:
   `vaenit`

3. When the system displays the Login menu:
   
   WebSphere Voice Response User Login
   1) Start WebSphere Voice Response Processes
   2) Do Not Start WebSphere Voice Response
   Enter choice (or <ENTER> for option list)

   select option 1.

4. When the system displays the Logon window, log in to WebSphere Voice Response as admin.

5. From the Welcome window, click Applications —> Applications to display the Applications window.

6. In the Applications window, click Applications —> Import —> Replace —> File.
   
   The system displays an Import File dialog box.

7. In the search string field, type:
   `/usr/lpp/dirTalk/sw/voip/sigproc_sip/*`
   
   The system displays a list of files in that path.

8. Select the `sigproc_sip.imp` file from the Files list.

9. Click OK.
   
   The system imports the files, then displays the Import Report window, showing information about the import process.

10. Press Enter to close the window.
Chapter 3. Configuring WebSphere Voice Response Voice over IP

This chapter tells you how to configure WebSphere Voice Response for use with Voice over IP. After completing the following instructions you will be able to run applications on your WebSphere Voice Response VoIP system.

The instructions are divided into the following sections:

- "Using an allowed host list"
- "Using a SIP Registrar" on page 32
- "Setting the country/region" on page 36
- "Configuring WebSphere Voice Response for Voice over IP" on page 37
- "Setting the SIP transport IP address" on page 44
- "Shutting down and restarting WebSphere Voice Response" on page 45
- "Activating your channels" on page 46

For a complete list of Voice over IP configuration parameters, see Appendix A, "System parameters," on page 131.

Using an allowed host list

WebSphere Voice Response for AIX supports the use of an allowed host list of IP addresses from which SIP Requests are accepted. A sample configuration file named allowedHostList.ini.orig, and a working copy named allowedHostList.ini are provided in directory $SYS_DIR/voip. This file contain comments explaining how to use the file, and also some example configurations.

To configure WebSphere Voice Response for AIX to use an allowed host list:

1. Edit allowedHostList.ini to list on separate lines, allowed individual IP addresses, or an allowed range of IP addresses, defined by a netmask in the format ipaddress;netmask.

   **ipaddress**
   The IP address whose subnet you want to allow

   **netmask**
   The part of the IP address that is considered to be the allowed subnet. IPv4 and IPv6 addresses are supported.
For example, a setting of 192.168.1.1;255.255.0.0 configures WebSphere Voice Response to accept SIP requests from any IP address beginning with 192.168.

A setting of 192.168.1.1;255.255.255.255 configures WebSphere Voice Response to accept SIP requests only from IP address 192.168.1.1. This has the same effect as omitting the netmask parameter.

If the network is IPv6, a setting of 192.168.1.1;255.255.0.0 would also accept SIP requests from any IP addresses that end in those values in other areas of the IPv6 range. The netmask to match IP addresses only in the IPv4 range would be FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.0.0 (assuming it is paired with an IPv4 address).

2. From the WebSphere Voice Response Welcome window:
   a. Click Configuration -> System Configuration -> Change -> VoIP SIP Signalling.
   b. Select Use allowed host list.
   c. Click Yes.
   d. Click File -> Save to save the configuration.

   See "Use allowed host list" on page 169 for further information on the system parameter.

3. Disable all trunks that use VoIP.

4. Re-enable VoIP trunks. The allowedHostList.ini configuration file is only read when VoIP starts up, and the Use allowed host list parameter is set to Yes.

   After configuration, SIP calls from IP addresses that are not on the allowed host list will be rejected and a SIP 403 Forbidden message issued. The reason given in the message is Requestor not from an accepted host.

   Yellow WebSphere Voice Response warning alarms stating Blocked request from non-accepted host are issued whenever a request from an IP address that is not on the accepted host list is rejected. The warning includes the originating IP address of the blocked host.

Using a SIP Registrar

WebSphere Voice Response for AIX supports a SIP register service that registers a list of ‘usernames’ to one or more registrars as defined in a user-configurable file $SYS_DIR/voip/master.ini and secondary files such as $SYS_DIR/voip/basicList.ini. The file master.ini defines a list of registrars. Each registrar defines a different server to which to send REGISTER attempts.
has an associated list of user names in one or more files, and may override default values for port, timeout of registration, user agent, and priority. For example:

Registrar 127.0.0.1
IniFile=/usr/lpp/dirTalk/db/sys_dir/voip/basicList.ini
Timeout=60
UserAgent=WebSphere Voice Response
Port=5061
Priority=0.7
Proxy=12.34.56.78
RegisterAs=93.0.0.4
HidePasswords=false
UseSIPS=false
TransportProtocol=udp
;

These registrations are triggered by putting one or more VoIP channels into service and will be unregistered if all VoIP channels go out of service. Registrations will be automatically reattempted when they expire.

To configure a SIP register service for your system, you must edit master.ini to define the appropriate registrar and attendant properties, and edit basicList.ini (or create your own secondary .ini file or files) to define the list of users to register.

The master.ini server definition parameters are as follows:

**Registrar**
- IP address or hostname of the registrar. (mandatory)

**IniFile**
- The path of a secondary .ini files that contain details of the actual REGISTER requests to make (mandatory). If multiple .ini files are listed (using multiple IniFile parameter definitions), the contents of each of the defined files are combined sequentially, in file name order, to form the final list of registrations for that registrar. If the same username is used in more than one secondary .ini file, the latest username added to the registrar definition overrides the entries in any files combined previously. To send exactly the same REGISTER requests to two different registrars, include IniFile=fileName in both Registrar definitions. File names are resolved relative to the $SYS_DIR/voip directory.

**Timeout**
- The period, in minutes, a registration should last before expiring and being reregistered (optional). This value can be overridden by lines in the secondary .ini file. The default value is the Timeout value defined in the system configuration.
**UserAgent**
Part of the register request and specifies who made the request (optional). If the system configuration parameter is set, this will have _hostname appended to it to make it easier to identify the machine a registration came from. This value can be overridden by lines in the secondary .ini file. The default value is the UserAgent value defined in the system configuration.

**Port**
The UDP port that the registrar uses (optional). Usually for UDP or TCP, the default is 5060. For TLS, the default is 5061. It is possible to change the port for any protocol if necessary.

**Priority**
The Contact 'q=' parameter referred to in the SIP specification rfc3261, which is by convention a value between 0 and 1 (optional). Some registrars support two machines registering the same name to themselves with differing priorities, so that if the main machine goes down and unregisters, a back-up machine can take over, for example. It is possible for a registrar to do load balancing using this facility by splitting the load proportionally according to priority. The default value is 0.5.

**Proxy**
Specifies a proxy IP address for the registrar (optional). The Registrar value will be used in all TO, FROM and CONTACT headers in the message, but the message will be sent to the proxy address.

**RegisterAs**
Specifies the IP address to use when identifying the WebSphere Voice Response machine that is doing the registration. By default this is inferred using the local host name.

**HidePasswords**
If set to true, prevents the password being added to the To and From headers of the REGISTER message. The password will still be used by the authentication challenge mechanism. The default value is false, which if a password is specified, results in the password being added to all instances of the address.

**UseSIPS**
Controls whether to register the contact as a sips rather than a sip address. Usually, if sips is specified, TransportProtocol is set to TLS, but a sips address can be registered using an unsecure SIP message if required.

**TransportProtocol**
Specifies the transport protocol to use to send the REGISTER message. Valid values are TCP, UDP, or TLS. The defaults is UDP. If set to TLS,
secure SIP must be set up to allow the protocol to work. See “Secure SIP configuration settings for outbound calls” on page 92 for more information.

The master.ini file can include comment lines beginning with the ‘#’ character.

The supplied example secondary .ini file, basicList.ini, shows how to override the timeout and user agent fields on a per-registration basis. You must supply the user name, but the timeout and user agent values are optional. You can provide either or both of those if you wish. The format of a line entry for a secondary .ini file is

Username[::Password][,Timeout][,UserAgent]

Or, if the Timeout is that specified in master.ini or the default:

Username[::Password],0,UserAgent

For example: 7777,13,MyUserAgent registers, without including any password, the URI sip:7777@thisMachine:5060 to the registrar with a timeout of 13 minutes and a requesting UserAgent of MyUserAgent, where thisMachine is the hostname of the resource where the secondary .ini file is located. To include a password of 2222 in the registration, the line entry would be 7777:2222,13,MyUserAgent.

7777,0,MyUserAgent registers the same URI to the registrar using the timeout value defined in master.ini if there is one, or if not, that defined in the system configuration and a requesting UserAgent of MyUserAgent. To include a password of 2222 in the registration, the line entry would be 7777:2222,0,MyUserAgent.

7777 registers the same URI to the registrar using the timeout value and UserAgent defined in master.ini if defined there, or if not, the values defined in the system configuration. To include a password of 2222 in the registration, the line entry would be 7777:2222.

The secondary .ini file parameters are as follows:

**Username**

The number at the WebSphere Voice Response machine registered to a registrar (mandatory). The username can be specified in several different ways:

- For a 480-channel system, each available channel could be defined as an individual available username, for example 001@mywvr to 480@mywvr
- To represent individual applications, (equivalent to a VRBE NumToApp application mappings representation). Registering one
or more applications that can be answered by a particular machine, for example: 12345@myWVR identifies that the WebSphere Voice Response system can handle calls for user 12345 and has an equivalent NumToApp mapping defined in the default.cff configuration file.

- A single username registered to represent the machine as available, for example: WVR61@xyz.ibm.com

Password
Optionally, a password can be specified in the registration.

Timeout
The period, in minutes, a registration should last before expiring and being reregistered (optional). The default value is the Timeout value defined in master.ini if there is one, or if not, that defined in the system configuration.

UserAgent
Part of the register request and specifies who made the request (optional). If the system configuration parameter is set, this will have _hostname appended to it to make it easier to identify the machine a registration came from. The default value is the UserAgent value defined in master.ini if there is one, or if not, that defined in the system configuration.

Secondary .ini file can include comment lines beginning with the ‘#’ character.

Default SIP register support is configured using the following VoIP SIP Signaling configuration parameters:
- Add Host Name To User Agents?
- Register Addresses on Startup
- Register Default Timeout (Minutes)
- Register Default User Agent

---

**Setting the country/region**

This section tells you how to set the country/region.

1. From the Welcome window, select Configuration > Pack Configuration > Change.
   The Pack Configuration window opens.
2. The country/region for which WebSphere Voice Response is currently configured is shown to the right of the **Country/Region** button at the top of the window. If the country/region is not correct, click this button. The Country/Region window opens.

3. If the name of your country/region is displayed in the list:
   a. Select the required country/region.
   b. Click **OK** to confirm your selection.

4. If the name of your country/region is not displayed:
   a. Select **Other (E1)** or **Other (T1)** as appropriate for your telephony system.
   b. Click **OK** to confirm your selection.
   
   The selected CAS type is displayed at the top of the Pack Configuration window.

---

**Configuring WebSphere Voice Response for Voice over IP**

This section tells you how to configure WebSphere Voice Response for Voice over IP.

1. To configure WebSphere Voice Response for Voice over IP, select **Configuration** → **Pack Configuration** → **Change** from the WebSphere Voice Response welcome window to display the **Pack Configuration** window:
Each pack has a corresponding **Trunk Parameters** button. To configure VoIP settings for an NPACK (for example, Pack 1), select this button for pack 1.

**Note:** The pack type shown in the Pack Configuration and System Monitor windows is:

- EPACK for DTEA
- NPACK for DTNA

2. For an NPACK, a Trunk Interface Parameters window similar to the following is displayed:
3. Click the drop-down button labeled Channels Assigned and select from it the number of channels that you want to assign to this trunk. Remember that your license allows you to use no more than a specified number of channels.

4. The Trunk Signaling Mode button will be set to VoIP.

5. Click OK.

6. The Trunk Interface Parameters window closes and you are returned to the Pack Configuration window.
7. The **Operating Status** determines the state in which the WebSphere Voice Response Voice Over IP interface is placed whenever WebSphere Voice Response is started. To change the operating status, click the **Operating Status** button.

The operating status selection window opens.

8. Click the required operating status. Possible values are:

   **Inservice**
   
   The required microcode is loaded and diagnostics are run and the system is ready to process calls. The channels are set to Inservice: ready to make or receive calls. This is the normal setting when you are using the system in production.

   **Enabled**
   
   The required microcode is loaded and diagnostics are run. Although the pack shows as In Service, it cannot be used until you put the relevant channels In Service (**Operations → System Monitor**).

   **Available**
   
   The DTNA digital trunk adapter is present but is not ready to communicate with the network. The system is not ready to process calls until you enable the pack and set the channels In Service (**Operations → System Monitor**). The relevant pack can be configured (using the buttons to the right of this one), but the required microcode is not loaded.

   **Defined**
   
   The pack and the associated configuration options are fully disabled. **Defined** is a place holder until the DTNA digital trunk adapter is physically installed in the pSeries computer.

9. Click **OK** to confirm your update.

   The **Operating Status** button displays your selection.

10. Use the **Channel Group** buttons to set up any channel groups you require. For more information on channel groups, see **WebSphere Voice Response for AIX: Configuring the System**.

11. Click on the **Line Signaling** button for the first pack.

   The Line Signaling window opens.
12. Select the SIP signaling protocol.
   a. If you are not working with a SIP proxy server, select **Standalone (no Proxy)**.
• If you are using a gateway, select Standalone (no Proxy) and Use Gateway, and enter the URI of the gateway.

b. If you are working with a SIP proxy server with a known fixed IP address, select Local proxy and enter the IP address and port number of the SIP proxy server.

c. Populate the DNS SRV file.

If you are working with manual services routing, select the Services Routing option then select Manual Services Routing Table defined in /usr/lpp/dirtalk/db/sys_dir/srv.init (where an example srv.init file can be found). This option allows SRV request records to be manually entered into the srv.init file. The information in this file will then be used for routing by SIP signaling.

If the request records contain weights and priorities, scheduling of services is maintained according to RFC 2782.

The srv.init file can be configured dynamically whilst VoIP is running. Any changes made will be reflected within approximately 60 seconds of the file being saved.

Note: The Automatic Service Lookup:DNS SRV option is not currently supported.

13. Click OK.

14. To set the call direction, select the Direction & Tones button for the relevant pack. The Call Direction window will show the settings of the channel groups. To change a channel group’s call direction, click the appropriate button.

15. To change the default caller ID, click the Channel IDs button for the relevant pack. The Channel Identification window opens.

Figure 6. Channel Identification window
The number that you type in the **Telephone number** field is the number of the default application that you want to use for all VoIP calls when an application profile ID cannot be determined from the incoming message. This number will be stored in the Default Caller ID (CLID) field (in **System Configuration → Browse → VoIP SIP Signalling → Default CLID for Incoming VoIP Calls**).

You can only have one default VoIP application profile, and this will apply for all VoIP channels. If a system also has DTTA adapters, the channel IDs that are set for these are independent from those set for VoIP.

16. Repeat steps 1 on page 37 to 15 on page 42 for each pack you want to configure.

17. An asterisk (*) will be displayed next to the pack identifier for the packs you have configured. Select the **Save** checkbox next to each asterisk.

18. Select **File → Save**.

19. Select **File → Close**.

Your configuration changes will take effect when you shut down and restart WebSphere Voice Response. See "Shutting down and restarting WebSphere Voice Response" on page 45 for instructions.
Setting the SIP transport IP address

You only need to follow the instructions in this section if you have multiple network adapters and you need to force the signaling stack to use a particular Ethernet card IP address.

1. From the Welcome window, select **Configuration → System Configuration → Change**.
   The System Configuration window opens.
2. Click on the **VoIP SIP Signaling** button.
   The VoIP SIP Signaling window opens:

   ![VoIP SIP Signaling window]

   *Figure 8. VoIP SIP Signaling window*

3. Click on the **Override SIP Transport IP Address** button.
   The VoIP SIP Signaling/Override SIP Transport IP Address window opens:
4. Enter the IP address of your pSeries computer Ethernet card.
5. Click **OK**.
   
   Your configuration changes will take effect if you disable all VoIP trunks and re-enable them, or when you shut down and restart WebSphere Voice Response.

### Shutting down and restarting WebSphere Voice Response

This section tells you how to shut down and restart WebSphere Voice Response.

1. Return to the Welcome window and select **Operations** → **Immediate Shutdown**.
   
   The system displays a dialog box.

2. Click **OK** to confirm.
   
   The system shuts down all windows, cleans up the database, and stops all foreground and background processes.

3. **Do not restart WebSphere Voice Response immediately.** Always wait until you see the message “Node Manager terminating” in the WebSphere Voice Response Status window before you try to restart WebSphere Voice Response, to allow all processes to finish properly. Then you can close the window by typing Ctrl+C.

4. Start WebSphere Voice Response by typing the following command, and pressing Enter:

   ```
   vaenit
   ```

5. When the system displays the Login menu:

   ```
   WebSphere Voice Response User Login
   1) Start WebSphere Voice Response Processes
   2) Do Not Start WebSphere Voice Response
   Enter choice (or <ENTER> for option list)
   ```

   select option 1.
6. When the system displays the Logon window, log in to WebSphere Voice Response.

   **Note:** If the display is not set, the system prompts for which display to use before displaying the Logon window.

   See *WebSphere Voice Response for AIX: Managing and Monitoring the System* for more information.

---

### Activating your channels

This section tells you how to activate your channels.

1. From the Welcome window, select **Operations → System Monitor** to display the System Monitor window.

2. If you have not already set the channels to **Inservice** in the Pack Config window, activate your channels now from the System Monitor window by clicking the **Trunk Status** button for each trunk you configured, then clicking **Enable**.

   The trunk operating status is changed to **Inservice**.

3. Click the **Trunk Status** button again for each trunk and click **Channels in Service**.

4. Select **File → Close** to close the window.

   See *WebSphere Voice Response for AIX: Managing and Monitoring the System* for more information.

You can now run applications on your WebSphere Voice Response VoIP system. You can continue to configure your system in the normal way using the instructions provided in *WebSphere Voice Response for AIX: Configuring the System*. 
Chapter 4. Problem determination

This chapter includes the following sections:

- "Diagnosing DTNA problems"
- "Setting Voice over IP trace levels" on page 54
- "Using VOIP_MONITOR" on page 55
- "Frequently asked questions" on page 56

For details of Voice over IP error messages and alarms, refer to the *WebSphere Voice Response for AIX: Problem Determination* book.

Diagnosing DTNA problems

Assuming that configuration instructions in Chapter 3, “Configuring WebSphere Voice Response Voice over IP,” on page 31 have been followed, it should be possible to enable WebSphere Voice Response channels and make a Voice Over IP call into WebSphere Voice Response DTNA. If you have problems in doing this, the following checks may be useful:

1. Ensure that you have run a `dt_setowner` command with the `-n` option to create the DTNA devices. Use the `-t T1` or `-t E1` options to set T1 (North America and Japan) or E1 (rest of world) as appropriate.
2. Use `lsdev -C | egrep "vnaio|dtline"` to ensure that you have created the devices and that they are all available.
3. Bring up WebSphere Voice Response and configure VoIP using the Pack Configuration window as described in section “Configuring WebSphere Voice Response for Voice over IP” on page 37.
4. Save your configuration and restart WebSphere Voice Response.
5. Open the WebSphere Voice Response System Monitor window. Enable at least one trunk and bring channels into service. (Alternatively, use the `wvrtrunk` command to do this. Refer to the *WebSphere Voice Response for AIX: Managing and Monitoring the System* book for details of the `wvrtrunk` command.)
6. Make a VoIP call into the WebSphere Voice Response machine from a ‘softphone’.
7. Check for any Red or Yellow alarms in the System Monitor window.
8. If the call is refused (the problem is at the SIP signalling level), use the `VOIP_MONITOR` command to trace SIP traffic flows. See “Using VOIP_MONITOR” on page 55.
9. Check that the SIP setup sequence is following the standard INVITE, 100, 180, 200, ACK sequence.

10. The most common error is 415 Media Not Supported which is probably due to a mismatch in codecs between what is being offered and what is supported.

   **Note:** This includes the Telephone Event media option (also known as RFC2833 or RTP Payload). If this is being offered by a remote endpoint, WebSphere Voice Response must be configured for RFC2833 (VoIP DTEA and DTNA group, DTMF Transmission Method).

11. If the call appears to be setup correctly (System Monitor shows an active channel) and you still hear no voice, the problem is probably either due to the application not sending any voice or some problem with the IP connection between WebSphere Voice Response and your softphone.

12. Using standard AIX tools, check IP connectivity between the two machines.

13. If you have a NAT/Firewall between your machine and WebSphere Voice Response, ensure that you have used the Use Symmetric RTP option (VoIP DTEA and DTNA Media group).

**Analysing errors**

All DTNA errors are written to the WebSphere Voice Response error log (file $OAM_LOG_PATH/errorlog shown with the System Monitor).

The WebSphere Voice Response console log ($OAM_LOG_PATH/DTstatus.out) shows any DTNA errors.

In some cases, the AIX error log (produced by typing `errpt -a`) might also contain some useful information.

**Summary of useful tools for debugging VoIP/SIP**

**WebSphere Voice Response VOIP_MONITOR**

Provided as part of WebSphere Voice Response. Allows all SIP sequences to be monitored. Become dtuser and type `VOIP_MONITOR`. (Use the Ctrl-C key combination to stop it). Pipe to `tee filename` to record to a file for subsequent analysis offline. See "Using VOIP_MONITOR" on page 55 for more information.

**WebSphere Voice Response debugmon**

`debugmon` is a WebSphere Voice Response tool for recording voice input and output on specific telephony channels. Use with `-n` option for DTNA Option 8 allows a record to be started and stopped on a specific channel (about 30 seconds wrap-around buffer). Also records channel commands and status showing internal operation. Use Option 9 to extract trace data into AIX file. Use option `v` to record voice input and output continuously (no 30 second limit).
WebSphere Voice Response dnd

*dnd* is a tool which was used during DTNA development and might be useful in diagnosing some problems. The stats/monitor option shows time spent in the DTNA interrupt handler and within DSP code sections. The report option gives a snapshot of the RTP streaming taking place on DTNA channels, and shows endpoint addresses/ports and packet counts.

**AIX Trace**

VoIP SIP and Media activity traced to AIX trace channel 1. Use trace -1 to start, trcstop -1 to stop, print_trace or trcrpt to print. The content is fairly ‘esoteric’ but is essential for IBM support to debug problems. The trace level for VoIP SIP/Media is controlled with -Tx option on SIP_VOIP Custom Server *main args()* properties. (Use -T3 for the maximum level). You can use *dt_alarmd* to stop tracing when WebSphere Voice Response error occurs. (Refer to the *WebSphere Voice Response for AIX: Problem Determination* book for further details.)

**AIX iptrace**

Traces all activity on system unit IP connection. Refer to AIX *man* command or publications for usage information. As root, start with iptrace /tmp/ip.rpt, for example. To stop tracing, locate and kill the iptrace process. At this point you have a file (/tmp/ip.rpt) that contains the binary trace information. You then have two options:

- To format the file into text output using ipreport /tmp/ip.rpt >ip.fmt, for example.
- To import the binary trace file into a trace analysis tool such as *etherreal*.

Alternatively, search for INVITE to see first SIP message of sequence. After that you should be able to see RTP messages (usually 214 bytes including IP/UDP/RTP headers). Look for messages such as DEST UNREACH, which indicates that AIX is throwing away incoming packets due to incorrect socket setup.

**AIX netstat**

To see allocated sockets, use netstat -f inet -a. To see the number of packets being sent received, use netstat -o. To see information about a specific interface use netstat -v en0, for example. Other *netstat* options may also be useful.

**AIX vmstat**

To see CPU utilization, use *vmstat* 1, for example.

Other CPU monitoring tools may be useful such as *smon*. 
**RTP and RTCP port allocation**

DTNA requires a contiguous block of 960 ports (480 even-numbered ports for RTP and 480 odd-numbered ports for RTCP) for a complete set of WebSphere Voice Response channels. These are allocated starting from the base value specified in the System Parameter (VoIP DTEA and DTNA Media group). For example, if the Minimum Port Number is set to 6000, Trunk 1 Channel 1 (RTP) is allocated to port 6000, Trunk 1 Channel 1 (RTCP) is allocated to port 6001, Trunk 1 Channel 2 (RTP) to port 6002, Trunk 2 Channel 1 (RTP) to port 6060 and so on, up to Trunk 16 Channel 30 (RTCP) to port 6959.

WebSphere Voice Response will start using the ports when a channel is enabled (brought into service from System Monitor). A failure will occur at this point if another application has already grabbed the same port and has not allowed it to be multiply used. In this case, the WebSphere Voice Response failure will indicate that Bind Failed with rc=67.

If another application attempts to use a port after it has been allocated by WebSphere Voice Response, the application is likely to fail in the same way unless it has specifically taken action to reuse the same port. However, in this case (with WebSphere Voice Response and another application both allocated to the same port) there are likely to be problems which may be hard to detect. Therefore care should be taken to ensure that no port overlap occurs.

Port allocation can be verified using the `netstat -f inet -a` command. One way to ensure that no overlap is likely to occur is to start all applications on your machine (including WebSphere Voice Response) but do not enable the DTNA trunks. You can then use `netstat` to examine port usage and ensure that the planned port range (starting at the default value of 6000) does not already have any allocated ports.

**Codecs**

The only codecs supported by DTNA are G.711 uncompressed voice (both A and µ law), and iLBC (Internet Low Bitrate Codec, RFC 3951), modes 20 and 30. WebSphere Voice Response offers one or more of these codecs (as defined in the VoIP DTNA and DTEA Media group) on an outgoing SIP Invite and accepts one of the offered codecs on an incoming SIP Invite.

**Using iLBC**

iLBC is a narrow-band speech audio compression format that is open-source and royalty-free. It provides audio quality better than G.729A with built-in error correction to provide improved robustness in networks with high packet loss.

iLBC compression is significantly more complex than G.711 and so, understandably, requires greater processing power. A system that can
currently support 480 channels of G.711 can typically support up to 120 iLBC channels. Systems with a higher/lower rPerf will support more/less concurrent channels, respectively, up to the maximum 480 per system/partition. See [http://www.ibm.com/systems/power/hardware/notices/rperf.html](http://www.ibm.com/systems/power/hardware/notices/rperf.html) for further information on rPerf.

To enable iLBC in VoIP calls, it must be added to the list of codec preferences found in: Configuration-&gt;System Configuration-&gt;Change-&gt;VoIP DTEA and DTNA Media-&gt;nth Codec Preference.

If iLBC is not included in any of these settings, WebSphere Voice Response will not negotiate iLBC in any calls, inbound or outbound. The order of the configured codecs (where 1st is most preferable and 4th is least preferable) determines the order in which WebSphere Voice Response supplies them in the SDP part of an INVITE message. This in turn determines the order of preference for outbound calls from WebSphere Voice Response, where the highest codec on the list supported by both parties is selected.

It is also important to set the DTNA channel allocation method to ensure the call load is spread over the adapters on the system by setting the following parameter to Allocate calls balanced across trunks: Configuration-&gt;System Configuration-&gt;Change-&gt;VoIP SIP Signalling-&gt;Inbound Call Channel Allocation Method.

When configured in WebSphere Voice Response, if iLBC is offered with no mode specified, WebSphere Voice Response defaults to iLBC 20. All audio is converted to the standard WebSphere Voice Response audio format before it is presented to the application.

Trombone is possible with iLBC 20 and iLBC 30, however, tromboning is not supported between a 30ms codec and a 20ms codec. Therefore, iLBC 30 cannot be tromboned with ALAW, MULAW or iLBC 20. However, it is possible to trombone between two iLBC 30 lines. Table 1 below shows a compatibility matrix for tromboned calls:

<table>
<thead>
<tr>
<th>Codec</th>
<th>PCMA</th>
<th>PCMU</th>
<th>iLBC 20</th>
<th>iLBC 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCMA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>PCMU</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>iLBC 20</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>iLBC 30</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1. Tromboning compatibility
Performance implications

Because DTNA is a software implementation of the DTEA adapter (with some codec limitations), there is an additional load placed on the system unit due to Voice over IP processing. This manifests itself in a higher than normal 'System' CPU time as indicated by performance monitoring tools such as `vmstat`. The additional load depends on a number of factors:

- The number of channels active.
- Whether or not the application is using compressed or uncompressed voice.

Where LPAR is in use, each DTNA system instance must be assigned a minimum of two processors or processor cores for each LPAR. (A development system with less than 30 channels may be supported by a single processor). An IBM Power 6 or Power 7 with multiple processors or multiple core processors can support 480 channels of a simple VoiceXML or state table application.

Note: This is compressed voice within the WebSphere Voice Response application (the 5:1 GSM compression algorithm) rather than compression on the IP network. Most WebSphere Voice Response applications use uncompressed voice, the notable exception being Unified Messaging for WebSphere Voice Response which can be configured to use either compressed or uncompressed voice.

- How the SIP call setup process has negotiated to handle (mainly detection but possible generation) DTMF keys. The three options are:
  - DTMF 'Inband' tones
  - RTP 'Payload' (known as RFC2833)
  - SIP INFO DTMF

The use of 'Inband' DTMF detection significantly increases the CPU load, therefore every attempt must be made to use the one of the other two methods.

See "Capacity planning for DTNA" on page 10 for details of DTNA CPU loadings.

DTMF handling

DTMF keys are sent and received with DTNA using the RTP channel as either in-band (audio tones) or RTP 'Payloads' (which is also known as RFC2833). For in-band audio, WebSphere Voice Response DTNA runs DTMF signal processing detectors on the signal, and for RFC2833, it simply decodes the received RTP packets and places the resulting DTMF keys into the queue just as if they were decoded with the signal processing code.

There is a DTMF Transmission Method system parameter in the VoIP DTEA and DTNA Media group which controls the way in which DTMF ‘Inband’ or ‘Payload’ is negotiated.
For outbound calls, WebSphere Voice Response offers RFC2833 in the SDP (Telephone Event) if the above System Parameter is set to RTP Payload. Otherwise (for DTMF 'Inband') the Telephone Event option is not offered in the INVITE SDP.

For inbound calls, WebSphere Voice Response accepts Telephone Events (if offered by the far end) only if the above System Parameter is set accordingly. If there is a mismatch either way (far end requests Telephone Event, but WebSphere Voice Response is not configured to accept it or the other way round), WebSphere Voice Response sends a 415 error code and refuses the call.

After it is negotiated, the DTMF method type (Inband or Payload) applies in both directions. (For example, if ‘inband’ is agreed by both ends, WebSphere Voice Response detects and send tones and if RTP payload has been agreed, WebSphere Voice Response sends and accepts RFC2833 packets.

NAT/Firewall considerations

In situations where a NAT (Network Address Translator) is located between WebSphere Voice Response and the remote endpoints, an additional option known as ‘Symmetric RTP’ is provided. This is to overcome the problem that, when a NAT is placed in the RTP message flow, WebSphere Voice Response needs to send RTP packets to an address within the NAT rather than to the far end IP address and port as specified in the SIP initial negotiation. When Symmetric RTP is enabled (in the Voip DTEA and DTNA Media System Configuration group), WebSphere Voice Response does not transmit RTP packets until it receives a first incoming packet at which time it extracts the source address/port within that packet and, from that point on, transmits all outgoing RTP packets to that address/port (which is likely to be in the NAT/firewall). The NAT translates the source/destination addresses/ports using tables that it maintains internally and forwards the packet to the remote endpoint.

If you experience the problem that audio is only flowing one way, the problem is likely to be due to the presence of a NAT and can be resolved by selecting the Symmetric RTP option in System Configuration.

Note: Only one endpoint (including WebSphere Voice Response) in a session can use the Symmetric RTP option as a ‘deadly embrace’ can result if both endpoints hold off transmitting until a first packet is received.

Tromboning with DTNA

Tromboning (connecting audio paths of two active channels) can be done between any pair of DTNA channels.
However, tromboning is not possible between DTNA and DTTA channels (an error is raised if an application attempt to do PSTN-VOIP tromboning).

**DTNA packet size interval**
WebSphere Voice Response always transmits RTP packets containing 20ms of audio (160 bytes). The additional MAC, IP, UDP and RTP headers mean that the actual packets size on the Ethernet are always 214 bytes.

WebSphere Voice Response does not implement silence suppression—packets are sent continuously regardless of the signal content.

For receive, WebSphere Voice Response accepts any packet size (even varying). However, the optimal size is 160 bytes (20ms).

WebSphere Voice Response Implements a ‘jitter buffer’ for received packets, which allows packets to be delayed or received out of sequence due to network loading without voice breakup. This implementation will also fill in ‘comfort silence’ if packets are found to be missing. This also allows the far end transmitter to implement silence suppression.

---

**Setting Voice over IP trace levels**

There are two types of system trace available for WebSphere Voice Response Voice over IP:

- SIP stack trace, for tracing signaling errors.
- Media trace, for tracing RTP voice stream errors that are carried over the adapter.

Three levels of trace are available for each. Level 1 gives the least trace information and level 3 gives the most. Level 1 is the default.

To increase the amount of trace information recorded for either the SIP stack or media:

1. From the Welcome window, select **Applications → Applications**.
2. Double-click on the **IBM_SIP_VOIP** entry.
3. Select **Custom Servers**.
4. Double-click on **SIP_VOIP**.
   - A Custom Server (SIP_VOIP) window opens.
5. Select **File → Properties**.
   - A Properties (SIP_VOIP) window opens.
6. In the Main() args field, enter one of the following:
   - **-TS3**
     - to increase the SIP signaling trace to level 3.
-TM3
to increase the media trace to level 3.

-T3
to increase both the SIP stack trace and the media trace to level 3.
To increase to only level 2, substitute a 2 in place of the 3 shown in these examples.

7. Click OK to close the window.
8. In the Custom Server (IBM_SIP_VOIP) window, select File → Save then File → Close.
9. Use the System Monitor window to disable your trunks and channels.
10. From the Welcome window, select Operations → Custom Server Manager to view your custom servers.
11. Stop and restart the VoIP custom server.
12. Return to the System Monitor window and activate your channels. See "Activating your channels" on page 46 for instructions.

Trace can be started and stopped in the normal way:

To start trace, use `trace -a -1`
To stop trace, use `trcstop -1`

See *WebSphere Voice Response for AIX: Problem Determination* for instructions.

Using VOIP_MONITOR

This section describes how you can use the VOIP_MONITOR to help with problem determination.

VOIP_MONITOR monitors all SIP messages that are sent or received from the network by the SIP signaling process and displays the contents of the messages to the screen.

Starting VOIP_MONITOR

To start VOIP_MONITOR, run it as `dtuser`. *WebSphere Voice Response for AIX* must be up and running with trunks enabled and with channels in service.

To start the VOIP_MONITOR from an AIX window command line, type the following and press Enter:

`VOIP_MONITOR`

VOIP_MONITOR output is written to stdout so that it can be monitored.
Stopping VOIP_MONITOR
You can stop VOIP_MONITOR at any time by using Ctrl+C.

Logging VOIP_MONITOR trace information to a file
If you want to run VOIP_MONITOR, and save the output to a file for later analysis, there are two ways to do this:

• If you want the output written to the screen and to a file (called trace1.out in this example) use the following command:
  VOIP_MONITOR | tee trace1.out
  When call rates are high, VOIP_MONITOR can lose trace information due to printing to the console being slow.

• If you want the output written to the file only, use the following command:
  VOIP_MONITOR 1 > trace1.out
  This method is recommended when call rates are high.

Analyzing the VOIP_MONITOR output
VOIP_MONITOR displays all sent and received SIP messages. When a message is displayed, a time stamp is added at the start of the displayed message. You can use the information in VOIP_MONITOR to examine the SIP call flows as sent and received by WebSphere Voice Response.

Frequently asked questions

Installation
I have just installed WebSphere Voice Response but when I open the Pack Configuration window, no packs are present.
  Check that you have created the 'dtlines' using the dt_setowner -x command. For more information on the dt_setowner command, see WebSphere Voice Response for AIX: Installation.

Configuration
Do I need to restart WebSphere Voice Response after making changes to Pack Configuration?
  For most changes made to Pack Configuration it is necessary to restart WebSphere Voice Response.

I keep making changes in the Pack Configuration window but the changes are not being saved. Why not?
  Before selecting File → Save, ensure that you have checked the boxes on the left side of the packs to which you have made configuration changes.

Why am I unable to select the packs in the Pack Configuration window?
  Check that your WebSphere Voice Response configuration is compatible with the country/region you have selected. For example, if
the country/region is set to E1, the configuration should match this. Use the `dt_setowner` command to change the channel associated signaling setting.

**I want to open a configuration window (Pack Configuration or System Configuration) in 'change' mode, but the 'change' option is non-selectable, why is this?**

Check that you do not have another configuration window (Pack Configuration or System Configuration) already open in 'change' mode. You cannot modify both configuration windows at once.

**State table applications**

I want to make a call from WebSphere Voice Response to a SIP phone. How do I write a state table application that can do this?

Create the following actions:

1. **AssignData:**
   - Operand=Put Tag
   - Operand 1=TO_HDR
   - Operand 2=sip: `sip address of client`
   - Result=SV541

2. **MakeCall**
   - Set Dialed Number and the format string to dummy values such as 0000 and ####

3. **PlayPrompt**

4. **CloseEverything**

I try to make a call into WebSphere Voice Response from a SIP phone, but the INVITE is not received.

Ensure that the stack has been started on the same Ethernet port that you are sending to. See “Setting the SIP transport IP address” on page 44.

**Miscellaneous**

**What is meant by the term 'recycle the trunks'?**

This term means to:

1. Disable all of the trunks.
2. Stop and restart the VoIP custom server.
3. Re-enable all of the trunks.

**Are loopback calls possible with SIP?**

No
Chapter 5. Security

This chapter describes WebSphere Voice Response secure SIP and secure RTP.

Security concepts and mechanisms, as they apply to any computer system, are presented first. The rest of the information gives details of how those security mechanisms are implemented in WebSphere Voice Response.

Security concepts and mechanisms

This collection of topics describes aspects of security to consider in your WebSphere Voice Response Voice over IP installation.

The commonly accepted aspects of security are as follows:

- Identification and authentication
- Authorization
- Confidentiality
- Data integrity

Security mechanisms are technical tools and techniques that are used to implement security services. A mechanism might operate by itself, or with others, to provide a particular service. Examples of common security mechanisms are as follows:

- Cryptography
- Message digests and digital signatures
- Digital certificates
- Public Key Infrastructure (PKI)

When you are planning a WebSphere Voice Response Voice over IP installation, consider which security mechanisms you require to implement those aspects of security that are important to you.

Identification and authentication

Identification is the ability to identify uniquely a user of a system or an application that is running in the system. Authentication is the ability to prove that a user or application is genuinely who that person or what that application claims to be.
For example, consider a user who logs on to a system by entering a user ID and password. The system uses the user ID to identify the user. The system authenticates the user at the time of logon by checking that the supplied password is correct.

**Authorization**

Authorization protects critical resources in a system by limiting access only to authorized users and their applications. It prevents the unauthorized use of a resource or the use of a resource in an unauthorized manner.

**Confidentiality**

Confidentiality refers to protecting sensitive information from unauthorized disclosure.

When SIP and RTP communications are transmitted over a network, especially an insecure network such as the Internet, there is a risk of the data being intercepted. Secure SIP and secure RTP can mitigate the risk of the data being viewed by encrypting the transmitted data.

**Data integrity**

Data integrity refers to ensuring there is no unauthorized modification of data.

There are two ways in which data might be altered: accidentally, through hardware and transmission errors, or because of a deliberate attack. Many hardware products and transmission protocols have mechanisms to detect and correct hardware and transmission errors. The purpose of the data integrity service is to detect a deliberate attack.

Both secure SIP and secure RTP use hash functions to ensure data integrity over a network.

**Cryptographic concepts**

This collection of topics describes the concepts of cryptography applicable to WebSphere Voice Response Voice over IP.

The term entity is used to refer to a client, an individual user, or any other system capable of exchanging messages.

**Cryptography**

Cryptography is the process of converting between readable text, called plaintext, and an unreadable form, called ciphertext.

Cryptography occurs as follows:
1. The sender converts the plaintext message to ciphertext. This part of the process is called encryption (sometimes encipherment).
2. The ciphertext is transmitted to the receiver.
3. The receiver converts the ciphertext message back to its plaintext form. This part of the process is called *decryption* (sometimes *decipherment*).

The conversion involves a sequence of mathematical operations that change the appearance of the message during transmission but do not affect the content. Cryptographic techniques can ensure confidentiality and protect messages against unauthorized viewing (eavesdropping), because an encrypted message is not understandable. Digital signatures, which provide an assurance of message integrity, use encryption techniques. For more information, see “Digital signatures in SSL and TLS” on page 75.

Cryptographic techniques involve a general algorithm, made specific by the use of keys. There are two classes of algorithm:

- Algorithms that require both parties to use the same secret key. Algorithms that use a shared key are known as *symmetric* algorithms. Figure 10 illustrates symmetric key cryptography.

- Algorithms that use one key for encryption and a different key for decryption. One of these keys must be kept secret but the other can be public. Algorithms that use public and private key pairs are known as *asymmetric* algorithms. Figure 11 on page 62 illustrates asymmetric key cryptography, which is also known as *public key cryptography*.

The encryption and decryption algorithms that are used can be public but the shared secret key and the private key must be kept secret.

---

**Figure 10. Symmetric key cryptography**
Figure 11 shows plaintext encrypted with the receiver’s public key and decrypted with the receiver’s private key. Only the intended receiver holds the private key for decrypting the ciphertext. The sender can also encrypt messages with a private key. Anyone who holds the sender’s public key can decrypt such messages, with the assurance that the messages must be from the sender.

With asymmetric algorithms, messages are encrypted with either the public or the private key, but can be decrypted only with the other key. Only the private key is secret. The public key can be known by anyone. With symmetric algorithms, the shared key must be known only to the two parties. This condition is called the key distribution problem. Asymmetric algorithms are slower but have the advantage that there is no key distribution problem.

Other terminology that is associated with cryptography is as follows:

**Block cipher algorithm**
These algorithms encrypt data by blocks. For example, the RC2 algorithm from RSA Data Security Inc. uses blocks 8 bytes long.

**Stream cipher algorithm**
These algorithms operate on each byte of data.

**Message digests and digital signatures**
A message digest is a fixed size numeric representation of the contents of a message, computed by a hash function. A message digest can be encrypted, forming a digital signature.

Messages are inherently variable in size. A message digest is a fixed size numeric representation of the contents of a message. A message digest is computed by a hash function, which is a transformation that meets two criteria:
• The hash function must be one way. It must not be possible to reverse the function to find the message corresponding to a particular message digest, other than by testing all possible messages.

• It must be computationally infeasible to find two messages that hash to the same digest.

The message digest is sent with the message itself. The receiver can generate a digest for the message and compare it with the digest of the sender. The integrity of the message is verified when the two message digests are the same. Any tampering with the message during transmission almost certainly results in a different message digest.

A message digest created by using a secret symmetric key is known as a Message Authentication Code (MAC) because it can provide assurance that the message has not been modified.

The sender can also generate a message digest and then encrypt the digest by using the private key of an asymmetric key pair, forming a digital signature. The signature must then be decrypted by the receiver before it is compared with a locally generated digest.

Related concepts:

"Digital signatures in SSL and TLS" on page 75

A digital signature is formed by encrypting a representation of a message. The encryption uses the private key of the signatory and, for efficiency, usually operates on a message digest rather than the message itself.

Digital certificates

Digital certificates protect against impersonation, certifying that a public key belongs to a specified entity. They are issued by a Certificate Authority.

Digital certificates provide protection against impersonation because a digital certificate binds a public key to its owner. Digital certificates are also known as public key certificates because they give assurances about the ownership of a public key when an asymmetric key scheme is used. A digital certificate contains the public key for an entity and is a statement that the public key belongs to that entity:

• When the certificate is for an individual entity, the certificate is called a personal certificate or user certificate.

• When the certificate is for a Certificate Authority, the certificate is called a CA certificate or signer certificate.

If public keys are sent directly by their owner to another entity, there is a risk that the message could be intercepted and the public key substituted by another. Such an action is known as a man in the middle attack. The solution to this problem is to exchange public keys through a trusted third party, giving
you a strong assurance that the public key really belongs to the entity with which you are communicating. Instead of sending your public key directly, you ask the trusted third party to incorporate it into a digital certificate. The trusted third party that issues digital certificates is called a Certificate Authority (CA), as described in “Certificate Authorities” on page 65.

What is in a digital certificate:

Digital certificates contain specific pieces of information, as determined by the X.509 standard.

Digital certificates that are used by WebSphere Voice Response comply with the X.509 standard, which specifies the information that is required and the format for sending it. X.509 is the Authentication framework part of the X.500 series of standards.

Digital certificates contain at least the following information about the entity that is being certified:

- The owner’s public key.
- The owner’s Distinguished Name.
- The Distinguished Name of the CA that issued the certificate.
- The date from which the certificate is valid.
- The expiry date of the certificate.
- The version number of the certificate data format as defined in X.509. The current version of the X.509 standard is Version 3, and most certificates conform to that version.
- A serial number, which is a unique identifier that is assigned by the CA that issued the certificate. The serial number is unique within the CA that issued the certificate. No two certificates that are signed by the same CA certificate have the same serial number.

An X.509 Version 2 certificate also contains an Issuer Identifier and a Subject Identifier, and an X.509 Version 3 certificate can contain a number of extensions. Some certificate extensions, such as the Basic Constraint extension, are standard, but others are implementation-specific. An extension can be critical, in which case a system must be able to recognize the field. If it does not recognize the field, it must reject the certificate. If an extension is not critical, the system can ignore it if does not recognize it.

The digital signature in a personal certificate is generated by using the private key of the CA that signed that certificate. Anyone who needs to verify the personal certificate can use the CA's public key to do so. The CA's certificate contains its public key.
Digital certificates do not contain your private key. You must keep your private key secret.

Requirements for personal certificates:

WebSphere Voice Response Voice over IP supports digital certificates that comply with the X.509 standard.

In addition to the standards that specify the data format for a digital certificate, there are also standards for determining whether a certificate is valid. These standards have been updated over time in to prevent certain types of security breach. For example, older X.509 version 1 and 2 certificates did not indicate whether the certificate could be legitimately used to sign other certificates. A malicious user could obtain a personal certificate from a legitimate source and create new certificates that were designed to impersonate other users.

When using X.509 version 3 certificates, the BasicConstraints and KeyUsage certificate extensions are used to specify which certificates can legitimately sign other certificates. The IETF RFC 5280 standard specifies a series of certificate validation rules that compliant application software must implement to prevent impersonation attacks. A set of certificate rules is known as a certificate validation policy.

Certificate Authorities:

A Certificate Authority (CA) is a trusted third party that issues digital certificates to provide you with an assurance that the public key of an entity truly belongs to that entity.

The roles of a CA are as follows:
- On receiving a request for a digital certificate, to verify the identity of the requester before building, signing and returning the personal certificate
- To provide the CA’s own public key in its CA certificate
- To publish lists of certificates that are no longer trusted in a Certificate Revocation List (CRL)
- To provide access to certificate revocation status by operating an OCSP responder server

Distinguished Names:

The Distinguished Name (DN) uniquely identifies an entity in an X.509 certificate.

The following attribute types are commonly found in the DN:
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERIALNUMBER</td>
<td>Certificate serial number</td>
</tr>
<tr>
<td>MAIL</td>
<td>Email address</td>
</tr>
<tr>
<td>E</td>
<td>Email address (Deprecated in preference to MAIL)</td>
</tr>
<tr>
<td>UID or USERID</td>
<td>User identifier</td>
</tr>
<tr>
<td>CN</td>
<td>Common Name</td>
</tr>
<tr>
<td>T</td>
<td>Title</td>
</tr>
<tr>
<td>OU</td>
<td>Organizational Unit name</td>
</tr>
<tr>
<td>DC</td>
<td>Domain component</td>
</tr>
<tr>
<td>O</td>
<td>Organization name</td>
</tr>
<tr>
<td>STREET</td>
<td>Street / First line of address</td>
</tr>
<tr>
<td>L</td>
<td>Locality name</td>
</tr>
<tr>
<td>ST (or SP or S)</td>
<td>State or Province name</td>
</tr>
<tr>
<td>PC</td>
<td>Postal code / zip code</td>
</tr>
<tr>
<td>C</td>
<td>Country</td>
</tr>
<tr>
<td>UNSTRUCTUREDNAME</td>
<td>Host name</td>
</tr>
<tr>
<td>UNSTRUCTUREDADDRESS</td>
<td>IP address</td>
</tr>
<tr>
<td>DNQ</td>
<td>Distinguished name qualifier</td>
</tr>
</tbody>
</table>

The X.509 standard defines other attributes that do not typically form part of the DN but can provide optional extensions to the digital certificate.

The X.509 standard provides for a DN to be specified in a string format. For example:

```
CN=John Smith, OU=Test, O=IBM, C=GB
```

The Common Name (CN) can describe an individual user or any other entity, for example a web server.

The DN can contain multiple OU and DC attributes. Only one instance of each of the other attributes is permitted. The order of the OU entries is significant: the order specifies a hierarchy of Organizational Unit names, with the highest-level unit first. The order of the DC entries is also significant.

**Related concepts:**

“*What is in a digital certificate*” on page 64

Digital certificates contain specific pieces of information, as determined by the X.509 standard.

**Obtaining personal certificates from a certificate authority:**

You can obtain a certificate from a trusted external certificate authority (CA).

You obtain a digital certificate by sending information to a CA, in the form of a certificate request. The X.509 standard defines a format for this information, but some CAs have their own format. Certificate requests are typically generated by the certificate management tool `wvrcert`. The information contains your Distinguished Name and your public key. When your certificate
management tool generates your certificate request, it also generates your private key, which you must keep secure. Never distribute your private key.

When the CA receives your request, the authority verifies your identity before it builds the certificate and returns it to you as a personal certificate.

Figure 12 illustrates the process of obtaining a digital certificate from a CA.

In the diagram:
- "User identification" includes your Subject Distinguished Name.
- "Certification Authority identification" includes the Distinguished Name of the CA that is issuing the certificate.

Digital certificates contain more fields than those fields shown in the diagram. For more information about the other fields in a digital certificate, see "What is in a digital certificate" on page 64.

**How certificate chains work:**

When you receive the certificate for another entity, you might need to use a certificate chain to obtain the root CA certificate.

The certificate chain, also known as the certification path, is a list of certificates that are used to authenticate an entity. The chain, or path, begins with the certificate of that entity. Each certificate in the chain is signed by the entity that is identified by the next certificate in the chain. The chain terminates with a root CA certificate. The root CA certificate is always signed by the certificate authority (CA) itself. The signatures of all certificates in the chain must be verified until the root CA certificate is reached.
Each certificate can contain one or more extensions. A certificate that belongs to a CA typically contains a BasicConstraints extension with the CA flag set to indicate that it is allowed to sign other certificates.

**Public Key Infrastructure (PKI)**
A Public Key Infrastructure (PKI) is a system of facilities, policies, and services that supports the use of public key cryptography for authenticating the parties that are involved in a transaction.

No single standard defines the components of a Public Key Infrastructure, but a PKI typically comprises certificate authorities (CAs) and Registration Authorities (RAs). CAs provide the following services:
- Issuing digital certificates
- Validating digital certificates
- Revoking digital certificates
Distributing public keys

The X.509 standards provide the basis for the industry standard Public Key Infrastructure.

For more information about digital certificates and certificate authorities (CAs), see “Digital certificates” on page 63. RAs verify the information that is provided when digital certificates are requested. If the RA verifies that information, the CA can issue a digital certificate to the requester.

A PKI might also provide tools for managing digital certificates and public keys. A PKI is sometimes described as a trust hierarchy for managing digital certificates, but most definitions include additional services. Some definitions include encryption and digital signature services, but these services are not essential to the operation of a PKI.

Cryptographic security protocols: SSL and TLS

Cryptographic protocols provide secure connections, enabling two parties to communicate with privacy and data integrity. The Transport Layer Security (TLS) protocol evolved from that of the Secure Sockets Layer (SSL). WebSphere Voice Response Voice over IP supports both SSL and TLS.

The primary goal of both protocols is to provide confidentiality (sometimes referred to as privacy), data integrity, identification, and authentication by using digital certificates.

Although the two protocols are similar, the differences are sufficiently significant that SSL 3.0 and the various versions of TLS do not interoperate.

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) concepts

The SSL and TLS protocols enable two parties to identify and authenticate each other and communicate with confidentiality and data integrity. The TLS protocol evolved from the Netscape SSL 3.0 protocol but TLS and SSL do not interoperate.

The SSL and TLS protocols provide communications security over the internet, and allow client/server applications to communicate in a way that is confidential and reliable. The protocols have two layers: a Record Protocol and a Handshake Protocol, and these protocols are layered above a transport protocol such as TCP/IP. They both use asymmetric and symmetric cryptography techniques.

An SSL or TLS connection is initiated by an application, which becomes the SSL or TLS client. The application that receives the connection becomes the SSL or TLS server. Every new session begins with a handshake, as defined by the SSL or TLS protocols.
A full list of CipherSpecs supported by WebSphere Voice Response Voice over IP/SIP is provided in the $SYS_DIR/voip/ciphers.ini file.

For more information about the SSL protocol, see the information that is provided at [https://developer.mozilla.org/en-US/docs/Introduction_to_SSL](https://developer.mozilla.org/en-US/docs/Introduction_to_SSL). For more information about the TLS protocol, see the information that is provided by the TLS Working Group on the website of the Internet Engineering Task Force at [http://www.ietf.org](http://www.ietf.org).

**An overview of the SSL or TLS handshake**

The SSL or TLS handshake enables the SSL or TLS client and server to establish the secret keys with which they communicate.

The steps that enable the SSL or TLS client and server to communicate with each other can be summarized as follows:

- Agree on the version of the protocol to use.
- Select cryptographic algorithms.
- Authenticate each other by exchanging and validating digital certificates.
- Use asymmetric encryption techniques to generate a shared secret key, which avoids the key distribution problem. SSL or TLS then uses the shared key for the symmetric encryption of messages, which is faster than asymmetric encryption.

Full details of the messages that are exchanged during the SSL handshake are not provided, but in overview, the steps that are involved in the SSL handshake are as follows:

1. The SSL or TLS client sends a “client hello” message that lists cryptographic information such as the SSL or TLS version and, in the client's order of preference, the CipherSuites supported by the client. The message also contains a random byte string that is used in subsequent computations. The protocol allows for the “client hello” to include the data compression methods that are supported by the client.
2. The SSL or TLS server responds with a “server hello” message that contains the CipherSuite chosen by the server from the list that is provided by the client, the session ID, and another random byte string. The server also sends its digital certificate. If the server requires a digital certificate for client authentication, the server sends a “client certificate request” that includes a list of the types of certificates that are supported and the Distinguished Names of acceptable Certification Authorities (CAs).
3. The SSL or TLS client verifies the server’s digital certificate. For more information, see “How SSL and TLS provide identification, authentication, confidentiality, and integrity” on page 72.
4. The SSL or TLS client sends the random byte string that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data. The random byte string itself is encrypted with the server’s public key.

5. If the SSL or TLS server sent a “client certificate request”, the client sends a random byte string encrypted with the client’s private key, together with the client’s digital certificate, or a “no digital certificate alert”. This alert is only a warning, but with some implementations the handshake fails if client authentication is mandatory.

6. The SSL or TLS server verifies the client’s certificate. For more information, see "How SSL and TLS provide identification, authentication, confidentiality, and integrity" on page 72.

7. The SSL or TLS client sends the server a “finished” message, which is encrypted with the secret key, indicating that the client part of the handshake is complete.

8. The SSL or TLS server sends the client a “finished” message, which is encrypted with the secret key, indicating that the server part of the handshake is complete.

9. For the duration of the SSL or TLS session, the server and client can now exchange messages that are symmetrically encrypted with the shared secret key.

Figure 14 on page 72 illustrates the SSL or TLS handshake.
How SSL and TLS provide identification, authentication, confidentiality, and integrity

During both client and server authentication there is a step that requires data to be encrypted with one of the keys in an asymmetric key pair and decrypted with the other key of the pair. A message digest is used to provide integrity.

How SSL and TLS provide authentication

For server authentication, the client uses the server’s public key to encrypt the data that is used to compute the secret key. The server can generate the secret key only if it can decrypt that data with the correct private key.

For client authentication, the server uses the public key in the client certificate to decrypt the data the client sends during step 5 on page 71 of the
handshake. The exchange of finished messages that are encrypted with the secret key (steps 7 on page 71 and 8 on page 71 in the overview) confirms that authentication is complete.

If any of the authentication steps fail, the handshake fails and the session terminates.

The exchange of digital certificates during the SSL or TLS handshake is part of the authentication process. For more information about how certificates provide protection against impersonation, refer to the related information. The certificates required are as follows, where CA \( X \) issues the certificate to the SSL or TLS client, and CA \( Y \) issues the certificate to the SSL or TLS server:

For server authentication only, the SSL or TLS server needs:
- The personal certificate issued to the server by CA \( Y \)
- The server’s private key

and the SSL or TLS client needs:
- The CA certificate for CA \( Y \)

If the SSL or TLS server requires client authentication, the server verifies the client’s identity by verifying the client’s digital certificate with the public key for the CA that issued the personal certificate to the client, in this case CA \( X \).

For both server and client authentication, the server needs:
- The personal certificate issued to the server by CA \( Y \)
- The server’s private key
- The CA certificate for CA \( X \)

and the client needs:
- The personal certificate issued to the client by CA \( X \)
- The client’s private key
- The CA certificate for CA \( Y \)

Both the SSL or TLS server and client might need other CA certificates to form a certificate chain to the root CA certificate. For more information about certificate chains, refer to the related information.

What happens during certificate verification

As noted in steps 3 on page 70 and 6 on page 71 of the overview, the SSL or TLS client verifies the server’s certificate, and the SSL or TLS server verifies the client’s certificate. There are three aspects to this verification:

1. The digital signature is checked (see "Digital signatures in SSL and TLS" on page 75).
2. The certificate chain is checked. You should have intermediate CA certificates (see “How certificate chains work” on page 67).

3. The expiry and activation dates and the validity period are checked.

**Secret key reset**

During an SSL or TLS handshake a secret key is generated to encrypt data between the SSL or TLS client and server. The secret key is used in a mathematical formula that is applied to the data to transform plaintext into unreadable ciphertext, and ciphertext into plaintext.

The secret key is generated from the random text sent as part of the handshake and is used to encrypt plaintext into ciphertext. The secret key is also used in the MAC (Message Authentication Code) algorithm, which is used to determine whether a message has been altered. See “Message digests and digital signatures” on page 62 for more information.

If the secret key is discovered, the plaintext of a message could be deciphered from the ciphertext, or the message digest could be calculated, allowing messages to be altered without detection. Even for a complex algorithm, the plaintext can eventually be discovered by applying every possible mathematical transformation to the ciphertext. To minimize the amount of data that can be deciphered or altered if the secret key is broken, the secret key can be renegotiated periodically. When the secret key has been renegotiated, the previous secret key can no longer be used to decrypt data encrypted with the new secret key.

**How SSL and TLS provide confidentiality**

SSL and TLS use a combination of symmetric and asymmetric encryption to ensure message privacy. During the SSL or TLS handshake, the SSL or TLS client and server agree an encryption algorithm and a shared secret key to be used for one session only. All messages transmitted between the SSL or TLS client and server are encrypted using that algorithm and key, ensuring that the message remains private even if it is intercepted. SSL supports a wide range of cryptographic algorithms. Because SSL and TLS use asymmetric encryption when transporting the shared secret key, there is no key distribution problem. For more information about encryption techniques, refer to “Cryptography” on page 60.

**How SSL and TLS provide integrity**

SSL and TLS provide data integrity by calculating a message digest.
Use of SSL or TLS does ensure data integrity, provided that the CipherSpec uses a hash algorithm. See the list of CipherSpecs in the $SYS_DIR/voip/ciphers.ini file.

In particular, if data integrity is a concern, you should avoid choosing a CipherSpec whose hash algorithm is listed as "None". Use of MD5 is also strongly discouraged as this is now very old and no longer secure for most practical purposes.

**CipherSpecs and CipherSuites**
Cryptographic security protocols must agree the algorithms used by a secure connection. CipherSpecs and CipherSuites define specific combinations of algorithms.

A CipherSpec identifies a combination of encryption algorithm and MAC algorithm. Both ends of an SSL or TLS connection must agree the same CipherSpec to be able to communicate.

For more information about CipherSpecs, see the related information.

A CipherSuite is a suite of cryptographic algorithms used by an SSL or TLS connection. A suite comprises three distinct algorithms:

- The key exchange and authentication algorithm, used during the handshake
- The encryption algorithm, used to encipher the data
- The MAC (Message Authentication Code) algorithm, used to generate the message digest

There are several options for each component of the suite, but only certain combinations are valid when specified for an SSL or TLS connection. The name of a valid CipherSuite defines the combination of algorithms used. For example, the CipherSuite SSL_RSA_WITH_RC4_128_MD5 specifies:

- The RSA key exchange and authentication algorithm
- The RC4 encryption algorithm, using a 128-bit key
- The MD5 MAC algorithm

Several algorithms are available for key exchange and authentication, but the RSA algorithm is currently the most widely used. There is more variety in the encryption algorithms and MAC algorithms that are used.

**Digital signatures in SSL and TLS**
A digital signature is formed by encrypting a representation of a message. The encryption uses the private key of the signatory and, for efficiency, usually operates on a message digest rather than the message itself.
Digital signatures vary with the data being signed, unlike handwritten signatures, which do not depend on the content of the document being signed. If two different messages are signed digitally by the same entity, the two signatures differ, but both signatures can be verified with the same public key, that is, the public key of the entity that signed the messages.

The steps of the digital signature process are as follows:
1. The sender computes a message digest and then encrypts the digest using the sender’s private key, forming the digital signature.
2. The sender transmits the digital signature with the message.
3. The receiver decrypts the digital signature using the sender’s public key, regenerating the sender’s message digest.
4. The receiver computes a message digest from the message data received and verifies that the two digests are the same.

Figure 15 illustrates this process.

If the digital signature is verified, the receiver knows that:
- The message has not been modified during transmission.
- The message was sent by the entity that claims to have sent it.

Digital signatures are part of integrity and authentication services. Digital signatures also provide proof of origin. Only the sender knows the private key, which provides strong evidence that the sender is the originator of the message.

Note: You can also encrypt the message itself, which protects the confidentiality of the information in the message.
Secure SIP

Session Initiation Protocol (SIP) is the Voice over IP signaling protocol that negotiates communications to and from WebSphere Voice Response. These communications can come from various SIP entities such as proxies, SIP registrars, phones, or other telephony servers. Secure SIP provides authentication, in addition to integrity and confidentiality of messages, and security for SIP communications on non-secure networks like the internet. Secure SIP can prevent attacks such as tampering with messages, or the impersonation of a user.

The concepts of authentication, integrity, and confidentiality are explained in Chapter 5, “Security,” on page 59. It is important to note that Secure SIP secures only the signaling data of a call. To secure the audio as well, you must enable Secure RTP.

Normally SIP uses one of UDP or TCP to send SIP messages, which are by default, sent to port 5060. Secure SIP uses TLS, itself built on top of TCP, which by default uses port 5061. If asked to choose a protocol for SIP messages, only TLS is secure.

SIPS URI scheme

Secure SIP uses a different URI scheme to non-secure SIP URIs.

A normal, non-secure SIP address has the form:

```
sip:user@ibm.com:5060
```

However, to specify the use of secure SIP use `sips:` in the address instead of `sip:` for example:

```
sips:user@ibm.com:5061
```

In these examples, the :5060 and :5061 parts of the message are optional. This is because :5060 is the default port for non-secure SIP messages, and :5061 is the default port for secure SIP messages.

Secure SIP minimal configuration

Configuring secure SIP systems across a network can be a complicated task and needs to be done with care. You must complete the following steps to enable secure SIP:

1. **Create a keyring.db database**
2. **Add trusted certificates to keyring.db**
3. **Configure the WebSphere Voice Response server certificate**
4. **Ensure that other SIP entities can communicate with WebSphere Voice Response**
5. **Enable secure SIP**
Create the keyring.db database
WebSphere Voice Response uses a specialized database for storing X.509 certificates and any associated private keys that are used in secure SIP communications. WebSphere Voice Response expects to find the database file at $SYS_DIR/voip/keyring.db. You need to create this database before you can start any certificate manipulation. If the keyring.db file is not currently in the $SYS_DIR/voip directory, you can create the database with the following command (replacing mypassword with your own password):

```
wvrcert -keydb -create -db $SYS_DIR/voip/keyring.db -pw mypassword -stash
```

Ensure you run this command as the user that is used to run WebSphere Voice Response (by default this user is dtuser). wvrcert is a command that is supplied with WebSphere Voice Response to manage keyring.db. Its capabilities are explained in detail in “Using the wvrcert utility” on page 85.

This command creates four files in $SYS_DIR/voip: three database files keyring.db, keyring.rdb and keyring.crl, and also a keyring.sth file, which stashes the database password in a file that is secured by the AIX file privileges system. This stashing allows WebSphere Voice Response to access keyring.db without knowledge of your password.

WebSphere Voice Response accesses the keyring.db database during startup, so WebSphere Voice Response will require a restart to load any database updates.

Add trusted certificates to keyring.db
The procedure for adding trusted certificates to WebSphere Voice Response is described here in detail.

For every SIP entity you want to trust, you must either add the Certificate Authority (CA) certificate that signed the entity’s certificate or, if the entity has a self-signed certificate, its self-signed certificate. These entities might be proxies, registrars, or other SIP entities. How you find their certificates depends on the software used.

If you intend to create a WebSphere Voice Response server certificate that is signed by a CA, you also need to add that CA’s certificate. This requirement does not apply if you are using a self-signed certificate. Setting up the WebSphere Voice Response server certificate is explained in “Configure the WebSphere Voice Response server certificate” on page 79.

If the certificate that you want to add is in a certificate chain, you must also add all the certificates that are above it in the chain. You must add the certificates in strictly descending order, starting from the root, followed by the CA certificate immediately below it in the chain, and so on. Certificate chains are explained in “How certificate chains work” on page 67.
Run the following command to add a CA certificate or the public part of a
self-signed certificate to the key repository:

```
wvrcert -cert -add -db $SYS_DIR/voip/keyring.db -stashed -label label -file filename -format ascii
```

where:

- **-label label**
  The label to be attached to the certificate in keyring.db. This label is
  for your own convenience only, and is not part of the certificate.

- **-file filename**
  The name of the file that contains the certificate.

- **-format**
  The format of the certificate. The value can be ascii for
  Base64-encoded ASCII or binary for Binary DER data. The default is
  ascii.

To list the certificates currently in the database, run the following command:

```
wvrcert -cert -list -db keyring.db -stashed
```

Output similar to the following is generated:

```
Certificates found
  * default, - personal, ! trusted
    ! Trusted_CA
    - wvr_server_certificate
```

Trusted certificates are labeled with an exclamation mark. Personal certificates,
which are labeled with a dash, are explained in the next topic, "Configure the
WebSphere Voice Response server certificate."

### Configure the WebSphere Voice Response server certificate

To establish a secure connection, you require a server certificate and an
associated private key in keyring.db. A certificate that has a private key in
keyring.db is also called a personal certificate.

When a secure SIP connection is made to WebSphere Voice Response,
WebSphere Voice Response presents a personal certificate to other SIP entities.
All certificates in the keyring.db database have a label that is associated with
them. These labels are for your own organizational purposes and are not part
of the certificate. WebSphere Voice Response uses the personal certificate
labeled wvr_server_certificate. Adding this certificate is mandatory.

### Configuring a certificate

How you add the personal certificate and private key to keyring.db depends
on your needs. You can choose one or more of the following options:

1. Create a self-signed certificate for a test situation.
2. Create a certificate signing request to be signed by a Certificate Authority. For more information on Digital Certificates and Certificate Authorities, see Digital Certificates.

3. Import a certificate into the database from a PKCS #12 file.

**Option 1 - Creating a self-signed certificate:**

You can use a self-signed certificate initially to test a secure SIP setup. The subsections describe how to create a self-signed certificate/key pair in the WebSphere Voice Response `keyring.db` database, and how to extract the certificate. Possible problems are also described.

After you extract the certificate, you must ensure other SIP entities that WebSphere Voice Response will communicate with trust this self-signed certificate. The method to use depends on the software and the platform. Please refer to the individual product’s information.

*Creating a certificate:* To create a certificate, you use a `wvrcert` command of the following form:

```
wvrcert -cert -create -db $SYS_DIR/voip/keyring.db -stashed -label wvr_server_certificate -dn distinguished_name -size key_size -expire days
```

where:

- **-dn distinguished_name**
  
  The X.500 distinguished name enclosed in double quotation marks. At least one attribute is required. You can supply multiple OU or DC attributes.

- **-size key_size**
  
  The key size. The value can be 512, 1024, 2048 or 4096. The default is 1024.

- **-expire days**
  
  The expiration time in days of the certificate. The default is 365 days for a certificate.

A typical use would be like the following instance of the command:

```
wvrcert -cert -create -db $SYS_DIR/voip/keyring.db -stashed -label wvr_server_certificate -dn CN='echo $HOSTNAME'
```

In this case, the Common Name (CN) value is set to be the hostname of the machine by using the AIX variable `$HOSTNAME`. The common name of a personal certificate must be the host name of the machine that will use it. Other SIP entities will be expecting the hostname in the certificate to match the hostname of the machine.
Extracting the self-signed certificate: You must extract the certificate and ensure that any SIP entity to be communicated with is configured to trust this certificate. This configuration is software and platform specific. To extract the certificate from WebSphere Voice Response, run the following command:

```
wrcert -cert -extract -db $SYS_DIR/voip/keyring.db -stashed -label wvr_server_certificate -target filename -format ascii
```

- **target filename**
  The name of the destination file.

- **format format**
  The format of the certificate. The value can be ascii for Base64-encoded ASCII or binary for Binary DER data. The default value is ascii.

Possible problems: Any errors that arise when you create the certificate are likely to occur for one or more of the following reasons:

- Not being the user who runs WebSphere Voice Response (by default, dtuser)
- A certificate with the label wvr_server_certificate already exists in the database
- A certificate request with the label wvr_server_certificate already exists in the database

You can view the certificates and certificate requests in the database by running the following commands:

```
wrcert -cert -list -db $SYS_DIR/voip/keyring.db -stashed
wrcert -certreq -list -db $SYS_DIR/voip/keyring.db -stashed
```

You can then delete certificates and certificate requests in the database by running the following command (changing the value of label where appropriate):

```
wrcert -cert -delete -label wvr_server_certificate -db $SYS_DIR/voip/keyring.db -stashed
wrcert -certreq -delete -label wvr_server_certificate -db $SYS_DIR/voip/keyring.db -stashed
```

Option 2 - Requesting a personal certificate: If you want a certificate to be signed by an external Certificate Authority (CA), you can use wrcert to create a server certificate request by running the following command:

```
wrcert -certreq -create -db $SYS_DIR/voip/keyring.db -stashed -label wvr_server_certificate -dn distinguished_name -file filename
```

where:

- **dn distinguished_name**
  Specifies the X.500 distinguished name, which is enclosed in double quotation marks. At least one attribute is required. You can supply multiple OU and DC attributes.
Specifies the file name for the certificate request. This file is the one that you provide to be signed by your certificate authority.

A typical use would be like the following instance of the command:

```
wrcert -certreq -create -db $SYS_DIR/voip/keyring.db -stashed -label wvr_server_certificate -dn CN='echo $HOSTNAME' -file $SYS_DIR/voip/cert_request.arm
```

Here the Common Name (CN) value is set to be the hostname of the machine by using the AIX variable $HOSTNAME. The common name of a personal certificate must be the host name of the machine that will use it.

Adding CA certificates: Before you receive your signed personal certificate into the database, it is important that you add the signing CA certificate. If the CA certificate that you want to add is in a certificate chain, you must also add all the certificates that are above it in the chain. You must add the certificates in strictly descending order, starting from the root, followed by the CA certificate immediately below it in the chain, and so on. Certificate chains are explained in "How certificate chains work" on page 67.

To add all appropriate CA certificates, follow the procedure in "Add trusted certificates to keyring.db" on page 78.

Receiving personal certificates: When you have a signed certificate from a CA, you must receive the certificate into the database by running a command similar to this one:

```
wrcert -cert -receive -db $SYS_DIR/voip/keyring.db -stashed -file filename -format format
```

- **-file filename**: `filename` must be the fully qualified file name of the file that contains the personal certificate.

- **-format format**: `format` can either be the default value ascii for Base64-encoded ASCII, or binary for Binary DER data.

Possible problems: Any errors that arise when you create the certificate request are likely to occur for one or more of the following reasons:

- Not being the user who runs WebSphere Voice Response (by default, dtuser)
- A certificate with the label wvr_server_certificate already exists in the database
- A certificate request with the label wvr_server_certificate already exists in the database

You can view the certificates and certificate requests in the database by running the following commands:
You can then delete certificates and certificate requests in the database by running the following command (changing the value of label where appropriate):

```
wvrcert -cert -delete -label wvr_server_certificate -db $SYS_DIR/voip/keyring.db -stashed
wvrcert -certreq -delete -label wvr_server_certificate -db $SYS_DIR/voip/keyring.db -stashed
```

**Option 3 - Adding a server certificate from a PKCS #12 file:** You can import a personal certificate into keyring.db from a PKCS #12 file by using the `wvrcert` command. Before you import a personal certificate in PKCS #12 format into the key database file, you must first add the full valid chain of issuing CA certificates to the keyring.db database file. If they are in the PKCS #12 file, you can use the import command. If not, see “Add trusted certificates to keyring.db” on page 78.

**Note:** Treat PKCS #12 files as temporary and deleted them after use.

The command to import is as follows:

```
wvrcert -cert -import -file filename -pw password -type pkcs12
          -target $SYS_DIR/voip/keyring.db -target_pw password -label label
```

where:

- `-file filename`  
  The fully qualified file name of the file that contains the PKCS #12 certificate.

- `-pw password`  
  The password for the PKCS #12 certificate.

- `-type pkcs12`  
  The type of the file.

- `-target filename`  
  The name of the destination CMS key database.

- `-target_pw password`  
  The password for the keyring.db database.

- `-label label`  
  The label of the certificate to import from the source key database.

- `-new_label label`  
  `label` is the label that the certificate is to be assigned in the target database. If you omit the `-new_label` option, the default is to use the same value as the `-label` option. WebSphere Voice Response uses the certificate that is labeled `wvr_server_certificate` as its personal certificate.
Possible problems: Any errors that arise when you import the certificate are likely to occur for one or more of the following reasons:

- Not being the user who runs WebSphere Voice Response (by default, dtuser)
- The label that you specified in the `wrcert -new_label` parameter is already in use by a certificate or a certificate request in the database.

You can view the certificates and certificate requests in the database by running the following commands:

```
wrcert -cert -list -db $SYS_DIR/voip/keyring.db -stashed
wrcert -certreq -list -db $SYS_DIR/voip/keyring.db -stashed
```

You can then delete certificates and certificate requests in the database by running the following command (changing the value of `label` where appropriate):

```
wrcert -cert -delete -label wvr_server_certificate -db $SYS_DIR/voip/keyring.db -stashed
wrcert -certreq -delete -label wvr_server_certificate -db $SYS_DIR/voip/keyring.db -stashed
```

Reviewing current certificates and certificate requests
To list the certificates currently in the database, run the following command:

```
wrcert -cert -list -db $SYS_DIR/voip/keyring.db -stashed
```

Output similar to the following is generated:

```
Certificates found
* default, - personal, ! trusted
  ! Trusted_CA
- wvr_server_certificate
```

Personal certificates with a private key are labeled with a dash, and can be used as server certificates. Trusted certificates are labeled with an exclamation mark.

To review a list of current certificate requests type:

```
wrcert -certreq -list -db $SYS_DIR/voip/keyring.db -stashed
```

Ensure that other SIP entities can communicate with WebSphere Voice Response
After you configure the WebSphere Voice Response server certificate and add the certificates that you want to trust into the keyring.db database, you must ensure other SIP entities trust WebSphere Voice Response. This task involves configuring the trusted certificates of the other software, and adding in certificates from WebSphere Voice Response.

If you have a self-signed certificate as WebSphere Voice Response’s personal certificate you must extract the certificate from keyring.db as explained in “Option 1 - Creating a self-signed certificate” on page 80. If you are using a
certificate that is signed by a Certificate Authority (CA), you must trust all of
the CA certificates in the chain that signed the certificate on WebSphere Voice
Response. You can also extract these certificates from keyring.db after they are
added.

To either extract a CA certificate or the public part of a personal certificate
from keyring.db, run the following command:

```
wvrcert -cert -extract -db $SYS_DIR/voip/keyring.db -stashed -label label
       -target filename -format format
```

where:

- **-label label**
  The label attached to the certificate.

- **-target filename**
  The name of the destination file.

- **-format format**
  The format of the certificate. The value can be the default ascii for
  Base64-encoded ASCII, or binary for Binary DER data.

**Enabling secure SIP**

After you configure your certificates in keyring.db, you can enable secure SIP
in the WebSphere Voice Response System Configuration by completing the
following steps:

1. From the Welcome window, select **Configuration → System Configuration → Change**.
   The System Configuration window opens.
2. Click on the **VoIP SIP Signaling** button.
   The VoIP SIP Signaling window opens.
3. Select **Secure SIP Enabled**, and change the value to **True**.

You then need to restart WebSphere Voice Response for these changes to take
effect.

Any future changes to this setting or to keyring.db will also require restarting
WebSphere Voice Response. Setting the **Secure SIP Enabled** parameter alone
does not allow WebSphere Voice Response to make secure SIP calls. You must
have previously set up the keyring.db certificate database. When enabled,
WebSphere Voice Response can accept secure inbound calls and can make
secure outbound calls.

**Using the wvrcert utility**

WebSphere Voice Response includes a command line utility called **wvrcert**.
This command manages the certificate database keyring.db in the
$SYS_DIR/voip directory. You must create this database yourself. For more information, see "Create the keyring.db database" on page 78. To ensure that the appropriate environment variables are set and files are created with appropriate permissions, you must run this command as the user WebSphere Voice Response is configured to be run by. (The default is dtuser). WebSphere Voice Response, along with various other IBM products, uses GSKit for making TLS connections. \texttt{wvrcert} is a wrapper for the GSKit command \texttt{gsk8capicmd}.

Each command specifies at least one object. Commands for key database, certificate, and certificate request objects also specify an action. The object can be one of the following:

- \texttt{-keydb}\hspace{1cm} Actions apply to a key database
- \texttt{-cert}\hspace{1cm} Actions apply to a certificate
- \texttt{-certreq}\hspace{1cm} Actions apply to a certificate request
- \texttt{-help}\hspace{1cm} Displays help
- \texttt{-version}\hspace{1cm} Displays version information

Before you run any \texttt{wvrcert} commands, see "Create the keyring.db database" on page 78.

\textbf{Certificate commands}

To list the current certificates in the database, run the following command:

\begin{verbatim}
  wvrcert -cert -list -db $SYS_DIR/voip/keyring.db -stashed
\end{verbatim}

To delete a certificate:

\begin{verbatim}
  wvrcert -cert -delete -label label -db $SYS_DIR/voip/keyring.db -stashed
\end{verbatim}

where \texttt{label} is the certificate that you want to delete.

To view the detailed information for a specific certificate:

\begin{verbatim}
  wvrcert -cert -details -db filename -pw password -label label
\end{verbatim}

To rename a certificate's label:

\begin{verbatim}
  wvrcert -cert -rename -label label1 -new_label label2 -db $SYS_DIR/voip/keyring.db -stashed
\end{verbatim}

To extract the public part of a certificate from a file:

\begin{verbatim}
  wvrcert -cert -extract -db $SYS_DIR/voip/keyring.db -pw password -label label
  -target filename -format format
\end{verbatim}
where format is either ascii or binary.

**Certificate request commands**
To list current certificate requests:

```
wvrcert -certreq -list -db $SYS_DIR/voip/keyring.db -stashed
```

To delete a certificate request:

```
wvrcert -certreq -delete -db filename -pw password -label label
```

**Database commands**
To change the database password:

```
wvrcert -keydb -changepw -db $SYS_DIR/voip/keyring.db -pw password -new_pw new_password -stash
```

To delete the database:

```
wvrcert -keydb -delete -db $SYS_DIR/voip/keyring.db -pw password
```

**Secure SIP configuration settings for Register**
The subsections identify the various configuration settings that can be modified to influence secure SIP usage both within the connection between WebSphere Voice Response and the registrar, and within the contacts being registered.

Certificate usage must previously have been set up as described in “Secure SIP minimal configuration” on page 77.

**Registrar connection**
Configuration settings for the connection that is established with a registrar are found in the $SYS_DIR/voip/master.ini file. The parameters that are used to determine the connection with the registrar are as follows:

**TransportProtocol**
This parameter can be set to UDP, TCP, TLS. It is an optional setting and if not specified, UDP is used as the transport protocol.

**Port**
The port the registrar is listening on. It defines the port WebSphere Voice Response uses to send the Register request and can be set to any numeric value. It is an optional setting and if not specified, defaults to 5060 when TransportProtocol is UDP or TCP and 5061 when TransportProtocol is TLS.

**UseSIPS**
This parameter is used to set the scheme for addresses within certain headers in the Register request sent to the registrar and can be set to true or false. This setting is optional, and if not specified, the default is false.

When the TransportProtocol is TLS:
If set to false, the sip: scheme is used for the addresses in the TO, FROM and REQUEST headers in the Register request.

If set to true, the sips: scheme is used for the addresses in the TO, FROM and REQUEST headers in the Register request.

When the TransportProtocol is UDP or TCP, the sip: scheme is used for these headers, regardless of the setting of UseSIPS.

See also "Using a SIP Registrar" on page 32 for full details of the WebSphere Voice Response SIP register service.

Contacts settings
Settings for the contacts to be registered are found in the $SYS_DIR/voip/master.ini file and in the WebSphere Voice Response VoIP Signalling configuration window. The contacts are passed to the registrar as contact headers within the Register message.

The parameters in $SYS_DIR/voip/master.ini that are used to determine the contact header settings within the register message sent to the registrar are as follows:

UseSIPS
This parameter can be set to true or false. This setting is optional, and if not specified, the default is false. If set to false, the sip: scheme is used for the address that is specified in the CONTACT header within the Register message. If set to true, the sips: scheme is used for the address that is specified in the CONTACT header within the Register message.

In the WebSphere Voice Response VoIP Signalling configuration window you can also configure the following parameters:

Call Signalling Secure Port
If UseSIPS is true and Call Signalling Secure Port is not set to 5061 (the default), the port that is specified by Call Signalling Secure Port is included in the contact header.

Call Signalling Port
If UseSIPS is false and Call Signalling Secure Port is not set to 5060 (the default), the Call Signalling Port is included in the contact header.

Configuring cipherspecs.ini
To perform secure SIP communications, two machines negotiate a cipher spec to use. A cipher spec is a combination of algorithms that are used in the underlying TLS connection that secure SIP uses. For more information, see "CipherSpecs and CipherSuites" on page 75.
The file $SYS_DIR/voip/cipherspecs.ini specifies the cipher specs that WebSphere Voice Response allows by default.

If you want to prevent WebSphere Voice Response using any of the default cipher specs, you can do so by prefixing each cipher spec in file $SYS_DIR/voip/cipherspecs.ini that you do not require with a hash character (#). WebSphere Voice Response will not make any TLS connections using those cipher specs.

To reinstate a cipher spec in file $SYS_DIR/voip/cipherspecs.ini, remove the hash character (#) prefix for that file.

You must restart WebSphere Voice Response for any changes to cipherspecs.ini to take effect.

**Problem determination**

Using secure SIP relies on a complicated setup of the underlying X.509 certificates. If you have a problem, complete the following steps:

1. When WebSphere Voice Response is first starting, it tests the security setup that it has. The output of these checks is made in $OAM_LOG_PATH/DTstatus.out. You can find the output of those checks underneath the line:
   
   NM: Secure SIP (TLS) is Enabled

   If Secure SIP is not enabled in the WebSphere Voice Response configuration, the following line is generated:

   NM: Secure SIP (TLS) is not Enabled

   Check the setup messages under those lines for errors and logs.

2. Confirm the presence of the database files keyring.db, keyring.rdb, keyring.crl, and keyring.sth in the $SYS_DIR/voip directory. If these files are not present, see "Create the keyring.db database" on page 78. Also, ensure that these files are owned by the user that WebSphere Voice Response is configured to be run by (default dtuser) and that this user has read and write permissions for the database files.

3. Run the following command:

   wvrcert -cert -list -db $SYS_DIR/voip/keyring.db -stashed

   Output similar to the following is generated:

   Certificates found
   * default, - personal, ! trusted
   ! Trusted_CA
   - wvr_server_certificate

   Ensure that you have a wvr_server_certificate listed with a dash next to it. If you cannot see the label, or it does not have a dash next to it, you have not configured the WebSphere Voice Response server certificate.
4. You must ensure that for every certificate in your database, you have all the certificates in the CA chain that signed that entity’s certificate. You must also check that your certificates have not expired. You can check this by running the following command, replacing `my_label` with your certificate label:

```
wvrcert -cert-validate -label "my_label" -db $SYS_DIR/voip/keyring.db -stashed
```

If all the necessary CA certificates are in the database and the certificate has not expired, an OK message is generated.

5. The labels that you add to your certificates are for your own convenience. It is possible to add a misleading label to a certificate accidentally, for example, to label a non-self-signed certificate "self_signed". You can check the details of your certificate with the following command, replacing `my_label` with your certificate label:

```
wvrcert -cert-details -label "my_label" -db $SYS_DIR/voip/keyring.db -stashed
```

Output similar to the following is generated:

```
Label : wvr_server_certificate
Key Size : 1024
Version : X509 V3
Serial : 382ec01d0cfa4e9a
Issuer : CN=MY_CA
Subject : CN=example-machine.ibm.com
Not Before : 1 November 2013 09:57:16 GMT-87:39:25
Not After : 21 October 2023 09:57:16 GMT-87:39:25
Public Key
30 ...... 01
Public Key Type : RSA (1.2.840.113549.1.1.1)
Fingerprint : SHA1 :
08 ...... 00
Signature Algorithm : SHA1WithRSASignature (1.2.840.113549.1.1.5)
Value
57 ...... 19
Trust Status : Enabled
```

6. If you have recently made any configuration changes, restart WebSphere Voice Response. Secure configuration changes will come into effect only on restart.

7. Restore the original `ciphers.ini`. If you modified `ciphers.ini` in the `$SYS_DIR/voip` directory, try using the original file that is stored as `$SYS_DIR/voip/ciphers.ini.orig`. Mistakes in `ciphers.ini` can lead to errors.

8. Errors can also be displayed in the WebSphere Voice Response errorlog in `$OAM_LOG_PATH`.
Secure SIP configuration settings for incoming calls

The various configuration settings that can be modified to influence secure SIP usage within incoming calls are described.

Certificate usage must have previously been set up as described in “Secure SIP minimal configuration” on page 77.

The configuration parameters that influence secure SIP usage within incoming calls are as follows:

Enable Secure SIP

When set to Yes, WebSphere Voice Response attempts to accept incoming calls that are made using TLS. When set to No, WebSphere Voice Response ignores incoming calls that are made using TLS. The default value is No.

Call Signalling Secure Port

Specifies the port that WebSphere Voice Response listens on for incoming calls made using TLS. The default value is 5061.

Incoming calls using TCP or UDP will always be accepted, regardless of these settings.

Secure SIP call transfer considerations

The following two types of call transfer are possible when secure SIP is used:

- Blind transfer
- Attended transfer

Blind transfer

WebSphere Voice Response initiates a transfer by sending a REFER message to the calling device, including the contact details for the party to be transferred to. It is then up to the calling device to establish the connection by using UDP, TCP, or TLS.

A WebSphere Voice Response application can specify a sip: or sips: address and this will be passed to the calling device. However, WebSphere Voice Response has no control over the transport protocol that the calling device uses to establish a connection to the party that the call is being transferred to. Also, WebSphere Voice Response has no information on the ability of the calling device to support secure SIP.

Attended transfer

In this case WebSphere Voice Response establishes a new call to the party to be transferred to. This call is set up as a standard outbound call with the same
secure SIP considerations as any other outbound call. When this second call is
established, WebSphere Voice Response passes details of it to the calling party
by using a REFER Replaces mechanism. The calling party then takes this call
from WebSphere Voice Response and drops the original call to WebSphere
Voice Response.

**Note:** WebSphere Voice Response cannot ensure that there is no secure SIP
mismatch between the original calling party and the party to which the call is
transferred.

**Secure SIP configuration settings for outbound calls**
The various configuration settings that can be modified to influence secure
SIP usage within outbound calls are described.

Certificate usage must have previously been set up as described in “Secure
SIP minimal configuration” on page 77.

The following configuration parameter influences secure SIP usage within
outbound calls:

**Transport Protocol**
Defines the default protocol that is used to make outbound calls when
the sip: scheme is used. When the sips: scheme is used, a TLS
(end-to-end) connection is used regardless of the setting of **Transport
Protocol**, which has a default value of **UDP**. For more information,
see “Transport Protocol” on page 168.

The use of the sip: scheme with **Transport Protocol** set to **TLS** guarantees the
use of TLS only for the first hop within the total end-to-end connection. If
end-to-end secure SIP is required, the sips: scheme must be used.

**Secure RTP**
Real-time Transport Protocol (RTP) is a protocol that is used with Voice over
IP to send audio data. Secure RTP provides confidentiality and message
authentication to RTP data. It can be used to prevent people from listening to
or tampering with the audio data sent over an unsecure network like the
internet. Secure RTP uses more computational resources than an unsecure,
uncompressed RTP stream due to the cryptographic operations involved.

To enable secure RTP, open the VoIP DTEA and DTNA Media Configuration
Settings window and change the **RTP Security Negotiation** setting. There are
three different options:

**Unsecured**
The default. WebSphere Voice Response does not accept secure RTP
for inbound or offer secure RTP for outbound calls. Inbound calls that offer secure RTP only, are rejected with a 488 Not Accepted Here response.

**Secure**

WebSphere Voice Response uses only secure RTP. Inbound calls not capable of secure RTP are rejected with a 488 Not Accepted Here response, and outbound calls that are made by WebSphere Voice Response offer secure RTP only.

**Both**

For inbound calls WebSphere Voice Response accepts secure RTP if offered, but also accepts calls if only RTP is offered. If both secure RTP and RTP are offered, secure RTP is used. For outbound calls, both secure RTP and RTP are offered.

When Secure RTP has been configured for either Secure or Both, partial support is provided for optional crypto session parameters (RFC 4568 section 6.3). See “RTP Security Negotiation” on page 146 for more information.

It is important to note that secure RTP does not offer any protection to the signaling data of a Voice over IP communication. The signaling data is handled by the SIP protocol, and it too can be made secure. See “Secure SIP” on page 77 for details. Secure RTP is independent of secure SIP and does not need secure SIP enabled to function. Using secure RTP on its own is not recommended because the negotiation of the cipher keys in the signaling messages is handled in the SIP messages. An attacker who reads these SIP messages could then decode and listen to the RTP stream. For complete security, secure SIP must also be configured and enabled.
Voice over IP using Session Initiation Protocol
Chapter 6. Programming SIP with DTNA

This chapter contains information on:

- “Programming SIP with VoiceXML”
- “Voice over IP tags” on page 104
- “SIP tags” on page 104
- “Transfer tags” on page 112
- “Call transfer” on page 114
- “SIP support of Message Waiting Indicator (MWI)” on page 128

Programming SIP with VoiceXML

In addition to using the VoiceXML <transfer> element to transfer calls using SIP (refer to the VoiceXML Programmer’s Guide book for details), it is possible to use a combination of base VoiceXML telephony primitives and calls to WebSphere Voice Response 'native' state tables to extract SIP specific information from the WebSphere Voice Response platform.

Although the ability to call State Tables from a VoiceXML application is extremely flexible and can be used to provide a link between existing WebSphere Voice Response applications (such as State Tables and Custom Servers) and newer VoiceXML applications, the most basic use of this mechanism is simply to extract signalling information from the base platform and pass it into the VoiceXML environment for incoming calls, and also to perform the reverse operation for outgoing calls (to provide signalling information to WebSphere Voice Response prior to a call being made).

SIP and Call Transfer tags

This section describes the use of the WebSphere Voice Response System Variables SV541, SV542, and SV543 which are used to pass additional signalling information between an application and the SIP stack and the SIP tags that are supported.

As described in “Voice over IP tags” on page 104, there are three WebSphere Voice Response System Variables that are used to pass additional signalling information between an application and the SIP stack. These are:

- SV541 - for supported tags that are sent to the SIP network (on outbound calls and transfer requests)
- SV542 - for supported tags that are received from the SIP network (on inbound calls)
- SV543 - for reviewing of far end hang up reasons
To ensure that these three System Variables contain the correct information, the **Call Information Type** parameter must be set to **Signalling Process** in the System Configuration GUI. Refer to the section “Call Information Type” in the *WebSphere Voice Response for AIX: Configuring the System* for more information.

For an inbound call the following tags are supported:

- TO_HDR, FROM_HDR, REQ_HDR, CALLINFO_HDR, CALLID_HDR, DIV_HDR, A_LANG_HDR, PRIVACY_HDR, SUBJECT_HDR

  Used in SV542 on an inbound INVITE.

- The first ten other (non-standard) headers in the configuration file `/usr/lpp/dirTalk/db/sys_dir/voip/siphdrtags.cfg` can be used in SV542 on an inbound INVITE.

For an outbound call the following tags are supported:

- TO_HDR, FROM_HDR, REQ_HDR, DIV_HDR, A_LANG_HDR, PRIVACY_HDR, PASSID_HDR, RPARTYID_HDR

  Used in SV541 on outbound INVITE (MakeCall action).

- The first ten other (non-standard) headers in the configuration file `/usr/lpp/dirTalk/db/sys_dir/voip/siphdrtags.cfg` can be used in SV541 on an outbound INVITE.

For a near-end hangup (TerminateCall action), the following tag used in SV541 is supported:

- SUBJECT_HDR

On a blind (REFER) transfer the following tags are supported:

- TO_HDR, FROM_HDR

On an attended (REFER/Replaces) transfer the same tags as outbound call are supported:

- TO_HDR, FROM_HDR, REQ_HDR, DIV_HDR, A_LANG_HDR, PRIVACY_HDR, PASSID_HDR, RPARTYID_HDR

In addition to the above, there are three tags that are used to control the internal operation of SIP transfer operations:

- **CONSULT**:
  - Used in SV541 prior to a TransferCall action being issued to set up the consultation leg of an attended transfer.

- **TRANSFER**:
  - Used in SV541 prior to a TransferCall being issued to actually perform the SIP transfer operation (REFER/REPLACES) for an attended transfer.

- **XFER TIMER**:
Used in SV541 prior to an outbound REFER (TransferCall action) for blind transfer.

See “SIP tags” on page 104 and “Transfer tags” on page 112 for details of how to implement Call Transfer under SIP.

Accessing SIP and Call Transfer tags from your voice application

This section describes how to access SIP tags (and how to control call transfer operations) by using the SV54x System Variables, and how to set and get these System Variables from State Table SIP applications and other types of SIP application.

Using state tables, the only way to access SIP tags (and to control call transfer operations) is by using the SV54x System Variables, and the only way to set and get these System Variables from State Tables is using the SetTag and GetTag options of the AssignData state table actions.

If your SIP application is coded in State Tables, the SV54x System Variables can be accessed directly from your application.

If your SIP application is coded in the Voice Response Beans Environment (VRBE), the SV54x system variables must be accessed using an intermediate State Table called using a VRBE Compatibility Bean.

If your SIP application is coded in VoiceXML, the SV542 system variable can be accessed using protocol-independent connection-related variables. For more information, refer to the section “Using WebSphere Voice Response call tags” in the WebSphere Voice Response for AIX: VoiceXML Programmer’s Guide. The SV541 and SV542 system variables must be accessed using an intermediate State Table called using the VoiceXML Object Tag.

The SV541 and SV542 system variables can be accessed using protocol-independent connection-related variables in CCXML applications. For more information, refer to the section “Using WebSphere Voice Response call tags” in the WebSphere Voice Response for AIX: Using the CCXML Browser.

Using an intermediate State Table to access tags

Although it is possible to create a single state table that sets a number of tags at the same time or to use a single state table to decode a parameter and to set one of a number of SIP tags accordingly, the simplest approach is to have a number of separate state tables to do individual things like “Set To Header”, “Set From Header” and so on.
The following is the ASCII source for a simple state table that sets the To_Header prior to doing a Call Transfer operation. Note that this state table has one entry point (Start) and is called with a single parameter (to_header):

```
DESCRIPTION ("Sets SIP To Header for transfer");
ENTRY_POINT (Start,Start);
INPUT STRING to_header;
Start: "Set SV541 to sip to header"
AssignData (SV541, "PUT_TAG", "TO_HDR", to_header);
ExitStateTable (0);
```

### Calling a State Table from VoiceXML

The VoiceXML Object Tag is used as shown in the following example to call a State Table. This calls the State Table SetToHdr at entry point Start with a single parameter sip:44409.20.114.23 (which is passed into the State Table as the to_header Input String).

```xml
<object name="sipTransfer" codetype="javacode-ext"
classid="method://com.ibm.wvr.vxml2.NativeAppSupport/invokeStateTable">
  <param name="setName" value="SetToHdr"/>
  <param name="setEntryPoint" value="Start"/>
  <param name="setEncoding" value="UTF-8"/>
  <param name="setParms" expr="new Array('sip:44409.20.114.23')"/>
</object>
```

For additional details on the use of the object tag to call a WebSphere Voice Response State Table, refer to the section “Invoking a State Table using Voice XML” in the WebSphere Voice Response: VoiceXML Programmer’s Guide.

### Implementing SIP Call Transfer operations

As described in “Call transfer” on page 114, two types of SIP call transfer are implemented by WebSphere Voice Response SIP support:

- Blind transfer
- Attended transfer

In the blind transfer mode using State Tables, WebSphere Voice Response initiates an immediate transfer using the SIP 'REFER' method, which will eventually result in the WebSphere Voice Response application’s calling party being connected to a third party and WebSphere Voice Response being disconnected. There is a single phase to this operation. A Blind Transfer is initiated by a State Table TransferCall action with SV541 containing the TO_HDR URI for the transfer target (No SIP CONSULT or SIP TRANSFER tags are set). The Ring Wait and Ring Time parameters must also be set to zero. XFER_TIMER can be set in SV541 to define a timeout value for the transfer. See “VoIP SIP blind transfer” on page 114, “How to write a SIP blind transfer application” on page 115, and “Transfer tags” on page 112 for details.
In the attended transfer mode using State Tables, WebSphere Voice Response initially starts a second application, which talks independently to the third party by using a SIP outbound call. This operation is what is known as the ‘Consultation Phase’. Assuming that it is still required to proceed with the transfer after the consultation, the actual call transfer phase then proceeds using the SIP ‘REFER/REPLACES’ sequence. As with the Blind Transfer, the result is that the WebSphere Voice Response application’s calling party is connected to the third party and WebSphere Voice Response is disconnected. See “VoIP SIP attended transfer” on page 114, “How to write a SIP attended transfer application” on page 115, and “Transfer tags” on page 112 for details.

**Blind transfer using VoiceXML**

For a blind transfer to work:

1. The calling device must allow the SIP REFER method indicated by the Invite message Allow header including “REFER”.

2. An on-hold method that is supported by the calling device must be used by WebSphere Voice Response. The on-hold method is set in the WebSphere Voice Response VOIP SIP Signalling window by the [RFC3264 media-on-hold method](http://www.ietf.org/rfc/rfc3264.txt) parameter.

To open the VOIP SIP Signalling window:

a. From the Welcome window, select **Configuration → System Configuration → Change**.
   The System Configuration window opens.

b. Click on the **VoIP SIP Signaling** button.
   The VoIP SIP Signalling window opens.

   See “RFC3264 Media on-hold method” on page 162 for details of the available on-hold methods.

The following example uses the VoiceXML `<transfer>` element to transfer a non-secure SIP call:
When the VoiceXML <transfer> element is executed, WebSphere Voice Response calls the DTJConsult state table followed by the DTJTransfer state table. The DTJConsult state table does a simple blind call transfer, and the DTJTransfer state table simply hangs up to complete the transfer. If necessary, the DTJConsult state table can be easily modified to change the transfer timeout value.

Using the <object> element:

There is an alternative way of implementing blind transfer from VoiceXML that does not use the VoiceXML <transfer> element, but instead uses the <object> element to call a state table that performs the SIP blind transfer. For more information, see "Accessing SIP and Call Transfer tags from your voice application" on page 97.

Attended transfer

An attended transfer consists of two steps, Consultation and Actual Transfer:
1. In the Consultation step, a second WebSphere Voice Response application is started up which then does a MakeCall such that the new application is talking to the intended transfer target.
2. In the Transfer step, the SIP REFER/REPLACES operation is used to connect second and third parties together and allow WebSphere Voice Response to disconnect from the call.
3. The Consultation Phase is triggered by a TransferCall State Table action with the CONSULT tag set to 0 in SV541, which must also contain the TO_HDR tag set to the URI of the transfer target.

4. The VOIP_Call_Transfer custom server is used to start the second application, which initiates an outbound call to the transfer target.

5. The application to be used on the Consultation phase must at least initially be a WebSphere Voice Response State Table (although a VoiceXML application can then be called). The name of the State Table to be initially called is defined in main args parameters of the VOIP_Call_Transfer state table (the default is SuperXferA, which is a State Table that is supplied in the VOIP_Call_Transfer application package).

6. The return code (edge) from the Consultation phase TransferCall action indicates whether it is required to proceed with the Transfer Phase or not.

7. The final Transfer Phase is triggered by a TransferCall State Table action with the TRANSFER tag set to zero. This operation causes the actual transfer to take place using the SIP ‘REFER/REPLACES’ method. When complete, the initial application is disconnected from the call.

See "Call transfer" on page 114 for more details of how to program SIP Call Transfer.

Attended transfer using VoiceXML

For an attended (consultation) transfer to work:

1. The calling device must allow the SIP REFER method indicated by the Invite message Allow header including “REFER”.

2. An on-hold method that is supported by the calling device must be used by WebSphere Voice Response. The on-hold method is set in the WebSphere Voice Response VOIP SIP Signalling window by the RFC3264 media-on-hold method parameter.

To open the VOIP SIP Signalling window:

a. From the Welcome window, select Configuration > System Configuration > Change.

   The System Configuration window opens.

b. Click on the VoIP SIP Signaling button.

   The VoIP SIP Signalling window opens.

See "RFC3264 Media on-hold method" on page 162 for details of the available on-hold methods.

3. The calling device must allow the SIP Replaces header indicated by the Invite message Supported header, including option tag "replaces". This last requirement can be overridden by setting the WebSphere Voice Response configuration option Ignore replaces option for Attended Transfer to Yes.
4. The VOIP_Call_Transfer custom server is installed and started. By default, the custom server calls the SuperXferA state table initially. This state table is supplied in the VOIP_Call_Transfer application package, and does all that is needed to complete the transfer. If required, the SuperXferA state table can be replaced by a customized state table, the name of which is specified in the main args parameters of the VOIP_Call_Transfer custom server. See "Attended transfer using the VOIP_Call_Transfer custom server" on page 116 for details of how to install and configure the VOIP_Call_Transfer custom server.

The following example uses the VoiceXML <transfer> element to transfer a non-secure SIP call:

```xml
<form id="Transfer1">
  <transfer name="Attended1" dest="sip:12345@uptonsip.hursley.ibm.com" type="consultation"
    connecttimeout ="33s">
    <filled>
      <log>
Sample Attended Transfer form filled. Attended1 (form variable) = <value expr="Attended1"/>
      </log>
      <if cond="Attended1 == 'busy'">
        <log>Sample Attended Transfer filled = busy.</log>
      </if>
      <elseif cond="Attended1 == 'noanswer'">
        <log>Sample Attended Transfer filled = no answer.</log>
      </elseif>
      <elseif cond="Attended1 == 'network_busy'">
        <log>Sample Attended Transfer filled = network busy.</log>
      </elseif>
      <elseif cond="Attended1 == 'unknown'">
        <log>Sample Attended Transfer filled = unknown.</log>
      </elseif>
    </filled>
  </transfer>
  <block>
    <log>Sample Attended Transfer goto exit_app</log>
    <goto next="#exit_app"/>
  </block>
</form>
```

in the VOIP SIP Signalling window. For more information, see "Ignore replaces option for Attended Transfer" on page 154.
Using the `<object>` element:

There is an alternative way of implementing attended transfer from VoiceXML that does not use the VoiceXML `<transfer>` element, but instead uses the `<object>` element to call a state table that performs the SIP attended transfer. For more information, see "Accessing SIP and Call Transfer tags from your voice application" on page 97.

Bridge Transfer

Bridge Transfer is not supported by WebSphere Voice Response.

Tromboning using VoIP/SIP

The standard IBM Trombone Custom Server can easily be used for SIP after adding one line to the Trombone Custom Server State Table. Refer to the WebSphere Voice Response for AIX: Application Development using State Tables for information on editing State Tables.

To modify the Trombone Custom Server State Table:
1. Open the Trombone Custom Server State Table in an editor.
2. Find line 78 (MakeCall).
3. Insert a new line AssignData, Put Tag, To_HDR, IPaddress with the label "outdial", where IPaddress is the address of the soft phone that is ‘tromboned to’, for example, SIP:fred@9.70.99.54.

   Note: The telephone number specified for the soft phone that is ‘tromboned to’ will be ignored.
Voice over IP tags

This section contains information on SIP tags and transfer tags.

This section contains information on:
• “SIP tags.”
• “Transfer tags” on page 112.

SIP tags

Voice applications can use SIP tags to send and receive information including Universal Resource Indicators (URIs) to and from the SIP stack. Information on how to use each SIP tag supported by WebSphere Voice Response is provided here.

The tags take the form tag name: value. For example:

TO_HDR= sip: user@uk.ibm.com
FROM_HDR= sip: wvr@uk.ibm.com

The value of the tag is always defined as a string and must follow the SIP standard for the header type that is used.

The tags can be used with MakeCall, AnswerCall, TransferCall, and TerminateCall state table actions. For more information on state table actions and system variables, see WebSphere Voice Response for AIX: Application Development using State Tables.

The system variables that are used are:
• SV541 - for supported tags that are sent to the SIP network.
• SV542 - for supported tags that are received from the SIP network.
• SV543 - for reviewing of far-end hang up reasons.

To ensure that these three System Variables contain the correct information, the Call Information Type parameter must be set to Signalling Process in the System Configuration GUI. Refer to the section “Call Information Type” in the WebSphere Voice Response for AIX: Configuring the System for more information.

SIP tag information contained in SV542 is also accessible from VoiceXML applications. For more information, refer to the section “Using WebSphere Voice Response call tags” in the WebSphere Voice Response for AIX: VoiceXML Programmer’s Guide.

SIP tag information contained in SV541 and SV542 is also accessible from CCXML applications. For more information, refer to the section “Using WebSphere Voice Response call tags” in the WebSphere Voice Response for AIX: Using the CCXML Browser.
The following sections describe each SIP tag supported by WebSphere Voice
Response and how to use it with the product:

TO_HDR

A SIP 'To Header' tag. This is the address to which the request is
sent. This corresponds to the called number in a traditional telephony
system. The standard character format for a SIP URI is UTF-8.

Size
The maximum length of the tag is 256 characters.

Valid attributes
None.

Valid usage
• In a SIP INVITE message on an inbound call in SV542.
• In an INVITE message on an outbound MakeCall action in
SV541.
• In a REFER message on a TransferCall action in SV541 for a
blind transfer.
• In an INVITE message on a TransferCall action in SV541 for an
attended transfer.

Examples
When using a proxy:
sip:john@mydomain.com

When not using a proxy:
sip:john@mycomputer.mydomain.com:5060 (the port is optional, the
default is 5060).

FROM_HDR

A SIP 'From Header' tag. This is the address from which the request is
sent. This relates to the calling number in a traditional telephony
system. The standard character format for a SIP URI is UTF-8.

Size
The maximum length of the tag is 256 characters.

Valid attributes
None.

Valid usage
• In a SIP INVITE message on an inbound call in SV542.
• In an INVITE message on an outbound MakeCall action in
SV541.
• In a REFER message on a TransferCall action in SV541 for a blind transfer.
• In an INVITE message on a TransferCall action in SV541 for an attended transfer.

**Example**

"harry" <sip:tom@mycomputer.mydomain.com>;

**PASSID_HDR**

A SIP 'P-Asserted-Identity' tag. Added by a SIP proxy when the SIP calling ID (in the From header) has been authenticated. For more information on the usage of 'P-Asserted-Identity', refer to SIP RFC3325.

**Size**
The maximum length of the tag is 256 characters.

**Valid attributes**
None

**Valid usage**
• In an INVITE message on an outbound MakeCall action in SV541.
• In an INVITE message on a TransferCall action in SV541 for an attended transfer.

**Example**

P-Asserted-Identity: "fred"<sip:123456@anyhost.com>

In this example, 123456 would be extracted as the calling number passed in SV186, and everything following P-Asserted-Identity: would be passed in tagged string SV542.

**PRIVACY_HDR**

A SIP 'Privacy' tag. Indicates whether presentation of the calling id (in the P-Asserted-Identity) header is to be suppressed or allowed.

**Size**
The maximum length of the tag is 256 characters.

**Valid attributes**
None.

**Valid usage**
• In a SIP INVITE message on an inbound call in SV542.
• In an INVITE message on an outbound MakeCall action in SV541.
In an INVITE message on a TransferCall action in SV541 for an attended transfer.

**Example**

Privacy: none

**REQ_HDR**

A SIP 'Request Header' tag. This is an actual address and defines the 'first hop' for an outbound request. This may be a proxy server or gateway which will then forward the request to the address defined in the TO_HDR. If a REQ_HDR tag is present it will override any routing information that has been configured on the system. The standard character format for a SIP URI is UTF-8.

**Size**

The maximum length of the tag is 256 characters.

**Valid attributes**

None.

**Valid usage**

- In a SIP INVITE message on an inbound call in SV542.
- In an INVITE message on an outbound MakeCall action in SV541.

**Example**

sip:john@99.23.45.7

**RPARTYID_HDR**

A SIP 'Remote-Party-ID' tag. Support provided for compatibility with existing gateways.

To determine Calling number (passed in SV186), WebSphere Voice Response looks for headers in the following order of priority:

1. P-Asserted-Identity
2. Remote-Party-ID
3. From

**Note:** Remote-Party-ID is included for compatibility with older SIP equipment and is now superseded by 'P-Asserted-Identity' and Privacy. SIP RFCs 3323 and 3325 refer.

**Size**

The maximum length of the tag is 256 characters.

**Valid attributes**

None.

**Valid usage**
• In an INVITE message on an outbound MakeCall action in SV541.
• In an INVITE message on a TransferCall action in SV541 for an attended transfer.

Example

Remote-Party-ID: "Bob"<sip:123456@anyhost.com>;party=calling;privacy=none

SUBJECT_HDR

A SIP 'Subject Header'. This is used to send information on the TerminateCall state table action. This may be information required to complete the call (to connect the user to a directory lookup number), and can be used as an alternative to call transfer (REFER). The standard character format for a SIP URI is UTF-8.

Size
The maximum length of the tag is 256 characters.

Valid attributes
None.

Valid usage
• In a SIP INVITE message on an inbound call in SV542.
• In a BYE message on a TerminateCall action in SV541.

Example

"1,5551234,555678"

DIV_HDR

A SIP 'Diversion Header'. The diversion header tag can contain a URI and certain attributes that describe the parameters that may be received with it (for example, the reason for the diversion). The standard character format for a SIP URI is UTF-8. A SIP message can contain multiple diversion headers. WebSphere Voice Response extracts the first of these (top) into the DIV_HDR string, but also uses the first to extract the Original Called Number into SV187 and the last (bottom) to extract the Last Redirecting Number into SV188.

Size
The maximum length of the tag is 256 characters.

Valid usage
• In a SIP INVITE message on an inbound call in SV542.
• In an INVITE message on an outbound MakeCall action in SV541.
• In an INVITE message on a TransferCall action in SV541 for an attended transfer.
Example
sip:john@hursley.ibm.com:5060;reason='do-not-disturb'

The reason code is added as an attribute.

DIVERT attributes

DIV_HDR.reason
A SIP 'Diversion Header' tag attribute.

Size
The maximum length of the tag is 20 characters.

Examples of the diversion reason attribute
"unknown"
"user-busy"
"no-answer"
"unavailable"
"unconditional"
"time-of-day"
"do-not-disturb"
"deflection"
"follow-me"
"out-of-source"
"away"

DIV_HDR.counter
A SIP 'Diversion Header' tag attribute. The standard character format for a SIP URI is UTF-8.

Size
The maximum length of the tag is 2 digit characters.

Example of diversion counter attribute
"15"

DIV_HDR.limit
A SIP 'Diversion Header' tag attribute. The standard character format for a SIP URI is UTF-8.

Size
The maximum length of the tag is 2 digit characters.

Example of diversion limit attribute
"20"

**DIV_HDR.privacy**

A SIP 'Diversion Header' tag attribute. The standard character format for a SIP URI is UTF-8.

**Size**

The maximum length of the tag is 20 characters.

**Examples of diversion privacy attribute**

"privacy=full"

"privacy=name"

"privacy=screen"

**DIV_HDR.screen**

A SIP 'Diversion Header' tag attribute. The standard character format for a SIP URI is UTF-8.

**Size**

The maximum length of the tag is 20 characters.

**Examples of diversion screen attribute**

"screen=yes"

"screen=no"

**DIV_HDR.ext**

A SIP 'Diversion Header' tag attribute. The standard character format for a SIP URI is UTF-8.

**Size**

The maximum length of the tag is 20 characters.

**Example of diversion extension attribute**

"quoted_string"

**A_LANG_HDR**

A SIP 'Accept Language Header' tag. This is used in requests to indicate the preferred language or languages to be used for reason phrases, session descriptions, or status responses included in the response. If not present, all languages should be acceptable.

**Size**

The maximum length of the tag is 256 characters.

**Valid attributes**

None.
Valid usage

- In a SIP INVITE message on an inbound call in SV542.
- In an INVITE message on an outbound MakeCall action in SV541.
- In an INVITE message on a TransferCall action in SV541 for an attended transfer.

Examples

da
en-gb;q=0.8
en;q=0.7

CALLID_HDR

A SIP 'Call-ID' tag. This is a unique identifier used to group together a series of messages.

Size
The maximum length of the tag is 256 characters.

Valid attributes
None.

Valid usage
- In a SIP INVITE message on an inbound call in SV542.

Example
f81d4fae-7dec-11d0-a765-00a0c91e6bf6@myco.com

CALLINFO_HDR

A SIP 'Call-Info' tag. This provides additional information about the caller.

Size
The maximum length of the tag is 256 characters.

Valid attributes
- purpose - the purpose of the URI.
- card - a business card, for example, in vCard or LDIF format
- icon - designates a digital image of the caller.

Valid usage
- In a SIP INVITE message on an inbound call in SV542.

Examples

<http://www.example.com/alice/photo.jpg>;purpose=icon
<http://www.example.com/alice/>;purpose=info
Transfer tags

This section gives information about the tags that can be used in call transfer applications.

CONSULT

A SIP 'CONSULT' tag. This is used to make the outbound call of an attended call transfer.

Size

The maximum length of the tag is 256 characters.

Valid attributes

None.

Valid usage

- The 'CONSULT' tag must be set to 0 in SV541 prior to issuing the TransferCall state table action to request that an outbound call be established. The VOIP_Call_Transfer custom server must be installed and running when using a 'CONSULT' tag.

Example

CONSULT=0

TRANSFER

A SIP 'TRANSFER' tag. This invokes the REFER/REPLACES call transfer action.

Size

The maximum length of the tag is 256 characters. The tag value should be set to '0'.

Valid attributes

None.

Valid usage

- To invoke a SIP call transfer using a TransferCall action, this must be set in SV541. This tag should be used either after the VOIP_Call_Transfer custom server has initiated an outbound call using CONSULT (and the consult phase is complete), or with any existing call as long as its call reference (SV237) is also assigned to SV541 in the CAL_REF tag.

Examples

TRANSFER=0

or

TRANSFER=0
CALL_REF=1234
CALL_REF
A SIP 'CALL_REF' tag. This is used with the TRANSFER tag to identify another existing call for which a SIP REFER/REPLACES transfer action is required. The VOIP_Call_Transfer custom server is not needed to establish the outbound call if the CALL_REF tag is used.

For example, '1234'.

Size
The maximum length of the tag is 12 digit characters.

Valid attributes
None.

Valid usage
- To invoke a SIP call transfer on a TransferCall action in SV541. This tag should be assigned to the call reference value (SV237) of the call to which the transfer is required. The TRANSFER tag must also be assigned (see "TRANSFER" on page 112 for more information).

Example
234568

PROTOCOL
A SIP 'PROTOCOL' tag contains the protocol type.

Size
The maximum length of the tag is 256 characters.

Valid attributes
PROTOCOL.VARIANT describes the protocol variant.

Valid usage
- In a SIP INVITE message on an inbound call in SV542.
- In a SIP BYE message on a far-end hang up in SV543.

Example
PROTOCOL=SIP
PROTOCOL.VARIANT=RFC3261

XFER_TIMER
A SIP 'XFER_TIMER' tag can be used to set the timeout for a blind transfer.

Size
The maximum length of the tag is 256 characters.

Valid attributes
None

Valid usage
- Set in SV541 prior to invoking a blind transfer.
• XFER_TIMER can be set to any value in the range 0 to 35. If it is set to a value greater than 35, a value of 35 will be used.
• Set XFER_TIMER to 0 for true blind transfer to occur.

Example
XFER_TIMER=0

Call transfer
This section describes how to transfer calls on WebSphere Voice Response using Voice over IP. WebSphere Voice Response supports both blind and attended transfer. The blind transfer is based on the SIP REFER method and the attended transfer uses the SIP REFER/REPLACES method.

This section contains the following sections:
• “VoIP SIP blind transfer.”
• “VoIP SIP attended transfer.”
• “How to write a SIP blind transfer application” on page 115.
• “How to write a SIP attended transfer application” on page 115.
• “Custom server functions” on page 122.
• “State table definitions” on page 123.

VoIP SIP blind transfer
Blind transfer is when a call is routed to a third party, the original call is ended, and no check is made to determine whether the transferred call is answered or if the number is busy.

To do this, the WebSphere Voice Response application places the call on hold and sends a REFER message containing the TO header address of the third party to the original caller. This call then disconnects from WebSphere Voice Response and connects to the third party. If the transfer is unsuccessful, the original call is lost. Blind transfer does not require WebSphere Voice Response to make a call to the third party.

Blind call transfer is implemented using the SIP REFER method based on RFC 3515. A Blind transfer can be used if the phones connected in the network support the REFER method. Most phones publicize their capabilities in the ALLOW header which is commonly present in a SIP INVITE message. Some phones use the OPTIONS message to indicate these capabilities.

VoIP SIP attended transfer
Attended transfer is when a call is routed to a third party only if the third party answers the call.
To do this, WebSphere Voice Response makes a call to the third party. When the application is ready to perform the transfer, a REFER message containing a TO header and a REPLACES header identifying the third party, is sent to the original caller. The original caller then sends an INVITE message containing the REPLACES information to the third party. If the transfer is accepted, the original caller notifies WebSphere Voice Response and both calls will disconnect from WebSphere Voice Response.

For attended transfer, phones must support both the REFER method and the REPLACES header.

**How to write a SIP blind transfer application**

The WebSphere Voice Response signaling model assumes that a blind transfer operation uses the same channel as the incoming call when it makes the transfer to a third party.

WebSphere Voice Response implements VoIP SIP blind call transfer in the following way:

1. A call arrives at WebSphere Voice Response and the application recognizes that the caller needs to be transferred to a third party.
2. The application uses the TransferCall state table action to initiate the transfer.
3. In preparation for the transfer SV541 must have a TO_HDR tag assigned to the URL address of the third party; for example:
   
   ```
   AssignData(SV541, "PUT_TAG", "TO_HDR","sip:chris@99.22.30.96:5060")
   ```

   XFER_TIMER can also be added to SV541 if required to set a timeout value for the transfer attempt. Set this to 0 if true blind transfer is required.

4. The TransferCall state table action must be placed after AssignData in the state table (this must be at the point that the transfer is required to be invoked). For a blind transfer the ring time and ring time parameters must be 0, for example, TransferCall("", "", 0, 0, 0).

   The TransferCall state table action can be used in its traditional form (by using numbers without setting SV541), however this would require the system to be configured to use a SIP proxy server.

5. The call is ended.

**How to write a SIP attended transfer application**

There are two ways of invoking an attended transfer using SIP:

- Using the VOIP_Call_Transfer custom server.
- Using call reference.
Attended transfer using the VOIP_Call_Transfer custom server
The VoIP call transfer application is supplied in the VoIP installp image that you install using the System Management Interface Tool (SMIT). The application components, include some source code to help you customize the VoIP call transfer capability to your specific needs. For a straightforward use of the VoIP call transfer capability, no customizing should be required.

The components of the application are:
- State tables (including source code): The following state tables are provided (for definitions for these state tables, see “State table definitions” on page 123):
  - VOIP_SupA_Xfer
  - VOIP_Xfer_C5
  - VOIP_Xfer_C10
  - VOIP_Xfer_Stat
  - VOIP_Xfer_Log
  The source code for these state tables is supplied in an ASCII file in the same directory as the custom server: $CUR_DIR/ca/VOIP_Call_Transfer_dir.
- Custom server: A custom server, named VOIP_Call_Transfer, is supplied to perform the functionality of the MakeCall operation (for details of the available calls, see “The MakeCallStatus custom server function” on page 122).

Note: Only the executable version (not the source code) of this custom server is supplied.

Before you install
Before you can install the VoIP call transfer application, you must have:
- A WebSphere Voice Response system running Version 6 Release 1 of WebSphere Voice Response for AIX.
- A connection to SIP telephones within a SIP network that support the REFER method with a REPLACES header.
- An application that performs a transfer operation using a screened transfer with simple call-answer supervision.
- Sufficient disk space for the components you are installing (WebSphere Voice Response for AIX: Installation explains how to check this).

Importing the VOIP_Call_Transfer custom server
If you intend to use the VoIP call transfer application, you must import the VOIP_Call_Transfer custom server and associated state table:
1. **Starting WebSphere Voice Response**: Start WebSphere Voice Response and log in to the user account you use for WebSphere Voice Response administration.

2. **Importing the custom server and state tables**: In the WebSphere Voice Response Welcome window, click **Applications → Application → Import → Replace → File**. Select the file `/usr/lpp/dirTalk/sw/VOIP/call_transfer/VOIP_Call_Transfer.imp`. The system displays the VOIP_Call_Transfer application icon.

3. **Creating an application profile**: Open the VOIP_Call_Transfer application. The system displays the Application (VOIP_Call_Transfer) window.

4. Click **Object → New → Application Profile**. The system displays the Application Profiles window, followed by the Application Profile window.

5. Type the name you want to allocate to the profile in the **Name** field.

6. Click **State Table**, then select your application that performs a call transfer. Click **OK**.

7. Click **File → Save**.

8. Specify the phone number to be used as the **Application Profile ID**. Click **OK**.

9. In the Application window, click **View → Refresh**. The system displays the Application Profiles folder, with the new Application Profile icon inside it.

10. **Starting the custom server**: Open the Custom Server Manager window, and click **Welcome → Operations → Custom Server Manager**. The system displays the available custom servers in a window.

11. To start the VOIP_Call_Transfer custom server click **Run Status → Start**. The Run Status button displays 'Waiting' after a short while.

12. **Running the Application**: How you run the application will depend on your application.

**Configuring the VOIP_Call_Transfer custom server**

The VOIP_Call_Transfer custom server has some command line parameters that you can set to help you debug problems, or to fine-tune the operation of the custom server.

The parameters are defined in Table 2. For information on how to set the parameters, see Setting configuration options.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>off</td>
<td>Send extra information alarms to the error log.</td>
</tr>
</tbody>
</table>
Table 2. Configuration options for the VOIP_Call_Transfer custom server (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d</td>
<td>off</td>
<td>Provide extra debugging information (in addition to the information that always gets sent).</td>
</tr>
<tr>
<td>-en</td>
<td>n=0</td>
<td>The event data (n) sent by the custom server if the third party hangs up (see CA_Report_Channel_Event() in the WebSphere Voice Response for AIX: Custom Servers book).</td>
</tr>
<tr>
<td>-in</td>
<td>n=0</td>
<td>The information field (n) sent by the custom server if the 3rd party hangs up (see CA_Report_Channel_Event() in the WebSphere Voice Response for AIX: Custom Servers book).</td>
</tr>
<tr>
<td>-s</td>
<td>off</td>
<td>Print debugging information to stdout, as well as to AIX system trace.</td>
</tr>
<tr>
<td>-z</td>
<td>off</td>
<td>This parameter is for test purposes only. It causes the custom server to enrol one of each type of error possible on custom server startup, then terminates. The enrolment of errors takes about 30s.</td>
</tr>
<tr>
<td>-Entry_point</td>
<td>entry_point = begin</td>
<td>The name of the state table entry point to use in the outbound state table. This must be the same for all of the outbound state tables used in VoIP transfer operations.</td>
</tr>
<tr>
<td>-file_name</td>
<td>file_name = null</td>
<td>Name of the log file to be used for logging debug information. If this is blank, no logging information will be sent.</td>
</tr>
<tr>
<td>-sttbl_name</td>
<td>sttbl_name = VOIP_SupA_Xfer</td>
<td>Name of the state table to use for attended transfers (that follow the WebSphere Voice Response signaling model for transfer operations).</td>
</tr>
</tbody>
</table>

Setting configuration options:

To set one or more of the configuration options, follow the procedure below:

1. From the Welcome window, select **Applications —> Custom Servers**
2. Setting a **command line parameter for the custom server**: Highlight the VOIP_Call_Transfer custom server.
3. Click **Server —> Open**.
   The system displays the Custom Server (VOIP_Call_Transfer) window.
4. Click **File —> Properties**.
   The system displays the Properties (VOIP_Call_Transfer) window.
5. Type your command line parameters in the panel titles **main() args**.
6. Click **OK**.
   The system closes the Properties (VOIP_Call_Transfer) window.
7. In the Custom Server (VOIP_Call_Transfer) window, click **File —> Save**.
8. **Restarting the custom server:** Open the Custom Server Manager window by clicking **Welcome —> Operations —> Custom Server Manager**. The system displays the available custom servers in a window.

9. If the **VOIP_Call_Transfer** custom server Run Status is set to Waiting, stop the custom server by clicking **Run Status —> Stop**. The Run Status button should display None, after a short while.

10. Start the **VOIP_Call_Transfer** custom server by clicking **Run Status —> Start**. After a short while the Run Status button should display Waiting. The new command line parameters are now in effect. If the Run Status remains at None, there is probably an error with one of the command line parameters. Check the error log for details.

### Before using the VoIP call transfer application

Before you can use the VoIP call transfer application, you must have:

- A connection to SIP telephones within a SIP network that support the REFER method with a REPLACES header.
- An application that performs a transfer operation using a screened transfer with simple call-answer supervision.
- Imported the custom server and state tables as described in “Importing the **VOIP_Call_Transfer** custom server” on page 116.

### Example state table

Here is an example state table showing how to perform an attended transfer using the SIP **VOIP_Call_Transfer** custom server:

```bash
#
# ==============================================================
# Description
DESCRIPTION("SIP Attended transfer using the VOIP_Call_Transfer custom server. AIX WebSphere Voice Response");
# The out bound call is answered in this state table
# using transfer with a CONSULT tag.
# ==============================================================
# Prompt Directory
PROMPT_DIRECTORY(Welcome);
#
# ==============================================================
# Entry Point(s)
ENTRY_POINT(begin,answer);
#
# ==============================================================
# Variables
#
# ==============================================================
# Actions
```
answer: "answer the phone"
AnswerCall()
edge EDGE_AC_NOT_RINGING: hup
;
"play 'Welcome' segment"
PlayPrompt("FALSE", Welcome)
edge EDGE_PP_LINE_PROBLEM: hup
dge EDGE_PP_NOTHING_PLAYED: hup
dge EDGE_HUP: hup
;
#This section sets up the out bound call leg.
"CONSULT TAG"
AssignData(SV541, "PUT_TAG", "CONSULT", "0")
;
"TO_HDR - ray"
AssignData(SV541, "PUT_TAG", "TO_HDR", "sip:ray@99.99.55.55:5060")
;
TransferCall(",", ",", 0, 0, 0, 30)
edge EDGE_TC_INVALID_PHONE_NO: hup
dge EDGE_TC_PHONE_BUSY: hup
dge EDGE_TC_NETWORK_BUSY: hup
dge EDGE_TC_NO_ANSWER: hup
dge EDGE_TC_OUTBOUND_LINE_PROBLEM: hup
dge EDGE_TC_UNEXPECTED_TONE: hup
dge EDGE_HUP: hup
;
# The original caller call can be consulted here.
# The outbound callee can be consulted from the
# SuperXferA state table.
good: "play 'Welcome' segment"
PlayPrompt("FALSE", Welcome)
edge EDGE_PP_LINE_PROBLEM: hup
dge EDGE_PP_NOTHING_PLAYED: hup
dge EDGE_HUP: hup
;
# The Transfer takes place when the transfer action
# is executed with a TRANSFER tag.
"TRANSFER TAG"
AssignData(SV541, "PUT_TAG", "TRANSFER", "0")
;
TransferCall(",", ",", 0, 0, 30)
edge EDGE_TC_SUCCESSFUL: hup
edge EDGE_HUP: hup
;
hup: "terminate call"
CloseEverything()
;
Attended transfer using a call reference
This method of transfer assumes that two calls are already active and are
being managed by a WebSphere Voice Response application. To invoke a SIP
transfer using a state table, the call reference of the call that is to accept the
transfer must be known to the state table requesting the TransferCall action.
The Call Reference is available in SV237.
The VOIP_Call_Transfer custom server is not required when using the call reference method.

**Example state table**

Here is an example state table using the call reference method of transfer:

```
# # ==============================================================
# Description state managing Call A
# DESCRIPTION("SIP Attended transfer using a call reference.
# AIX WebSphere Voice Response");
# ==============================================================
# Prompt Directory
PROMPT_DIRECTORY(Welcome);
#
# ==============================================================
# Entry Point(s)
ENTRY_POINT(begin,answer);
#
# ==============================================================
# Variables
#
# ==============================================================
# Actions
answer: "answer the phone"
    AnswerCall()
    edge EDGE_AC_NOT_RINGING: hup
; "play 'Welcome' segment"
    PlayPrompt("FALSE", Welcome)
    edge EDGE_PP_LINE_PROBLEM: hup
    edge EDGE_PP_NOTHING_PLAYED: hup
    edge EDGE_HUP: hup
;
# The original caller call can be consulted here.
# The outbound callee can be consulted from the
# state table controlling call B.
# The Transfer takes place when the transfer action
# is executed with a TRANSFER tag and a CALLREF tag
# set to the call reference value of call B.
# The application must provide this CALLREF value. This could
# for example be passed from call B to call A using a custom server.
"TRANSFER TAG"
    AssignData(SV541, "PUT_TAG", "TRANSFER", "0")
    ;
    AssignData(SV541, "PUT_TAG", "CALLREF", "654321")
    ;
    TransferCall("", ",", 0, 0, 30)
    edge EDGE_TC_SUCCESSFUL: hup
    edge EDGE_HUP: hup
```
Custom server functions

This section describes the custom server functions supplied with the VoIP call transfer application.

The MakeCallStatus custom server function

The MakeCallStatus function is provided with the VOIP_Call_Transfer custom server. The MakeCallStatus function is used by the outbound state table to return information about the success of the outbound call to the third party.

For more information on the MakeCallStatus function see *WebSphere Voice Response for AIX: Designing and Managing State Table Applications*.

Sending user data to the outbound call

User information can be sent to the outbound call by a tag set within SV541. This must be assigned using the AssignData PUT_TAG action ahead of the TransferCall action in the user application. For example:

```plaintext
AssignData(SV541, "PUT_TAG", "USERPW", "12345")
```

This data can be retrieved into, for example, userpw string variable ahead of making the outbound call in the VOIP_SupA_Xfer state table using:

```plaintext
AssignData(userpw, "GET_TAG_VALUE", tags, "USERPW")
```

where tags is an input parameter to VOIP_SupA_Xfer which is set to the value of SV541 assigned ahead of the TransferCall action in the user application.

In each case the USERPW tag is ignored by SIP.

Further details of all the input parameters for VOIP_SupA_Xfer can be found at "VOIP_SupA_Xfer" on page 123

Messages from the custom server

The VOIP_Call_Transfer custom server issues the following types of alarm messages:

- Red messages (for fatal problems)
- Yellow messages (warnings of any problems that may cause future failures)
- White messages (give information)
The messages are issued as standard custom server messages. The VOIP_Call_Transfer custom server error title and ID are shown in parameters 1 and 2 of the standard custom server errors. The error IDs have the form VOIP_XREFERnnn. The remaining parameters contain information specific to the particular error.

With any of these errors, if you have followed the suggestions in the User Response, and you are unable to solve the problem, contact IBM Support. When you call you will need details of any errors, your system setup, and details of the conditions that cause the problem. You may also require an AIX system trace taken when the problem occurs. Before taking the trace, set the -d option in the custom server. For more information on the command line options see “Configuring the VOIP_Call_Transfer custom server” on page 117.

Error messages generated by the VOIP_Call_Transfer custom server can be found here: WebSphere Voice Response for AIX: Problem Determination.

State table definitions

This section describes the state tables supplied with the VOIP_Call_Transfer application. Six state tables are provided as compiled state tables and ASCII state table source code. The compiled state tables are compiled directly from the ASCII state table source code. The source code for the state tables is provided in the directory $CUR_DIR/ca/VOIP_Call_Transfer_dir. The state tables fall into two groups:

Outbound state tables

An outbound state table provides functions to make the outbound call for the transfer operation:

- "VOIP_SupA_Xfer."

Helper state tables

The helper state tables encapsulate code that is used to provide interaction between the inbound and outbound state tables during the transfer, and other utility functions:

- "VOIP_Xfer_CS" on page 124.
- "VOIP_Xfer_C10" on page 125.
- "VOIP_Xfer_Data" on page 126.
- "VOIP_Xfer_Log" on page 126.
- "VOIP_Xfer_Stat" on page 127.

The following sections describe the state tables.

VOIP_SupA_Xfer

The VOIP_SupA_Xfer state table is used as the default outbound state table for attended transfers when using DTEA, hardware. The state table attempts to perform a MakeCall and returns the success or failure to the VOIP Signaling Process using the VOIP_Call_Transfer custom server. After the
results of the MakeCall have been reported, if the action was successful, the state table enters into a WaitEvent for up to 30 seconds to allow the transfer to complete. When the transfer completes, the transfer channel is hung up and the WaitEvent exited. If the call does not hang up, the WaitEvent times out and the state table finishes (this hangs up the call). If the MakeCall was not successful, the call hangs up immediately.

You can customize the application, but you must not change the basic functions. Comments in the source code show where you can insert code to perform a consultation with the third party.

**Parameters**

**String phone_number (maximum of 40 characters)**

The number to dial for the outgoing part of the transfer. This can contain a NULL string if a SIP header is passed using SV541. For more information, see the definition of phone_number in the description of the MakeCall state table action in *WebSphere Voice Response for AIX: Application Development using State Tables*.

**String format (maximum of 50 characters)**

The format string for the number to dial for the outgoing part of the transfer. For more information, see the definition of format in the description of the MakeCall state table action in *WebSphere Voice Response for AIX: Application Development using State Tables*.

**String log_filename (maximum of 64 characters)**

If this parameter is blank, no event logging is performed. If you specify a file name, event logging of the transfer calls is performed and the results are logged in the file you specify. The logging is performed by the VOIP_Xfer_Log state table.

**String tags**

Contains a copy of the contents of SV541 sent from the application. This can contain SIP tags such as T0_HDR (see “Voice over IP tags” on page 104), or user-defined tags. User defined tags are ignored by the SIP stack.

**VOIP_Xfer_C5**

The VOIP_Xfer_C5 state table encapsulates the states needed to concatenate up to 5 strings. It returns the result as a single string.

**Parameters**

**String in0**

The first string.

**String in1**

The second string.
String in2
  The third string.
String in3
  The fourth string.
String in4
  The fifth string.
String out (this value is returned)
  The concatenation of the input strings (in0 + in1 + in2 + in3 + in4).

VOIP_Xfer_C10
The VOIP_Xfer_C10 state table encapsulates the states needed to concatenate up to 10 strings. It returns the result as a single string.

Parameters
String in0
  The first string.
String in1
  The second string.
String in2
  The third string.
String in3
  The fourth string.
String in4
  The fifth string.
String in5
  The sixth string.
String in6
  The seventh string.
String in7
  The eighth string.
String in8
  The ninth string.
String in9
  The tenth string.
String out (this value is returned)
  The concatenation of the input strings (in0 + in1 + in2 + in3 + in4 + in5 + in6 + in7 + in8 + in9).
VOIP_Xfer_Data
The VOIP_Xfer_Data state table encapsulates the actions necessary to use the SetUserData custom server function. It is called by the incoming state table to set up some user-defined data to be supplied to the outbound state table during a transfer. This call must be made before the TransferCall action is performed. You can customize the application, but you must not change the basic functions.

Parameters

String log_filename (maximum of 64 characters)
If this parameter is blank, no event logging is performed. If you specify a file name, event logging of the supplied data is performed and the results are logged in the file you specify. The logging is performed by the VOIP_Xfer_Log state table.

String user_data1 (maximum of 64 characters), (this value is returned)
User data supplied to the SetUserData custom server call.

String user_data2 (maximum of 16 characters), (this value is returned)
User data supplied to the SetUserData custom server call.

String user_data3 (maximum of 16 characters), (this value is returned)
User data supplied to the SetUserData custom server call.

String user_data4 (maximum of 16 characters), (this value is returned)
User data supplied to the SetUserData custom server call.

String user_data5 (maximum of 16 characters), (this value is returned)
User data supplied to the SetUserData custom server call.

Number return_code, (this value is returned)
This value is returned to indicate whether or not the call to the VOIP_Xfer_Data state table was successful. The possible return values are:

0 The state table ran successfully
1 The OHSL to the VOIP_Call_Transfer custom server failed.
2 The SendData to the VOIP_Call_Transfer custom server failed.

VOIP_Xfer_Log
The VOIP_Xfer_Log state table encapsulates the actions needed to log debug data in a file.

The state table first checks the logging_on flag to see if the supplied data should be sent to the log file. The data to be logged consists of a header string, followed by six general strings; these are concatenated before being logged using a LogEvent action. The state table always returns edge 0, regardless of any errors.
Parameters

Number logging_on
   A value of 1 means perform data logging; any other value causes the state
table to exit without logging anything.

String header
   The header string for the logging function. This must contain at least the
   file name, in a format required by the LogEvent state table action.

String in_1
   The first string to log.

String in_2
   The second string to log.

String in_3
   The third string to log.

String in_4
   The fourth string to log.

String in_5
   The fifth string to log.

String in_6
   The sixth string to log.

VOIP_Xfer_Stat
The VOIP_Xfer_Stat state table encapsulates the actions necessary to use the
GetUserStatus custom server function. It is used by the incoming state table to
receive the user-specified status values returned by the outbound state table
after the make call. This call must be made after the TransferCall action and
before the TerminateCall or ReconnectCall actions. You can customize the
application, but you must not change the basic functions.

Parameters

String log_filename (maximum of 64 characters):
   If this parameter is blank, no event logging is performed. If you specify a
   file name, event logging of the supplied data is performed and the results
   are logged in the file you specify. The logging is performed by the
   VOIP_Xfer_Log state table.

Number user_status
   The status of the call to get the user status information. The possible
   values are:

   0   User status data is present.

   1   Request to get user status is not valid at this time.
No user status data is available.

char user_status1[64]
User-defined status field that was sent by the outbound state table during a transfer after the outbound call was made.

char user_status2[16]
User-defined status field that was sent by the outbound state table during a transfer after the outbound call was made.

char user_status3[16]
User-defined status field that was sent by the outbound state table during a transfer after the outbound call was made.

char user_status4[16]
User-defined status field that was sent by the outbound state table during a transfer after the outbound call was made.

char user_status5[16]
User-defined status field that was sent by the outbound state table during a transfer after the outbound call was made.

Number return_code (this value is returned)
This value is returned to indicate whether or not the call to the VOIP_Xfer_Data state table was successful. The possible return values are:

0  The state table ran successfully
1  The OHSL to the VOIP_Call_Transfer custom server failed.
2  The SendData to the VOIP_Call_Transfer custom server failed.
3  The ReceiveData failed with a Timeout edge.
4  The ReceiveData failed with a No More Data edge.
5  The ReceiveData failed with a Data Not Found edge.
6  The ReceiveData failed with a Host Problem edge.
7  The ReceiveData failed with a Host Not Open edge.

SIP support of Message Waiting Indicator (MWI)
WebSphere Voice Response for AIX provides support for sending a blind Notify message for controlling a Message Waiting Indicator. Blind Notify means that a SIP Notify message can be sent without the need for a Subscribe method.

To use this method of controlling a Message Waiting Indicator:
1. From the Welcome window, select Configuration —> System Configuration —> Change.
2. Click **Exchange Data Link**.

3. Set the **MWI Automatically Set** to Yes.

4. Close the Exchange Data Link window to save your changes.

**Note:** Only numeric mail box IDs are supported and therefore WebSphere Voice Response must be configured to use a proxy so that the Notify message can be correctly routed over a network.

The following example shows the contents of the Notify message:

```
16:20:11 NOTIFY sip:123456@192.168.0.10:5060 SIP/2.0
To: <sip:123456@acme.com>
From: <sip:WVRuser@192.168.0.110:5060>; tag=2b085750c1-192.168.0.110
Content-Type: application/simple-message-summary
Content-Length: 23
Call-ID: 26734cea84fad0_192.168.0.10
Max-Forwards: 70
CSeq: 7627 NOTIFY
Via: SIP/2.0/UDP 192.168.0.10:5060 branch=z9hG4bK2456725f63732f66f-192.168.0.110
Contact: <sip:WVRuser@192.168.0.110:5060;transport=UDP>
Route: <sip:123456@acme.com>
Messages-Waiting: yes
```
Appendix A. System parameters

This section lists all the Voice over IP system parameters. It gives the following information about each. All system parameters, for both DTEA and DTNA media are contained within the same system parameters group (VoIP DTEA and DTNA Media). Not all parameters apply to both DTEA and DTNA media. See each parameter description for applicability.

Note: The DTEA hardware adapter in now no longer available but is still supported by WebSphere Voice Response Version 6.1. For this reason, some of the following parameter descriptions still refer to DTEA.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Context</th>
<th>Authorization</th>
<th>Parameters</th>
<th>Defaults</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The group to which the parameter belongs, and the name on the button in the System Configuration window that gives access to the parameter values.</td>
<td>Admin or Field.</td>
<td>The range, or a list of possible values, to which you can set the parameter.</td>
<td>The value to which the parameter at WebSphere Voice Response installation.</td>
<td>A brief explanation of the function of the parameter and, if necessary, the meaning of the values.</td>
</tr>
</tbody>
</table>
VoIP DTEA and DTNA Media parameters

1st Codec Preference

Parameter group
VoIP DTEA and DTNA Media

Applicability
DTEA and DTNA

Access level
Admin

Possible values
G711 ALAW (64Kb/s)
G711 MULAW (64Kb/s)
G723
G729
iLBC 20
iLBC 30

Defaults
G711 ALAW (64Kb/s)

Explanation
The codec that will be the first choice when negotiating media streaming. For DTEA, this value can be set to any of the above values except iLBC 20, or iLBC 30. For DTNA, this value should only be set to None, G711, A law, µ law, iLBC 20, or iLBC 30 otherwise an error message will be issued when the trunk is enabled.

2nd Codec Preference

Parameter group
VoIP DTEA and DTNA Media

Applicability
DTEA and DTNA
Access level

Admin

Possible values

None
G711 ALAW (64Kb/s)
G711 MULAW (64Kb/s)
G723
G729
iLBC 20
iLBC 30

Defaults

G711 ALAW (64Kb/s)

Explanation

The codec that will be the second choice when negotiating media streaming. If this is set to None all subsequent codecs will be ignored. For DTEA, this value can be set to any of the above values except iLBC 20, or iLBC 30. For DTNA, this value should only be set to None, G711, A-law, µ-law, iLBC 20, or iLBC 30 otherwise an error message will be issued when the trunk is enabled.

3rd Codec Preference

Parameter group

VoIP DTEA and DTNA Media

Applicability

DTEA only

Access level

Admin

Possible values

None
G711 ALAW (64Kb/s)
G711 MULAW (64Kb/s)
G723
G729
iLBC 20
iLBC 30

Defaults

None

Explanation

The codec that will be the third choice when negotiating media streaming. If this is set to None all subsequent codecs will be ignored. For DTEA, this value can be set to any of the above values except iLBC 20, or iLBC 30. For DTNA, this value should only be set to None, G711, A law, µ law, iLBC 20, or iLBC 30 otherwise an error message will be issued when the trunk is enabled.

4th Codec Preference
Parameter group

VoIP DTEA and DTNA Media

Applicability

DTEA only

Access level

Admin

Possible values

None
G711 ALAW (64Kb/s)
G711 MULAW (64Kb/s)
G723
G729
iLBC 20
iLBC 30

Defaults

None
Explanation

The codec that will be the fourth choice when negotiating media streaming. For DTEA, this value can be set to any of the above values except iLBC 20, or iLBC 30. For DTNA, this value should only be set to None, G711, A law, μ law, iLBC 20, or iLBC 30 otherwise an error message will be issued when the trunk is enabled.

DTMF Transmission Method
Parameter group

VoIP DTEA and DTNA Media

Applicability

DTEA only

Applicability

DTEA and DTNA

Access level

Admin

Possible values

In-band
RTP payload

Defaults

In-band

Explanation

Determines which method is used to send DTMF keys. This can be used when sending compressed voice to ensure that DTMF keys can be detected.

Enable Echo Cancellation
Parameter group

VoIP DTEA and DTNA Media
Applicability
DTEA only

Access level
Admin

Possible values
No
Yes

Defaults
No

Explanation
Setting this parameter to ‘yes’ will enable an echo canceller to be built as part of the DSP processing resource.

Note: DTNA does not provide echo cancellation.

G711 Voice activity det/comfort noise gen

Parameter group
VoIP DTEA and DTNA Media

Applicability
DTEA only

Access level
Admin

Possible values
No
Yes

Defaults
No
Explanation

Setting this parameter will enable detection of silence in the incoming voice and the resulting transmission of special short RTP packets containing silence indication.

Note: DTNA will always send RTP packets, regardless of this setting.

G711 Packet Voice Interval (ms)
Parameter group

VoIP DTEA and DTNA Media

Applicability

DTEA only

Access level

Admin

Possible values

10 through 200

Defaults

20

Explanation

Time between the transmission of voice packets when using G711 codec.

Note: DTNA always uses 20ms packet voice interval for G711 (the only codec supported for DTNA).

G729 Voice activity det/comfort noise gen
Parameter group

VoIP DTEA and DTNA Media

Applicability

DTEA only
**Access level**

Admin

**Possible values**

No
Yes

**Defaults**

No

**Explanation**

Setting this parameter will enable detection of silence in the incoming voice and the resulting transmission of special short RTP packets containing silence indication.

**G729 Packet Voice Interval (ms)**

**Parameter group**

VoIP DTEA and DTNA Media

**Applicability**

DTEA only

**Access level**

Admin

**Possible values**

20 through 200

**Defaults**

20

**Explanation**

Time between the transmission of voice packets when using G729 codec.
**G723 Voice activity det/comfort noise gen**  
Parameter group  
VoIP DTEA and DTNA Media

**Applicability**  
DTEA only

**Access level**  
Admin

**Possible values**  
No  
Yes

**Defaults**  
No

**Explanation**  
Setting this parameter will enable detection of silence in the incoming voice and the resulting transmission of special short RTP packets containing silence indication.

---

**G723 Data Transfer Rate**  
Parameter group  
VoIP DTEA and DTNA Media

**Applicability**  
DTEA only

**Access level**  
Admin

**Possible values**  
5.3 Kb/s (ACELP)  
6.3 Kb/s (MP_MLP)
Defaults

5.3 Kb/s (ACELP)

Explanation

The data rate when using G.723 codec. (5.3 Kb/s is 12:1 compression, 6.3 Kb/s in 10:1).

G723 Packet Voice Interval (ms)
Parameter group

VoIP DTEA and DTNA Media

Applicability

DTNA only

Access level

Admin

Possible values

30 through 210

Defaults

30

Explanation

Time between the transmission of voice packets when using G723 codec.

Inbound DTMF Method Override
Parameter group

VoIP DTEA and DTNA Media

Applicability

DTNA only

Access level

Admin
Possible values

Use negotiated values
DTMF using tones (‘in-band’ audio tones in the RTP stream)
DTMF using payload (RFC2833 payload)
DTMF using SIP info (using the SIP INFO method)

Defaults

Use negotiated values (what is agreed as part of the SIP protocol)

Explanation

Normally, the DTMF method is limited to RTP methods only (in-band/payload). However, using the System Parameter Inbound DTMF Method Override, it is possible to force the DTMF transmission type to be in-band, payload or (the new option) SIP Info method. In this case, the State Table 'GetData' action will allow collection of DTMF keys sent using the SIP INFO method. If you want the DTMF transfer method to be agreed as part of the SIP call setup process (by the SDP in the INVITE and 200 OK messages), set this option to Use Negotiated Values. See also "Outbound DTMF Method Override."

Outbound DTMF Method Override

Parameter group

VoIP DTEA and DTNA Media

Applicability

DTNA only

Access level

Admin

Possible values

Use negotiated values
DTMF using tones (‘in-band’ audio tones in the RTP stream)
DTMF using payload (RFC2833 payload)
DTMF using SIP info (using the SIP INFO method)

Defaults

Use negotiated values (what is agreed as part of the SIP protocol)
**Explanation**

Normally, the DTMF method is limited to RTP methods only (in-band/payload). However, using the System Parameter Outbound DTMF Method Override, it is possible to force the DTMF transmission type to be in-band, payload or (the new option) SIP Info method. In this case, any State Table ‘Dial’ action will cause DTMF keys to be sent using the SIP INFO method. If you want the DTMF transfer method to be agreed as part of the SIP call setup process (by the SDP in the INVITE and 200 OK messages), set this option to *Use Negotiated Values*. See also “Inbound DTMF Method Override” on page 140.

**Override DTNA RTP Transport IP Address**

**Parameter group**

VoIP DTEA and DTNA Media

**Applicability**

DTNA only

**Access level**

Admin

**Possible values**

Null, or an IPv4 ‘dotted’ IP address such as 9.20.38.97

**Defaults**

Null

**Explanation**

This parameter controls the IP address used for DTNA Media (RTP). If left at the default setting (Null) or set to Null, WebSphere Voice Response selects the first valid IP address on the system unit. If set to a ‘dotted’ IP address, WebSphere Voice Response uses this address for sending and receiving RTP Media. The specified IP address must exist as a network address on the system unit otherwise WebSphere Voice Response will issue an error message when the VoIP channels are enabled.
**RTCP Enable Sender Report**

*Parameter group*

VoIP DTEA and DTNA Media

**Applicability**

DTEA and DTNA

**Access level**

Admin

**Possible values**

No
Yes

**Defaults**

No

**Explanation**

Defines whether to enable the sending of RTCP reports.

**RTCP Sender Report Interval**

*Parameter group*

VoIP DTEA and DTNA Media

**Applicability**

DTEA and DTNA

**Access level**

Admin

**Possible values**

1 through 60.

**Defaults**

5
Explanation

The interval at which RTCP reports will be sent (if RTCP Enable Sender Report is enabled).

RTP Base Port Number

Parameter group

VoIP DTEA and DTNA Media

Applicability

DTEA and DTNA

Access level

Admin

Possible values

2 through 65534

Defaults

6000

Explanation

The base port number to be used by the adapter to assign to a media channel. The adapter will use this value as the base for assigning 480 ports for RTP and RTCP. The base port number should always be an even number.

Note: For DTNA, ports can be calculated as follows:

For trunk T (1...16) and Channel C (1...24 or 1...30)
RTP port = ((T-1)*30 + C-1)*2 + Base Port
RTCP port = ((T-1)*30 + C-1)*2 + Base Port + 1

RTP IP TOS Byte (TOS)

Parameter group

VoIP DTEA and DTNA Media

Applicability

DTEA and DTNA
**Access level**

Admin

**Possible values**

0 through 63

**Defaults**

0

**Explanation**

The Type of Service (TOS) setting for RTP packets being sent across the network. A higher setting will give RTP packets priority over standard IP packets.

**RTP IP Time to Live (TTL)**

**Parameter group**

VoIP DTEA and DTNA Media

**Applicability**

DTEA and DTNA

**Access level**

Admin

**Possible values**

1 through 255

**Defaults**

20

**Explanation**

Time to Live (TTL) for RTP packets sent over the network.
RTP Security Negotiation
Parameter group
VoIP DTEA and DTNA Media

Applicability
DTNA

Access level
Admin

Possible values
Unsecured
Secure
Both

Defaults
Unsecured

Explanation
Secure RTP provides confidentiality and message authentication to RTP data. It can be used to prevent people from listening to or tampering with the audio data sent over an unsecure network like the internet. In WebSphere Voice Response, there are three possible RTP security configuration settings:

Unsecured
The default. WebSphere Voice Response does not accept secure RTP for inbound or offer secure RTP for outbound calls. Inbound calls that only offer secure RTP will be rejected with a 488 Not Accepted Here response.

Secure
WebSphere Voice Response only uses secure RTP. Inbound calls not capable of secure RTP are rejected with a 488 Not Accepted Here response, and outbound calls made by WebSphere Voice Response will only offer secure RTP.

Both
For inbound calls WebSphere Voice Response accepts secure RTP if offered, but will also accept calls if only RTP is offered. In the case of both secure RTP and RTP being offered, secure RTP will be used. For outbound calls, both secure RTP and RTP will be offered.
Partial support is provided for optional crypto session parameters [RFC 4568 section 6.3]. This support is enabled when Secure RTP has been configured for either Secure or Both.

The following session parameters are fully supported:
- UNENCRYPTED_SRTP
- UNENCRYPTED_SRTCP
- UNAUTHENTICATED_SRTP

All other session parameters are parsed, but are not supported. Any crypto lines containing the unsupported parameters are ignored, and treated as unsuitable matches. If there are no other suitable matches (which can be either unsecure RTP/AVP, or crypto attributes with supported session parameters) the SDP is rejected. This will result in a SIP response of 488 Not Acceptable Here.

Session parameters will never be presented on outbound SIP requests/responses. This includes outbound INVITE (make call or on hold requests) and responses to OPTIONS.

See “Secure RTP” on page 92 for more information.

VoIP SIP Signaling parameters

Accept Inbound Transfer Requests
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
No
Yes

Defaults
Yes
Explanation

This determines whether WebSphere Voice Response will accept a request to transfer a call to a different SIP endpoint.

Add Host Name To User Agents?
Parameter group

VoIP SIP Signalling

Applicability

DTEA and DTNA

Access level

Admin

Possible values

No
Yes

Defaults

Yes

Explanation

If enabled, appends the hostname of the machine to the end of all User Agents used by registrations. The main purpose of this is to denote which machine an individual registration has come from.

User Agents are sent as specified by the following in order of priority, with the hostname appended to the result if enabled:

1. The row of the .ini file that describes the destination.
2. The definition of Registrar in $SYS_DIR/voip/master.ini, if not specified in the row.
3. The entry in the System Configuration window, if neither of the above are specified.

Call Signalling Port
Parameter group

VoIP SIP Signalling
Applicability
DTEA and DTNA

Access level
Admin

Possible values
1024 through 65535

Defaults
5060

Explanation
The local port used for all SIP signalling.

CHP available call reject threshold
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
0
A integer greater than 0

Default
0

Explanation
To allow sufficient time for handling new incoming calls while other CHPs are cleaning up from previous calls, this parameter determines the number of
additional WebSphere Voice Response Channel Processes (CHPs) that must be available before an incoming call can be accepted. If the arrival rate of new incoming calls exceeds the rate at which CHPs are being cleaned up, SIP calls may not get answered within the required time causing timeouts and recovery actions to be instigated.

If set to zero (default), no check is made as to whether at least one CHP is available before accepting the SIP Call. If set to any other value, the number of free (unallocated) CHP's must be more than that value for the SIP call to be accepted.

If a call is rejected for this reason, a '486 Busy' is returned as a response to the SIP incoming call. This causes the far end to retry rather than timeout, cancel and restart the call.

**Default CLID for Incoming VoIP Calls**

**Parameter group**

VoIP SIP Signalling

**Applicability**

DTEA and DTNA

**Access level**

Admin

**Possible values**

The number of the default application.

**Defaults**

6661234

**Explanation**

The default calling line ID (CLID) to use for application profile selection when a suitable identifier cannot be found in an incoming call.

**Default Destination URI**

**Parameter group**

VoIP SIP Signalling
Applicability
DTEA and DTNA

Access level
Admin

Possible values
Null

Destination URI
Defaults
Null

Explanation
The routing address for messages when proxy mode is set to None. This may be the address of a SIP gateway. If this field is left blank then calls will be sent directly to the endpoint specified in the To Header. Example formats are:

- sip:gateway@anyplace.com
- sip:gateway@9.20.38.97
- 9.20.38.97

Default Destination Port
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
1024 through 65535
Defaults
5061

Explanation
The port associated with the default destination URI.

DNSSRV Server address
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
Null
Address of the DNSSRV Server.

Defaults
Null.

Explanation
The address of the DNSSRV Server to be used if proxy mode is set to 'Automatic Routing: DNSSRV'. The format of this address could be, for example, dnssrv.ibm.com or 9.20.38.97.

DNSSRV Server Port
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA
Access level
Admin

Possible values
1024 through 65535

Defaults
5061

Explanation
The IP address of the DNSSRV Server.

E164 Prefixes to Strip
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
Prefixes separated by a comma

Defaults
Null

Explanation
Defines which leading digits should be stripped from SIP numeric user parts for E.164 country codes, for example, ‘1,44,393’ will strip +1, +44 or +393 from the number, ‘+’ will strip only the +, leaving the country code.
Ignore replaces option for Attended Transfer

Parameter group

VoIP SIP Signalling

Applicability

DTEA and DTNA

Access level

Admin

Possible values

No
Yes

Defaults

No

Explanation

Specifies whether or not the presence of an inbound REPLACES header should be used to control attended transfers. If set to No, attended transfer will not proceed unless a REPLACES header has been received inbound. If set to Yes, attended transfer will continue regardless of a REPLACES header being received inbound.

See VoIP SIP attended transfer for more information on attended transfer.

Inbound Call Channel Allocation Method

Parameter group

VoIP SIP Signalling

Applicability

DTEA and DTNA

Access level

Admin
Possible values

Allocate calls from highest available channel (using 'linear' allocation)
Allocate calls using 'round robin' method
Allocate calls for D2IS
Allocate calls balanced across trunks

Defaults

Allocate calls from highest available channel

Explanation

This determines how WebSphere Voice Response allocates an incoming SIP call to a trunk and channel.

The default is by searching for the first free channel starting from the highest numbered channel and working downwards (from Trunk 16, channel 30 to Trunk 1 Channel 1). For some systems which use CTI (Computer Telephony Integration) this means that highest-numbered channels are reused very quickly and also there is an uneven spread of active channels across the complete range.

For more even channel allocation, the Allocate calls using 'round robin' search method begins at the next lower-numbered channel from the one last allocated (wrapping first to last).

To ensure the call load is spread over the adapters on the system, specify Allocate calls balanced across trunks. You must use this method if using iLBC compression.

The Allocate calls for D2IS method is for use with the Genesys supplied D2IS custom server. The allocated trunk and channel are based on the dialled number. The first three digits of the dialled number are ignored. All digits after this are converted into a trunk and channel number. The trunk is derived from the remaining digits of the dialled number divided by 100, the channel is the remaining digits of the dialled number, modulo 100. For example:

- 4440205 is trunk 2, channel 5
- 444000205 is trunk 2, channel 5
- 11100430 is trunk 4, channel 30

Note: Using the 'D2IS' method, it is not possible to specify an application profile number to state table mapping. However, Number-to-application (NumToApp) mappings are possible if the first three digits map to a
particular application, for example, 123001001 could be defined as 123* and
987001001 could be defined as 987* in NumToApp mappings.

**Message Header Format**

**Parameter group**

VoIP SIP Signalling

**Applicability**

DTEA and DTNA

**Access level**

Admin

**Possible values**

Compact
Long

**Defaults**

Long

**Explanation**

This determines the format of the message headers when a message is
created. Using compact headers will reduce the size of the message.

**Organization Name**

**Parameter group**

VoIP SIP Signalling

**Applicability**

DTEA and DTNA

**Access level**

Admin
Possible values
Null
Organization Name

Defaults
Null

Explanation
If this field is completed, the value can be added to the call setup messages to identify the group from which the call is being made.

Outbound SIP INFO
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
application/vnd.nortelnetworks.digits
application/dtmf-relay

Defaults
application/vnd.nortelnetworks.digits

Explanation
The format that is used to send outbound SIP INFO header information for DTMFs initiated by WebSphere Voice Response.

Override SIP Transport IP Address
Parameter group
VoIP SIP Signalling
Applicability
DTEA and DTNA

Access level
Admin

Possible values
Null
SIP transport IP address

Defaults
Null

Explanation
The IP address on which the SIP signalling stack will run if multiple network connections are available on a single machine.

Proxy Mode

Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
None
Local Proxy
Automatic Services DNSSRV
Proxy Routing Table

Defaults
None.
**Explanation**

Specifies the method that will be used to route the SIP calls.

*None* specifies that WebSphere Voice Response is not to use a proxy. If a gateway is to be used, the address should be entered in the Default Destination URI field. If this field is left blank calls are sent directly to the endpoint.

*Local Proxy* specifies that WebSphere Voice Response is to route all SIP messages through a single Proxy, which is defined in the Proxy Address field.

*Automatic Services DNNSRV* specifies that the routing address is to be determined from a list of values returned by a DNNSRV lookup. This allows multiple proxies to be defined. Proxies can be assigned to specific services and be allocated a weighting to support load balancing.

**Note:** The Automatic Services DNNSRV method is not currently supported. If selected, the Manual Services Routing Table method will be used in its place.

**Proxy Routing Table:** The Manual Services Routing Table defined in /usr/lpp/dirTalk/db/sys_dir/srv.init (where an example srv.init file can be found) allows SRV request records to be manually entered into the srv.init file. The information in this file will then be used for routing by SIP signaling. If the request records contain weights and priorities, scheduling of services is maintained according to RFC 2782. The srv.init file can be configured dynamically whilst VoIP is running. Any changes made will be reflected within approximately 60 seconds of the file being saved.

**Proxy Address**

Parameter group

VoIP SIP Signalling

**Applicability**

DTEA and DTNA

**Access level**

Admin

**Possible values**

Null

Address of the proxy.
Defaults
Null.

Explanation
The address of the proxy to be used if proxy mode is set to 'Local Proxy'. The format of this address should be a SIP URI, for example, proxy@uk.ibm.com or proxy@9.20.2.200.

Proxy Port
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
1024 through 65535

Defaults
5060

Explanation
The IP port of the local proxy.

Register Addresses on Startup
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA
Access level

Admin

Possible values

No
Yes

Defaults

No

Explanation

Specifies whether or not the .ini files located in $SYS_DIR/voip are used to register SIP addresses at one or more registrars. If set to Yes, .ini files will be scanned and the register processes started up when a SIP trunk is enabled, and stopped when all SIP trunks are disabled. If set to No, no registrations will be made at all.

Register Default Timeout (Minutes)

Parameter group

VoIP SIP Signalling

Applicability

DTEA and DTNA

Access level

Admin

Possible values

Integer in the range 1 through 10080

Defaults

60
**Explanation**

Specifies the default expiry length of a SIP registration. This can be overridden by the $SYS_DIR/voip/master.ini file, the line of the secondary ini files pointed to by master.ini, or both.

**Register Default User Agent**

**Parameter group**

VoIP SIP Signalling

**Applicability**

DTEA and DTNA

**Access level**

Admin

**Possible values**

A String

**Defaults**

“IBM Websphere Voice Response”

**Explanation**

Specifies the default User Agent (which represents an end system) for a SIP registration. This can be overridden by the $SYS_DIR/voip/master.ini file, or the line of the secondary ini files pointed to by master.ini.

**RFC3264 Media on-hold method**

**Parameter group**

VoIP SIP Signalling

**Applicability**

DTEA and DTNA

**Access level**

Admin
Possible values

ip=0.0.0.0 media-on-hold method
RFC3264 defined media-on-hold method
ip=0.0.0.0 and RFC3264 defined media-on-hold methods

Defaults

ip=0.0.0.0 media-on-hold method

Explanation

Specifies whether WebSphere Voice Response should use the RFC3264 defined method for media on-hold (rather than the older ip=0.0.0.0 method), or a combination of both methods.

Secure SIP Enabled

Parameter group

VoIP SIP Signalling

Applicability

DTEA and DTNA

Access level

Admin

Possible values

False
True

Defaults

False

Explanation

This controls whether or not WebSphere Voice Response can accept secure inbound calls and can make secure outbound calls. Setting the Secure SIP Enabled parameter alone will not allow WebSphere Voice Response to make secure SIP calls. You must have previously set up the keyring.db certificate database. See “Secure SIP minimal configuration” on page 77 for instructions on how to do this.
Session Timer Allow Update For Refresh  
Parameter group  
VoIP SIP Signalling  

**Applicability**  
DTEA and DTNA  

**Access level**  
Admin  

**Possible values**  
No  
Yes  

**Defaults**  
No  

**Explanation**  
This controls whether reINVITE or UPDATE are to be used for refresh (assuming that far-end has said that it will allow the use of UPDATE using the ALLOW header).

Session Timer Enable  
Parameter group  
VoIP SIP Signalling  

**Applicability**  
DTEA and DTNA  

**Access level**  
Admin
**Possible values**

Disabled
Inbound only
Outbound only
Enabled

**Defaults**

Disabled

**Explanation**

Allows the Session Timer to be disabled completely, enabled for inbound calls only, enabled for outbound calls only, or enabled for both directions.

**Session Timer Inbound Refresher Default**

**Parameter group**

VoIP SIP Signalling

**Applicability**

DTEA and DTNA

**Access level**

Admin

**Possible values**

UAC (Remote Endpoint) - the call originator
UAS (WebSphere Voice Response = Local Endpoint)

**Defaults**

UAC (Remote End Point)

**Explanation**

Specifies whether, on an inbound call (WebSphere Voice Response acting as UAS=Uase Agent Server) and when no refresher is specified, on the Session Timer offer, whether the UAS (WebSphere Voice Response) or the UAC (Remote End Point) will be selected as the session timer refresher.
Session Timer Maximum Session Time
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
90 through 86400 (24 hours)

Defaults
1800 (30 minutes)

Explanation
Maximum allowed session time in seconds.

Session Timer Minimum Session Time
Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
90 through 86400 (24 hours)

Defaults
90
Explanation

Minimum allowed session time in seconds. This controls the value put in the 'Min-SE' header of a Session Timer offer.

Session Timer Outbound Calls Refresher Default
Parameter group

VoIP SIP Signalling

Applicability

DTEA and DTNA

Access level

Admin

Possible values

None Requested
UAC (WebSphere Voice Response = Local Endpoint) Requested
UAS (Remote Endpoint) Requested

Defaults

None Requested

Explanation

Specifies whether, on an outbound call (WebSphere Voice Response acting as UAC=User Agent Client) the choice of refresher will be none (left to remote endpoint to make decision), UAC (WebSphere Voice Response will be refresher) or UAS (forces far end to be refresher).

T.38 Fax Refer URI
Parameter group

VoIP SIP Signalling

Applicability

DTEA and DTNA
**Access level**
Admin

**Possible values**
URI String, for example, 10.20.31.42

**Defaults**
Null

**Explanation**
Specifies the URI to which a SIP REFER will be sent when T.38 fax is detected in an incoming INVITE.

**Transport Protocol**

**Parameter group**
VoIP SIP Signalling

**Applicability**
DTEA and DTNA

**Access level**
Admin

**Possible values**
UDP
TCP
TLS

**Defaults**
UDP

**Explanation**
The protocol to be used for sending all signaling messages for calls initiated by WebSphere Voice Response. Inbound calls will be accepted from both UDP and TCP protocols.
Use allowed host list

Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
No
Yes

Defaults
No

Explanation
Specifies whether or not to use an allowed host list to exclude SIP Requests from unlisted IP addresses.

Use SIP REQHDR for Application Profile Selection

Parameter group
VoIP SIP Signalling

Applicability
DTEA and DTNA

Access level
Admin

Possible values
No
Yes
**Defaults**

No

**Explanation**

Specifies which of TO or Request headers is used for called number (and Application Profile) selection.

---

**VoIP Media-Adapters parameter group**

The parameters in this group define how the DTEA cards connect to the IP network.

The media adapter parameters must be defined once for each installed DTEA card, with a maximum of four cards. Select the media adapter that you want to define and open it.

To make the parameter values effective, restart WebSphere Voice Response.

**IP Address**

**Context**

VoIP Media - Adapters

**Authorization**

Admin

**Parameters**

0.0.0.0 through 255.255.255.255

**Defaults**

0.0.0.0

**Comments**

Records the IPv4 IP address of a DTEA card. There are four instances of this parameter, one for each card.

**Subnet Mask**

**Context**

VoIP Media - Adapters
Authorization
Admin

Parameters
0.0.0.0 through 255.255.255.255

Defaults
255.255.255.0

Comments
Records the IPv4 IP address subnet mask of a DTEA card. There are four instances of this parameter, one for each card.

Default RTP router
Context
VoIP Media - Adapters

Authorization
Admin

Parameters
0.0.0.0 through 255.255.255.255

Defaults
0.0.0.0

Comments
Records the IPv4 IP address of the default RTP router for a DTEA card. There are four instances of this parameter, one for each card.
Appendix B. SIP-specific

This appendix describes several SIP-specific aspects of using Voice over IP/SIP:

- “SIP and IP support”
- “TOS byte” on page 174
- “Session timer” on page 176
- “Subscribe/Notify” on page 177
- “Related SIP RFCs” on page 178

SIP and IP support

WebSphere Voice Response supports IPSEC on signaling information. IPSEC is a method of making internet protocol as secure as possible by using, authentication, integrity checking, and encryption.

IPSEC allows protection of:
- Broad communications (messages entering and leaving an interface)
- Singular connections (communications between a single TCP connection).

The three main services that IPSEC provides are:

Authentication
The method by which a process may identify and verify a known host or end point.

Integrity checking
Allows a receiver of an IP packet to check that the packet has not been modified between being sent and being received.

Encryption
Securely hides the data and IP address information of a packet from anyone attempting to discover the content of the packet.

Each of these services can be configured by an administrator to give a flexible security facility.

These services are generally implemented and configured using one of two protocols:
- Authentication Header (AH) which provides services for Authentication and Integrity Checking.
- Encapsulating Security Payload (ESP) which provides confidentiality services (encryption) as well as services for Authentication and Integrity Checking.

Both protocols are suitable for IPv4 and IPv6.

Security Associations (SAs) exist to help provide IPSEC services. Specific sets of security parameters are mapped to a particular packet flow, such that a Security Association is established between a pair of hosts or gateways in a one-way connection. Security parameters include:

- IP address information
- An identifier known as the Security Parameters Index (SPI)
- The encryption and authentication methods in use

The Security Association provides all the information needed to set up a secure session using either AH or ESP.

Generally, IPSEC uses a Virtual Tunnel between hosts to provide a secure connection and to initiate Security Associations; this may be over a Virtual Private Network (VPN). Virtual Tunnels can also be used between network subnets and allow filter rules to be built allowing packets to be accepted or rejected based on these rules.

Generally in IPSEC, ESP is used in conjunction with an IPSEC tunnel. In AIX an IKE tunnel is frequently used.

WebSphere Voice Response supports IPSEC on signaling information (using the AIX IP stack), but not on media information as this is routed through DTEA.

**TOS byte**

A well managed network is capable of differentiating between different network traffic. This means that a network may offer a better service to traffic types which require high reliability and throughput, with minimal delays. Similarly other less essential traffic can be downgraded in the service they receive, since they are likely to be more tolerant of network delays. Type of Service (ToS), also known as Differentiated Services (DS) is part of the Quality of Service (QoS) model. ToS can be used by components on the network to define how they should prioritize the packet they have received. ToS provides a mechanism to do this by providing the following sub-fields:

- Precedence
- Delay
- Throughput
- Reliability
All of these subfields are set to 'normal' by default.

By setting these individual sub-fields, packets can be separated into classes, each packet class can then be handled by the network in an appropriate manner (as defined by your network definitions). This means that each network component will attempt to satisfy, but not guarantee, the requirements specified by the ToS byte fields.

The ToS byte forms part of Internet Protocol (IP) packet header, it exists as a 8 bit field value. The relationship to sub-fields and bits is as follows:

Table 3. ToS byte sub-fields and bits.

This table describes for each bit position, the purpose, possible values and their meaning.

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Purpose</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 0-2</td>
<td>Specifies precedence.</td>
<td>(0 = low, 7 = high)</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Specifies delay</td>
<td>(0 = normal delay, 1 = low delay)</td>
</tr>
<tr>
<td>Bit 4</td>
<td>Specifies throughput</td>
<td>(0 = normal throughput, 1 = high throughput)</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Specifies reliability.</td>
<td>(0 = normal reliability, 1 = high reliability)</td>
</tr>
<tr>
<td>Bits 6-7</td>
<td>Are currently reserved.</td>
<td></td>
</tr>
</tbody>
</table>

By default, all values are set to 0 (Normal). The WebSphere Voice Response VoIP feature allows users to set the ToS byte as a value between 0 - 64, representing bits 0 - 5 (inclusive), for all RTP media streams leaving all adapter cards.

Example

To turn on high Reliability, low Delay, normal Throughput and medium Precedence you would enter a value of 43 (out of 64). In the IP header, the field values would be set as follows:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Medium Precedence</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Low Delay</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>Normal Throughput</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>High Reliability</td>
</tr>
<tr>
<td>Bits</td>
<td>Value</td>
<td>Result</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>----------------------</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>Currently Reserved</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>Currently Reserved</td>
</tr>
</tbody>
</table>

To set the ToS byte for the VoIP signalling, users must edit the AIX policyd and rsvpd daemons configuration files and make use of the AIX QoS commands `mkqos` and `rmqos`. For more details please refer to your AIX Systems Management Guide: *Communications and Networks*.

**Note:** ToS byte will only ever take effect if your network is enabled to respect ToS Byte settings.

---

**Session Timer**

RFC4028 defines the SIP Session Timer mechanism (which is a way to prevent SIP sessions hanging up indefinitely). A brief description of its operation is as follows:

- SIP Session Timer is a mechanism in which a ‘heartbeat’ message sequence is used within a SIP session (a call) to protect against the unexpected loss of an endpoint leading to a SIP session continuing to remain up.

- There is an initial negotiation (actually a simple offer/answer like SDP) which determines:
  - If the session is to be refreshed
  - How often the refresh is to be performed
  - Which end does the refresh
  - The refresh mechanism to be used

Two new headers are defined to support Session Timer: ‘Session Expires’ and ‘Min-SE’. These are used to define the initial offer and the subsequent answer.

One new response code (422) is defined to report offered session timer duration is too small i.e. ‘Session Expires’ is less than ‘Min-SE’ supported.

In a SIP session with Session Timer active, either end can do the refresh and this can be done with either a ‘reINVITE’ or by using the SIP ‘UPDATE’ method.

WebSphere Voice Response implements RFC4028. To control Session Timer operation, WebSphere Voice Response provides the following additional System Parameters in the VoIP SIP Signalling group:

**Session Timer Enable**

- Allows the Session Timer to be disabled completely, enabled for
inbound calls only, enabled for outbound calls only, or enabled for both directions. See “Session Timer Enable” on page 164.

**Session Timer Minimum Session Time**
Minimum allowed session time in seconds. This controls the value put in the ‘Min-SE’ header of a Session Timer offer. See “Session Timer Minimum Session Time” on page 166.

**Session Timer Maximum Session Time**
Maximum allowed session time in seconds. See “Session Timer Maximum Session Time” on page 166.

**Session Timer Inbound Refresher Default**
See “Session Timer Inbound Refresher Default” on page 165.

**Session Timer Outbound Calls Refresher Default**
See “Session Timer Outbound Calls Refresher Default” on page 167.

**Session Timer Allow Update For Refresh**
This controls whether reINVITE or UPDATE are to be used for refresh. See “Session Timer Allow Update For Refresh” on page 164.

---

**Subscribe/Notify**

WebSphere Voice Response and Unified Messaging together support the use of the SIP Subscribe/Notify methods for notification of MWI (Message Waiting Indications) according to RFC3265 (SIP - Specific Event Notification).

Subscribe/Notify is automatically enabled by starting the UM IMC_Subscribe custom server, which takes over the role of handling MWI notifications from the standard UM MWISERVER custom server.

---

**Communicating over SIP with a switch using DTMF digits**

Normally, DTMF digits are communicated over the RTP (Media) stream using either inband (audio) or payloads (RFC2833). The problem in this case is that, with SIP, the media flow is between endpoints (the caller/called and WebSphere Voice Response) whereas the information needs to be sent to or received from a switch. The only way to do this is using the SIP Signalling path and the only defined way to do this is by using the SIP INFO method defined by RFC2976.

The SIP INFO Method is controlled by WebSphere Voice Response using a System Parameter in the VoIP DTEA and DTNA Media parameter group. Normally, the DTMF method is limited to RTP methods only (inband/payload). However, using the System Parameter Outbound DTMF Method Override, it is possible to force the DTMF transmission type to be inband, payload or (the new option) SIP Info method. In this case, any State
Table ‘Dial’ action will cause DTMF keys to be sent using the SIP INFO method. Similarly, using the System Parameter Inbound DTMF Method Override, it is possible to enable the DTMF reception method of SIP Info. In this case, the State Table ‘GetData’ action will allow collection of DTMF keys sent using the SIP INFO method.

SIP INFO Method is implemented for DTNA only. It is not implemented for DTEA.

### Related SIP RFCs

Details of the Session Initiation Protocol (SIP) RFCs related to WebSphere Voice Response SIP support are provided in this section.

The SIP RFCs related to WebSphere Voice Response SIP support include the following:

- 2833  DTMF Payload over RTP
- 3261  Session Initiation Protocol
- 3262  Reliability of provisional responses in SIP
- 3264  Session Description Protocol Offer/Answer model
- 3265  Specific Event Notification
- 3311  UPDATE method
- 3323  Privacy mechanism for SIP
- 3325  Private extensions to SIP for trusted networks
- 3515  SIP Refer Method
- 3891  REPLACES Header
- 4028  Session Timer
- 4566  Session Description Protocol
- 5589  Call Control - Transfer
Appendix C. TCP and UDP Network Configuration

This section describes how to optimize network configuration parameters for default TCP and UDP send and receive socket buffers sizes to improve VoIP performance and reliability under extreme load.

At startup a number of network configuration parameters are set by the AIX kernel. These parameters include the default TCP and UDP send and receive socket buffers sizes. These buffers are used to store messages that have been sent or received from the network, if these buffers become full, then this can result in failures sending TCP/UDP messages, TCP/UDP messages being lost with messages having to be resent. For the WebSphere Voice Response VoIP feature this could result in performance and reliability issues under extreme load. It is therefore recommend that the default values for these parameters are increased. By default the four parameters have the following values:

- `udp_sendspace = 9k`
- `udp_recvspace = 42k`
- `tcp_sendspace = 16k`
- `tcp_recvspace = 16k`

The `no` (network options) command can be used to view and change default network attributes. To view these values, as root, simply type `no -o parameter` where `parameter` can be one of the following:

**udp_sendspace**

To review the udp send buffer size in bytes.

For example, `no -o udp_sendspace` returns `udp_sendspace=9216`.

**udp_recvspace**

To review the udp receive buffer size in bytes.

For example, `no -o udp_recvspace` returns `udp_recvspace=9216`.

To reset these values, simply type `no -o parameter=newValue` where `parameter` refers to the network parameter you wish to change, and `newValue` is the new value to which you want to set it. For example, to set the `tcp_sendspace` to be 64k, type the following:

`no -o tcp_sendspace=64000`

The confirmation message Setting tcp_sendspace to 64000 is displayed.
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

The IBM Director of Licensing, IBM Corporation,
North Castle Drive,
Armonk,
NY 10504-1785,
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing,
Legal and Intellectual Property Law,
IBM Japan, Ltd.,
19-21, Nihonbashi-Hakozakicho, Chuo-ku,
Tokyo 103-8510, Japan.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY
OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow
disclaimer of express or implied warranties in certain transactions, therefore,
this statement may not apply to you.

This information could include technical inaccuracies or typographical errors.
Changes are periodically made to the information herein; these changes will
be incorporated in new editions of the publication. IBM may make
improvements and/or changes in the product(s) and/or the program(s)
described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for
convenience only and do not in any manner serve as an endorsement of those
Web sites. The materials at those Web sites are not part of the materials for
this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it
believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the
purpose of enabling: (i) the exchange of information between independently
created programs and other programs (including this one) and (ii) the mutual
use of the information which has been exchanged, should contact: IBM UK
Limited, Department 88013, 4NW, 76/78 Upper Ground, London, SE1 9PZ,
England. Such information may be available, subject to appropriate terms and
conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material
available for it are provided by IBM under terms of the IBM Customer
Agreement, IBM International Programming License Agreement, or any
equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of
those products, their published announcements or other publicly available
sources. IBM has not tested those products and cannot confirm the accuracy
of performance, compatibility or any other claims related to non-IBM
products. Questions on the capabilities of non-IBM products should be
addressed to the suppliers of those products.

COPYRIGHT LICENSE: This information contains sample application
programs in source language, which illustrate programming techniques on
various operating platforms. You may copy, modify, and distribute these
sample programs in any form without payment to IBM, for the purposes of
developing, using, marketing or distributing application programs conforming
to the application programming interface for the operating platform for which
the sample programs are written. These examples have not been thoroughly
tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

For country-specific notes on the use of WebSphere Voice Response, refer to the README file located in the directory /usr/lpp/dirTalk/homologation. The file name is in the format README_homologation.xxxx, where xxxx is the country/region identifier.

**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at http://www.ibm.com/legal/copytrade.shtml.

Adobe, is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks of Oracle and/or its affiliates.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Glossary

The following terms and abbreviations are defined as they are used in the context of WebSphere Voice Response Voice over IP. If you do not find the term or abbreviation you are looking for, see IBM Dictionary of Computing, McGraw-Hill, 1994, or the AIX Version 4.3: Glossary.

A

action  See state table action

application
  A (usually) customer-written program or set of programs which might comprise one or more state tables or custom servers running on WebSphere Voice Response, along with associated voice segments.

B

bandwidth
  A measure of the capacity of a communication transport medium to convey data.

C

call transfer
  A series of actions that directs a call to another telephone number.

cas  See channel associated signaling

channel associated signaling (cas)
  A method of communicating telephony supervisory or line signaling (on-hook and off-hook) and address signaling on T1 and E1 digital links. The signaling information for each traffic (voice) channel is transmitted in a signaling channel permanently associated with the traffic channel. On T1 links, supervisory signaling is sent in the traffic channel using robbed-bit signaling (RBS). On E1 links, a separate channel is used to send signaling. Address signaling can be transmitted either in the signaling channel (out-of-band) or in the traffic channel (in-band).

Call Control eXtensible Markup Language (CCXML)
  Language designed to provide telephony call control support for VoiceXML or other dialog systems. Refer to the CCXML forum web site at http://www.w3.org/TR/ccxml

codec  Pertaining to adapters that compress and decompress video files. The letters "codec" represent "compression/decompression"; in the past, they represented "coder/decoder."

D

DNSSRV
  A variation of DNS (Domain Name System) for identifying the location of specific SERVICES (for example, a SIP Proxy capable of supporting SIP signaling using TCP/IP) within a network.

Digital Signal Processor (DSP)
  A specialized processor for computing the algorithms and procedures used to process digital electronic signals.
Digital Trunk Extended Adapter (DTXA)
The IBM ARTIC960Rx4 Quad Digital Trunk PCI Adapter. In WebSphere Voice Response, this adapter is known as a DTXA. It allows you to connect directly to the telephony network a pSeries computer that has a PCI bus; it does not require an external pack. The DTXA that is not supported by WebSphere Voice Response Version 6.1.

Digital Trunk Ethernet Adapter (DTEA)
A Radysis adapter card that provides the audio streaming (RTP) interface between the WebSphere Voice Response internal H.100 bus and Ethernet for a maximum of 120 channels using uncompressed (G.711) voice, and compressed G.723.2 and G.729A compressed voice.

Digital Trunk No Adapter (DTNA)
A device driver that supports uncompressed (G.711) voice RTP streaming.

DTMF
See dual-tone multifrequency (DTMF) The signals sent by pressing one of the telephone keys. Each signal is composed of two different tones.

dual-tone multifrequency (DTMF)
The signals sent by pressing one of the telephone keys. Each signal is composed of two different tones.

E1
A digital trunking facility standard used in Europe and elsewhere, capable of transmitting and receiving 30 digitized voice or data channels. Two additional channels are used for synchronization, framing, and signaling. The transmission rate is 2048 kilobits per second. Contrast with T1.

E164
A Telecommunication Standardization Sector (ITU-T) recommendation for the format of international dialing prefixes, which usually begin with a + character and can include up to 15 digits.

endpoint
A place where calls are originated and terminated.

error message
Any message displayed by WebSphere Voice Response in the System Monitor as an alarm and optionally written to the WebSphere Voice Response error log, or to the AIX error log (as an alert). Strictly speaking, the term error message should include only red (immediate attention) and yellow (problem situation) messages but it is also used to refer to green (a red or yellow message has been cleared) and white (informational) messages.

Ethernet
A 10/100 network connection between the VoIP gateway and the Speech Server that supports VoIP.
Fax over IP (FoIP)
The sending of Fax data over IP networks.

G.711
Specification for uncompressed voice for PSTN and Voice over IP access.

G.723.1
Compressed audio codecs used on Voice over IP connection for voice.

G.729A
Compressed audio codecs used on Voice over IP connection for voice.

gateway
A component of a Voice over IP network. A gateway provides a bridge between VoIP and circuit-switched environments.

Integrated Services Digital Network (ISDN)
A digital end-to-end telecommunication network that supports multiple services including, but not limited to, voice and data.

Internet Protocol (IP)
In the Internet suite of protocols, a connectionless protocol that routes data through a network or interconnected networks and acts as an intermediary between the higher protocol layers and the physical network.

IP address
The unique 32-bit (IPv4) or 64-bit (IPv6) address that specifies the location of each device or workstation on the Internet. IPv4 addresses are usually written in dot-decimal notation, for example, 9.67.97.103. IPv6 addresses are usually written as eight groups of four hexadecimal digits, with each group separated by a colon. URIs that include numeric IPv6 format addresses, must have the numeric part within [ ] brackets, for example:

rtsp://[2002:0914:fc12:195:0:0:0:8a2e]:7354/media/recognition

This applies to all protocols.

ISDN
See Integrated Services Digital Network (ISDN).

LAN
See local area network.

local area network (LAN)
A network in which computers are connected to one another within a limited geographical area.

packet
A sequence of binary digits, including data and control signals, that is transmitted and switched as a composite whole.

private automatic branch exchange (PABX)
An automatic private switching system that services an organization and is usually located on a customer's premises. Often used as another name for private branch exchange (PBX).

private branch exchange (PBX)
A switch inside a private business that concentrates the number of inside lines into a smaller number of outside lines (trunks). Many PBXs also provide advanced voice and data communication features. Often used as another name for private automatic branch exchange.
PSTN  An ITU-T abbreviation for public switched telephone network.

Pulse Code Modulation (PCM)  Variation of a digital signal to represent information.

Quality of Service  
For an asynchronous transfer mode (ATM) virtual channel or a Networking BroadBand Services (NBBS) network connection, a set of communication characteristics such as end-to-end delay, jitter, and packet loss ratio.

Real-time Transport Protocol (RTP)  
A protocol that provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery.

Session Initiation Protocol  
A signaling protocol used for internet conferencing, telephony, presence, events notification and instant messaging.

signaling  
The exchange of control information between functional parts of the system in a telecommunications network.

signaling process  
A WebSphere Voice Response component that controls signaling for an exchange data link or common-channel signaling protocol. Some signaling processes are supplied with WebSphere Voice Response, and others can be custom-written.

signaling System Number 7 (SS7)  
The international high-speed signaling backbone used for the public-switched telephone network.

silence  
A short pause between utterances.

state table  
A list of all the actions used in a particular voice application. A component of WebSphere Voice Response.

state table action  
One instruction in a set of instructions contained in a WebSphere Voice Response state table that controls how WebSphere Voice Response processes various operations such as playing voice prompts or recording voice messages.

T1  A digital trunking facility standard used in the United States and elsewhere, capable of transmitting and receiving 24 digitized voice or data channels. Signaling can be
imbedded in the voice channel transmission when robbed-bit signaling is used. The transmission rate is 1544 kilobits per second. Contrast with TCP/IP

TCP/IP
See Transmission Control Protocol/Internet Protocol

TDM
See time-division multiplex bus

time-division multiplex bus
A method of transmitting many channels of data over a smaller number of physical connections by multiplexing the data into timeslots, and demultiplexing at the receiving end. In this document one such channel can be considered to be a half-duplex unidirectional stream of 64 kilobits per second.

Transmission Control Protocol/Internet Protocol (TCP/IP)
A communication subsystem that is used to establish local area and wide area networks.

trombone
A connected voice path which enters an IVR from a switch on one circuit, then returns to the same switch on a parallel circuit. Two IVR ports and two circuits are consumed, but in some circumstances this might be the only way to make a connection between two callers if the attached switch does not support a Call Transfer function. Also known as double-trunking.

trunk
A telephone connection between two central offices or switching devices. In WebSphere Voice Response, a trunk refers to 24 or 30 channels carried on the same T1 or E1 digital interface.

User Agent
A SIP endpoint

User Agent Client (UAC)
The SIP endpoint that initiates call setup (the caller)

User Agent Server (UAS)
The SIP endpoint that receives a call setup request the called party

User Datagram Protocol (UDP)
In the Internet suite of protocols, a protocol that provides unreliable, connectionless datagram service. It enables an application program on one machine or process to send a datagram to an application program on another machine or process. UDP uses the Internet Protocol (IP) to deliver datagrams.

Universal Resource Identifier (URI)
Means to access a resource on the Internet and is a more up-to-date alternative to URL. URIs that include numeric IPv6 format addresses, must have the numeric part within [ ] brackets, for example:


This applies to all protocols.

Voice over Internet Protocol (VoIP)
Sending telephony voice over Internet Protocol (IP) data connections rather than existing dedicated voice networks, switching and transmission equipment.

VoiceXML
VoiceXtensible Markup Language. An XML-based markup language for creating distributed voice applications. Refer to the VoiceXML forum web site at www.voicexml.org
VoIP  See *Voice over Internet Protocol*
List of WebSphere Voice Response and associated documentation

Here is a list of the documentation for WebSphere Voice Response for AIX and associated products. PDF and HTML versions of the documentation are available from the IBM Publications Center at http://www.ibm.com/shop/publications/order. Hardcopy books, where available, can be ordered through your IBM representative or at this Web site.

WebSphere Voice Response for AIX documentation can also be found by going to the IBM Pervasive software Web site at http://www.ibm.com/software/pervasive, selecting the WebSphere Voice products link, and then selecting the library link from the WebSphere Voice Response page.

PDF and HTML versions of the WebSphere Voice Response for AIX publications are available on the CD-ROM supplied with the product. In addition, WebSphere Voice Response for AIX, WebSphere Voice Response for Windows, Unified Messaging, and other WebSphere Voice publications are available together in PDF and HTML formats on a separately-orderable CD-ROM (order number SK2T-1787).

Note: To read PDF versions of books you need to have the Adobe Acrobat Reader (it can also be installed as a plug-in to a Web browser). It is available from Adobe Systems at http://www.adobe.com.

WebSphere Voice Response software

- WebSphere Voice Response for AIX: General Information and Planning, GC34-7084
- WebSphere Voice Response for AIX: Installation, GC34-7095
- WebSphere Voice Response for AIX: User Interface Guide, SC34-7091
- WebSphere Voice Response for AIX: Configuring the System, SC34-7078
- WebSphere Voice Response for AIX: Managing and Monitoring the System, SC34-7085
- WebSphere Voice Response for AIX: Designing and Managing State Table Applications, SC34-7081
- WebSphere Voice Response for AIX: Application Development using State Tables, SC34-7076
- WebSphere Voice Response for AIX: Developing Java applications, GC34-7082
IBM hardware for use with WebSphere Voice Response

- IBM Quad Digital Trunk Telephony PCI Adapter (DTTA): Installation and User’s Guide, part number 00P3119 (DTTA card)

WebSphere Voice Response related products

WebSphere Voice Server

The documentation for Version 5.1 of WebSphere Voice Server is provided in the form of an HTML-based information center, and can be found at:


Unified Messaging for WebSphere Voice Response

- Unified Messaging: General Information and Planning, GC34-6398
- Unified Messaging: Subscriber’s Guide (Types 0, 1, 2, 3, 4 and 9), SC34-6403
- Unified Messaging: Subscriber’s Guide (Types 5, 6, 7 and 8), SC34-6400
- Unified Messaging: Administrator’s Guide, SC34-6399
- Unified Messaging: Voice Interface, GC34-6401
- Unified Messaging: Web Services Voicemail API, SC34-6975

Unified Messaging publications can be found by going to the IBM Pervasive software Web site at http://www.ibm.com/software/pervasive, selecting the products link, and then selecting the library link from the Unified Messaging page.
AIX and the IBM pSeries computer

For information on AIX Version 6.1, refer to the AIX V6.1 infocenter.

For information on System p5 and BladeCenter computers, refer to the IBM Power hardware infocenter.

HACMP

- HACMP for AIX: HACMP 5.4 Concepts and Facilities, SC23-4864-09
- HACMP for AIX: HACMP 5.4 Planning Guide, SC23-4861-09
- HACMP for AIX: HACMP 5.4 Installation Guide, SC23-5209-00
- HACMP for AIX: HACMP 5.4 Administration Guide, SC23-4862-09
- HACMP for AIX: HACMP 5.4 Smart Assist for DB2, SC23-5179-03
- HACMP for AIX: HACMP 5.4 Troubleshooting, SC23-5177-03
- HACMP for AIX: Enhanced Scalability Installation and Administration Guide, Volume 1, SC23-4284

For more information on HACMP, refer to the HACMP Library and the AIX V6.1 infocenter.

SS7


IBM SS7 Support for WebSphere Voice Response observes the applicable parts of the following specifications for ISUP:

- ITU-T (formerly CCITT) Recommendations Q.700 - Q.716, Volume VI Fascicle VI.7
- ITU-T (formerly CCITT) Recommendations Q.721 - Q.725, Volume VI Fascicle VI.8
- ITU-T (formerly CCITT) Recommendations Q.771 - Q.775, Q.791, Volume VI Fascicle VI.9

ADC

Integrated Services Digital Network

WebSphere Voice Response ISDN support observes the applicable parts of the following standards for User Side protocol:

Custom ISDN Standards:
- Northern Telecom DMS/250 Primary Rate Interface NIS A211-4 Release 8, July 1995. (IEC05 level)
- Northern Telecom DMS/100 Primary Rate Interface NIS A211-1 Release 7.05, May 1998. (NA007 & RLT)
- AT&T 5ESS Switch. ISDN Primary Rate Interface Specification. 5E7 and 5E8 Software Release AT&T 235-900-332. Issue 2.00 December 1991
- AT&T 5ESS Switch. ISDN Primary Rate Interface Specification. 5E9 Software Release AT&T 235-900-342. Issue 1.00 November 1993 (National ISDN only)
- Lucent 5ESS-2000 Switch ISDN Primary Rate Interface, Interface Specification, 5E9(2) and Later Software Releases, 235-900-342. Issue 5.00 January 1997 (National ISDN only)
- AT&T ISDN Primary Rate Specification TR41449 July 1989
- AT&T ISDN Primary Rate Specification TR41459 August 1996

Euro-ISDN

The following documents refer to the specifications required for observing ISDN:
- TBR4-ISDN; Attachment Requirements For Terminal Equipment To Connect To An ISDN Using ISDN Primary Rate Access, Edition 1, Nov. 95, English
- CTR 4 - European Communities Commission Decision 94/796/EC published in the Official Journal of the European Communities L 329, 20 December 94 (ISDN PRA)

National ISDN

National ISDN is described in the following publications:
- National ISDN, SR-NWT-002006, Issue 1, August 1991, published by Bellcore

INS Net Service 1500

INS Net Service is described in the following publications:
• Interface for the INS Net Service Volume 1 (Outline), 7th Edition, published by Nippon Telegraph and Telephone Corporation
• Interface for the INS Net Service Volume 3 (Layer 3 Circuit Switching), 5th Edition, published by Nippon Telegraph and Telephone Corporation

**Bellcore Specifications for ADSI Telephones**

The following Bellcore specification documents contain technical details of the requirements for ADSI telephones, and the interface to voice response systems such as WebSphere Voice Response:

- SR-INS-002461: CustomerPremises Equipment Compatibility Considerations for the Analog Display Services Interface
- TR-NWT-001273: Generic Requirements for an SPCS to Customer Premises Equipment Data Interface for Analog Display Services
Index

A
access control
   introduction 60
   accessibility x
activating channels 46
adding CA certificates 82
adding personal certificates PKCS #12 file 83
adding trusted certificates to keyring.db 78
allowed host list, configuring an 31
asymmetric cryptography algorithm 60
attended transfer
   secure SIP 91
attended transfer using a call reference 120
attended transfer, VoIP SIP 114
authentication
   digital signature 76
   introduction 60
   SSL 72

B
blind transfer
   secure SIP 91
blind transfer, VoIP SIP 114
block cipher algorithm 60

C
CA 63
CA certificates
   adding 82
call management 6
call reference, using for attended transfer 120
call transfer 114
call transfer considerations
   secure SIP 91
capacity planning for DTNA 10
certificate
   chain 67
Certificate Authority
   digital certificates 63
   introduction 65
   public key infrastructure (PKI) 68
certificate commands 86
certificate request commands 87
certificates
   configuring 79
   creating 80
   extracting self-signed 81
   certificates and certificate requests
      reviewing current 84
   certificates, personal
      receiving 82
   certification path 67
   channel group 40
   channels 10
   cipher algorithm
      block 60
      stream 60
   cipher strength 60
   CipherSpec
      introduction 75
cipherspecs.ini
   configuring 88
   CipherSuite
      introduction 75
ciphertext 60
codecs 50
commands
   certificate 86
   certificate request 87
   database 87
   confidentiality
   cryptography 60
   introduction 60
   SSL 72
   configuration settings for register
      secure SIP 87
   configuring 31
      allowed host list 31
      pack configuration 36
      SIP registrar 32
      system configuration 37
   configuring certificates 79
   configuring cipherspecs.ini 88
   configuring secure SIP 77
   configuring the VOIP_Call_Transfer custom server 117
   configuring the WebSphere Voice Response server
   certificate 79
   connecting to a Registrar 87
   considerations, NAT/Firewall 53
   contacts settings 88
   country/region 36
   creating a certificate 80
   creating certificate requests
      problems 82
   creating certificates 80
      problems 81
   creating the keyring.db database 78
cryptography
algorithm 60
introduction 60
current certificate requests
reviewing 84
current certificates
reviewing 84
custom server 29
custom server, VOIP_Call_Transfer 116
D
data integrity
cryptography 60
message digests 62
SSL 72
database commands 87
debugging VoIP/SIP, useful tools for 48
decipherment 60
decryption 60
definitions, State table 123
device drivers 24
diagnosing DTNA problems 47
digital certificate
Certificate Authority 65
certificate chain 67
certificate 64
Distinguished Name (DN) 65
introduction 63
public key infrastructure (PKI) 68
SSL authentication 72
digital signature
introduction 76
SSL integrity 72
Distinguished Name (DN)
introduction 65
DN 65
DTEA 24
DTMF handling with DTNA 52
DTNA problems, diagnosing 47
DTNA, capacity planning 10
DTNA, DTMF handling with 52
DTNA, performance implications 52
DTNA, tromboning with 53
dtuser 28
E
eavesdropping 60
enabling secure SIP 85
encipherment 60
encryption
introduction 60
SSL confidentiality 72
endpoint 5
ensuring other SIP entities can communicate with 84
ensuring other SIP entities can communicate with
WebSphere Voice Response 84
extracting self-signed certificates 81
F
file permissions 28
frequently asked questions 56
functions, VOIP_Call_Transfer custom server 122
G
gateway 6
H
handshake, SSL 70
hash function
 overview 62
headers
 outband arbitrary SIP 18
I
identification
introduction 60
impersonation 72
importing certificates
problems 84
importing the VOIP_Call_Transfer custom server 116
incoming calls
 secure SIP configuration settings 91, 92
installing 23
software 23, 27
K
key 60
key distribution problem
 symmetric cryptography 60
keyring.db
 adding trusted certificates 78
keyring.db database, creating 78
L
line signaling 40
M
MAC 62
man in the middle attack
 introduction 63
Message Authentication Code (MAC)
 introduction 62
 part of CipherSuite 75
message context
 introduction 60
message digest 62, 76
Message Waiting Indicator (MWI), SIP support of 128
mutual authentication
 definition 60