Asterisk V1.6.1.1 OSP Module User Guide

July 15, 2009

Revision History

Revision	Date of Issue	Description
1.2	6 Jul 2005	OSP Module User Guide for Asterisk V1.2
1.4	16 Jun 2006	OSP Module User Guide for Asterisk V1.4
1.6.0	13 Dec 2006	OSP Module User Guide for Asterisk V1.6
1.6.1	4 Jan 2007	Clarifying edits, add revision history, add general purpose extensions.conf example
1.6.2	9 Feb 2007	Replace OSP Toolkit site from SIPfoundry with SourceForge
1.6.3	21 Oct 2008	Minor changes
1.6.4	21 Nov 2008	Add H.323 module built instructions
1.6.4.1	22 June 2009	Update for Asterisk 1.6.1.1
1.6.4.2	15 July 2009	Change location for OpenOSP.

1 Introduction

This document provides instructions on how to build and configure Asterisk V1.6.1.1 with OSP Toolkit to enable secure, multi-lateral peering. OSP Toolkit is an open source implementation of the OSP peering protocol and is freely available from https://sourceforge.net/projects/osp-toolkit. The OSP standard defined by the European Telecommunications Standards Institute (ETSI TS 101 321) http://www.etsi.org. Additional information on the OSP protocol is available at http://en.wikipedia.org/wiki/Open_Settlement_Protocol. If you have questions or need help to build Asterisk with OSP Toolkit, please post your question on the OSP mailing list at https://lists.sourceforge.net/lists/listinfo/osp-toolkit-client.

2 OSP Toolkit

Please reference the OSP Toolkit document "How to Build and Test the OSP Toolkit" available from https://sourceforge.net/projects/osp-toolkit.

2.1 Build OSP Toolkit

The software listed below is required to build and use OSP Toolkit:

- **OpenSSL** (required for building) - Open Source SSL protocol and Cryptographic Algorithms (version 0.9.8i recommended) from http://www.openssl.org. Pre-compiled OpenSSL binary packages are not recommended because of the binary compatibility issue.
- **Perl** (required for building) - A programming language used by OpenSSL for compilation. Any version of Perl should work. One version of Perl is available from http://www.activestate.com/activeperl. If pre-compiled OpenSSL packages are used, Perl package is not required.
- **C compiler** (required for building) - Any C compiler should work. The GNU Compiler Collection from http://www.gnu.org is routinely used for building OSP Toolkit for testing.
- **OSP Server** (required for testing) - Access to any OSP server should work. An open source reference OSP server developed by Cisco System is available at https://sourceforge.net/projects/openosp. RAMS, a java based open source OSP server is available at https://sourceforge.net/projects/rams. A free version of the TransNexus commercial OSP server may be downloaded from http://www.transnexus.com/OSP%20Toolkit/Peering_Server/VoIP_Peering_Server.htm.

2.1.1 Unpacking OSP Toolkit

After downloading OSP Toolkit (version 3.4.2, Asterisk 1.6.1.1 does not support OSP Toolkit 3.5.0 and later versions) from https://sourceforge.net/projects/osp-toolkit, perform the following steps in order:

1) Copy the OSP Toolkit distribution into the directory where it will reside, say ***/usr/src***.
2) Un-package the distribution file by executing the following command:

```
gunzip –c OSPToolkit-###.tar.gz | tar xvf –
```

Where ### is the version number separated by underlines. For example, if the version is 3.4.2, then the above command would be:

```
gunzip –c OSPToolkit-3_4_2.tar.gz | tar xvf –
```

A new directory (***TK-3_4_2-20071227***) will be created within the same directory as the tar file.

3) Go to the ***TK-3_4_2-20071227*** directory by running this command:

```
cd TK-3_4_2-20071227
```

Within this directory, you will find directories and files similar to what is listed below if the command "ls -F" is executed):

```
ls –F
bin/  crypto/  enroll/  include/  lib/  LICENSE.txt  README.txt  RELNOTES.txt  src/
test/
```

2.1.2 Preparing to Build

4) Compile OpenSSL according to the instructions provided with the OpenSSL distribution (You would need to do this only if you don't have openssl already).
5) Copy the OpenSSL header files (the *.h files) into the OSP Toolkit ***crypto/openssl*** directory. The OpenSSL header files are located under the ***openssl/include/openssl*** directory.
6) Copy the OpenSSL library files (libcrypto.a and libssl.a) into the OSP Toolkit ***lib*** directory. The OpenSSL library files are located under the openssl directory.
   Note: Since the Asterisk requires the OpenSSL package. If the OpenSSL package has been installed, steps 4~6 are not necessary.
7) Optionally, change the install directory of OSP Toolkit. Open Makefile in the OSP Toolkit ***src*** directory, look for the install path variable – **INSTALL_PATH**, and edit it to be anywhere you want (defaults ***/usr/local***).
   Note: Please change the install path variable only if you are familiar with both OSP Toolkit and Asterisk. Otherwise, it may case Asterisk does not support the OSP protocol.

2.1.3 Building OSP Toolkit

8) From within the OSP Toolkit directory (***/usr/src/TK-3_4_2-20071227***), start the compilation script by executing the following commands:

```
cd src
make clean; make build
```

2.1.4 Installing OSP Toolkit

The header files and the library of OSP Toolkit should be installed. Otherwise, you must specify the OSP Toolkit path for Asterisk.

9) Use the make script to install OSP Toolkit.

```
make install
```

The make script is also used to install the OSP Toolkit header files and the library into the **INSTALL_PATH** directory specified in Makefile.

Note:

- Please make sure you have the rights to access the **INSTALL_PATH** directory. For example, in order to access ***/usr/local*** directory, root privileges are required.
- By default, OSP Toolkit is compiled in the production mode. The following table identifies which default features are activated with each compile option:

Default Feature	Production	Development
Debug Information Displayed	No	Yes

The "Development" option is recommended for a first time build. The **CFLAGS** definition in Makefile must be modified to build in development mode.

2.1.5 Building Enrollment Utility

Device enrollment is the process of establishing a trusted cryptographic relationship between the VoIP device and the OSP Server. The Enroll program is a utility application for establishing a trusted relationship between an OSP client and an OSP server. Please see the document "Device Enrollment" at http://www.transnexus.com/OSP%20Toolkit/OSP%20Toolkit%20Documents/Device_Enrollment.pdf for more information about the enroll application.

10) From within the OSP Toolkit directory (***/usr/src/TK-3_4_2-20071227***), execute the following commands at the command prompt:

```
cd enroll
make clean; make linux
```

Compilation is successful if there is not any error in the compiler output. The enroll program is now located in the OSP Toolkit ***bin*** directory.

2.2 Obtain Crypto Files

The OSP module in Asterisk requires three crypto files containing a local certificate (localcert.pem), private key (pkey.pem), and CA certificate (cacert_0.pem). Asterisk will try to load the files from the Asterisk public/private key directory - ***/var/lib/asterisk/keys***. If the files are not present, the OSP module will not start and Asterisk will not support the OSP protocol. Use the enroll.sh script from the toolkit distribution to enroll Asterisk with an OSP server and obtain the crypto files. Documentation explaining how to use the enroll.sh script (Device Enrollment) to enroll with an OSP server is available at http://www.transnexus.com/OSP%20Toolkit/OSP%20Toolkit%20Documents/Device_Enrollment.pdf. Copy the files generated by the enrollment process to ***/var/lib/asterisk/keys*** directory.

Note: The OSP test server, osptestserver.transnexus.com, is configured only for sending and receiving non-SSL messages, and issuing signed tokens. If you need help, post a message on the OSP mailing list at https://lists.sourceforge.net/lists/listinfo/osp-toolkit-client.

The enroll.sh script takes the domain name or IP addresses of the OSP servers that OSP Toolkit needs to enroll with as arguments, and then generates pem files – cacert_#.pem, certreq.pem, localcert.pem, and pkey.pem. The '#' in the cacert file name is used to differentiate the ca certificate file names for the various SP's (OSP servers). If only one address is provided at the command line, cacert_0.pem will be generated. If 2 addresses are provided at the command line,

2 files will be generated – cacert_0.pem and cacert_1.pem, one for each SP (OSP server). The example below shows the usage when the client is registering with osptestserver.transnexus.com. . If all goes well, the following text will be displayed. The gray boxes indicate required input.

```
./enroll.sh osptestserver.transnexus.com
Generating a 512 bit RSA private key
.......................++++++++++++
.........++++++++++++
writing new private key to 'pkey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

Error Code returned from openssl command : 0

CA certificate received
[SP: osptestserver.transnexus.com]Error Code returned from getcacert command : 0

output buffer after operation: operation=request
output buffer after nonce: operation=request&nonce=1655976791184458
X509 CertInfo context is null pointer
Unable to get Local Certificate
depth=0 /CN=osptestserver.transnexus.com/O=OSPServer
verify error:num=18:self signed certificate
verify return:1
depth=0 /CN=osptestserver.transnexus.com/O=OSPServer
verify return:1
The certificate request was successful.
Error Code returned from localcert command : 0
```

The files generated should be copied to the ***/var/lib/asterisk/keys*** directory.
Note: The script enroll.sh requires AT&T korn shell (ksh) or any of its compatible variants. The ***/usr/src/TK-3_4_2-20071227/bin*** directory should be in the **PATH** variable. Otherwise, enroll.sh cannot find the enroll file.

3 Asterisk

In Asterisk, all OSP support is implemented as dial plan functions. The detailed descriptions of the OSP dial plan functions are listed in section 4. In Asterisk V1.6.1.1, all combinations of routing between OSP and non-OSP enabled networks using any combination of SIP, H.323 and IAX protocols are fully supported. Section 3.1 describes the steps to add OSP support to Asterisk:

1) Build Asterisk with OSP Toolkit
2) Configure modules.conf file
3) Configure osp.conf file
4) Configure extensions.conf
5) Configure channels

Some configuration examples are provided in section 3.2. The detailed information provided in sections 3.2 is not required for operating Asterisk with OSP, but may be helpful to developers who want to customize their Asterisk OSP implementation.

3.1 Build Asterisk

We assume the audiences are familiar with the Asterisk installation. We only briefly discuss the installation steps.

3.1.1 Build with OSP Toolkit

1) Obtain Asterisk source code. There are two ways to obtain Asterisk source code. The stable released version can be downloaded from http://www.asterisk.org/downloads. The branch version can be downloaded by the following command,

```
svn co http://svn.digium.com/svn/asterisk/tags/1.6.1.1 asterisk-1.6.1.1
```

2) Build all necessary packages. The instructions to build OSP Toolkit are listed in section 2.1. pwlib and openh323 are used for Asterisk H.323 channel module. The instructions to build pwlib and openh323 libraries are listed in section 5.1.
3) Apply H.323 channel OSP patch. When this documentation is processing, the H.323 OSP patch has not been committed into Asterisk yet. The instructions to apply the patch are listed in section 5.2.
4) Optionally change SIP channel connection timeout. The timeout value can be changed from 32s to 2s by modifying **MAX_RETRNAS** from 6 to 1 in chan_sip.c.
5) Configure Asterisk compiling environment. Configure must be run before compiling Asterisk. If OSP Toolkit is installed in the default install directory, ***/usr/local***, no additional option is required.

```
./configure
```

If OSP Toolkit is installed in other directory, say ***/myosp***, Asterisk must be configured with the location of OSP Toolkit.

```
./configure --with-osptk=/myosp
```

Note: Please change the install path only if you familiar with both OSP Toolkit and Asterisk. Otherwise, the change may result in Asterisk not supporting the OSP protocol.

6) Cleanup the directory

```
make clean
```

7) Select Asterisk modules. In Application page, app_osplookup should be selected by default. If it cannot be select, please check if OSP Toolkit has been installed properly. Normally, chan_sip, chan_h323 and chan_iax2 channel modules should be selected too.

```
make menuselect
```

8) Compile and install Asterisk

```
make install
```

9) Install Asterisk default configuration files

```
make samples
```

Now, Asterisk has been installed. Some configuration files should be changed for the running environment.

3.1.2 modules.conf

The /etc/asterisk/modules.conf file contains the modules will be loaded.

```
;
; Asterisk configuration file
;
; Module Loader configuration file
;

[modules]
autoload=no

; Resources
load => res_adsi.so
load => res_smdi.so

; PBX
load => pbx_config.so

; Functions
load => func_callerid.so
load => func_cut.so
load => func_logic.so
load => func_channel.so

; Channels
load => chan_sip.so
;load => chan_h323.so
;load => chan_ooh323.so
;load => chan_iax2.so
```

```
; Codecs
load => codec_alaw.so
load => codec_ulaw.so
load => codec_gsm.so

; Formats
load => format_gsm.so
load => format_pcm.so
load => format_wav_gsm.so
load => format_wav.so

; Applications
load => app_dial.so
load => app_macro.so
load => app_osplookup.so
```

3.1.3 osp.conf

The /etc/asterisk/osp.conf file contains configuration parameters for using OSP. Two parameters, `servicepoint` and `source` must be configured. The default values for all other parameters will work well for standard OSP implementations.

```
;
; Open Settlement Protocol Sample Configuration File
;
; This file contains configuration of OSP server providers that are used by the
; Asterisk OSP module.  The section "general" is reserved for global options.
; All other sections describe specific OSP Providers.  The provider "default"
; is used when no provider is otherwise specified.
;
; The "servicepoint" and "source" parameters must be configured.  For most
; implementations the other parameters in this file can be left unchanged.
;
[general]
;
; Enable cryptographic acceleration hardware.
; The default value is no.
;
accelerate=no
;
; Defines the status of tokens that Asterisk will validate.
; 0 - signed tokens only
; 1 - unsigned tokens only
; 2 - both signed and unsigned
; The default value is 0, i.e. the Asterisk will only validate signed tokens.
;
tokenformat=0
;
[default]
;
; List all service points (OSP servers) for this provider.
; Use either domain name or IP address.  Most OSP servers use port 1080.
;
;servicepoint=http://osptestserver.transnexus.com:1080/osp
```

```
servicepoint=http://[127.0.0.1]:1080/osp
;
; Define the "source" device for requesting OSP authorization.
; This value is usually the domain name or IP address of the the Asterisk server.
;
;source=domain name or [IP address in brackets]
source=[127.0.0.1]
;
; Define path and file name of crypto files.
; The default path for crypto file is /var/lib/asterisk/keys.  If no path is
; defined, crypto files will in /var/lib/asterisk/keys directory.
;
; Specify the private key file name.
; If this parameter is unspecified or not present, the default name will be the
; osp.conf section name followed by "-privatekey.pem" (for example:
; default-privatekey.pem)
;
privatekey=pkey.pem
;
; Specify the local certificate file.
; If this parameter is unspecified or not present, the default name will be the
; osp.conf section name followed by "- localcert.pem " (for example:
; default-localcert.pem)
;
localcert=localcert.pem
;
; Specify one or more Certificate Authority key file names.  If none are listed,
; a single Certificate Authority key file name is added with the default name of
; the osp.conf section name followed by "-cacert_0.pem " (for example:
; default-cacert_0.pem)
;
cacert=cacert_0.pem
;
; Configure parameters for OSP communication between Asterisk OSP client and OSP
; servers.
;
; maxconnections: Max number of simultaneous connections to the provider OSP
;                 server (default=20)
; retrydelay:     Extra delay between retries (default=0)
; retrylimit:     Max number of retries before giving up (default=2)
; timeout:        Timeout for response in milliseconds (default=500)
;
maxconnections=20
retrydelay=0
retrylimit=2
timeout=500
;
; Set the authentication policy.
; 0 - NO        - Accept all calls.
; 1 - YES       - Accept calls with valid token or no token. Block calls with
;                 invalid token.
; 2 - EXCLUSIVE - Accept calls with valid token. Block calls with invalid token
;                 or no token.
; Default is 1,
;
authpolicy=1
;
```

```
; Set the default destination protocol. The OSP module supports SIP, H323, and
; IAX protocols.  The default protocol is set to SIP.
;
defaultprotocol=SIP
```

3.1.4 extensions.conf

OSP functions are implemented as dial plan functions in the /usr/etc/asterisk/extensions.conf file. To add OSP support to your Asterisk server, simply copy and paste the text box below to your extensions.conf file. These functions will enable your Asterisk server to support all OSP call scenarios.

```
;----------Asterisk/OSP Configuration by Transnexus----------
[GeneralProxy]
exten => _XXXX.,1,NoOp(OSPGeneralProxy)
exten => _XXXX.,n,Macro(inbound)
exten => _XXXX.,n,OSPAuth()
exten => _XXXX.,n,GotoIf($["${OSPAUTHSTATUS}"!="SUCCESS"]?done)
exten => _XXXX.,n,Set(OSPINTIMELIMIT=$[${OSPINTIMELIMIT}])
exten => _XXXX.,n,OSPLookup(${EXTEN})
exten => _XXXX.,n,GotoIf($["${OSPLOOKUPSTATUS}"!="SUCCESS"]?done)
exten => _XXXX.,n(next),Macro(outbound)
exten => _XXXX.,n,Dial(${OSPDIALSTR},60,oL($[${OSPOUTTIMELIMIT}*1000]))
exten => _XXXX.,n,NoOp(${DIALSTATUS})
exten => _XXXX.,n,OSPNext(${HANGUPCAUSE})
exten => _XXXX.,n,GotoIf($["${OSPNEXTSTATUS}"!="SUCCESS"]?done)
exten => _XXXX.,n,Goto(next)
exten => _XXXX.,n(done),Hangup
exten => h,1,NoOp(OSPGeneralProxy-Hangup)
exten => h,n,OSPFinish(${HANGUPCAUSE})

[macro-inbound]
exten => s,1,NoOp(inbound)
exten => s,n,Set(CHANTECH=${CUT(CHANNEL,/,1)})
exten => s,n,GotoIf($["${CHANTECH}"="H323"]?h323)
exten => s,n,GotoIf($["${CHANTECH}"="IAX2"]?iax)
exten => s,n,GotoIf($["${CHANTECH}"="SIP"]?sip)
exten => s,n,Hangup
; H323 -------------------------------------------------------
exten => s,n(h323),Set(OSPPEERIP=${CHANNEL(peerip)})
exten => s,n,Set(OSPINTOKEN=${CHANNEL(osptoken)})
exten => s,n,GoTo(done)
; IAX ---------------------------------------------------------
exten => s,n(iax),Set(OSPPEERIP=${IAXPEER(CURRENTCHANNEL)})
exten => s,n,Set(OSPINTOKEN=${CHANNEL(osptoken)})
exten => s,n,GoTo(done)
; SIP ---------------------------------------------------------
exten => s,n(sip),Set(OSPPEERIP=${CHANNEL(peerip)})
exten => s,n,Set(OSPINTOKEN=${SIP_HEADER(P-OSP-Auth-Token)})
; ------------------------------------------------------------
exten => s,n(done),MacroExit

[macro-outbound]
exten => s,1,NoOp(outbound)
exten => s,n,Set(CALLERID(number)=${OSPCALLING})
```

```
exten => s,n,GotoIf($["${OSPTECH}"="H323"]?h232)
exten => s,n,GotoIf($["${OSPTECH}"="IAX2"]?iax)
exten => s,n,GotoIf($["${OSPTECH}"="SIP"]?sip)
exten => s,n,Hangup
; H323 -------------------------------------------------------
exten => s,n(h232),Set(CHANNEL(callid)=${OSPOUTCALLID})
exten => s,n,Set(CHANNEL(osptoken)=${OSPOUTTOKEN})
exten => s,n,GoTo(done)
; IAX ---------------------------------------------------------
exten => s,n(iax),Set(CHANNEL(osptoken)=${OSPOUTTOKEN})
exten => s,n,GoTo(done)
; SIP ---------------------------------------------------------
exten => s,n(sip),NoOp()
; -------------------------------------------------------------
exten => s,n(done),MacroExit
;----------Asterisk/OSP Configuration by Transnexus----------
```

3.1.5 dahdi/sip/iax/h323/ooh323.conf

There is no configuration required for OSP except to use the GeneralProxy context.

- For SIP channel
  In general context, the default options, "context=default" and "srvlookup=yes", should be commented out and the following statements should be inserted.

```
;----------Asterisk/OSP Configuration by Transnexus----------
context=GeneralProxy        ; General Proxy

allowguest=yes
nat=no
canreinvite=no
bindport=5060
srvlookup=no
trustrpid=yes
sendrpid=yes
disallow=all
allow=alaw
allow=ulaw
allow=gsm
allow=g722
allow=g723
allow=g726
allow=g729
;----------Asterisk/OSP Configuration by Transnexus----------
```

- For H.323 channel
  In general context, the following statements should be inserted.

```
;----------Asterisk/OSP Configuration by Transnexus----------
context=GeneralProxy

disallow=all
allow=alaw
allow=ulaw
allow=gsm
```

```
allow=g722
allow=g723
allow=g726
allow=g729
;----------Asterisk/OSP Configuration by Transnexus----------
```

3.2 extensions.conf Examples

The extensions.conf file example provided in section 3.1.4 is designed to handle the OSP call scenarios when Asterisk is used as a B2BUA between VoIP networks. The extenstion.conf examples in this section are designed for specific use cases.

3.2.1 Source Gateway

The examples in this section apply when the Asterisk server is being used as a TDM to VoIP gateway. Calls originate on the TDM network and are converted to VoIP by Asterisk. In these cases, the Asterisk server queries an OSP server to find a route to a VoIP destination. When the call ends, Asterisk sends a CDR to the OSP server.

- For SIP protocol.

```
[SIPSrcGW]
exten => _XXXX.,1,NoOp(SIPSrcGW)
; Set calling number if necessary
exten => _XXXX.,n,Set(CALLERID(numner)=CallingNumber)
; OSP lookup using default provider
exten => _XXXX.,n(lookup),OSPLookup(${EXTEN})
; If anyting wrong, hangup fails the call
exten => _XXXX.,n,GotoIf($["${OSPLOOKUPSTATUS}"!="SUCCESS"]?done)
; Set calling number which may be translated
exten => _XXXX.,n,Set(CALLERID(number)=${OSPCALLING})
; Dial to destination, 60 timeout, with call duration limit
exten => _XXXX.,n,Dial(${OSPDIALSTR},60,oL($[${OSPOUTTIMELIMIT}*1000]))
; Hangup
exten => _XXXX.,n(done),Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})
```

- For IAX protocol.

```
[IAXSrcGW]
exten => _XXXX.,1,NoOp(IAXSrcGW)
; Set calling number if necessary
exten => _XXXX.,n,Set(CALLERID(numner)=CallingNumber)
; OSP lookup using default provider
exten => _XXXX.,n(lookup),OSPLookup(${EXTEN})
; If anyting wrong, hangup fails the call
exten => _XXXX.,n,GotoIf($["${OSPLOOKUPSTATUS}"!="SUCCESS"]?done)
; Set outbound OSP token
exten => _XXXX.,n,Set(IAXCHANINFO(osptoken)=${OSPOUTTOKEN})
; Set calling number which may be translated
exten => _XXXX.,n,Set(CALLERID(number)=${OSPCALLING})
; Dial to destination, 60 timeout, with call duration limit
```

```
exten => _XXXX.,n,Dial(${OSPDIALSTR},60,oL($[${OSPOUTTIMELIMIT}*1000]))
; Hangup
exten => _XXXX.,n(done),Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})
```

- For H.323 protocol.

```
[H323SrcGW]
exten => _XXXX.,1,NoOp(H323SrcGW)
; Set calling number if necessary
exten => _XXXX.,n,Set(CALLERID(numner)=CallingNumber)
; OSP lookup using default provider
; “h” parameter is used to generate a call id
; Cisco OSP gateways use this call id to validate OSP token
exten => _XXXX.,n(lookup),OSPLookup(${EXTEN},,h)
; If anyting wrong, hangup fails the call
exten => _XXXX.,n,GotoIf($["${OSPLOOKUPSTATUS}"!="SUCCESS"]?done)
; Set outbound call id
exten => _XXXX.,n,Set(OH323CHANINFO(callid)=${OSPOUTCALLID})
; Set outbound OSP token
exten => _XXXX.,n,Set(OH323CHANINFO(osptoken)=${OSPOUTTOKEN})
; Set calling number which may be translated
exten => _XXXX.,n,Set(CALLERID(number)=${OSPCALLING})
; Dial to destination, 60 timeout, with call duration limit
exten => _XXXX.,n,Dial(${OSPDIALSTR},60,oL($[${OSPOUTTIMELIMIT}*1000]))
; Hangup
exten => _XXXX.,n(done),Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})
```

3.2.2 Destination Gateway

The examples in this section apply when Asterisk is being used as a VoIP to TDM gateway. VoIP calls are received by Asterisk which validates the OSP peering token and completes to the TDM network. After the call ends, Asterisk sends a CDR to the OSP server.

- For SIP protocol

```
[SIPDstGW]
exten => _XXXX.,1,NoOp(SIPDstGW)
; Get peer IP
exten => _XXXX.,n,Set(OSPPEERIP=${SIPCHANINFO(peerip)})
; Get OSP token
exten => _XXXX.,n,Set(OSPINTOKEN=${SIP_HEADER(P-OSP-Auth-Token)})
; Validate token using default provider
exten => _XXXX.,n(auth),OSPAuth()
; If anything wrong, hangup fails the call
exten => _XXXX.,n,GotoIf($["${OSPAUTHSTATUS}"!="SUCCESS"]?done)
; Ringing
exten => _XXXX.,n,Ringing
; Dial phone, timeout 15 seconds, with call duration limit
exten => _XXXX.,n,Dial(${DIALOUTANALOG}/${EXTEN:1},15,oL($[${OSPINTIMELIMIT}*1000]))
```

```
; Hangup
exten => _XXXX.,n(done),Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})
```

- For IAX protocol

```
[IAXDstGW]
exten => _XXXX.,1,NoOp(IAXDstGW)
; Get peer IP
exten => _XXXX.,n,Set(OSPPEERIP=${IAXPEER(CURRENTCHANNEL)})
; Get OSP token
exten => _XXXX.,n,Set(OSPINTOKEN=${IAXCHANINFO(osptoken)})
; Validate token using default provider
exten => _XXXX.,n(auth),OSPAuth()
; If anything wrong, hangup fails the call
exten => _XXXX.,n,GotoIf($["${OSPAUTHSTATUS}"!="SUCCESS"]?done)
; Ringing
exten => _XXXX.,n,Ringing
; Dial phone, timeout 15 seconds, with call duration limit
exten => _XXXX.,n,Dial(${DIALOUTANALOG}/${EXTEN:1},15,oL($[${OSPINTIMELIMIT}*1000]))
; Hangup
exten => _XXXX.,n(done),Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})
```

- For H.323 protocol

```
[H323DstGW]
exten => _XXXX.,1,NoOp(H323DstGW)
; Get peer IP
exten => _XXXX.,n,Set(OSPPEERIP=${OH323CHANINFO(peerip)})
; Get OSP token
exten => _XXXX.,n,Set(OSPINTOKEN=${OH323CHANINFO(osptoken)})
; Validate token using default provider
exten => _XXXX.,n(auth),OSPAuth()
; If anything wrong, hangup fails the call
exten => _XXXX.,n,GotoIf($["${OSPAUTHSTATUS}"!="SUCCESS"]?done)
; Ringing
exten => _XXXX.,n,Ringing
; Dial phone, timeout 15 seconds, with call duration limit
exten => _XXXX.,n,Dial(${DIALOUTANALOG}/${EXTEN:1},15,oL($[${OSPINTIMELIMIT}*1000]))
; Hangup
exten => _XXXX.,n(done),Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})
```

3.2.3 Proxy

The example in this section applies when Asterisk is a proxy between two VoIP networks.

```
[GeneralProxy]
exten => _XXXX.,1,NoOp(GeneralProxy)
; Get peer IP and inbound OSP token
; SIP, un-comment the following two lines.
;exten => _XXXX.,n,Set(OSPPEERIP=${SIPCHANINFO(peerip)})
;exten => _XXXX.,n,Set(OSPINTOKEN=${SIP_HEADER(P-OSP-Auth-Token)})
; IAX, un-comment the following 2 lines
;exten => _XXXX.,n,Set(OSPPEERIP=${IAXPEER(CURRENTCHANNEL)})
;exten => _XXXX.,n,Set(OSPINTOKEN=${IAXCHANINFO(osptoken)})
; H323, un-comment the following two lines.
;exten => _XXXX.,n,Set(OSPPEERIP=${OH323CHANINFO(peerip)})
;exten => _XXXX.,n,Set(OSPINTOKEN=${OH323CHANINFO(osptoken)})
;-----------------------------------------------------------------
; Validate token using default provider
exten => _XXXX.,n(auth),OSPAuth()
; If anything wrong, hangup fails the call
exten => _XXXX.,n,GotoIf($["${OSPAUTHSTATUS}"!="SUCCESS"]?done)
; OSP lookup using default provider
; “h” parameter is used to generate a call id for H.323 destinations
; Cisco OSP gateways use this call id to validate OSP token
exten => _XXXX.,n(lookup),OSPLookup(${EXTEN},,h)
; If anything wrong, hangup fails the call
exten => _XXXX.,n,GotoIf($["${OSPLOOKUPSTATUS}"!="SUCCESS"]?done)
; Set outbound call id and OSP token
; IAX, un-comment the following line.
;exten => _XXXX.,n,Set(IAXCHANINFO(osptoken)=${OSPOUTTOKEN})
; H323, un-comment the following two lines.
;exten => _XXXX.,n,Set(OH323CHANINFO(callid)=${OSPOUTCALLID})
;exten => _XXXX.,n,Set(OH323CHANINFO(osptoken)=${OSPOUTTOKEN})
;-----------------------------------------------------------------
; Set calling number which may be translated
exten => _XXXX.,n,Set(CALLERID(number)=${OSPCALLING})
; Dial to destination, 14 timeout, with call duration limit
exten => _XXXX.,n,Dial(${OSPDIALSTR},14,oL($[${OSPOUTTIMELIMIT}*1000]))
; Hangup
exten => _XXXX.,n(done),Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})
```

4 Asterisk OSP Dial Plan Functions

This section provides a description of each OSP dial plan function.

4.1 OSPAuth

OSP token validation function.

Input:

- OSPPEERIP: last hop IP address
- OSPINTOKEN: inbound OSP token
- provider: OSP service provider configured in osp.conf. If it is empty, default provider is used.

Output:

- OSPINHANDLE: inbound OSP transaction handle

- OSPINTIMELIMIT: inbound call duration limit
- OSPAUTHSTATUS: OSPAuth return value. SUCCESS/FAILED/ERROR

4.2 OSPLookup

OSP lookup function.
Input:

- OSPPEERIP: last hop IP address
- OSPINHANDLE: inbound OSP transaction handle
- OSPINTIMELIMIT: inbound call duration limit
- exten: called number
- provider: OSP service provider configured in osp.conf. If it is empty, default provider is used.
- callidtypes: Generate call ID for the outbound call. h: H.323; s: SIP; i: IAX. Only h, H.323, has been implemented.

Output:

- OSPOUTHANDLE: outbound transaction handle
- OSPTECH: outbound protocol
- OSPDEST: outbound destination IP address
- OSPCALLED: outbound called nummber
- OSPCALLING: outbound calling number
- OSPOUTTOKEN: outbound OSP token
- OSPRESULTS: number of remaining destinations
- OSPOUTTIMELIMIT: outbound call duration limit
- OSPOUTCALLIDTYPES: same as input callidtypes
- OSPOUTCALLID: outbound call ID. Only for H.323
- OSPDIALSTR: outbound dial string
- OSPLOOKUPSTATUS: OSPLookup return value. SUCCESS/FAILED/ERROR

4.3 OSPNext

OSP lookup next function.
Input:

- OSPINHANDLE: inbound transaction handle
- OSPOUTHANDLE: outbound transaction handle
- OSPINTIMELIMIT: inbound call duration limit
- OSPOUTCALLIDTYPES: types of call ID generated by Asterisk.
- OSPRESULTS: number of remain destinations
- cause: last destination disconnect cause

Output:

- OSPTECH: outbound protocol
- OSPDEST: outbound destination IP address
- OSPCALLED: outbound called number
- OSPCALLING: outbound calling number
- OSPOUTTOKEN: outbound OSP token
- OSPRESULTS: number of remain destinations

- OSPOUTTIMELIMIT: outbound call duration limit
- OSPOUTCALLID: outbound call ID. Only for H.323
- OSPDIALSTR: outbound dial string
- OSPNEXTSTATUS: OSPLookup return value. SUCCESS/FAILED/ERROR

4.4 OSPFinish

OSP report usage function.
Input:

- OSPINHANDLE: inbound transaction handle
- OSPOUTHANDLE: outbound transaction handle
- OSPAUTHSTATUS: OSPAuth return value
- OSPLOOKUPTSTATUS: OSPLookup return value
- OSPNEXTSTATUS: OSPNext return value
- cause: last destination disconnect cause

Output:

- OSPFINISHSTATUS: OSPLookup return value. SUCCESS/FAILED/ERROR

5 Appendix

5.1 Build pwlib and openh323

5.1.1 Download Source Code

Let's say we use $WORKDIR to store pwlib and openh323 source trees.

```
cd $WORKDIR
cvs -z3 -d:pserver:anonymous@openh323.cvs.sourceforge.net:/cvsroot/openh323 co -P -r
v1_10_0 ptlib_unix
cvs -z3 -d:pserver:anonymous@openh323.cvs.sourceforge.net:/cvsroot/openh323 co -P
pwlib/tools/asnparser
cvs -z3 -d:pserver:anonymous@openh323.cvs.sourceforge.net:/cvsroot/openh323 co -P -r
v1_18_0 openh323
```

5.1.2 Export Environment Variables

These environment variables should be set from profit file.

```
export PWLIBDIR=$WORKDIR/pwlib
export OPENH323DIR=$WORKDIR/openh323
export LD_LIBRARY_PATH=$PWLIBDIR/lib:$OPENH323DIR/lib
export PATH=$PATH:$PWLIBDIR/tools/asnparser/obj_linux_x86_r
```

Note: Since Asterisk H.323 channel module needs pwlib and openh323 libraries, it is necessary to set LD_LIBRARY_PATH with the correct paths. An easy way is to run the profile file.

5.1.3 Build Libraries

- From $PWLIBDIR

```
./configure
make clean
make opt
cd tools/asnparser
make clean
make opt
```

- From $OPENH323DIR

```
./configure --disable-transnexusosp
make clean
make opt
```

Note: There may be a compiler.h issue. The solution is to erase the include statement.

5.2 Apply H.323 OSP Patch

5.2.1 h323_osp.patch

```
Index: channels/h323/ast_h323.cxx
===================================================================
--- channels/h323/ast_h323.cxx  (revision 150382)
+++ channels/h323/ast_h323.cxx  (working copy)
@@ -753,6 +753,14 @@
           if (opts->transfer_capability >= 0) {
                  transfer_capability = opts->transfer_capability;
           }
+          if (!ast_strlen_zero(opts->callId)) {
+                 callIdentifier = OpalGloballyUniqueID(opts->callId);
+          }
+          if (!ast_strlen_zero(opts->ospToken)) {
+                 ospToken.SetSize(OSPTOKENSIZE);
+                 int size = ast_base64decode(ospToken.GetPointer(), opts->ospToken,
OSPTOKENSIZE);
+                 ospToken.SetSize(size);
+          }
    }
    tunnelOptions = opts->tunnelOptions;
 }
@@ -830,6 +838,28 @@

           /* Don't show local username as called party name */
           SetDisplayName(cd->call_dest_e164);
+
+          if (setupPDU.m_h323_uu_pdu.m_h323_message_body.GetTag() ==
H225_H323_UU_PDU_h323_message_body::e_setup) {
+                 const H225_Setup_UUIE& setup =
setupPDU.m_h323_uu_pdu.m_h323_message_body;
+                 if (setup.HasOptionalField(H225_Setup_UUIE::e_tokens)) {
+                        const H225_ArrayOf_ClearToken& clearTokens = setup.m_tokens;
+                        PINDEX tokenCount = clearTokens.GetSize();
+                        for (PINDEX i = 0; i < tokenCount; i++) {
+                               H235_ClearToken& clearToken = clearTokens[i];
+                               if (clearToken.m_tokenOID == OpalOSP::ETSIXMLTokenOID
&&
```

```
+
    clearToken.HasOptionalField(H235_ClearToken::e_nonStandard) &&
+
    clearToken.m_nonStandard.m_nonStandardIdentifier == OpalOSP::ETSIXMLTokenOID)
+                               {
+                                       PBYTEArray token =
clearToken.m_nonStandard.m_data;
+                                       cd->ospToken = (char*)malloc(OSPTOKENSIZE);
+                                       if (cd->ospToken != NULL) {
+                                               ast_base64encode(cd->ospToken,
token.GetPointer(), token.GetSize(), OSPTOKENSIZE - 1);
+                                       }
+                                       break;
+                               }
+                       }
+                }
+          }
    }

    /* Convert complex strings */
@@ -1394,6 +1424,18 @@
           }
    }

+   if (ospToken.GetSize() != 0) {
+          setup.IncludeOptionalField(H225_Setup_UUIE::e_tokens);
+          H225_ArrayOf_ClearToken& clearTokens = setup.m_tokens;
+          PINDEX tokenCount = clearTokens.GetSize();
+          clearTokens.SetSize(tokenCount + 1);
+          H235_ClearToken& clearToken = clearTokens[tokenCount];
+          clearToken.m_tokenOID = OpalOSP::ETSIXMLTokenOID;
+          clearToken.IncludeOptionalField(H235_ClearToken::e_nonStandard);
+          clearToken.m_nonStandard.m_nonStandardIdentifier =
OpalOSP::ETSIXMLTokenOID;
+          clearToken.m_nonStandard.m_data = ospToken;
+   }
+
 #ifdef H323_TRANSNEXUS_OSP
    // check for OSP server (if not using GK)
    if (gatekeeper == NULL) {
Index: channels/h323/chan_h323.h
===================================================================
--- channels/h323/chan_h323.h   (revision 150382)
+++ channels/h323/chan_h323.h   (working copy)
@@ -43,6 +43,9 @@
 #define H323_HOLD_Q931ONLY     (1 << 1)
 #define H323_HOLD_H450         (1 << 2)

+#define CALLIDSIZE                     64
+#define OSPTOKENSIZE           4096
+
 /** call_option struct holds various bits
  *         of information for each call */
 typedef struct call_options {
@@ -67,6 +70,8 @@
    int                         tunnelOptions;
    int                         holdHandling;
```

```
    struct ast_codec_pref       prefs;
+   char                 callId[CALLIDSIZE];
+   char            ospToken[OSPTOKENSIZE];
 } call_options_t;

 /* structure to hold the valid asterisk users */
@@ -121,6 +126,7 @@
    int type_of_number;
    int transfer_capability;
    char *sourceIp;
+   char* ospToken;
 } call_details_t;

 typedef struct rtp_info {
Index: channels/h323/ast_h323.h
===================================================================
--- channels/h323/ast_h323.h    (revision 150382)
+++ channels/h323/ast_h323.h    (working copy)
@@ -125,6 +125,8 @@

    RTP_DataFrame::PayloadTypes dtmfCodec[2];
    int dtmfMode;
+
+   PBYTEArray ospToken;
 };

 class MyH323_ExternalRTPChannel : public H323_ExternalRTPChannel
Index: channels/chan_h323.c
===================================================================
--- channels/chan_h323.c (revision 150382)
+++ channels/chan_h323.c (working copy)
@@ -440,6 +440,10 @@
           ast_free(cd->redirect_number);
           cd->redirect_number = NULL;
    }
+   if (cd->ospToken) {
+          free(cd->ospToken);
+          cd->ospToken = NULL;
+   }
 }

 static void __oh323_destroy(struct oh323_pvt *pvt)
@@ -584,6 +588,7 @@
    struct oh323_pvt *pvt = (struct oh323_pvt *)c->tech_pvt;
    const char *addr;
    char called_addr[1024];
+   const char* value;

    if (h323debug) {
           ast_debug(1, "Calling to %s on %s\n", dest, c->name);
@@ -640,6 +645,20 @@

    pvt->options.transfer_capability = c->transfercapability;

+   value = pbx_builtin_getvar_helper(c, "~OH323~callid");
+   if (ast_strlen_zero(value)) {
+          pvt->options.callId[0] = '\0';
```

```
+   } else {
+           strncpy(pvt->options.callId, value, sizeof(pvt->options.callId));
+   }
+
+   value = pbx_builtin_getvar_helper(c, "~OH323~osptoken");
+   if (ast_strlen_zero(value)) {
+           pvt->options.ospToken[0] = '\0';
+   } else {
+           strncpy(pvt->options.ospToken, value, sizeof(pvt->options.ospToken));
+   }
+
    /* indicate that this is an outgoing call */
    pvt->outgoing = 1;

@@ -3170,6 +3189,80 @@
    .set_rtp_peer = oh323_set_rtp_peer,
 };

+/*! \brief ${OH323CHANINFO()} Dialplan function - reads oh323 channel data */
+static int oh323_chaninfo_read(struct ast_channel *chan, char *cmd, char *varname,
char *buf, size_t len)
+{
+   struct oh323_pvt *p;
+
+   *buf = 0;
+
+   if (!varname) {
+           ast_log(LOG_WARNING, "This function requires a parameter name.\n");
+           return -1;
+   }
+
+   ast_channel_lock(chan);
+   if (chan->tech != &oh323_tech) {
+           ast_log(LOG_WARNING, "This function can only be used on OH323
channels.\n");
+           ast_channel_unlock(chan);
+           return -1;
+   }
+
+   if (!(p = chan->tech_pvt)) {
+           /* If there is no private structure, this channel is no longer alive */
+           ast_channel_unlock(chan);
+           return -1;
+   }
+
+   if (!strcasecmp(varname, "peerip")) {
+           ast_copy_string(buf, p->cd.sourceIp ? p->cd.sourceIp : "", len);
+   } else  if (!strcasecmp(varname, "osptoken")) {
+           ast_copy_string(buf, p->cd.ospToken ? p->cd.ospToken : "", len);
+   } else {
+           ast_channel_unlock(chan);
+           return -1;
+   }
+   ast_channel_unlock(chan);
+
+   return 0;
+}
```

```
+
+/*! \brief ${OH323CHANINFO()} Dialplan function - writes oh323 channel data */
+static int oh323_chaninfo_write(struct ast_channel *chan, char *cmd, char *varname,
const char *value)
+{
+   char tmp[256];
+
+   if (!varname) {
+           ast_log(LOG_WARNING, "This function requires a parameter name.\n");
+           return -1;
+   }
+
+   if (!strcasecmp(varname, "callid")) {
+           snprintf(tmp, sizeof(tmp), "_~OH323~%s", varname);
+           pbx_builtin_setvar_helper(chan, tmp, value);
+   } else  if (!strcasecmp(varname, "osptoken")) {
+           snprintf(tmp, sizeof(tmp), "_~OH323~%s", varname);
+           pbx_builtin_setvar_helper(chan, tmp, value);
+   } else {
+           return -1;
+   }
+
+   return 0;
+}
+
+/*! \brief Structure to declare a dialplan function: OH323CHANINFO */
+static struct ast_custom_function oh323_chaninfo_function = {
+    .name = "OH323CHANINFO",
+    .synopsis = "Gets the specified H.323 parameter from the current channel",
+    .syntax = "OH323CHANINFO(item)",
+    .read = oh323_chaninfo_read,
+    .write = oh323_chaninfo_write,
+    .desc = "Valid items are:\n"
+    "- peerip        The IP address of the source peer. Read only.\n"
+    "- callid        The call identifier to the destination. Write only.\n"
+    "- osptoken      The OSP token from the source or to the destination. Read and
write.\n"
+};
+
 static enum ast_module_load_result load_module(void)
 {
    int res;
@@ -3267,6 +3360,10 @@
                         res = AST_MODULE_LOAD_SUCCESS;
                  }
           }
+
+       /* Register dial plan functions */
+       ast_custom_function_register(&oh323_chaninfo_function);
+
           /* And start the monitor for the first time */
           restart_monitor();
    }
@@ -3277,6 +3374,9 @@
 {
    struct oh323_pvt *p, *pl;
```

```
+   /* Unregister dial plan functions */
+   ast_custom_function_unregister(&oh323_chaninfo_function);
+
    /* unregister commands */
    ast_cli_unregister_multiple(cli_h323, sizeof(cli_h323) / sizeof(struct
ast_cli_entry));
    ast_cli_unregister(&cli_h323_reload);
```

5.2.2 Apply Patch

Run the following command from Asterisk directory to apply the patch.

```
patch -p0 < h323_osp.patch
```