



OSP User Guide for Asterisk V1.6

9 February 2007

Table of Contents

Revision History	3
1 Introduction.....	4
2 OSP Toolkit	4
2.1 Build OSP Toolkit.....	4
2.1.1 Unpacking the Toolkit	4
2.1.2 Preparing to build the OSP Toolkit.....	5
2.1.3 Building the OSP Toolkit.....	5
2.1.4 Installing the OSP Toolkit	6
2.1.5 Building the Enrollment Utility	6
2.2 Obtain Crypto Files.....	6
3 Asterisk	8
3.1 Configure for OSP Support.....	8
3.1.1 Build Asterisk with OSP Toolkit	8
3.1.2 osp.conf.....	8
3.1.3 extensions.conf.....	10
3.1.4 zapata/sip/iax/h323/ooh323.conf	13
3.2 OSP Dial Plan Functions	13
3.2.1 OSPAuth	13
3.2.2 OSPLookup.....	13
3.2.3 OSPNext	14
3.2.4 OSPFinish	15
3.3 extensions.conf Examples.....	15
3.3.1 Source Gateway	15
3.3.2 Destination Gateway.....	16
3.3.3 Proxy.....	18

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 Introduction

This document provides instructions on how to build and configure Asterisk V1.6 with the OSP Toolkit to enable secure, multi-lateral peering. This document is also available in the Asterisk source package as doc/osp.txt. The 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) www.etsi.org. If you have questions or need help, building Asterisk with the 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 the OSP Toolkit:

- OpenSSL (required for building) - Open Source SSL protocol and Cryptographic Algorithms (version 0.9.7g recommended) from 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 www.activestate.com/Products/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 www.gnu.org is routinely used for building the 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 <http://www.vovida.org/applications/downloads/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 the Toolkit

After downloading the OSP Toolkit (version 3.3.6 or later release) from www.sourceforge.net, perform the following steps in order:

- 1) Copy the OSP Toolkit distribution into the directory where it will reside. The default directory for the OSP Toolkit is /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 3.3.6, then the above command would be:

```
gunzip -c OSPToolkit-3_3_6.tar.gz | tar xvf -
```

A new directory (TK-3_3_6-20060303) will be created within the same directory as the tar file.

- 3) Go to the TK-3_3_6-20060303 directory by running this command:

```
cd TK-3_3_6-20060303
```

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

```
ls -F
enroll/
RelNotes.txt
README.txt
bin/
crypto/
include/
lib/
license.txt
src/
test/
```

2.1.2 Preparing to build the OSP Toolkit

- 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 crypto/openssl directory within the osptoolkit 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 lib directory within the osptoolkit 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 through 6 are not necessary.

- 7) Optionally, change the install directory of the OSP Toolkit. Open the Makefile in the /usr/src/TK-3_3_6-20060303/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 the OSP Toolkit and the Asterisk.

2.1.3 Building the OSP Toolkit

- 8) From within the OSP Toolkit directory (/usr/src/TK-3_3_6-20060303), start the compilation script by executing the following commands:

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

2.1.4 Installing the OSP Toolkit

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

9) Use the make script to install the Toolkit.

```
make install
```

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

2.1.5 Building the 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 (example: `/usr/src/TK-3_3_6-20060303`), execute the following commands at the command prompt:

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

Compilation is successful if there are no errors in the compiler output. The enroll program is now located in the OSP Toolkit/bin directory (example: `/usr/src/TK-3_3_6-20060303/bin`).

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 the 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 the Asterisk `/var/lib/asterisk/keys` directory.

Note: The `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 the 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 ospstestserver.transnexus.com.

```
./enroll.sh ospstestserver.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: ospstestserver.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=ospstestserver.transnexus.com/O=OSPSPServer
verify error:num=18:self signed certificate
verify return:1
depth=0 /CN=ospstestserver.transnexus.com/O=OSPSPServer
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_3_6-20060303/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. In Asterisk V1.6, 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 three easy steps to add OSP support to Asterisk:

1. Build Asterisk with OSP Toolkit
2. Configure `osp.conf` file
3. Cut and paste to `extensions.conf`

Sections 3.2 and 3.3 provide a detailed explanation of OSP dial plan functions and configuration examples. The detailed information provided in Sections 3.2 and 3.3 is not required for operating Asterisk with OSP, but may be helpful to developers who want to customize their Asterisk OSP implementation.

3.1 Configure for OSP Support

3.1.1 Build Asterisk with OSP Toolkit

The first step is to build Asterisk with the OSP Toolkit. If the OSP Toolkit is installed in the default install directory, `/usr/local`, no additional configuration is required. Compile Asterisk according to the instructions provided with the Asterisk distribution.

If the OSP Toolkit is installed in another directory, such as `/myosp`, Asterisk must be configured with the location of the OSP Toolkit. See the example below.

```
--with-osptk=/myosp
```

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

3.1.2 `osp.conf`

The `/etc/asterisk/osp.conf` file, shown below, 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.
;
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://OSP server IP: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=[host IP]
;
; 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 should be 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

```

3.1.3 extensions.conf

OSP functions are implemented as dial plan functions in the 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. Configuration of your Asterisk server for OSP is now complete.

```

[globals]
DIALOUT=Zap/1

[SrcGW] ; OSP Source Gateway
exten => _XXXX.,1,NoOp(OSPSrcGW)
; Set calling number if necessary
exten => _XXXX.,n,Set(CALLERID(numner)=1234567890)
; OSP lookup using default provider, if fail/error jump to lookup+101
exten => _XXXX.,n(lookup),OSPLookup(${EXTEN}||j)
; Deal with outbound call according to protocol
exten => _XXXX.,n,Macro(outbound)
; Dial to destination, 60 timeout, with call duration limit
exten =>
_XXXX.,n,Dial(${OSPDIALSTR},60,oL(${OSPOUTTIMELIMIT}*1000))
; Wait 1 second

```

```

exten => _XXXX.,n,Wait,1
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPLookup fail/error
exten => _XXXX.,lookup+101,Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})

[DstGW] ; OSP Destination Gateway
exten => _XXXX.,1,NoOp(OSPDstGW)
; Deal with inbound call according to protocol
exten => _XXXX.,n,Macro(inbound)
; Validate token using default provider, if fail/error jump to
auth+101
exten => _XXXX.,n(auth),OSPAuth(|j)
; Ringing
exten => _XXXX.,n,Ringing
; Wait 1 second
exten => _XXXX.,n,Wait,1
; Check inbound call duration limit
exten => _XXXX.,n,GoToIf(${${OSPINTIMELIMIT}=0}?100:200)
; Without duration limit
exten => _XXXX.,100,Dial(${DIALOUT},15,o)
exten => _XXXX.,n,Goto(1000)
; With duration limit
exten => _XXXX.,200,Dial(${DIALOUT},15,oL(${${OSPINTIMELIMIT}*1000}))
exten => _XXXX.,n,Goto(1000)
; Wait 1 second
exten => _XXXX.,1000,Wait,1
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPAAuth fail/error
exten => _XXXX.,auth+101,Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})

[GeneralProxy] ; Proxy
exten => _XXXX.,1,NoOp(OSP-GeneralProxy)
; Deal with inbound call according to protocol
exten => _XXXX.,n,Macro(inbound)
; Validate token using default provider, if fail/error jump to
auth+101
exten => _XXXX.,n(auth),OSPAuth(|j)
; OSP lookup using default provider, if fail/error jump to lookup+101
exten => _XXXX.,n(lookup),OSPLookup(${EXTEN}||j)
; Deal with outbound call according to protocol
exten => _XXXX.,n,Macro(outbound)
; Dial to destination, 14 timeout, with call duration limit
exten =>
_XXXX.,n,Dial(${OSPDIALSTR},14,oL(${OSPOUTTIMELIMIT}*1000))
; OSP lookup next destination using default provider, if fail/error
jump to next1+101
exten => _XXXX.,n(next1),OSPNext(${HANGUPCAUSE}||j)
; Deal with outbound call according to protocol

```

```

exten => _XXXX.,n,Macro(outbound)
; Dial to destination, 15 timeout, with call duration limit
exten =>
_XXXX.,n,Dial(${OSPDIALSTR},15,oL(${OSPOUTTIMELIMIT}*1000))
; OSP lookup next destination using default provider, if fail/error
jump to next2+101
exten => _XXXX.,n(next2),OSPNext(${HANGUPCAUSE}||j)
; Deal with outbound call according to protocol
exten => _XXXX.,n,Macro(outbound)
; Dial to destination, 16 timeout, with call duration limit
exten =>
_XXXX.,n,Dial(${OSPDIALSTR},16,oL(${OSPOUTTIMELIMIT}*1000))
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPAuth fail/error
exten => _XXXX.,auth+101,Hangup
; Deal with OSPLookup fail/error
exten => _XXXX.,lookup+101,Hangup
; Deal with OSPNext fail/error
exten => _XXXX.,next1+101,Hangup
; Deal with OSPNext fail/error
exten => _XXXX.,next2+101,Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})

[macro-inbound]
exten => s,1,NoOp(inbound)
; Get inbound protocol
exten => s,n,Set(CHANTECH=${CUT(CHANNEL,/,1)})
exten => s,n,GoToIf("${CHANTECH}"="H323"?100)
exten => s,n,GoToIf("${CHANTECH}"="IAX2"?200)
exten => s,n,GoToIf("${CHANTECH}"="SIP"?300)
exten => s,n,GoTo(1000)
; H323 -----
; Get peer IP
exten => s,100,Set(OSPPEERIP=${H323CHANINFO(peerip)})
; Get OSP token
exten => s,n,Set(OSPINTOKEN=${H323CHANINFO(osptoken)})
exten => s,n,GoTo(1000)
; IAX -----
; Get peer IP
exten => s,200,Set(OSPPEERIP=${IAXPEER(CURRENTCHANNEL)})
; Get OSP token
exten => s,n,Set(OSPINTOKEN=${IAXCHANINFO(osptoken)})
exten => s,n,GoTo(1000)
; SIP -----
; Get peer IP
exten => s,300,Set(OSPPEERIP=${SIPCHANINFO(peerip)})
; Get OSP token
exten => s,n,Set(OSPINTOKEN=${SIP_HEADER(P-OSP-Auth-Token)})
exten => s,n,GoTo(1000)
; -----
exten => s,1000,MacroExit

[macro-outbound]

```

```

exten => s,1,NoOp(outbound)
; Set calling number which may be translated
exten => s,n,Set(CALLERID(number)=${OSPCALLING})
; Check destination protocol
exten => s,n,GoToIf("${OSPTECH}"="H323"?100)
exten => s,n,GoToIf("${OSPTECH}"="IAX2"?200)
exten => s,n,GoToIf("${OSPTECH}"="SIP"?300)
; Something wrong
exten => s,n,Hangup
exten => s,n,GoTo(1000)
; H323 -----
; Set call id
exten => s,100,Set(H323CHANINFO(callid)=${OSPOUTCALLID})
; Set OSP token
exten => s,n,Set(H323CHANINFO(osptoken)=${OSPOUTTOKEN})
exten => s,n,GoTo(1000)
; IAX -----
; Set OSP token
exten => s,200,Set(IAXCHANINFO(osptoken)=${OSPOUTTOKEN})
exten => s,n,GoTo(1000)
; SIP -----
exten => s,300,GoTo(1000)
; -----
exten => s,1000,MacroExit

```

3.1.4 zapata/sip/iax/h323/ooh323.conf

There is no configuration required for OSP.

3.2 OSP Dial Plan Functions

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

3.2.1 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.
- priority jump

Output:

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

3.2.2 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.
- priority jump
- 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 number
- 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

3.2.3 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
- priority jump

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

3.2.4 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
- priority jump

Output:

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

3.3 extensions.conf Examples

The extensions.conf file example provided in Section 3.1 is designed to handle all OSP call scenarios when Asterisk is used as a source or destination gateway to the PSTN or as a proxy between VoIP networks. The extension.conf examples in this section are designed for specific use cases only.

3.3.1 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, if fail/error jump to lookup+101
exten => _XXXX.,n(lookup),OSPLookup(${EXTEN}||j)
; Set calling number which may be translated
exten => _XXXX.,n,Set(CALLERID(number)=${OSPCALLING})
; Dial to destination, 60 timeout, with call duration limit
exten => _XXXX.,n,Dial(${OSPDIALSTR},60,oL(${OSPOUTTIMELIMIT}*100))
; Wait 3 seconds
exten => _XXXX.,n,Wait,3
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPLookup fail/error
exten => _XXXX.,lookup+101,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, if fail/error jump to lookup+101
exten => _XXXX.,n(lookup),OSPLookup(${EXTEN}||j)
; Set outbound OSP token
exten => _XXXX.,n,Set(IAXCHANINFO(osptoken)=${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(${OSPOUTTIMELIMIT}*1000))
; Wait 3 seconds
exten => _XXXX.,n,Wait,3
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPLookup fail/error
exten => _XXXX.,lookup+101,Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${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, if fail/error jump to lookup+101
; "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(${EXTEN}||jh)
; Set outbound call id
exten => _XXXX.,n,Set(OH323CHANINFO(callid)=${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))
; Wait 3 seconds
exten => _XXXX.,n,Wait,3
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPLookup fail/error
exten => _XXXX.,lookup+101,Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})

```

3.3.2 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, if fail/error jump to auth+101
exten => _XXXX.,n(auth),OSPAuth(|j)
; Ringing
exten => _XXXX.,n,Ringing
; Wait 1 second
exten => _XXXX.,n,Wait,1
; Dial phone, timeout 15 seconds, with call duration limit
exten =>
_XXXX.,n,Dial(${DIALOUTANALOG}/${EXTEN:1},15,oL(${OSPINTIMELIMIT}*1000))
; Wait 3 seconds
exten => _XXXX.,n,Wait,3
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPAuth fail/error
exten => _XXXX.,auth+101,Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${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, if fail/error jump to auth+101
exten => _XXXX.,n(auth),OSPAuth(|j)
; Ringing
exten => _XXXX.,n,Ringing
; Wait 1 second
exten => _XXXX.,n,Wait,1
; Dial phone, timeout 15 seconds, with call duration limit
exten =>
_XXXX.,n,Dial(${DIALOUTANALOG}/${EXTEN:1},15,oL(${OSPINTIMELIMIT}*1000))
; Wait 3 seconds
exten => _XXXX.,n,Wait,3
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPAuth fail/error
exten => _XXXX.,auth+101,Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})
```

For H.323 protocol

```
[H323DstGW]
exten => _XXXX.,1,NoOp(H323DstGW)
; Get peer IP
exten => _XXXX.,n,Set(OSPPEERIP=${H323CHANINFO(peerip)})
; Get OSP token
exten => _XXXX.,n,Set(OSPINTOKEN=${H323CHANINFO(osptoken)})
; Validate token using default provider, if fail/error jump to auth+101
exten => _XXXX.,n(auth),OSPAuth(|j)
; Ringing
exten => _XXXX.,n,Ringing
; Wait 1 second
exten => _XXXX.,n,Wait,1
; Dial phone, timeout 15 seconds, with call duration limit
exten =>
_XXXX.,n,Dial(${DIALOUTANALOG}/${EXTEN:1},15,oL(${OSPINTIMELIMIT}*1000))
; Wait 3 seconds
exten => _XXXX.,n,Wait,3
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPAAuth fail/error
exten => _XXXX.,auth+101,Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})
```

3.3.3 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, if fail/error jump to auth+101
exten => _XXXX.,n(auth),OSPAuth(|j)
; OSP lookup using default provider, if fail/error jump to lookup+101
; "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(${EXTEN}||jh)
; Set outbound call id and OSP token
; IAX, un-comment the following line.
;exten => _XXXX.,n,Set(IAXCHANINFO(osptoken)=${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(${OSPOUTTIMELIMIT}*1000))
; OSP lookup next destination using default provider, if fail/error jump to
next1+101
exten => _XXXX.,n(next1),OSPNext(${HANGUPCAUSE}||j)
; 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, 15 timeout, with call duration limit
exten => _XXXX.,n,Dial(${OSPDIALSTR},15,oL(${OSPOUTTIMELIMIT}*1000))
; OSP lookup next destination using default provider, if fail/error jump to
next2+101
exten => _XXXX.,n(next2),OSPNext(${HANGUPCAUSE}||j)
; Set outbound call id and OSP token
; IAX, un-comment the following line.
;exten => _XXXX.,n,Set(IAXCHANINFO(osptoken)=${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)=${OSPCALLING})
; Dial to destination, 16 timeout, with call duration limit
exten => _XXXX.,n,Dial(${OSPDIALSTR},16,oL(${OSPOUTTIMELIMIT}*1000))
; Hangup
exten => _XXXX.,n,Hangup
; Deal with OSPAuth fail/error
exten => _XXXX.,auth+101,Hangup
; Deal with OSPLookup fail/error
exten => _XXXX.,lookup+101,Hangup
; Deal with 1st OSPNext fail/error
exten => _XXXX.,next1+101,Hangup
; Deal with 2nd OSPNext fail/error
exten => _XXXX.,next2+101,Hangup
exten => h,1,NoOp()
; OSP report usage
exten => h,n,OSPFinish(${HANGUPCAUSE})

```