OSP Routing &
CDR Collection Server
with
OpenSIPS 1.5.1

Revision History

Revision	Date of Issue	Changes
0.1	Aug 12, 2005	Initial draft
0.2	Aug 23, 2005	Minor clarifications and edits
0.3	Nov 4, 2005	Added Appendix 1 and other edits
1.0.0	Nov 15, 2005	Edited for OpenSER V1.0.0
1.1.0	July 6, 2006	Edited for OpenSER V1.1.0
1.1.x.0	Sep 4, 2006	Edited for OpenSER V1.1.x
1.1.x.1	Dec 13, 2006	1. Increased service point number 2. Changed device_port
1.1.x.2	Dec 26, 2006	Corrected typographic errors
1.1.1	Feb 6, 2007	Changed OSP Toolkit site from SIPfoundry to SourceForge
1.2.x	May 17, 2007	Updated for OpenSER V1.2.x
1.3	July 23, 2008	Updated for OpenSER V1.3
1.4.0	Nov 25, 2008	Updated for OpenSIPS V1.4
1.4.1	Jan 30, 2009	Added instructions for Solaris 10 x86
1.4.2	Mar 4, 2009	Updated for Toolkit 3.5.0
1.5.0	Mar 25, 2009	Updated for OpenSIPS 1.5
1.5.1	June 25, 2009	Updated for OpenSIPS 1.5.1
1.5.1.1	July 2, 2009	Updated for Toolkit 3.5.2
1.5.1.2	July 15, 2009	Change title, edit location of OpenOSP project and add link to wikipedia.

Website: http://www.transnexus.com
Mailing list: https://lists.sourceforge.net/lists/listinfo/osp-toolkit-client

Contents

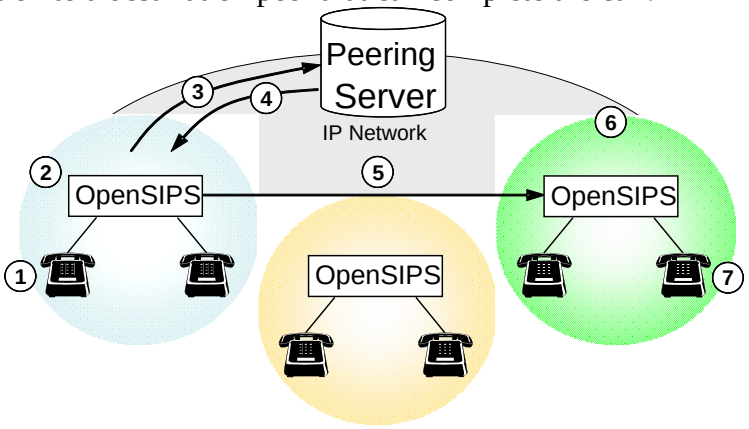
Introduction

Secure multi-lateral peering uses Public Key Infrastructure (PKI) services to secure, direct peering among an anonymous group of SIP peers. In a multi-lateral peering architecture, each peer trusts a common peering authority that enforces routing and access policies on behalf of each peer. The benefits of multi-lateral peering are increased peering security and the elimination of burdensome bilateral peering agreements and access control lists which are difficult to administer in a large peering network.

This document provides build instructions on how to build OpenSIPS V1.5.1 with OSP Toolkit. OSP Toolkit, which is freely available from http://sourceforge.net/projects/osp-toolkit, contains an implementation of 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. OSP Toolkit enables OpenSIPS for secure multi-lateral peering.

Multi-lateral SIP Peering

A peering server is a simple and efficient solution for managing routing, access control and CDR collection for VoIP calls among a network of OpenSIPS devices. OSP can be used to securely manage wholesale VoIP peering among independent SIP networks, or by an enterprise to create a secure VoIP virtual private network for calling among branch offices using SIP PBXs. The diagram below illustrates a call scenario between OpenSIPS networks using OSP peering. Each OpenSIPS proxy manages calls within its own domain. However, when a call must be completed outside its own network, an OpenSIPS proxy can query a peering server for routing and access information to a destination peer that can complete the call.

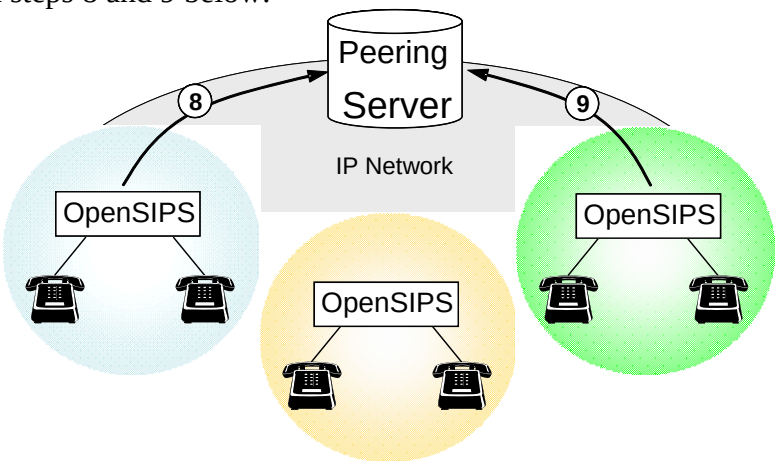


1. The calling party makes a call.
2. The source OpenSIPS cannot complete the call within its domain.
3. Peering Request. The source OpenSIPS queries the peering server for the IP addresses of other peers that can complete the call to the dialed number.
4. Peering Response. The peering server returns a list of IP addresses of destination peers and digitally signed peering tokens authorizing access to each destination peer.
5. The source OpenSIPS routes the call to the destination OpenSIPS returned by the peering server. Included in the SIP Invite message is the peering access token signed by the peering server.

6. The destination OpenSIPS receives the call and validates the peering token. If the token is valid, the destination OpenSIPS routes the call to the called telephone number.
7. The call is completed to calling party.

Call Detail Record Collection

When the call is over, both the source and destination peers send call detail records to the peering server as shown in steps 8 and 9 below.



Step 1 - Build OSP Toolkit

OSP Toolkit, when compiled, is a library comprised of OSP client functions that simplify sending and receiving OSP peering messages. It is this library, which will be integrated into OpenSIPS. OSP Toolkit uses third party software (by default OpenSSL) for cryptographic algorithms and for secure internet transactions (HTTPS). OSP Toolkit also includes the application Enroll which enables the OSP client device to generate its own public-private key pair, get the public key from an OSP peering server, send a certificate request to a peering server and receive the resulting signed certificate from the peering server.

In order to successfully compile and use OSP Toolkit, the following list of software is required:

- **OpenSSL** (required) - Open Source SSL protocol and Cryptographic Algorithms (version 0.9.8i recommended) from http://www.openssl.org. Pre-compiled OpenSSL packages are not recommended because of the binary compatibility issue.
- **Perl** (required) - A programming language used by OpenSSL for compilation. Any version of Perl will 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) - 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) - Two open source OSP server projects are available. OpenOSP, an OSP server written in C code, is located at http://sourceforge.net/projects/openosp/. RAMS, a Java based OSP server, is located at http://sourceforge.net/projects/rams. Also, a free version of the TransNexus commercial OSP server can be downloaded from http://www.transnexus.com/OSP%20Toolkit/Peering_Server/VoIP_Peering_Server.htm.

Unpacking OSP Toolkit

After downloading OSP Toolkit (version 3.5.2 or later release) from http://sourceforge.net/projects/osp-toolkit, perform the following steps in order:

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

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

  Where ### is the version number separated by dots. For example, if the version is 3.5.2, then the above command would be

```
gunzip –c OSPToolkit-3.5.2.tar.gz | tar xvf –
```

  A new OSP Toolkit directory, ***TK-3_5_2-20090702***, will be created within the same directory as the tarball file.
- Go to the OSP Toolkit directory by running this command

```
cd /usr/src/TK-3_5_2-20090702
```

  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/
```

Preparing to Build OSP Toolkit

- 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).
- 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.
- 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: If the OpenSSL package has been installed, above steps are not necessary.
- 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 OpenSIPS. Otherwise, it may make OpenSIPS does not support the OSP protocol.

Building OSP Toolkit

- From the OSP Toolkit directory, ***/usr/src/TK-3_5_2-20090702***, start the compilation script by executing the following commands

```
cd src
```

```
make clean; make build
```

- Use the make script to install OSP Toolkit

```
make install
```

The make script installs 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.

- By default, OpenSIPS does not use the security features that OSP Toolkit provides. If the security features are used, the enroll utility must be built and OpenSIPS must be enrolled with peering servers. The instructions are listed in Appendix 1 – To Use Security Features.

Step 2 - Compile OpenSIPS with OSP module

Obtaining OpenSIPS Source Code

Download OpenSIPS V1.5.1 release from http://www.opensips.org and un-package it.

```
gunzip -c opensips-1.5.1-notls_src.tar.gz |tar xvf -
```

It will create a sub-folder ***opensips-1.5.1-notls***. The distribution comes with a variety of modules (under modules folder), including the OSP module.

Building OpenSIPS with OSP Module

OpenSIPS V1.5.1 default does not compile the OSP module. You have to change the OpenSIPS Makefile file to remove osp from the exclude_modules list. After that, log in as root user, go to the OpenSIPS directory.
For Linux plarforms, run the following command

```
make clean; make all; make install
```

For Solaris 10 x86 platform, run the following command

```
gmake clean; gmake prefix=/usr/local LOCALBASE=/usr/local all; gmake
prefix=/usr/local LOCALBASE=/usr/local install
```

Note:

- OpenSIPS should be built by gmake.
- OpenSIPS should be installed by ginstall. For Solaris 10 x86, GNU install may be included in coreutils or fileutils package. A symbol link "ginstall" should be created for this GNU install if coreutils package is used.
- ***/usr/local/include*** and ***/usr/local/lib*** are not the default include and library pathes in Solaris 10 x86. They should be set by "LOCALBASE=/usr/local" in gmake command lines.
- ***/usr/local*** is not the default install path. It should be set by "prefix=/usr/local" in gmake command lines.

The commands will build the OpenSIPS with all modules (including OSP), and distribute executable and configuration files around the system.

Step 3 - Configure OpenSIPS

opensips.cfg is the main configuration file for OpenSIPS. It includes instructions for configuring OpenSIPS, loading external modules, configuring modules and CPL (call processing language) instructions. By default, the OpenSIPS looks for the configuration file in ***/usr/local/etc/opensips*** directory. OpenSIPS includes a sample configuration file, but it does not include instructions for using the OSP module. Instead of using the default OpenSIPS configuration file, use the sample configuration file, sample-osp-opensips.cfg, under OpenSIPS ***modules/osp/etc*** directory. Copy this file to overwrite the default OpenSIPS configuration file

```
cp modules/osp/etc/sample-osp-opensips.cfg
/usr/local/etc/opensips/opensips.cfg
```

The only parameter that must be configured in the OSP supporting opensips.cfg is the location of a peering server. Edit the configuration file and update the value of "sp1_uri" parameter to indicate the URL of the peering server. Configuring other "spN_uri" parameters to load share and add redundancy with other peering servers is recommended. Another parameter which is recommended for editing is "device_ip". More details on these parameters are provided in sample-osp-opensips.cfg. Appendix 2 - OSP Module Parameters provides detail on all OSP module parameters.

Note: For a 64-bit system, the OpenSIPS module directory in opensips.cfg should be changed from ***mpath="/usr/local/lib/opensips/modules"*** to ***mpath="/usr/local/lib64/opensips/modules"***.

An additional tuning adjustment for optimizing OpenSIPS is to decrease the number of TCP synchronization retries made by the Linux operating system when a TCP connection fails. Setting this parameter to 0 or 1 will decrease post dial delay when a TCP connection to a destination peer is not possible. Reducing TCP synchronization retries by the operating system will speed call fail-over to the next destination peer. To change this parameter, open the file ***/etc/sysctl.conf*** and add the entry "net.ipv4.tcp_syn_retries = 1". Reboot the computer and check the parameter entry was accepted by the following command,

```
cat /proc/sys/net/ipv4/tcp_syn_retries
```

It should display "1".

Step 4 - Run OpenSIPS

By default, running OpenSIPS requires root privileges. To start OpenSIPS, use the following command,

```
/usr/local/sbin/opensipsctl start
```

Note: Additional share memory may be necessary for heavy load systems. Add "-m 256" for STARTOPTIONS in ***/usr/local/etc/opensips/opensiosctlrc***.

To stop the OpenSIPS, use the following command,

```
/usr/local/sbin/opensipsctl stop
```

Sometimes, OpenSIPS can be run by a user other than root. The following commands can be used to start/stop OpenSIPS,

```
/usr/local/sbin/opensips    # start opensips
pkill -9 opensips           # stop opensips
```

Note:

- Additional share memory may be necessary for heavy load systems. Add "-m 256" in start command line.
- It needs the privileges of reading OpenSIPS configuration file to run OpenSIPS. OpenSIPS installs the configuration file in read and write only by root mode. To start OpenSIPS by other users, the directory and file modes may have to be changed using following commands by root,

```
chmod 755 /usr/local/etc/opensips
chmod 644 /usr/local/etc/opensips/opensips.cfg
```

For Solaris 10 x86, to log OpenSIPS log messages, syslog should be configured. Adding the following line in /etc/syslog.conf

```
daemon.debug    -/var/adm/messages
```

OpenSIPS log messages should be logged into /var/adm/messages file.
Note: For some Solaris versions, the asynch mode syslog may not work. You may have to remote the "-" symbol before "/var/adm/messages".

Frequently Asked Questions

This section includes common questions about the OSP module with OpenSIPS along with answers. If your question was not answered here, please submit your question to the OSP mailing list at https://lists.sourceforge.net/lists/listinfo/osp-toolkit-client.

Where are OSP module log messages written?

Log messages are sent to the sys-log application and will be logged according to the logger's configuration. Look for the messages in the /var/log/messages file for Linux or /var/adm/messages file for Solaris.

Can multiple instances of OpenSIPS be run on the same machine?

It is possible to run more than one OpenSIPS on the same computer. The simplest way is to create a second configuration file and edit the port number from 5060 to a different port number. When start the second OpenSIPS, instead of using the opensips.init script, explicitly specify where the configuration file is using the "–f config-file" option. If the second OpenSIPS requires a different set of certificates, uncomment and update the private_key, local_certificate and ca_certificates variables in the configuration file.

Can I use both TCP and UDP for communication?

OpenSIPS supports both UDP and TCP. A peering response does not explicitly indicate which transport protocol to use for communicating to the terminating SIP proxy or user agent. By default, OpenSIPS uses to UDP. Insure that the terminating proxy or UA is defined on the peering server as a SIP device. This will instruct the peering server to use a compressed peering token, small enough to fit into a single UDP packet.

How is OSP authentication different from password or access list authentication?

The OSP module implements Public Key Infrastructure (PKI) services to securely authenticate and authorize peer to peer calling among SIP proxies. If the SIP INVITE received by OpenSIPS includes a peering authorization token, the OSP module will validate the peering token was digitally signed by the OpenSIPS's peering authority. If the token is valid the OpenSIPS will accept the call. If not, OpenSIPS will reject the call.

The OSP module uses the public key of its peering authority to verify that its peering authority signed the peering token (standard asymmetric cryptography). When compared to access lists or passwords for peer to peer authentication and authorization, PKI services offer improved security and scalability. Passwords, shared secrets or symmetric keys are prone to security breaches. Access lists have limited scalability. Managing an access list that contains hundreds of addresses and changes frequently is a major operations challenge. Secure peering using PKI services eliminates these limitations.

Why is time synchronization with the OSP peering server important?

Peering authorization tokens have a limited life – usually five minutes. If the time of day of OpenSIPS differs by more than five minutes from its peering server, peering tokens from SIP INVITES received from other peers will be invalid and calls will be rejected. The simple solution is for all peers to use Network Time Protocol (NTP) to maintain time synchronization.

Appendix 1 – To Use Security Features

If the security features that OSP Toolkit provides are not used, it is not necessary to build enroll utility and enroll OpenSIPS with peering servers.

Building Enroll Utility

The Enroll program is a utility application for establishing a trusted relationship between an OpenSIPS OSP client and an OSP peering server or certificate authority. The following steps will build the Enroll utility included within OSP Toolkit.

- For Linux platforms, from within the OSP Toolkit directory, ***/usr/src/TK-3_5_2-20090702***, execute the following commands at the command prompt

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

- For Solaris 10 x86 platform, from within the OSP Toolkit directory, ***/usr/src/TK-3_5_2-20090702***, execute the following commands at the command prompt

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

  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. For more information on the Enroll utility, please see http://www.transnexus.com/OSP%20Toolkit/OSP%20Toolkit%20Documents/Device_Enrollment.pdf.

Enroll OpenSIPS with Peering Servers

Overview

It requires three crypto files to establish a secure relationship between an OSP peering server and the OSP module in OpenSIPS. These files are:

- localcert.pem - The local certificate for OpenSIPS signed by the OSP server.
- pkey.pem - The private key generated by the Enroll utility for OpenSIPS.
- cacert_#.pem - The Certificate Authority (CA) certificate from an OSP server. OpenSIPS may enroll with multiple certificate authorities or peering servers. The # represents an integer indicating the CA certificate from different peering servers.

The Enroll utility automates the process of enrolling OpenSIPS with peering servers and creating the crypto files. By default, the OSP Module of OpenSIPS will load the crypto files from the default configuration directory, ***/usr/local/etc/opensips***. If the security features are used and the certificate files are not present, the OSP Module of OpenSIPS will not start.

Using Enroll Script

The script enroll.sh requires AT&T korn shell (ksh) or any of its compatible variants. The OSP Toolkit ***bin*** directory should be in the **PATH** environment variable.

From the command line, type enroll.sh followed by the IP addresses or domain name of the peering serveres. Below is an example of the Enroll utility being used to enroll OpenSIPS with a peering server named osptestserver.transnexus.com. The gray boxes indicate optional input which will be included in the OpenSIPS's certificate. Error Code 0 indicates the operation was successful with no error.

```
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=6096834216798074
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
```

Note: localcert.pem, pkey.pem and cacert_#.pem files generated by the Enroll utility must be copied to the OpenSIPS configuration directory (default ***/usr/local/etc/opensips***).

Appendix 2 - OSP Module Parameters

This Appendix provides a detailed explanation of all OSP module parameters that may be included in the configuration file. Information on OSP module functions can be found in the "OSP Module for Secure, Multi-Lateral Peering" document located at: http://www.opensips.org/html/docs/modules/1.5.x/osp.html.

sp1_uri, sp2_uri, …, sp16_uri

These sp_uri (string) parameters define OSP peering servers which may be queried for peering authorization and routing information. At least one OSP peering server must be configured. Configuring additional peering servers is recommended for redundancy, but not required. Each peering server address takes the form of a standard URL, and consists of up to four components:

- An optional indication of the protocol to be used for communicating with the peering server. Both HTTP and HTTP secured with SSL/TLS are supported and are indicated by "http://" and "https://" respectively. If the protocol is not explicitly indicated, the OpenSIPS defaults to HTTP secured with SSL.
- The Internet domain name for the peering server. An IP address may also be used, provided it is enclosed in square brackets such as [172.16.1.1].
- An optional TCP port number for communicating with the peering server. If the port number is omitted, the OpenSIPS defaults to port 1080 (for HTTP) or port 1443 (for HTTP secured with SSL).
- The uniform resource identifier for requests to the peering server. This component is not optional and must be included.

Examples:

```
modparam("osp", "sp1_uri",
"http://osptestserver.transnexus.com:1080/osp")
modparam("osp", "sp2_uri", "https://[1.2.3.4]:1443/osp")
```

sp1_weight, sp2_weight, …, sp16_weight

These sp_weight (integer) parameters are used for load balancing peering requests to peering servers. These parameters are most effective when configured as factors of 1000. For example, if sp1_uri should manage twice the traffic load of sp2_uri, then set sp1_weight to 2000 and sp2_weight to 1000. Shared load balancing between peering servers is recommended. However, peering servers can be configured as primary and backup by assigning a sp_weight of 0 to the primary server and a non-zero sp_weight to the back-up server. The default values for sp1_weight and sp2_weight are 1000.

Example:

```
modparam ("osp", "sp1_weight", 1000)
```

device_ip

The device_ip (string) is a recommended parameter that explicitly defines the IP address of OpenSIPS in a peering request message (as SourceAlternate type=transport). The IP address must be in brackets as shown in the example below.

Example:

```
modparam ("osp", "device_ip", "[1.1.1.1]")
```

device_port

The device_port (string) parameter is an optional field which includes the SIP port being used by OpenSIPS in the peering request (as SourceAlternate type=network) to the peering server. If is not configured, this information is not included in the peering request. This field is useful if multiple OpenSIPS's are running on the same Linux computer since it enables the peering server to administer different peering policies based on different SIP proxies. This parameter has not been implemented yet.

Example:

```
modparam ("osp", "device_port", "5060")
```

use_security_features

This parameter is used to tell OSP module if the OSP security features should be used. The default value is 0.

- 0 - OSP module will not use the OSP security features.
- 1 - OSP module will use the OSP security features.

Example:

```
modparam ("osp", "use_security_features", 0)
```

token_format

When OpenSIPS receives a SIP INVITE with a peering token, the OSP module will validate the token to determine whether or not the call has been authorized by a peering server. Peering tokens may, or may not, be digitally signed. The token_format (integer) parameter defines if OpenSIPS will validate signed or unsigned tokens or both. The values for token format are defined below. The default value is 2.

If use_security_features parameter is set to 0, signed toekns cannot be validated.

- 0 - Validate only signed tokens. Calls with valid signed tokens are allowed.
- 1 - Validate only unsigned tokens. Calls with valid unsigned tokens are allowed.
- 2 - Validate both signed and unsigned tokens are allowed. Calls with valid tokens are allowed.

Example:

```
modparam ("osp", "token_format", 2)
```

private_key, local_certificate, ca_certificates

These parameters identify crypto files are used for validating peering authorization tokens and establishing a secure channel between OpenSIPS and a peering server using SSL. The files are generated using the 'Enroll' utility from the OSP Toolkit. By default, the proxy will look for pkey.pem, localcert.pem, and cacart_0.pem in the default configuration directory. The default configuration directory is set at compile time using "CFG_DIR" and defaults to ***/usr/local/etc/opensips***. The files may be copied to the expected file location or the parameters below may be changed.
If use_security_features parameter is set to 0, these parameters will be ignored.
Example: If the default "CFG_DIR" value was used at compile time, the files will be loaded from:

```
modparam("osp", "private_key",
"/usr/local/etc/opensips/pkey.pem")
modparam("osp", "local_certificate",
"/usr/local/etc/opensips/localcert.pem")
modparam("osp", "ca_certificates",
"/usr/local/etc/opensips/cacert_0.pem")
```

enable_crypto_hardware_support

The enable_crypto_hardware_support (integer) parameter is used to set the cryptographic hardware acceleration engine in the OpenSSL library. The default value is 0 (no crypto hardware is present). If crypto hardware is used, the value should be set to 1.
Example:

```
modparam ("osp", "enable_crypto_hardware_support", 0)
```

ssl_lifetime

The ssl_lifetime (integer) parameter defines the lifetime, in seconds, of a single SSL session key. Once this time limit is exceeded, the OSP module will negotiate a new session key. Communication exchanges in progress will not be interrupted when this time limit expires. This is an optional field with default value is 200 seconds.
Example:

```
modparam ("osp", "ssl_lifetime", 200)
```

persistence

The persistence (integer) parameter defines the time, in seconds, that an HTTP connection should be maintained after the completion of a communication exchange. The OSP module will maintain the connection for this time period in anticipation of future communication exchanges to the same peering server.
Example:

```
modparam ("osp", "persistence", 1000)
```

retry_delay

The retry_delay (integer) parameter defines the time, in seconds, between retrying connection attempts to an OSP peering server. After exhausting all peering servers the OSP module will delay for this amount of time before resuming connection attempts. This is an optional field with default value is 1 second.

Example:

```
modparam ("osp", "retry_delay", 1)
```

retry_limit

The retry_limit (integer) parameter defines the maximum number of retries for connection attempts to an OSP peering server. If no connection is established after this many retry attempts to all peering servers, the OSP module will cease connection attempts and return appropriate error codes. This number does not count the initial connection attempt, so that a retry_limit of 1 will result in a total of two connection attempts to every peering server. The default value is 2.

Example:

```
modparam ("osp", "retry_limit", 2)
```

timeout

The timeout (integer) parameter defines the maximum time in milliseconds, to wait for a response from an OSP peering server. If no response is received within this time, the current connection is aborted and the OSP module attempts to contact the next peering server. The default value is 10 seconds.

Example:

```
modparam ("osp", "timeout", 10)
```

max_destinations

The max_destinations (integer) parameter defines the maximum number of destinations that OpenSIPS requests the peering server to return in a peering response. The default value is 5.

Example:

```
modparam ("osp", "max_destinations", 5)
```

validate_call_id

The validate_call_id (integer) parameter instructs the OSP module to validate call id in the peering token. If this value is set to 1, the OSP module validates that the call id in the SIP INVITE message matches the call id in the peering token. If they do not match the

INVITE is rejected. If this value is set to 0, the OSP module will not validate the call id in the peering token. The default value is 1
Example:

```
modparam ("osp", "validate_call_id", 1)
```

use_rpid_for_calling_number

The use_rpid_for_calling_number (integer) parameter instructs the OSP module to use the calling number in the Remote-Party-ID of the SIP INVITE message. If this value is set to 1, the OSP Module uses the calling number in the Reomte-Party-ID header of the INVITE message when a Remote-Party-ID exists. If this value is set to 0, the OSP Module will use the calling number in the From header of the INVITE message. The default value is 1
Example:

```
modparam ("osp", "use_rpid_for_calling_number", 1)
```

redirection_uri_format

The redirection_uri_format (integer) parameter instructs the OSP module to use the different URI format in the SIP redirection message. If this value is set to 0, the OSP Module uses "xxxxxxxxxx@xxx.xxx.xxx.xxx" URI in the SIP redirection messages. If this value is set to 1, the OSP Module will use "<xxxxxxxxxx@xxx.xxx.xxx.xxx>" URI in the SIP redirection messages. The default value is 0
Example:

```
modparam("osp", "redirection_uri_format", 1)
```

source_networkid_avp

The source_networkid_avp (string) parameter instructs the OSP module to use the defined AVP to pass the source network ID value. The default value is "$avp(s:_osp_source_networkid_)". Then the source network ID can be set by "$avp(s:_osp_source_networkid_) = pseudo-variables". All pseudo variables are described in http://www.opensips.org/index.php?n=Resources.DocsCoreVar#varpv.
Example:

```
modparam("osp", "source_networkid_avp", "$avp(s:snid)")
```

Appendix 3 – OpenSIPS Log Issues

OpenSIPS logging may cause some performance issues in heavy load environments. The reason is the systems waits file I/O.

Phenomenon

- OpenSIPS is configured to log at level 3.
- Syslog is configured to log OpenSIPS log messages to /var/log/messages in sync mode.
- OpenSIPS sends the first 100 Trying messages several seconds after receiving INVITE messages.
- OpenSIPS can only handle ~10 cps.

Solution

The solution is to configure syslog to log OpenSIPS log messages in async mode by following steps,

- Edit /etc/syslog.conf to use async mode for /var/log/messages

```
*.info;mail.none;authpriv.none;cron.none          -/var/log/messages
```

- Restart syslog

```
/etc/init.d/syslog restart
```

  Note: For different OS systems, the command may be a little different.
- Restart OpenSIPS