OpenOSP

Installation & Configuration Guide

10 April 2001

Document Version 1.5

Data Connection Manual MOU-100-0105

Notice

Data Connection Limited
100 Church Street
Enfield
Middlesex
EN2 6BQ
England

+44 20 8366 1177
http://www.datcon.co.uk

Contents

1 Introduction

This document contains release documentation for the V2.0 final release of OpenOSP.

OpenOSP is an open source Open Settlement Protocol (OSP) server protocol stack jointly developed by Cisco Systems and Data Connection Limited (DCL). Before starting to use these release notes, you should read the *OpenOSP Product Overview* for general information about OpenOSP.

The remainder of this document is structured as follows.

- Section 2 provides a summary of this release, including its licensing, functionality, target environment, and any known issues and limitations.
- Section 3 describes the changes required to migrate from the OpenOSP V2.0 alpha release to the OpenOSP V2.0 final release.
- Section 4 explains how to build the OpenOSP libraries and sample application.
- Section 5 explains how to configure and run OpenOSP.

2 Release Summary

2.1 Licensing

Licensing information for this release of the OpenOSP distribution is available at http://www.vovida.org or in the README file included in the distribution.

2.2 Functionality

This release includes all the functionality described in the *OpenOSP Product Overview*. For reference, this level of function is as follows.

- All OSP messages – AuthorizationRequest, AuthorizationIndication, AuthorizationResponse, AuthorizationConfirmation, ReauthorizationRequest, ReauthorizationResponse, PricingIndication, PricingConfirmation, UsageIndication, UsageConfirmation, SubscriberAuthenticationRequest, SubscriberAuthenticationResponse, CapabilitiesIndication, CapabilitiesConfirmation.
- Token signing with all the ciphers specified in the *OpenOSP Product Overview*.
- SSL/TLS security with all the ciphers specified in the *OpenOSP Product Overview*.
- S/MIME security with all the ciphers specified in the *OpenOSP Product Overview*.
- Full SCEP support.
- Full multi-threaded operation including support for symmetric multi-processor systems.
- Full support for Cisco extensions to allow the use of supported protocol types in Client CapabilitiesIndications and routing decisions.
- Stack support for Cisco extensions for authorization of prepaid subscribers.

2.3 Target environment

This release has been developed and tested using the Solaris 7 operating system on Sun Ultra 10 workstations, using the "slapd" directory server from OpenLDAP version 1.2.10.

It has also been verified to a limited extent running on a 4-way Sun MP machine, using the Solaris 8 operating system.

The behaviour of the OpenOSP server on other operating systems, and on other versions of SunOS or Solaris, has not been verified.

2.4 Current Issues

This subsection lists known issues to be aware of when using the current OpenOSP release.

2.4.1 Transnexus extensions used by Cisco routers

By default, the Cisco 3640 uses Transnexus-defined extensions in the messages that it generates, and marks these extensions as critical. For interoperation with OpenOSP, these extensions should be switched off in the router configuration.

2.4.2 Private extensions used by the Transnexus client

By default, the Transnexus OSP client sends UsageIndication requests containing a private extension element without setting the critical flag to False. The OpenOSP server doesn't understand this element and so, in accordance with the OSP specification, rejects the requests.

The workaround for this problem is to adjust the Transnexus client config file to set the client id to 0. With this setting, the client does not include any private extensions in the requests that it generates, and so OpenOSP processes and responds to the requests successfully.

2.4.3 Advanced authorization and authentication testing

This release implements support for several modes of authorization and authentication that are not yet supported by Cisco routers and other OSP clients. Consequently interoperability testing of this function is incomplete.

2.5 Acknowledgement

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

3 Migrating from OpenOSP V2.0 alpha

The OpenOSP V2.0 final release contains no major functional changes from the V2.0 alpha release, and no changes are required to any existing V2.0 alpha configuration.

4 Building OpenOSP

4.1 Prerequisites

OpenOSP uses a number of other open source libraries, which must be built before beginning to build OpenOSP itself:

- Expat – an XML parser library
- OpenSSL – a toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols and a general purpose cryptography library
- Sleepycat Berkeley DB – an embedded database backend
- OpenLDAP – LDAP client libraries and directory server.

Note that perl is required when building OpenSSL. The building and installation of perl is not covered by these release notes.

4.1.1 Disk space requirements

The disk space required for building Expat, OpenSSL, OpenLDAP and OpenOSP on Sparc Solaris is estimated as follows:

- Expat: 1Mb
- OpenLDAP: 12Mb
- OpenSSL: 25Mb
- Sleepycat: 10Mb
- OpenOSP: 25Mb
- TOTAL: 73Mb.

Note that the estimated total does not include space for the certificates and routing and pricing data stored in the directory, which can be as large as the complexity of your routing and pricing tables demands. For the sample tables that are included in the OpenOSP distribution, this data occupies less than 50kb.

4.1.2 Build directory hierarchy

The OpenOSP makefiles assume that the distributions for these libraries have been unpacked into a directory hierarchy like this:

$EXTROOT/expat top-level Expat directory

$EXTROOT/expat/expat.html

$EXTROOT/expat/gennmtab
$EXTROOT/expat/makefile
… etc …

$EXTROOT/openssl-0.9.5a — top-level OpenSSL directory

$EXTROOT/openssl-0.9.5a/CHANGES
$EXTROOT/openssl-0.9.5a/CHANGES.SSLeay
$EXTROOT/openssl-0.9.5a/Configure
… etc …

$EXTROOT/openldap-1.2.10 — top-level OpenLDAP directory

$EXTROOT/openldap-1.2.10/ANNOUNCEMENT
$EXTROOT/openldap-1.2.10/CHANGES
$EXTROOT/openldap-1.2.10/COPYRIGHT
… etc …

The Sleepycat distribution is not referenced directly by the OpenOSP makefiles – it is only used indirectly via OpenLDAP. Therefore Sleepycat may be unpacked at any filesystem location.

Begin, therefore, by choosing a directory location for $EXTROOT. Then follow the instructions in the following subsections to unpack and build the Expat, OpenSSL, Sleepycat and OpenLDAP distributions.

4.1.3 Expat

Expat can be downloaded from ftp://ftp.jclark.com/pub/xml/expat.zip.

Unpack the distribution below $EXTROOT so that the directory hierarchy is as illustrated above. Build the Expat object files by running "make" in the top-level Expat directory.

The Expat makefile assumes that the GCC compiler will be used to build Expat. To compile using an alternative compiler, change the CC and CFLAGS variables accordingly. For example, for the Sun compiler, change CC=gcc to CC=cc and CFLAGS=–O2 to CFLAGS=–O.

4.1.4 OpenSSL

The current OpenOSP release has been tested with OpenSSL version 0.9.5a, which can be downloaded from http://www.openssl.org/source/openssl-0.9.5a.tar.gz or ftp://ftp.openssl.org/source/openssl-0.9.5a.tar.gz.

Unpack the distribution below $EXTROOT so that the directory hierarchy is as illustrated above.

OpenOSP includes a patch for OpenSSL to add support for the directoryName CRLDP format (which is not yet supported by the standard OpenSSL distribution). To apply this patch, replace the OpenSSL file crypto/x509v3/v3_alt.c by the file patch/v3_alt.c from the OpenOSP distribution.

Then follow the instructions in the top-level INSTALL file to build and install OpenSSL.

4.1.5 Sleepycat Berkeley DB

OpenLDAP requires a third party backend database implementation for data storage. The current OpenOSP release has been tested with Sleepycat Berkeley DB version 2.7.7, which can be downloaded from http://www.sleepycat.com/update/2.7.7/db-2.7.7.tar.gz.

Unpack the distribution at any convenient filesystem location. Go to the build_unix subdirectory, then build and install using these commands:

```
$ env CC=gcc ../dist/configure

$ make –e prefix=/usr

$ su root –c 'make –e prefix=/usr install'
```

If not using the GCC compiler, omit the "CC=gcc" from the configure command. To install in /usr/local rather than /usr, omit the "–e prefix=/usr" from the make commands.

4.1.6 OpenLDAP

The current OpenOSP release has been tested with OpenLDAP version 1.2.10, which can be downloaded from http://www.openldap.org/software/download/.

Unpack the distribution below $EXTROOT so that the directory hierarchy is as illustrated above. Follow the instructions in the top-level INSTALL file to build and install OpenLDAP. The configuration command for building OpenLDAP using the Sleepycat Berkeley DB as the backend database is:

```
$ env LIBS="–lpthread –lposix4" CPPFLAGS="–I/usr/BerkeleyDB/include"
LDFLAGS="–L/usr/BerkeleyDB/lib" ./configure --with-ldbm-api=db2
```

4.2 OpenOSP

4.2.1 Unpacking the OpenOSP distribution

Choose a directory location in which to unpack the OpenOSP distribution. Change to that directory, and copy in the OpenOSP distribution file from its distribution media.

```
$ cd <directory>

$ cp <OpenOSP distribution file> .
```

Unpack the distribution file.

```
$ uncompress openosp-2.0-alpha.tar.Z

$ tar xvf openosp-2.0-alpha.tar
```

4.2.2 Preparing the build environment

Before building OpenOSP, you should set

- the EXTROOT environment variable to the directory beneath which the Expat, OpenSSL and OpenLDAP distributions were unpacked, as described above in section 4.1

- the OSPROOT environment variable to <directory>/openosp, where <directory> is the directory where you chose to unpack the OpenOSP distribution, as described above in section 4.2.1.

For Bourne-type shells, use "=" and "export". For example

```
$ EXTROOT=/usr/local/src

$ export EXTROOT

$ OSPROOT=/usr/local/src/openosp

$ export OSPROOT
```

For Csh-type shells, use "setenv". For example

```
% setenv EXTROOT /usr/local/src

% setenv OSPROOT /usr/local/src/openosp
```

4.2.3 Building OpenOSP

Now you are ready to build OpenOSP. Change to the OSPROOT directory.

```
$ cd $OSPROOT
```

To build normal copies of the OpenOSP libraries and sample application executable, type:

```
$ make release
```

To build debugging ("traced") copies of the OpenOSP libraries and sample application executable, type:

```
$ make debug
```

To build using the GCC compiler, modify these make commands to include the compiler name and any required compiler flags, like this:

```
$ make –e CC="gcc" release

$ make –e CC="gcc –g" debug
```

To build OpenOSP without SCEP (PKI) support, edit the top level make.common file to remove the "–DOSP_SUPPORT_SCEP" flag on line 49.

All files should compile cleanly, except for several expected "statement not reached" warnings. The binaries are placed in the locations shown below.

OpenOSP stack library:	$OSPROOT/stack/rel/libosp.a (normal copy) $OSPROOT/stack/dbg/libosp.a (debugging copy)
OpenOSP sample application executable:	$OSPROOT/samplapp/rel/ospd (normal copy) $OSPROOT/samplapp/dbg/ospd (debugging copy)
Certificate revocation tool executable:	$OSPROOT/crtool/rel/crladmin (normal copy) $OSPROOT/crtool/dbg/crladmin (debugging copy)

4.2.4 Installation

It is not necessary to install OpenOSP before running it, and the current release does not include installation support in the form of a "make install" target. For convenience when developing an OpenOSP application (or modifying the sample provided), however, you may find it convenient to copy the following files to standard system locations:

- $OSPROOT/stack/openosp.h – to /usr/include/openosp.h
- $OSPROOT/stack/rel/libosp.a – to /usr/lib/libosp.a.

When running the sample application frequently, it may be convenient to copy $OSPROOT/samplapp/rel/ospd to a location that is in the super-user's default path, such as /usr/bin.

5 Configuring and Running OpenOSP

Configuring and running OpenOSP breaks down into the following steps:

- configuring the OpenOSP stack
- configuring the OpenOSP sample application
- configuring the security/PKI environment
- configuring routing and pricing information
- notes for interoperability configuration
- starting or stopping related server applications
- starting the OpenOSP sample application.

Section 5.1 presents an overview of OpenOSP configuration, which is relevant to all of these steps. The subsequent subsections describe each step in detail.

5.1 Configuration overview

OpenOSP is configured by entries in an OpenSSL-style configuration file, a sample of which is included in the OpenOSP distribution – see samples/openosp.cnf or Appendix A. When OpenOSP runs, it determines the location of its configuration file by looking up the value of the environment variable OSP_CONFIG_FILE.

Since later parts of the configuration involve further files, such as those containing private keys for the security/PKI environment, we recommend creating a configuration directory dedicated to OpenOSP that is accessible only to the super-user.

In the following subsections, we suppose this directory is /usr/openosp, and that the OpenOSP configuration file (which could initially have been created by coping from samples/openosp.cnf) is named /usr/openosp/openosp.cnf.

5.2 Configuring the OpenOSP stack

The OpenOSP stack is configured by the following entries in the [openosp] group.

non_ssl_port (default value **80**)

The TCP/IP port number on which OpenOSP listens for non-secure incoming connections.

ssl_port (default value **443**)

The TCP/IP port number on which OpenOSP listens for secure (SSL) incoming connections.

Note that OpenOSP will process both PKI (SCEP) and OSP requests that are received on either the non-SSL port or the SSL port. This is because both SCEP and OSP are wrapped using HTTP, and HTTP may be either secure or non-secure.

Which of these two ports is actually used for SCEP and OSP requests from a particular client – such as a router – is determined by that client's configuration.

ssl_identity (no default, must be configured)

The DN of the cryptographic identity that OpenOSP uses for SSL transport-level security. Cryptographic identities are described in section 5.4.2.

ssl_cipher_list (default value **DEFAULT**)

Specifies the list of cipher suites allowed when SSL is in use. The value of this configuration entry is passed directly to the OpenSSL library: for further details please refer to OpenSSL documentation.

ssl_verify_client (default value **no**)

Specifies whether or not OpenOSP should attempt to verify the certificates of clients that connect via SSL. The permitted values for this option are **no** and **yes**. If **yes** is specified, it may be followed by one or both of the following modifiers to change the behaviour of the client verification function.

require The client is required to provide a certificate. If it does not, the connection will be refused. Without this option, clients may still connect if they do not provide a certificate, although this may be overridden by the application in the client verification callback.

once The server will only request the client's certificate when establishing a new SSL session. The server will not request the client's certificate again if the SSL session is renegotiated or if the session is re-used on a subsequent connection. Without this option, the server will request (and verify) the client's certificate each time the connection is renegotiated or the session is re-used.

The modifiers may be separated by colons, commas, spaces or tabs.

ca_identity (no default, must be configured)

The DN of the cryptographic identity that OpenOSP uses when acting as a certification authority (CA) in order to process SCEP requests. Cryptographic identities are described in section 5.4.2.

When acting as a certificate authority, OpenOSP maintains a persistent serial number to ensure that the certificates issued are unique. The serial number is stored in a file whose location is specified by the "ca_serial_file" entry. The certificates that OpenOSP generates and issues are also stored in an LDAP-accessible directory.

ca_serial_file (no default, must be configured)

The name of the file where the OpenOSP certificate serial number is stored. Before running OpenOSP for the first time, the file specified here must be initialized to contain an arbitrary number in hex digits. A sample serial number file, which may be copied to the file named here, is included in the OpenOSP distribution – see samples/serial.txt.

ca_valid_duration (no default, must be configured)

The number of days for which the certificates issued by OpenOSP remain valid.

ca_directory (no default, must be configured)

The name of the host where an LDAP-accessible directory is running, in which OpenOSP should store the certificates that it generates and issues. The directory specified here may be set up without access control, such that it can be accessed without needing to supply a user name and password; otherwise, the ca_directory_user and ca_directory_password must also be configured. If the certificate directory server is running on the same computer as OpenOSP, this entry may be set to "localhost".

Optionally, the value for this entry may have the form <hostname>**:**<port>, for example **open-osp:400**, where the number after the colon specifies the port number to connect to. If the colon and port number are omitted, OpenOSP uses the default LDAP port number 389.

Note that the directory specified here may be the same as that specified below by the routing_directory entry in the [sample_application] group.

ca_directory_user (no default, must be configured **if using directory access control**)

The user name, as a DN, that should be used to bind to the certificate directory, if the directory is configured with access control.

ca_directory_password (no default, must be configured **if using directory access control**)

The password that should be used to bind to the certificate directory, if the directory is configured with access control.

ca_directory_dn_suffix (default value empty string)

A distinguished name suffix that OpenOSP appends to all DNs when accessing the certificate directory. This is normally left empty, but could be used to restrict the certificate directory to a subtree of some larger directory hierarchy.

ca_rsa_signing_digest (default value **md5**)

The digest that OpenOSP will use with the RSA algorithm to sign new certificates. The permitted values for this option are **md5** and **sha1**.

ca_subject_name_suffix (default value empty string)

The distinguished name suffix that OpenOSP appends to the subject name in a certificate request before creating a new certificate. This is normally set to a subset of the certificate authority's distinguished name.

For example, if the CA's distinguished name is 'cn=OpenOSP, ou=VOIP, o=Cisco, c=US' then ca_subject_name_suffix would be set to 'ou=VOIP, o=Cisco, c=US'. Then, a certificate request containing the relative distinguished name 'unstructuredName=router1' would result in a certificate having the fully-qualified distinguished name 'unstructuredName=router1, ou=VOIP, o=Cisco, c=US'.

smime_identity (no default, must be configured)

The DN of the cryptographic identity that OpenOSP uses when S/MIME signing OSP server responses. Cryptographic identities are described in section 5.4.2. The cryptographic identity used for S/MIME signing may be the same as that used for CA and SSL operations.

smime_default_digest (default value **sha1**)

The default digest algorithm that OpenOSP uses when S/MIME signing OSP server responses, in cases where the corresponding request was not signed. Valid values are **sha1** and **md5**.

Note that if the corresponding request was signed, OpenOSP signs the response using the digest algorithm that was used for the request.

random_seed_file (no default, must be configured)

The name of a file containing random data that OpenOSP uses to seed the random number generator in the crypto library. Note that this file must have write permissions, because OpenOSP writes the state of the random number generator back to this file when it exits.

threads_per_processor (default value **2**)

The number of connection handling threads that OpenOSP creates per processor. Normally it should not be necessary to change this from its default value. However, if you use OpenOSP with a custom application whose design is such that it can block the OpenOSP callback threads, you may achieve better overall performance by increasing this number.

5.3 Configuring the OpenOSP sample application

The OpenOSP sample application is configured by the following entries in the [sample_application] group.

token_signing_enabled (default value **yes**)

Controls whether the XML tokens generated during call authorization are signed. To disable XML token signing, set the value of this configuration entry to **no**.

token_signing_identity (no default, must be configured)

The DN of the cryptographic identity that the sample application uses when signing and verifying the tokens in AuthorizationRequest and AuthorizationResponse messages. Cryptographic identities are described in section 5.4.2.

routing_directory (no default, must be configured)

The name of the host where the LDAP-accessible directory containing the routing and pricing information database is running. The directory specified here **must** be set up without access control, such that it can be accessed without needing to supply a user name and password. If the routing directory server is running on the same computer as OpenOSP, this entry may be set to "localhost".

Optionally, the value for this entry may have the form <hostname>**:**<port>, for example **open-osp:400**, where the number after the colon specifies the port number to connect to. If the colon and port number are omitted, OpenOSP uses the default LDAP port number 389.

Note that the directory specified here may be the same as that specified above by the ca_directory entry in the [openosp] group.

usage_record_file (no default, must be configured)

The name of the file to which the sample application should write usage records, to record the receipt and processing of UsageIndication messages. The named file must be writable by the super-user.

max_usage_records (default value **1000**)

The maximum number of records that the sample application will write to the usage record file. When the number of records written reaches this number, the sample application discards the accumulated records and starts writing at the beginning of the usage record file. This is to avoid using an arbitrarily large amount of disk space.

authorized_call_duration (default value **NONE**)

The maximum call duration, in seconds, for calls authorized in an AuthorizationRequest/Response exchange. If this entry is not set, no usage information is included in the authorization, which means that call duration is unrestricted.

authorization_validity (default value **30**)

The time in seconds for which a call authorization, obtained during an AuthorizationRequest/Response exchange, remains valid. The validity period of the authorization is from

(T – authorization_validity)

until

(T + authorization_validity)

where T is the time that the authorization is issued.

service_url (no default, must be configured)

The URL that a client should use to send OSP requests to this server. Typically this has the form

http://<hostname>/settle

where <hostname> should be replaced by the name of the computer on which the OpenOSP server is running. The value of this configuration entry is included as the data for the <OSPServiceURL> elements in a CapabilitiesConfirmation response.

5.4 Configuring the security/PKI environment

In the current OpenOSP release, security and crypto functionality is based on function provided by the OpenSSL libraries. Much of the following description of how to bootstrap the security environment for OpenOSP therefore uses OpenSSL tools or contains OpenSSL-specific configuration details.

5.4.1 Seeding the random number generator

OpenSSL uses a value of at least 512 bytes in length to seed its random number generator. It reads this value at start up from a file that is configured by the random_seed_file entry in the OpenOSP configuration file, and also saves a new random seed to the same file when terminating.

For genuine randomness, it is important that every OpenOSP installation creates the initial contents of this file in some non-predetermined way. Hence the procedure is as follows.

- Choose a file name for the random number seed file, such as /usr/openosp/random.
- Copy any data into this file, as long as the data is at least 512 bytes long. One way of doing this is to choose any existing file larger than 512 bytes at random from the installation system, and to copy this to the random number seed file.
- Ensure that the random_seed_file entry in the OpenOSP configuration file points to this random number seed file.

5.4.2 Setting up cryptographic identities

The Security API component of the OpenOSP stack assumes a number of "cryptographic identities" when performing security operations. Each of the identities referenced by the following OpenOSP configuration entries

- [openosp] ssl_identity
- [openosp] ca_identity
- [sample_application] token_signing_identity

must be correctly set up before OpenOSP can run successfully. For a minimum PKI configuration, you need to configure a single identity and set all of the above configuration entries to the DN for this identity.

Available cryptographic identities are configured by the following entries in the [identities] group of the OpenOSP configuration, where N is replaced by a unique non-negative integer, starting from 0. See Appendix A for an example.

identity_N_dn (no default, must be configured)

> The name, in LDAP DN format, of the identity being configured, such as "CN=security,O=DCL" or "unstructuredName=tony.datcon.co.uk". This DN identifies the location of the identity's X.509 certificate, which includes the identity's public key, stored in the directory identified by ca_directory.

identity_N_privkey (no default, must be configured)

> The name of the file, including the complete path, which contains the private key for this identity stored in Privacy Enhanced Mail (PEM) format.

To configure an identity, you need to

- create a private key and certificate (described later in this section)
- store the certificate in the directory, using the DN of your choice (also described later in this section)
- add the above entries to the [identities] section of the file for this identity
- reference this identity's DN from one or more of ssl_identity, ca_identity and token_signing_identity.

5.4.3 CA key and certificate generation

For testing purposes a self-signed CA certificate and corresponding private key may be generated by the make_ca.sh script, distributed in the OpenOSP test subdirectory.

make_ca.sh generates a 1024 bit RSA private key in Privacy Enhanced Mail format, in the file cakey.pem, and an X.509 certificate in ASN.1 DER encoding, in the file cacert.der, that has a serial number of 1 and is valid for 10 years (3652 days).

Note: because of inconsistencies in the implementation of the OpenSSL utilities used by make_ca.sh, this utility requires the random seed file (described in section 5.4.1 above) to be configured in **two different ways**.

- The environment variable RANDFILE must be set to the random seed file name, and exported.
- The RANDFILE entry on line 9 of the OpenSSL configuration file (openssl.cnf) must also be set to the random seed file name.

As specified in section 5.4.1 above, you should ensure that the referenced random seed file exists and is at least 512 bytes long.

In order to generate the CA certificate correctly, you should first decide the directory DN where you intend to store the generated certificate, and then specify this DN in response to the prompts that make_ca.sh gives you. If you chosen DN does not include a value for one of the possible fields, such as locality, enter a single period in response to the corresponding prompt.

For example, suppose that you choose to store your CA certificate under the DN:

cn=OpenOSP, ou=VOIP, o=Cisco, c=US

The interaction with make_ca.sh should proceed as follows (user input is in bold):

$ **./make_ca.sh**
Using configuration from /usr/local/ssl/openssl.cnf
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'cakey.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]:**US**
State or Province Name (full name) [Some-State]:**.**
Locality Name (eg, city) []:**.**
Organization Name (eg, company) [Internet Widgits Pty Ltd]:**Cisco**
Organizational Unit Name (eg, section) []:**VOIP**
Common Name (eg, YOUR name) []:**OpenOSP**
Email Address []:**.**

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:**.**
An optional company name []:**.**
Signature ok
subject=/C=US/O=Cisco/OU=VOIP/CN=OpenOSP
Getting Private key
Signature ok
subject=/C=US/O=Cisco/OU=VOIP/CN=OpenOSP
Getting CA Private Key

Having generated a CA certificate, you should use ldapadd to add it to the certificate directory, as described below.

5.4.4 CA certificate extensions

When using the OpenSSL distribution crypto libraries, the certificates that OpenOSP generates when acting as a certificate authority can be further customized by entries in the [ca_cert_extensions] group. The sample configuration file shows a typical set of certificate extensions, including a CRLDP. For further details on configuring certificate extensions, see the file doc/openssl.txt in the OpenSSL distribution.

If you built OpenSSL with the directoryName format patch as described in section 4.1.4, the DirName entry in the [crldp1] group of the sample configuration file can be uncommented. OpenOSP will then add this value to new certificates as a CRLDP in directoryName format. Without the patch to OpenSSL, only the URI format may be used.

5.4.5 Storing certificates in the directory

The certificates that OpenOSP uses are stored in the directory in one of two ways.

- The X.509 certification authority certificate that OpenOSP uses when processing a SCEP request should be stored in binary format as a "cACertificate" attribute in an object with object class "certificationAuthority". OpenOSP must have a CA certificate stored in this way in order to run. (A minimal installation will have all the configurable identities using this certificate.)

- An X.509 certificate for a cryptographic identity other than the certification authority identity is stored in binary format as a "userCertificate" attribute in an object with object class "strongAuthenticationUser". The X.509 certificates generated by OpenOSP when acting as a certification authority are stored in this way. In a minimum installation, there is no requirement to configure any of these certificates.

The term "binary" refers here to LDAP binary format, as opposed to a syntax-checked format such as a number. Internally, the binary certificate data is organized according to the ASN.1 DER encoding format, as stated above.

The details of the mechanism for actually storing the certificate will vary from one directory product to another. If you are using the OpenLDAP directory, then you can use an LDIF file to load data into the directory.

Note that Cisco routers filter on 'cn' when retrieving a CRL from the CA's directory entry, so if you are using CRLs you should ensure that the CA's directory entry has a 'cn' attribute whose value matches that of the 'cn' component of its distinguished name. The sample LDIF file described below illustrates this.

Sample OpenLDAP schema and import files

Appendices B and C present a sample schema configuration file and LDIF data import file for the OpenLDAP directory server. These files are also included in the OpenOSP distribution – see samples/openldap.conf and samples/openldap.ldif.

Note that the standard LDAP directory schema includes the object classes certificationAuthority and strongAuthenticationUser and the attributes cACertificate and userCertificate, so no security-related schema extension is required.

5.5 Certificate revocation tool

OpenOSP includes a command line tool to allow revocation of a certificate previously issued by the sample certificate authority. This tool, called "crladmin", allows you to create, view and change the contents of the certificate revocation list (CRL) stored in your certificate authority's directory entry. It is invoked like this:

crladmin –c <*config_file*> [–n <*subject_name*>] [–u]
[–d <*days_to_next_crl*>] [–h <*hours_to_next_crl*>]
[–m <*minutes_to_next_crl*>]

–c <*config_file*>
The location of your OpenOSP configuration file. This parameter is always required because crladmin uses the configuration file to get information about your CA and the LDAP certificate directory.

–n <*subject_name*>
The subject name of a certificate to be revoked. This should be a distinguished name in standard LDAP text format (defined in RFC2253). For example:

CN=router1,OU=VOIP,O=Cisco,C=US

If the subject name contains spaces, it must be enclosed in double quotes.

–u
Forces an existing CRL to be updated with a new expiry date, without revoking any certificates.

–d <*days_to_next_crl*> –h <*hours_to_next_crl*> –m <*minutes_to_next_crl*>
The number of days, hours and minutes until the next CRL is due. These values are combined to calculate the expiry date of a new or updated CRL. The default validity period is 30 days.

5.5.1 Creating a new CRL

If your CA's directory entry does not yet contain a CRL, crladmin will generate an empty one for you. You must do this if you have configured OpenOSP to verify clients, because it will fail to verify a certificate if it cannot retrieve a CRL. The following example shows a new CRL being generated, with a validity of 30 days:

$ **./crladmin –c /usr/local/openosp.cnf –d 30 –h 0**

Looking up current CRL for CA 'CN=OpenOSP, OU=VOIP, O=Cisco, C=US'...
LDAP search returned no attribute values

CA does not have a CRL in the directory.
Do you want to create a new, empty CRL? [y] y

Using CA private key from '/usr/local/openosp/ospcakey.pem'
Looking up CA certificate...
The new CRL will expire in 30 days, 0 hours and 0 minutes

```
The signature digest will be MD5
The changes are about to be written to the directory. OK? [n] y
Storing CRL in directory...

Successfully created new CRL for 'CN=OpenOSP, OU=VOIP, O=Cisco, C=US'
```

If there are any problems, the output from crladmin should be enough to help you diagnose the problem. Once you have an empty CRL in your CA's directory entry, you can use crladmin either to display the contents of the current CRL or to revoke a certificate.

5.5.2 Displaying the current CRL

To display the contents of the current CRL, invoke crladmin using just the –c parameter as in the following example, which shows that an empty CRL is present.

```
$ ./crladmin –c /usr/local/openosp.cnf
Looking up current CRL for CA 'CN=OpenOSP, OU=VOIP, O=Cisco, C=US'...
Looking up CA certificate to verify CRL...
Verifying CRL...
CRL signature is OK
Certificate Revocation List (CRL):
        Version 1 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: /C=US/O=Cisco/OU=VOIP/CN=OpenOSP
        Last Update: Jul 19 16:50:45 2000 GMT
        Next Update: Aug 18 16:50:45 2000 GMT
No Revoked Certificates.
    Signature Algorithm: md5WithRSAEncryption
        <hex data>
```

5.5.3 Revoking a certificate

To revoke a certificate, specify the –n parameter together with the full distinguished name of the end entity whose certificate you want to revoke. You may also specify the number of days, hours and minutes until the next CRL is due using the –d, –h and –m parameters, with the default being 30 days. The following example shows how this is done.

```
$ ./crladmin –c /usr/local/openosp.cnf
        –n "unstructuredName=router1, ou=VOIP, o=Cisco, c=US"  –d 14  –h 0
Looking up current CRL for CA 'CN=OpenOSP,OU=VOIP,O=Cisco,C=US'...
Looking up CA certificate to verify CRL...
Verifying CRL...
CRL signature is OK

Using CA private key from '/usr/local/openosp/ospcakey.pem'
Looking up certificate to be revoked for 'unstructuredName=router1, ou=VOIP, o=Cisco,
c=US'...
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 11 (0xb)
        Signature Algorithm: md5WithRSAEncryption
```

```
        Issuer: C=US, O=Cisco, OU=VOIP, CN=OpenOSP
        Validity
            Not Before: Jul 18 13:55:17 2000 GMT
            Not After : Jul 18 13:55:17 2001 GMT
        Subject: C=US, O=Cisco, OU=VOIP, unstructuredName=router1
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (512 bit)
                Modulus (512 bit):
                    <hex data>
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 CRL Distribution Points:
                URI:ldap://buffy/CN=OpenOSP,OU=VOIP,O=Cisco,C=US?certificateRevocationList

    Signature Algorithm: md5WithRSAEncryption
        <hex data>

The above certificate is to be revoked
Are you sure? [n] y
Updating CRL...
The new CRL will expire in 14 days, 0 hours and 0 minutes
The signature digest will be MD5
The changes are about to be written to the directory. OK? [n] y
Storing CRL in directory...
Deleting revoked certificate...
Successfully revoked certificate for 'unstructuredName=router1, ou=VOIP, o=Cisco,
c=US'
```

Note that when crladmin revokes a certificate, it deletes that certificate from the LDAP directory to avoid it being used inadvertently and to allow a new certificate to take its place.

5.6 Configuring routing and pricing information

The OpenOSP sample application responds to OSP AuthorizationRequest messages by referring to a database of routing and pricing information that is stored in an LDAP-accessible directory.

The detailed procedure for populating an appropriate database depends on your directory server software, but in most cases it consists of two steps:

- extending the directory schema so that it can accommodate the information to be stored
- importing data according to the extended schema.

For the OpenLDAP directory server, these steps are accomplished by modifying the OpenLDAP configuration file, and by importing data using an LDIF file. Examples of both files for OpenLDAP are included in Appendices B and C and also in the OpenOSP distribution – see samples/openldap.conf and samples/openldap.ldif.

The following subsection specifies the schema requirements for the routing and pricing information database.

5.6.1 Schema requirements for routing and pricing information

Routing and pricing information is associated with gateways provided by internet telephony service providers (ITSPs). Therefore this information is arranged hierarchically in the directory in "gateway" objects that are immediately subordinate to "itsp" objects. For example:

arbitrary object in directory tree

itsp object with itspname=British Telecom

gateway object with gatewayname=london1

gateway object with gatewayname=cardiff1

etc.

itsp object with itspname=MCI

gateway object with gatewayname=eastcoast1

gateway object with gatewayname=westcoast1

etc.

etc.

The **itsp** object class contains

- a mandatory itspname attribute, which names the ITSP
- an optional description attribute, which contains an extended description of the ITSP.

The **gateway** object class contains

- a mandatory gatewayname attribute, which names the gateway
- a mandatory ip_address attribute, which contains the gateway's IP address
- an optional description attribute, which contains an extended description of the gateway
- an e164 attribute, which contains one or more E164 telephone number prefixes for destinations that this gateway can reach

- a pricing attribute, which contains the pricing rate in USD/min and an optional validity period for each of the E164 telephone number prefixes specified by the e164 attribute

- an optional almost_out attribute, with value “true” or “false”, indicating whether or not the gateway is almost out of resources

- optional protocol_type attributes, with values indicating the protocols that are supported by the gateway

- optional service_type attributes, with values indicating the services that are supported by the gateway.

All these attributes are case-insensitive strings except for e164, which is a telephone number.

See the gateway entries in the sample LDIF file of Appendix C for examples of different gateway configurations.

The value of the ip_address attribute may be either the fully qualified domain name of the gateway, or the IP address enclosed in square brackets. Either form may optionally be followed by the port, separated with a colon. For example,

- “londonnorth.bt.com” specifies the domain name of a gateway using the default port

- “[192.168.1.12]:81” specifies the IP address of the gateway explicitly, and that it is listening on port 81.

A value for the e164 attribute may be any initial substring of an E164 telephone number. For example,

- “4”, to match telephone numbers in all countries with country code beginning with “4”

- “367”, to match any E164 telephone number beginning “367”, even if “367” only includes a partial country or area code.

Each value for the pricing attribute consists of

<e164>@<rate>F<validity start time>U<validity end time>

where

<e164> is the corresponding e164 attribute value

<rate> is the pricing rate is USD/min

<validity start time> is the (optional) time before which this price is invalid, in OSP Timestamp format

<validity end time> is the (optional) time after which this price is invalid, in OSP Timestamp format.

The OSP Timestamp format is defined by the OSP specification. It takes the form

<YYYY>–<MM>–<DD>T<HH>:<MM>:<SS>Z

for example “2000-06-01T13:16:16Z”. Hence a complete pricing attribute looks like this:

0114420@5F2000-06-01T13:16:16ZU2001-12-31T23:59:59Z

If no validity information is present, the pricing information is assumed to be valid indefinitely.

The OpenOSP sample applications uses these gateway attributes to generate a list of possible destination gateways in response to a routing request. The algorithm is as follows.

- OpenOSP searches the directory to get a list of candidate gateways.
    - A gateway is a candidate if it has an e164 attribute which is a prefix of the destination number in the routing request. For example, for a target telephone number of 40852560, e164=408525 is a candidate, while e164=408526 is not.
    - If the routing request included a <ServiceType> element, a candidate gateway must, in addition to the above, either have no service_type attribute values at all, or must have a service_type attribute value matching the specified <ServiceType>.
    - If the routing request included a <ProtocolType> element, a candidate gateway must, in addition to the above, have a protocol_type attribute value matching the specified <ProtocolType>.
- If any DestinationAlternates were specified by the routing request, candidate gateways that are not in the DestinationAlternate list are immediately discarded.
- The remaining candidate gateways are ordered first by price, then by length of the matching prefix. Thus if the prices are equal, a gateway with e164=408525 will come ahead of one with e164=408. Where there is no pricing attribute value that corresponds to a matched e164 attribute value, the price is considered to be “very expensive”.
- OpenOSP generates a response including information about the “best” candidate gateways, up to the requested maximum number of gateways given by the <MaximumDestinations> element.

Note that, in the OpenOSP V2.0 server, existing gateway objects can be updated by the OSP CapabilitiesIndication and PricingIndication messages.

- CapabilitiesIndication messages must contain a DeviceInfo of type transport containing the gateway's IP address as stored in the ip_address attribute.
- PricingIndication messages must include a SourceInfo of type transport containing the gateway's IP address as stored in the ip_address attribute.

Changes to gateway capabilities and pricing are only possible for existing gateways and destination phone numbers – it is not possible to add a new gateway or destination through the CapabilitiesIndication and PricingIndication mechanisms.

5.6.2 Schema requirements for subscriber information

Subscriber information is stored in "subscriber" objects that, like gateways, are immediately subordinate to "itsp" objects.

A **subscriber** object contains

- an objectclass attribute with value "subscriber"
- a subscriber attribute containing the subscriber's name
- a subscriber_id attribute containing the subscriber's ID.

The OpenOSP sample application implements only a very basic subscriber authentication algorithm. It searches for a **subscriber** object whose subscriber_id attribute value matches the data supplied in the OSP authentication request's <SourceAlternate type="subscriber"> element. If such an object is found, the subscriber is authenticated.

5.7 Usage notes for the OpenLDAP directory server

5.7.1 Starting the OpenLDAP directory server

The OpenLDAP directory server application is "slapd".

To start it, type the following command as root:

$ /usr/local/libexec/slapd –f <OpenLDAP config file> –p <port number>

where <OpenLDAP config file> is replaced by the full path to your OpenLDAP configuration file (the file that defines the schema for the directory server), and <port number> is replaced by the port number that slapd should listen on for LDAP connections. "–p <port number>" may be omitted, in which case slapd defaults to listening on port 389.

5.7.2 Importing data into the directory server

Note that the directory server must already be running when importing data.

To import data specified in an LDIF file into the directory server, use "ldapadd" like this:

$ ldapadd –v –b –D "cn=root, dbname=OpenOSP" –w password < [LDIF file]

where [LDIF file] is replaced by the path to the LDIF file containing the data that you want to import. The options in this command line are as follows:

–v provide verbose output

–b interpret file names beginning with a slash in the LDIF file as referring to data contained in the referenced files

–D, –w bind to the directory server using the specified name and password; these are only required if you have configured your server to use access control (which is not recommended for use with OpenOSP) or if the data to be imported includes a top-level node (as does, for example, the sample LDIF file in Appendix C).

Data, once imported, is persistent – it is retained if you stop slapd and then restart it. Therefore you shouldn't run ldapadd twice with the same file. If you do run ldapadd twice, it can result in all your attributes being multi-valued, which could cause later problems for the OpenOSP server.

5.7.3 LDIF file format

The following notes on the required format of an LDIF file may not be exhaustive; they are simply what we have inferred from experience.

- An LDIF file cannot contain any comments.
- Each group of lines beginning "dn:" specifies an object to add to the directory and the attributes for that object. These groups of lines must be separated from each other by a blank line. Without the blank separator lines, ldapadd adds the whole LDIF file as a single object.

5.7.4 Searching for directory entries

To search for entries in the OpenLDAP directory, use the ldapsearch command.

For example, to search for all objects in the directory:

```
$ ldapsearch –L –b "dbname=OpenOSP" "(objectClass=*)"
```

Or to search for all objects with an itspname attribute:

```
$ ldapsearch –L –b "dbname=OpenOSP" "(itspname=*)"
```

The options in these command lines are as follows:

–L	display search results in a modified format
–b	use the specified DN as the root point for the search
filter	search for objects that match the specified filter

5.7.5 Deleting directory entries

To delete a particular directory entry, use the ldapdelete command.

Example 1 – deleting a certificate:

```
$ ldapdelete –D "cn=root, dbname=OpenOSP" –w password "unstructuredName=open-osp.cisco.com, dbname=OpenOSP"
```

Example 2 – deleting a gateway entry:

```
$ ldapdelete –D "cn=root, dbname=OpenOSP" –w password "gatewayname=tickle,
itspname=BT, dbname=OpenOSP"
```

To delete the entire directory contents:

- stop the directory server using “kill [slapd process ID]”
- delete all the .dbb files in the slapd database directory (this directory is identified by the “directory” entry in the OpenLDAP configuration file)
- restart the directory server as per section 5.7.1 above.

Data may then be reimported as required using ldapadd.

5.7.6 Getting further help with OpenLDAP

For further help with OpenLDAP issues, please refer to the extensive online OpenLDAP documentation at http://www.openldap.org/faq/data/cache/1.html.

The documentation for ldapadd is at http://www.openldap.org/software/man.cgi?query=ldapadd.

5.8 Notes for interoperability configuration

This section lists additional notes that relate to configuration for interoperability testing between the OpenOSP server and various clients. The list is not exhaustive; it simply reflects our experiences to date.

5.8.1 Requirements for clock synchronization

When testing interoperability with Cisco routers, we have found that the clocks of the server and the routers must be very closely synchronized with each other.

For example, in call setup for a telephone call, if the terminating gateway (TGW) clock is behind the server's clock, the TGW will decide that the token it receives on an AuthorizationResponse is not yet valid, trace an error accordingly, and refuse to permit the call to continue.

5.9 Starting or stopping related server applications

Before starting OpenOSP, make sure that

- there are no other applications running – typically web servers – that are listening on the ports that OpenOSP is configured to listen on (normally 80 and 443)
- all the LDAP directory servers referenced in the OpenOSP configuration file are running.

5.10 Starting the OpenOSP sample application

Start the OpenOSP stack and sample application as follows.

- Use “su” to become root if you are not already logged in as root. Note that OpenOSP must run as root so that it can listen on the privileged TCP/IP port numbers for HTTP (80) and secure HTTP (443).

- Set and export the OSP_CONFIG_FILE variable in your environment so that it references the OpenOSP configuration file. For example: “OSP_CONFIG_FILE=/usr/openosp/osp.cnf; export OSP_CONFIG_FILE”.

- Run the OpenOSP sample application executable.

In normal operation, OpenOSP prints the following lines to standard output, to indicate that it has initialized successfully.

```
ospd starting…
ospd ready
Command options:
q - terminate server
s - connection stats
```

In this state, OpenOSP processes OSP and SCEP requests that are received on the ports specified in the configuration file.

To terminate OpenOSP, press q followed by <Return>. OpenOSP then prints the following lines to indicate successful termination.

```
ospd stopping…
ospd stopped
```

While OpenOSP is in its normal processing (ready) state, it is also possible to obtain a summary of server statistics equivalent to that returned by the advertized osp_get_stack_statistics() interface. To do this, press s followed by <Return>. OpenOSP responds by printing output in the following form.

```
Stack statistics:
    Non-SSL connections:    1
    SSL connections:        5
    OSP requests:           11
    SCEP requests:          0
```

5.10.1 OpenOSP trace output and simple debugging

OpenOSP generates a trace file called osptrc in its working directory. The debug build of OpenOSP produces a lot of detailed trace output to help in debugging any problems that may arise. The release build produces trace output only in case of error or unexpected occurrences.

If something appears to be going wrong, load the trace file into an editor and look for lines in this file that are marked with an exclamation mark (!) or an asterisk (*) – these indicate problems. The text on these lines should give you a clue about what is wrong.

References

The following references provide further information on

- related OpenOSP documents
- the OSP protocol.

MSM-0005-0103	OpenOSP Product Overview
MOM-101-0102	OpenOSP Interface Specification
ETSI TS 101 321 v2.1.0	Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Open Settlement Protocol (OSP) for Inter-Domain pricing, authorization, and usage exchange

Appendix A Sample OpenOSP configuration file

This sample configuration file is a copy of the file samples/openosp.cnf in the OpenOSP distribution.

```
# openosp.cnf
#
# OpenOSP configuration file
#
# (C) COPYRIGHT DATA CONNECTION LIMITED 2000
#
# $Revision::   1.3               $ $Modtime::   Jul 19 2000 13:20:22   $
#

#
# The identity list binds distinguished names to private keys
#
[ identities ]
identity_0_dn = CN=buffy,O=DCL,C=GB
identity_0_privkey = /usr/openosp/ospcakey.pem

#
# OpenOSP stack configuration
#
[ openosp ]
random_seed_file = /usr/local/openosp/random

non_ssl_port = 80
ssl_port = 443

ssl_identity = CN=buffy,O=DCL,C=GB
ssl_verify_client = no
ssl_cipher_list = DEFAULT

smime_identity = CN=buffy,O=DCL,C=GB
smime_default_digest = sha1

ca_identity = CN=buffy,O=DCL,C=GB
ca_serial_file = /usr/openosp/serial.txt
ca_valid_duration = 365
ca_rsa_signing_digest = md5
ca_subject_name_suffix = O=DCL,C=GB

ca_directory = buffy
ca_directory_user = ""
ca_directory_password = ""
ca_directory_dn_suffix = ""

#
# Extensions that the CA function will add to new certificates;
# see doc/openssl.txt in the OpenSSL distribution for details.
# Note that a semicolon, not a comma, must be used in crlDistributionPoints
#
[ ca_cert_extensions ]
basicConstraints = CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment
crlDistributionPoints = @crldp1
```

```
[ crldp1 ]
#DirName = CN=buffy,O=DCL,C=GB
URI = ldap://buffy/CN=buffy,O=DCL,C=GB?certificateRevocationList

#
# Sample application configuration
#
[ sample_application ]
routing_directory = buffy
token_signing_identity = CN=buffy,O=DCL,C=GB
authorized_call_duration = 3600
authorization_validity = 300
usage_record_file = /usr/openosp/usage.txt
max_usage_records = 1000
service_url = buffy.datcon.co.uk
```

Appendix B Sample OpenLDAP configuration file

This sample OpenLDAP configuration file is a copy of the file samples/openldap.conf in the OpenOSP distribution.

```
#
# Standard attributes and object classes
# This assumes that you're running from servers/slapd/release
#
include         ../slapd.at.conf
include         ../slapd.oc.conf

#
# Object classes required for OpenOSP database
#
attribute       dbname                  cis
attribute       itspname                cis
attribute       gatewayname             cis
attribute       ip_address              cis
attribute       e164                    tel
attribute       almost_out              cis
attribute       pricing                 cis
attribute       protocol_type           cis
attribute       service_type            cis
attribute       subscriber              cis
attribute       subscriber_id           cis

#
# An ITSP (for the sample application)
#
objectClass itsp
        requires
                objectClass,
                itspname
        allows
                description

#
# A gateway (for the sample application)
#
objectClass gateway
        requires
                objectClass,
                gatewayname,
                ip_address
        allows
                description,
                e164,
                almost_out,
                pricing,
                protocol_type,
                service_type,
```

```
#
# A subscriber (for the sample application)
#
objectClass subscriber
        requires
                objectClass,
                subscriber,
                subscriber_id


########################################################################
# ldbm database definitions
########################################################################
database        ldbm
directory       /usr/openosp
suffix          "dbname=OpenOSP"
suffix          "C=GB"

sizelimit       1000
schemacheck     off

# 'e164' is the attribute on which the sample application searches
# most of the time, so we ensure that the database is indexed on it
index           e164

rootdn          "cn=root, dbname=OpenOSP"
rootpw          password

access          to * by * write
access          to * by * read
```

Appendix C Sample OpenLDAP data import file

This sample configuration file is a copy of the file samples/openldap.ldif in the OpenOSP distribution.

```
dn: dbname=OpenOSP
objectClass: top
dbname: OpenOSP

dn: itspname=BT, dbname=OpenOSP
objectClass: itsp
itspname: BT
description: British Telecom

dn: itspname=MCI, dbname=OpenOSP
objectClass: itsp
itspname: MCI
description: MCI WorldCom

dn: itspname=Sprint, dbname=OpenOSP
objectClass: itsp
itspname: Sprint
description: Sprint Telecom

dn: gatewayname=tickle, itspname=BT, dbname=OpenOSP
objectClass: gateway
gatewayname: tickle
ip_address: tickle.bt.co.uk
e164: 44
pricing: 44@1F2000-06-01T13:16:16ZU2001-12-31T23:59:59Z
protocol_type: sip
protocol_type: h323

dn: gatewayname=patch, itspname=BT, dbname=OpenOSP
objectClass: gateway
gatewayname: patch
ip_address: patch.bt.co.uk
e164: 44171
e164: 44181
e164: 4420
pricing: 44171@0.5
pricing: 4420@0.8
protocol_type: sip
protocol_type: h323

dn: subscriber=oking, itspname=BT, dbname=OpenOSP
objectClass: subscriber
subscriber: Ollie King
subscriber_id: 887

dn: gatewayname=ren, itspname=MCI, dbname=OpenOSP
objectClass: gateway
gatewayname: ren
ip_address: ren.mci.com
e164: 1
pricing: 1@3
protocol_type: sip
```

```
dn: gatewayname=stimpy, itspname=MCI, dbname=OpenOSP
objectClass: gateway
gatewayname: stimpy
ip_address: stimpy.mci.com
e164: 1703
e164: 1508
e164: 1492
pricing: 1703@5
pricing: 1508@6
pricing: 1492@7.5

dn: gatewayname=milo, itspname=sprint, dbname=OpenOSP
objectClass: gateway
gatewayname: milo
ip_address: milo.sprint.com:82
e164: 44802
e164: 447713
e164: 447713019
pricing: 44802@5F2000-06-01T13:16:16ZU2001-12-31T23:59:59Z
pricing: 447713@3F2000-06-01T13:16:16ZU2001-12-31T23:59:59Z
pricing: 447713019@2F2000-06-01T13:16:16ZU2001-12-31T23:59:59Z

dn: gatewayname=jake, itspname=sprint, dbname=OpenOSP
objectClass: gateway
gatewayname: jake
ip_address: [192.168.1.12]
e164: 44976
e164: 447000
e164: 447713019543
protocol_type: h323

dn: subscriber=njerram, itspname=sprint, dbname=OpenOSP
objectClass: subscriber
subscriber: Neil Jerram
subscriber_id: 1005

dn: subscriber=mrichards, itspname=sprint, dbname=OpenOSP
objectClass: subscriber
subscriber: Martin Richards
subscriber_id: 1089

dn: C=GB
objectClass: top
objectClass: country
c: Great Britain

dn: O=DCL,C=GB
objectClass: top
objectClass: organization
o: Data Connection Ltd.

dn: CN=buffy,O=DCL,C=GB
objectClass: top
objectClass: certificationAuthority
cn: buffy
cACertificate: /usr/openosp/ospcacrt.der
```