OEM Customization

This page describes how you can customize OpenOSP to suit your implementation requirements. The areas you may want to customize are:

- the sample settlement application
- the security support
- the resource management
- the Secure Sockets Layer (SSL)

Sample Settlement Application

This section describes the sample settlement application which is included in the OpenOSP distribution.

The sample application provides basic billing functionalities such as call detail record (CDR) generation, subscriber authentication and authorization, client verification and non-repudiation. It is fully tested to the same standard as the rest of the stack. The reason this application is considered a "sample" is that it implements functionalities that an original equipment manufacturer (OEM) is likely to replace when integrating the OpenOSP implementation with existing systems for billing, routing, certificate management, etc.

The following sections describe the components of the sample application, and provide guidelines for OEMs who may want to modify them.

Usage Metering

The usage metering component writes a CDR containing the complete contents of each received UsageIndication message to a text file. The file format is similar to that used in Annex F of the v2 OSP specification, but is extended to include all fields in the received message, including the enhanced usage parameters defined in Annex C of the specification. CDRs in the text file can be exported to a third party billing application for customer bill generation.

It is expected that an OEM will replace this with code that either interacts dynamically with a settlement back office system, or writes OEM-specific format CDRs.

Authorization, Routing, Pricing and Capabilities

This component handles authorization, routing, pricing and capabilities exchanges, using a Light Weight Access Protocol (LDAP) directory to store information required by the routing algorithm. The key objects in the directory include the following:

- Service providers, which are hierarchically above the gateways in the directory — each gateway appears in the directory under the service provider that owns it.
- Gateways, which have the following attributes:

- An IP address that is used to identify the gateway
- One or more partial e164 phone numbers indicating the destination numbers that it can reach — these partial numbers may be of any length e.g., 1 for the US, 1703 for Virginia, 1703715 for Reston, which is a town in Virginia
- Pricing information (in US$) for call services terminated by the gateway for a given destination number

The directory also includes information used in subscriber authentication. See the next section for details.

Destination Ranking

On receipt of an authorization request, the server finds all entries in the directory that match the required destination, and ranks them in the following order, with the highest priority first:

1. Gateways with plenty of resources are ranked ahead of those that have exhausted their resources
2. Gateways that the client requested are ranked ahead of gateways that were not requested
3. Price to terminate the call, the cheapest are ranked first
4. Quality of match on destination address: i.e. a gateway that advertises "703715" comes ahead of "703" for matching on destination number "7037154914".

The server then returns signed XML tokens for these routes to the client.

Subscriber Authentication

This component provides a mechanism to authenticate the identity of a subscriber. It uses an extension to the aforementioned LDAP directory . For each service provider, there may be a list of subscribers configured. Each subscriber entry is identified by a "subscriber ID" attribute.

On receipt of a SubscriberAuthenticationRequest, the sample application searches the directory for an entry with a subscriber ID matching the request, and returns a signed XML token to the client. If no match is found, it returns a failure response.

It is expected that OEMs will replace this scheme with some alternate scheme for remotely querying the private subscriber records of each service provider.

Client Verification and Non-Repudiation

This component provides a mechanism for OSP clients (OSP enabled gateways) to subscribe to an OSP server. It listens to notifications from the OpenOSP stack each time a client connects or disconnects. It does not take any action to verify clients. All client connections are accepted.

In addition, this component provides a basic non-repudiation scheme where all signed requests from OSP clients are recorded in a file. Records of all OSP requests can be used to reconcile differences in call records from difference OSP servers or even service providers.

Security Support

The sample cryptography and certificates manager (CCM) provides a basic mechanism to manage a pool of secure certificates. It is implemented using the facilities of the OpenSSL distribution. It also makes use of OpenLDAP to store and retrieve certificates and certificate revocation lists (CRLs) to and from an LDAP-accessible directory. As such, its Certificate Authority (CA) function may be suitable for large-scale deployment if the directory schema, described below, is deemed appropriate for a particular OEM.

End entity certificates are stored and retrieved using the subject's distinguished name (DN) and the `userCertificate;binary` attribute, which is defined in the standard LDAP schema. Certificate authority (CA) certificates are treated similarly, except that the `cACertificate;binary` attribute is used. Only one certificate is expected to be stored per end entity or CA. CRLs are stored and retrieved using the issuer's DN and the `certificateRevocationList;binary` attribute, which is also defined in the standard LDAP schema.

The sample CCM provides the following configuration options:

- It can add a suffix to a DN while using it for LDAP access purposes. This allows certificates and CRLs to use a naming hierarchy that is a subset of the full directory hierarchy.
- When issuing new certificates, it can expand a relative distinguished name (RDN) in the certificate request's subject field to a fully-qualified distinguished name (FQDN) which is inserted into the issued certificate. This allows CCM to define the name space in which certificates will be issued. The LDAP access suffix described above is independent of this name expansion.
- It can include a CRL Distribution Point (CRLDP) in each newly-issued certificate. The CRLDP points to a location from which a suitable CRL may be retrieved, and will typically be an LDAP URL to the CA's directory entry.
- If RSA certificates are in use, new certificates can be signed using either the MD5 or the SHA-1 digest algorithm.

If an OEM wants to replace the sample CCM in order to integrate with some existing PKI system, then it should ensure that any certificates issued by CCM via SCEP are put into an LDAP-accessible directory (with a schema compatible with that described above), as well as in whatever underlying storage is used for the PKI. Alternatively, the OEM could implement the alternate (but non-preferred) solution of supporting the GetCert SCEP

message. Similarly, the OEM should ensure that there is an LDAP-accessible CRL distribution point, or implement the alternate (non-preferred) GetCRL message.

The sample CCM uses OpenSSL's random number generator as a random source. If a different random number generator is required (for example, a hardware-based source), OpenSSL can be configured to route all random number requests to externally-supplied callbacks. The external random source will then be used for all random data required by any part of OpenOSP.

Resource Management

The sample resource manager implements the most simple policy.

- When resources are requested, it allows the OS to determine whether the request should be satisified (for example, a request for memory is passed directly to `malloc()`).
- When resources are released, they are released directly to the OS.

OEMs implementing the OpenOSP server on a system that also has other functions may want to implement some kind of threshold system to prevent the OSP server from consuming all the system's resources.

Secure Sockets Layer

The SSL API is implemented by OpenSSL. This handles SSL and Transport Layer Security (TLS) connections, including renegotiation and session ID re-use, as well as providing other security-related routines for the sample cryptography and certificates management library.

OEMs can replace OpenSSL with an alternative SSL / TLS implementation if required, for example when integrating into an existing system.