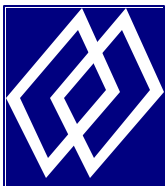

TransNexus

OSP Toolkit

Introduction

Release 2.5.5

09 February 2002

OSP Toolkit

Introduction

Release 2.5.5

09 February 2002

Document 0300-1211-0009

TransNexus
1140 Hammond Drive, Building E
Suite 5250
Atlanta, GA 30328
USA

Phone: +1 770 671 1888
Fax: +1 770 671 1188

E-mail: support@transnexus.com

Contents

Introduction

This document provides an overview of release 2.5.5 of the Open Settlement Protocol (OSP) Toolkit. That Toolkit, freely available under license from TransNexus, contains an implementation of the standard settlement protocol endorsed by the European Telecommunications Standards Institute (ETSI) and the International Multimedia Teleconferencing Consortium's Voice over IP (VoIP) Forum. The Toolkit also implements, as an option, extensions to the standard that allow access to enhanced services.

Five main sections comprise this document. The first summarizes the functionality the Toolkit provides. It is followed by a description of the software that makes up the Toolkit, including language, development methodologies, and internal architecture. The document then describes additional components that must be combined with the Toolkit for a complete implementation. That section also identifies potential sources for the additional components. The fourth section summarizes the programming interface exposed by the Toolkit. This interface is the means by which other software components can access the Toolkit. The final section is a document roadmap. It identifies the other documents that are part of the Toolkit, and offers suggestions on how users of the Toolkit can gain proficiency most effectively.

Functionality

The Open Settlement Protocol Toolkit provides complete authorization and usage reporting between Internet Telephony gateways or their controllers (e.g. H.323 gatekeepers, SIP proxy servers, SGCP call agents, …) and settlement service providers. The protocol conforms to ETSI and VoIP Forum standards for settlement services. The Toolkit includes a software library designed for integration into gateway and controller products. That library includes functionality to:

- Securely query a settlement service provider for authorization tokens and, optionally, call routing information needed to complete a telephone or facsimile call.
- Verify the authenticity of authorization tokens presented to a system as part of a call setup request from a peer system.
- Securely report usage information for a call to a settlement service provider.

In addition, the Toolkit library can optionally support extensions to the standard protocol. These extensions, which are fully compliant with the associated protocol standards allow for enhanced services. The additional functions include:

- The ability to perform a real-time assessment of network quality of service to various potential peer devices and re-prioritize the peer selection based on that assessment.
- Reporting of failed as well as successful call details.
- Processing of network performance statistics gathered during a call.

- Reporting of additional usage details (principally network performance statistics) to a settlement service provider.

- Enabling or disabling the generation of digitally signed audits of usage details.

Provided Software

The software development kit includes full, well-documented source code for a complete implementation of the Open Settlement Protocol. The Toolkit libraries have been exhaustively tested in Solaris 2.7, Solaris 2.8, and Windows NT 4.0 environments. This section highlights the language and conventions, environmental dependencies, and internal architecture of the Toolkit software.

Language and Conventions

Software provided in the OSP Toolkit is written entirely in the ANSI standard C programming language. All source modules compile without warnings and are checked using additional quality tools (e.g. commercial lint packages).

The Toolkit source code is completely thread-safe and can be employed in a multi-threaded environment.

Environmental Dependencies

The Toolkit software requires some services from an underlying operating system or real time environment. Those services include basic memory management (both for local data structures and for message buffers), network input and output, thread management, and time-of-day services. The Toolkit includes simple functions that implement these services based on POSIX-compliant libraries. These functions are developed as separate modules in the Toolkit, and can be easily replaced by services of other operating environments.

Internal Architecture

Although implemented entirely in ANSI C, the OSP Toolkit software relies on an object-oriented design. The major objects comprising the library include:

- **Settlement Provider**. The provider object serves as the parent for all other objects in the library. It abstracts the information needed to communicate with a single settlement service provider, including communication and cryptographic parameters. The Toolkit programming interface exposes some member functions of the provider object.

- **Settlement Transaction**. The transaction object abstracts a particular settlement transaction, most typically a telephone or facsimile call. Member functions of this object are exposed by the Toolkit programming interface, and applications use a transaction object to request and report information to a settlement provider.

- **MIME Message**. The MIME message object provides methods to construct and parse MIME messages, combining or extracting XML documents and SMIME signatures.

- **XML Document**. The XML document object encompasses the actual messages passed between applications and settlement providers. This object includes methods to convert canonical XML to and from binary data structures.

- **SMIME Signature**. The SMIME signature object provides methods to create and to validate SMIME-format digital signatures.

- **Communications Manager**. The communications manager object abstracts the communication interface between the application and a settlement service provider.

- **SSL Session**. The SSL session object, a child of the communication manager object, maintains information about SSL sessions established with the settlement provider. Under appropriate conditions, these sessions may persist beyond the life of an HTTP connection object, allowing re-use of negotiated SSL parameters.

- **HTTP Connection**. The HTTP connection object represents a HTTP connection with a settlement provider. HTTP connections have (configurable) persistence, so that a single HTTP connection may be used for multiple settlement transactions.

Additional Components

In addition to the Toolkit itself, a complete implementation of the Open Settlement Protocol requires two additional software components. Those components are necessary to provide Secure Socket Layer (SSL) services and cryptographic algorithms. TransNexus does not provide that software as part of its Toolkit. The following subsections detail those components, and they identify potential sources.

Secure Sockets Layer

Secure Sockets Layer (SSL) is a protocol that provides a secure channel for communications between clients and servers. SSL defines mechanisms to authenticate both parties in a communication and to encrypt and decrypt their exchanges. The Open Settlement Protocol requires SSL version 3.0, and an SSL implementation is a required addition to the Toolkit. At the time this document was written, TransNexus was aware of the following sources for SSL implementations (listed in alphabetical order):

- **Baltimore**. Baltimore offers both C and Java implementations of SSL in its C/SSL and J/SSL Developer Toolkits. Information is available at http://www.baltimore.com.

- **Certicom**. Certicom Corporation offers a commercial implementation of SSL known as SSL Plus. Licensing information is available at http://www.certicom.com.

- **Eric Young**. Eric Young has developed an implementation of SSL known as SSLeay that, in most circumstances, may be licensed free of charge. As of this writing, SSLeay was available at ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL.

- **Microsoft Corporation**. Microsoft includes an SSL implementation as part of its WinInet control. As of this writing, more information on this is available from Microsoft at http://msdn.microsoft.com/workshop/networking/wininet/wininet.asp.

In addition, Microsoft has unofficially indicated that it will soon make available a public interface to SCHANNEL.DLL, the SSL implementation used by Microsoft applications such as IIS.

- **OpenSSL**. OpenSSL is an open source implementation of the SSL protocol. More information is available at http://www.openssl.org.

- **RSA Data Security**. RSA Data Security offers a commercial implementation of SSL known as SSL-C. Licensing information is available from RSA at http://www.rsa.com.

- **Spyrus/Terisa**. Spyrus/Terisa offers commercial implementations of SSL (and HTTP) in its TLS Gold SSL Toolkit, and Device SSL Toolkit for embedded systems. For details, refer to http://www.spyrus.com.

As delivered, the TransNexus Open Settlement Protocol Toolkit supports the programming interface of OpenSSL. The Toolkit's SSL interface is modularized; however, to permit easy porting to other SSL libraries.

Cryptographic Algorithms

The Open Settlement Protocol relies on special cryptographic algorithms for message digesting, digital signature generation and verification, and encryption and decryption. At the time this document was written, sources for those cryptographic algorithms include the following.

- **Baltimore**. Baltimore offers cryptographic algorithms in its J/Crypto and Crypto Systems Toolkit products, in Java and C respectively. Licensing information is available at http://www.baltimore.ie.

- **Microsoft Corporation**. Microsoft includes a complete implementation of the required cryptographic algorithms as part of Windows 98, Windows NT 4.0 (beginning with SP3) and Internet Explorer 4.0 for Windows NT, Windows 98, and Windows 95. These algorithms are exposed by version 2.0 of the CryptoAPI interface, which is described at http://www.microsoft.com/security.

- **RSA Data Security**. RSA Data Security provides a reference implementation of some cryptographic algorithms (RSARef) and a commercial software library of cryptographic algorithms (BSAFE). Availability and licensing information for these products is available at http://www.rsa.com.

- **OpenSSL**. OpenSSL is an open source implementation of the SSL protocol. More information is available at http://www.openssl.org.

As delivered, the Open Settlement Protocol Toolkit supports the OpenSSL programming interface. Cryptographic interfaces are modularized, however, to permit easy porting to other libraries.

Programming Interface

The Open Settlement Protocol Toolkit presents a programming interface that reflects its object-oriented design. External applications view the Toolkit library as two main objects—providers and transactions. The provider object abstracts an interface to a specific settlement provider, while the transaction object represents a complete settlement transaction.

Provider Object

Before an application can make use of the Toolkit library services, it must initialize and configure a provider object. As part of this initialization, the application performs the following functions:

- Identify settlement provider servers.
- Install cryptographic keys and specify cryptographic algorithms.
- Configure network parameters (e.g. connection and session timeouts, persistence, retry limits, maximum number of simultaneous connections, etc.).

Transaction Object

Once an application has initialized a provider object, it executes settlement transactions by creating and using transaction objects. In general, the application creates one transaction object for each call. Originating gateways, gatekeepers, SIP proxies, or softswitches typically use the following process when they wish to use a settlement provider to complete a call:

1) Create a new transaction object.
2) Request authorization for the destination phone number.
3) Retrieve authorization and/or routing information for a destination gateway.
4) If first destination gateway is unavailable, retrieve authorization/routing for subsequent gateways.
5) Optionally, provide statistical information during the call.
6) After the call is released, report usage information.
7) Delete the transaction object.

A terminating gateway or gatekeeper typically uses the following process in its interaction with the OSP library. That process starts when the gateway receives a setup message from a peer gateway that it cannot independently authorize.

1) Create a new transaction object.

2) Request validation of the authorization tokens present in the setup message.

3) Optionally, provide statistical information during the call.

4) After the call is released, report usage information.

5) Delete the transaction object.

Document Roadmap

The following additional documents are part of the Open Settlement Protocol Software Development Kit.

OSP Toolkit: Implementation Guide

This document describes the use of the Toolkit in various operational scenarios. It includes details for using the Toolkit with various call control protocols (e.g. H.323, IPDC, SGCP, SIP), in various applications (voice, real-time fax, store-and-forward fax). This guide also describes service variations such as broadcast fax, voice conferencing, and roaming users. The document also provides practical implementation guidelines for supporting multiple, simultaneously active settlement providers, usage-limited authorisation, and transit settlements.

OSP Toolkit: How to Build and Test the OSP Toolkit

This document describes in detail how to build the OSP Toolkit in Unix and Windows NT. It also describes how to test the OSP Toolkit with an OSP Test Server to ensure that the OSP Toolkit has been compiled properly.

OSP Toolkit: Errorcode List

This is a list of all the errorcodes that the OSP Toolkit may report along with an explanation. This documentation was derived directly from the osperrno.h header file.

OSP Toolkit: Programming Interface

The programming interface is a detailed description of the external programming interface to the Toolkit. It is focused on software developers who must actually use the Toolkit, including, for example, gateway or gatekeeper developers. The programming interface is also the ultimate authority on the functionality that the Toolkit provides.

OSP Toolkit: Cisco Interoperability Example

Since the most common OSP devices are made by Cisco, this document contains examples of the messages exchanged for a VoIP call. It also contains information about token format required for interoperability with Cisco devices.

OSP Toolkit: Device Enrollment

This document describes a sample, stand-alone application included in the software development kit. This application supports the enrollment of devices with OSP servers. The application relies on industry-standard cryptographic and networking protocols; and may be useful or easily adopted for use with other applications.

OSP Toolkit: Internal Architecture

This guide provides a detailed examination of the internal architecture of the Toolkit. It is most useful for developers that need to modify the Toolkit. Such modifications may include, for example, enhancements to the Toolkit functionality or performance improvements for particular environments.

OSP Toolkit: Porting Guide

The porting guide provides instructions and suggestions for porting the Toolkit to various environments. It is intended primarily for those users that wish to support the Toolkit in environments other than UNIX and Windows NT.

OSP Toolkit: Protocol Extensions

This guide describes extensions to the open settlement protocol. Use of these extensions is optional, but they do allow full support of value added settlement services. This guide is useful as an addition to the protocol standard specification.

ETSI Technical Specification TS 101 321

This document is the European Telecommunications Standards Institute's (ETSI) specification of Profiles for Inter-domain Pricing, Authorisation, and Usage Exchange for Basic Internet Telephony Service. This standard forms the base specification for the Open Settlement Protocol.