Oracle® Communications Tunneled Session Controller

SDK Guide

Version 1.3.0

Formerly Net-Net Tunneled Services Client

September 2014

ORACLE

Contents

About this Guide

Introduction

The *Oracle® Communications Tunneled Session Controller SDK Guide* describes the client-side SDK (software development kit) that facilitates the creation of secure tunnels between a client application and an Oracle Communications Tunneled Session Controller (TSC) server. A client is typically a softphone application that utilizes the SDK software libraries and source code to create TLS tunnels to a TSC service, thus achieving secure real time communications and ubiquitous firewall traversal.

This document specifically describes the SDK, functional libraries, and source code supplied with the SDK SDK Version 1.3.0.

Note: This document describes the SDK, functional libraries, and source code that accompanies the Oracle Communications Tunneled SEssion Controller S-CX6.4.6 software release.

Server-side TSC capabilities and operations are described in the accompanying document, Oracle® Communications Tunneled Session Controller Essentials Guide.

Audience

This guide is written for experienced software developers who are well versed in standard software development practices and procedures, the C++ programming language, and common operating systems to include Windows and UNIX.

Related Documentation

The following table lists related documents.

Document Name	Document Description
Oracle® Communications Tunneled Session Controller Essentials Guide	Contains information for setting up Tunneled Service Control Function Server that provides tunneled Session Initiation Protocol (SIP) and Message Session Relay Protocol (MSRP) secure transport services.

Revision History

This section contains the revision history for this document.

Date	Description
September 2014	Initial Release

1 Overview

Introduction

Tunnel Session Management (TSM) is a new feature on the Acme Packet 4500. Tunnel Session Management dramatically improves firewall traversal for real time communications for OTT VoIP applications, reduces the dependency on SIP/TLS and SRTP by encrypting access-side VoIP within standardized VPN tunnels. As calls or sessions traverse a TSM tunnel, the SBC will route all SIP and RTP traffic from within the TSM tunnel to the core (or appropriate destination).

Oracle Communications is working with other telecom providers and vendors to standardize TSM. Within the 3GPP, TSM is called a Tunneled Services Control Function (TSCF). Currently the 3GPP Technical Requirement draft is TR 33.8de V0.1.3 (2012-05) as a standardized approach for overcoming non-IMS aware firewall issues with supporting companies including China Mobile, Ericsson, Huawei, Intel, RIM, Vodafone, ZTE. Beyond the standard, we provide exceptional tunnel performance & capacity within the Session Border Controller as well as high availability, DDoS protection and our patent pending TSM Tunnel Redundancy to improve audio quality in lossy networks such as the Internet.

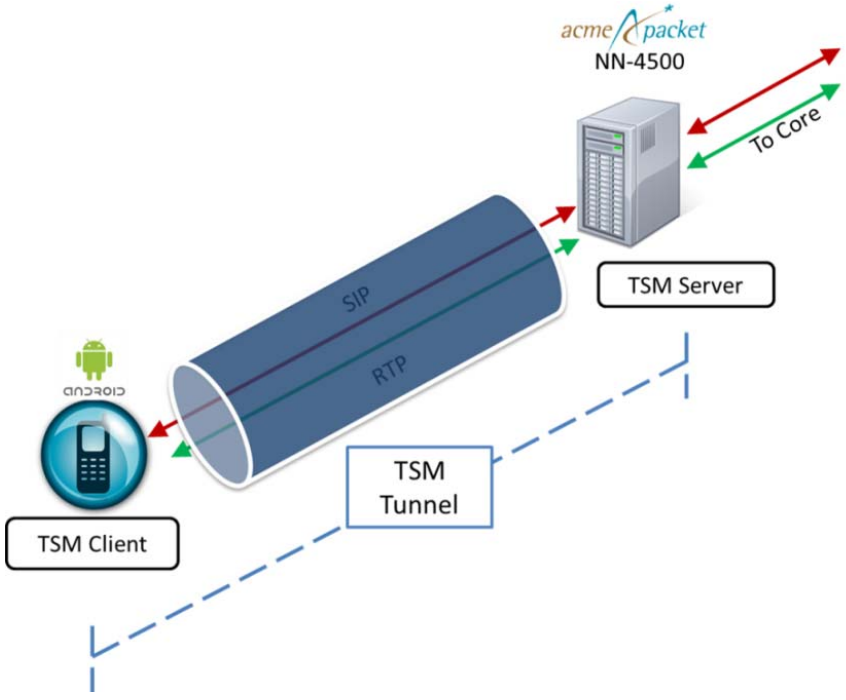


Figure 1 - 1: Basic TSM Setup

As seen in Figure 1 - 1, “Basic TSM Setup,” on page 7, TSM consists of two parts: a) the TSM server (often referred to as a TSCF or Tunneled Services Control Function) and b) the TSM client. The TSM server resides and runs on the Acme Packet 4500 and the TSM client runs within applications that reside on workstations, laptops, tablets and mobile devices (ex. Android, iPhone or iPad) and even network elements. To deploy TSM enabled-clients (such as softphones, SIP-enabled iOS/Android applications or contact center agent applications), customers and 3rd party ISVs will need to incorporate the ’s open source software libraries into their applications which will establish SSL VPNs (TLS or DTLS) to the TSM server.

To deploy TSM enabled-clients such as softphones, SIP-enabled iOS/Android applications or contact center agent applications, customers and 3rd party ISVs will need to incorporate the open source TSM software libraries into their applications which will establish tunnels to the TSM server.

TSM Tunnel

The following diagram briefly explains the various IP addresses utilized during the TSM session. Where required, references have been made to related configuration as described in the *TSCF Essentials Guide, Version SCX-6.4.6FCS*.

- TSCF External IP

  This IP address is visible to any endpoint on the Internet and is used to initiate the TSM session between the TC and the TSCF. This may be configured under “Security ? tscf ? tscf-interface” as described in the following section.

- TC Source IP

  This IP address corresponds to the source address of the TC in its respective access network or it could be the IP of the Proxy behind which it is located.

- Internal Tunnel IP

  This IP address will be assigned to the TC (once TLS authentication is successful) from a configured pool of IP addresses on the TSCF. It will be used to facilitate communication with the core (P-CSCF). The address pool can be configured under **security -> tscf -> tscf-address-pool**.

- TC Application IP

  This is the IP address associated with the respective application (SIP / RTP / other) at the TC.

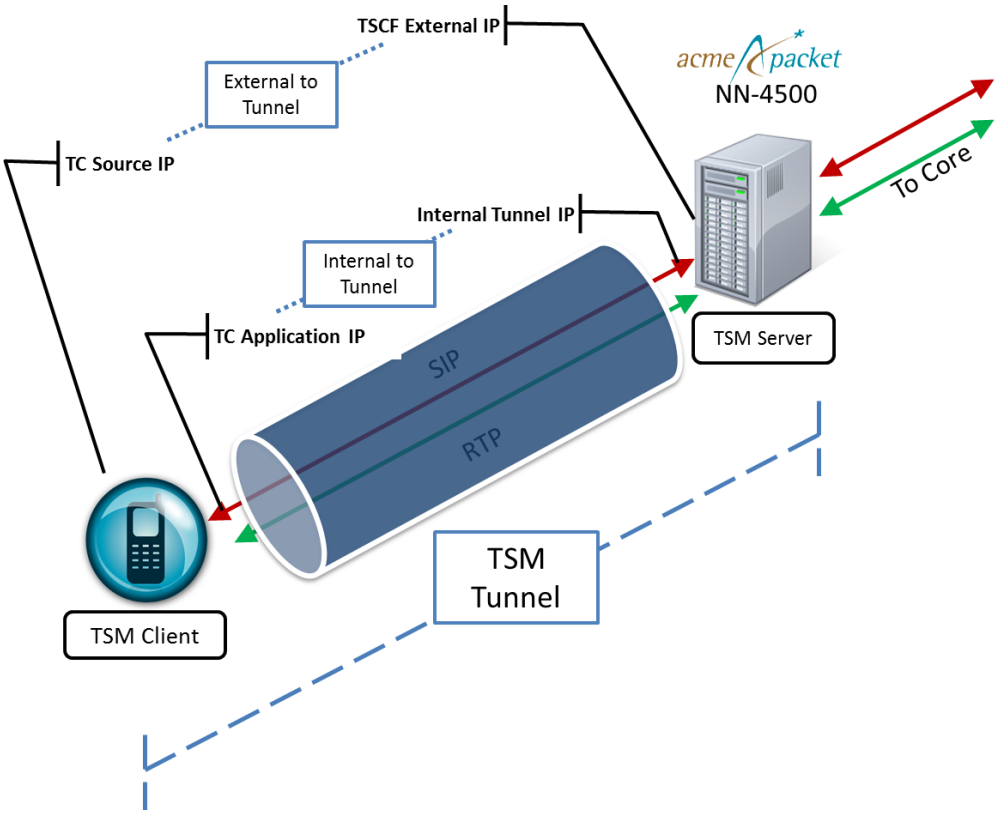


Figure 1 - 2: TSM Tunnel Description

Using The SDK To Create A Tsm Tunnel

The following steps provide an outline on integrating a SIP client (in this example *tsc_sip_client*) with the TSM SDK. Please refer to the file *tsc_sip_client.c* (located at *tsm/sdk/apps/tsc_sip/tsc_sip_client*) which contains working code references on establishing a TSM tunnel and making a SIP/RTP based call.

Initialize the TSCF-Client side API

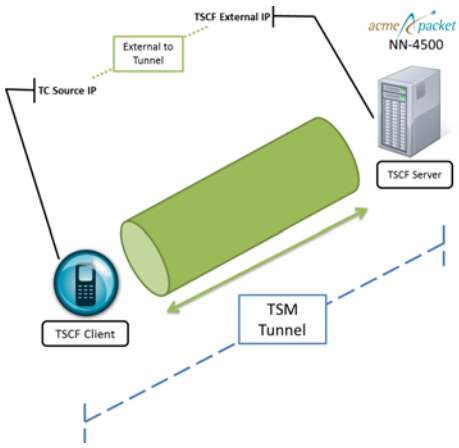
As you initialize you will need information such as the TSCF Server IP address, port, transport type, wireshark tracing and certificate parameters (in the case of TLS / DTLS). You will need to populate the required information in a tsc_tunnel_params type structure.

Search for the text: “tsc_ctrl_init()” in the reference file for actual code implementation

Create a TSM Tunnel

Once initialized, you can proceed with creating a TSM tunnel between the TSCF client and server, register for callbacks and obtain specific information such as the SIP server IP address.

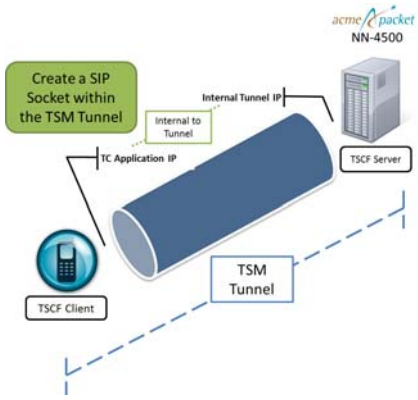
Search for the text: “Create a tunnel” in the reference file for actual code implementation.



Create a SIP Socket within the TSM Tunnel

With the TSM tunnel established, you should proceed with creating a SIP socket and binding the local address (which is assigned by the TSCF server) to it.

Search for the text: “SIP SOCKET CREATION” in the reference file for actual code implementation.

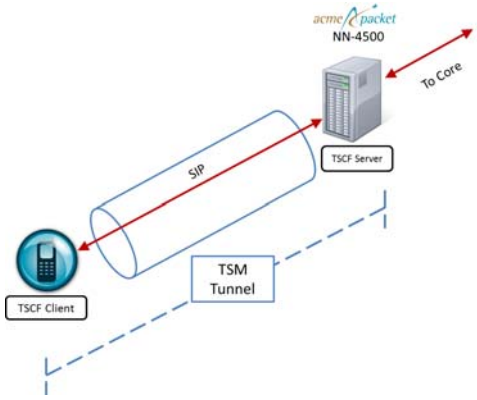


Send SIP messages over the TSM Tunnel

You can now send SIP messages over the TSM tunnel. Based on the kind of SIP applications, you may need to send a REGISTER message or directly initiate a peer-to-peer call through an INVITE message.

You can find examples of both messages being sent by looking for the following text. Since this is only sample code, please use the same as reference for sending and receiving SIP messages via the TSM tunnel.

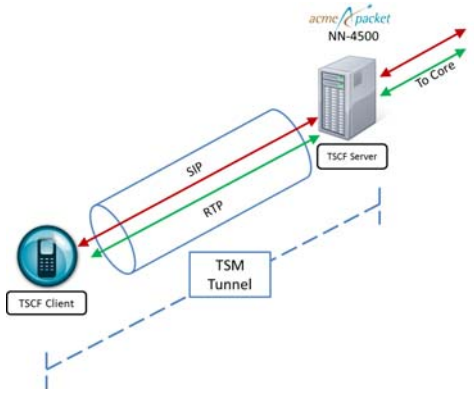
- **"REGISTER TRANSACTION": Constructs and sends a REGISTER message to the tsc_sip_server. The code currently doesn't check for a 200 OK message.**
- **"INVITE TRANSACTION": Constructs and sends an INVITE message and handles a 200 OK.**
- **"Build and send ACK (to INVITE transaction)" : This code builds and sends an ACK to the INVITE received.**



Send RTP message over the TSM Tunnel

As part of the call, there is also sample code to send and receive RTP packets over the TSM Tunnel – search for the text "RTP EXCHANGE" within the reference file.

The code builds RTP packets purely for the purpose of simulation. You would be expected to rely on your SIP/RTP applications to do the needful regarding RTP packet construction.



At this point you should not be able to securely make a call through the TSM tunnel

Terminate a SIP call over the TSM Tunnel

After you are done with the call, you may terminate it using sample code in the reference file – look for the text "BYE TRANSACTION".

Terminate a TSM Tunnel

If your application has determined that it doesn't require the use of a TSM tunnel, you may terminate the tunnel using the sample code – search for "Finished, Socket, Tunnel Cleanup".

Enabling Redundancy

TSM enables an application to improve media quality under adverse network packet loss through the tunnel redundancy feature. You can enable and make use of this feature through the following steps:.

Decide on a Redundancy Factor

Once enabled, tunnel redundancy comes choices around two options:
Factor 1: This includes the original TSM tunnel + 1 redundant tunnel
Factor 2: This includes the original TSM tunnel + 2 redundant tunnels

You will see improved media quality with an increase in the number of tunnels but with the obvious impact on network bandwidth. Hence you will need to take a decision on what kind of redundancy you require before enabling the feature.

Enable Tunnel Redundancy

Tunnel redundancy can be enabled on a per-socket basis, by following these steps:

- Create a notification handler function. Once requested for, the TSCF feature will notify the application to inform if the redundancy was enabled successfully.
- Set a socket option with the type of redundancy factor (1 or 2)
- -You can also choose the kind of redundancy method based on the underlying transport protocol (Refer to the TSCF Essentials Guide, Version SCX-6.4.6FCS for additional configuration details.
  - UDP / DTLS only employ one kind of redundancy
  - TCP / TLS, on the other hand, give you the option of performing either “Load Balancing” or “Fan Out”

Search for the text: “RTP socket created” and “TSC_REDUNDANCY” immediately following it in the reference file for actual code implementation.
The code creates redundant tunnels for RTP packets.

Compiling the TSM Library and Documentation

The following documents, present within the TSM SDK, will help you get up and running with the API. Based on the operating system that you are developing for, you will find updated instructions, on how to generate the TSM SDK library, in the README files located below:

Operating System	Description	Location
	This file provides information on how to compile the TSM SDK for the Android OS	/tsm/sdk/lib/android/README_openssl.tsc
	This file provides information on how to compile the TSM SDK for the iOS	/tsm/sdk/lib/README.ios
	This file provides information on how to compile the TSM SDK for the Linux OS	/tsm/sdk/lib/README
	This file provides information on how to compile the TSM SDK for the Windows OS	/tsm/sdk/lib/README.WIN

In addition, you can generate the API documentation using the following steps:

1. Go to the SDK library folder: *cd tsm/sdk/lib*
2. Auto generate an updated version of the API documentation: *make document*
3. Open the documentation using a browser pointing to: */../docs/html/index.html*

Provided Functionality

This initial SDK release provides the following support.

- Server Assigned Configuration mode
- Tunnel Transport
  - TLS
  - DTLS
- Platform Support
  - Linux
    - Ubuntu 10.4 LTS (64bit)
    - CentOS 5.5 (64bit)
  - Windows
    - Win32
  - Android
    - 2.3.x (Gingerbread)
    - 3.2.x (Honey Comb)
    - 4.0.X (Ice Cream Sandwich)
- Tunnel IP version
  - IPv4
- Inner IP version
  - IPv4
- Payload multiplexing within a tunnel
  - Up to 3 concurrent voice calls
  - Up to 10 MSRP chat sessions
  - 1 MSRP file transfer session

Error Codes

The following are error codes you can check against when calling the TSC layer to ensure you can handle all success and failure scenarios:

Return Value	TSC_ERROR_CODE
0	tsc_error_code_ok
1	tsc_error_code_error
2	tsc_error_code_not_logged
3	tsc_error_code_cannot-connect
4	tsc_error_code_cannot_configure
5	tsc_error_code_keepalive_failure
6	tsc_error_code_service_failure
7	tsc_error_code_cannot_recv_data
8	tsc_error_code_no_data
9	tsc_error_code_cannot_send_data
10	tsc_error_code_authenticate
11	tsc_error_code_cannot_release
12	tsc_error_code_queue_overflow

SDK/Host Operating System Relationship

The following illustrations depict the relationship between the SDK and the host operating system

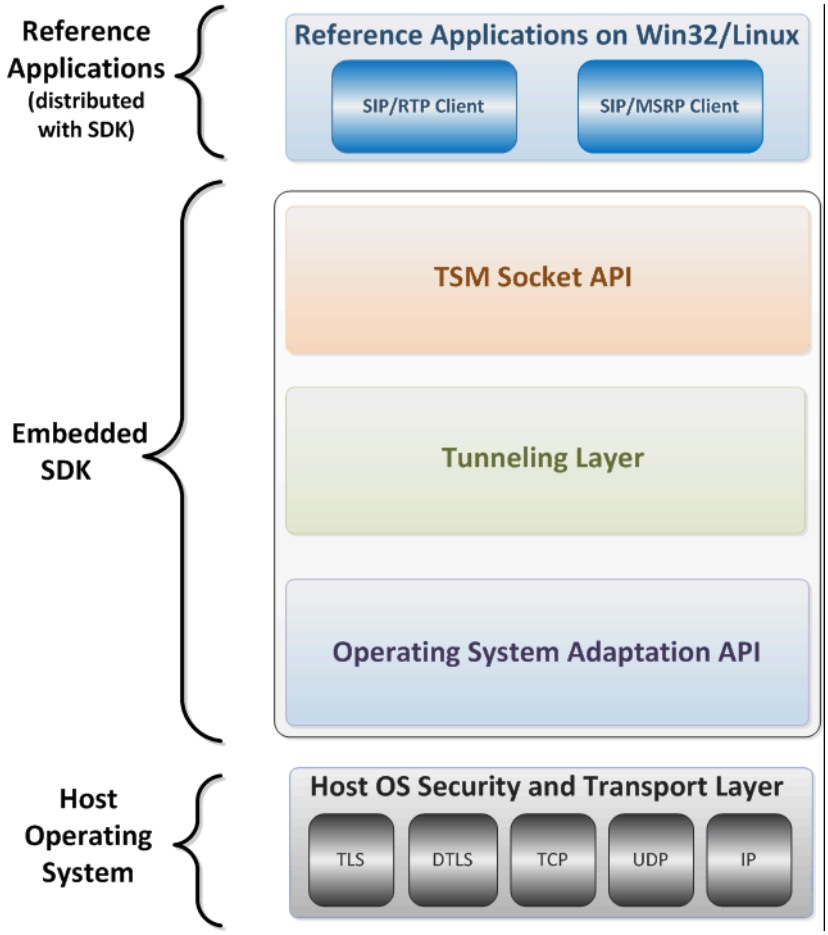


Figure 1 - 3: SDK/Host OS Relationship (Simplified View)

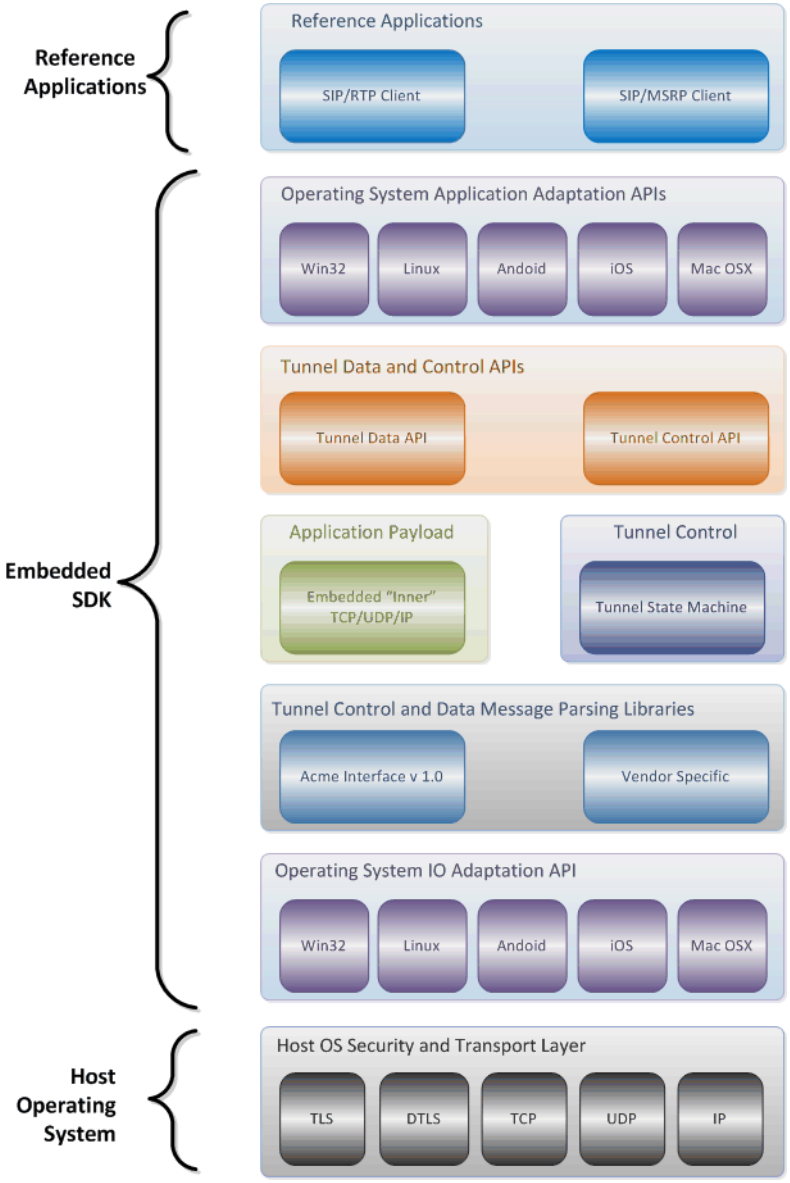


Figure 1 - 4: SDK/Host OS Relationship (Detailed View)

2 SDK Directory Structure

SDK Directories

SDK directories are shown below. Note that not all listed directories may be present (or supported) in the current release.

SDK:	
+---apps	SDK based applications
| +---linphone	Open source VoIP application utilizing TCP/TLS/DTLS/UDP for tunnel transport that has been ported to the SDK. Platform support limited to Windows. Refer to www.linphone.org for additional details.
| +---linphone-android	Open source VoIP application utilizing TCP/TLS/DTLS/UDP for tunnel transport that has been ported to the SDK. Platform support limited to Android. Refer to www.linphone.org for additional details.
| +---linphone-ios	Open source VoIP application utilizing TCP/TLS/DTLS/UDP for tunnel transport that has been ported to the SDK. Platform support limited to iOS (iPhone, iTouch, iPad). Refer to www.linphone.org for additional details.
| +---sipp	Open source SIP traffic generator application utilizing TCP tunnel transport that has been ported to the SDK. Platform support limited to Linux. Refer to www.sourceforge.net for additional details.
| +---tsc_sip	Reference demonstration/development guide app (tsc_sip_client.c)
| +---tsc_sip_http_proxy	Reference demonstration app for http proxy traversal.
| +---tsc_sip_inner_tcp	Reference demonstration app for tunneling of TCP traffic (HTTP)
| +---tsc_sip_serve	Reference demonstration/development guide app (tsc_sip_client.c)
+---docs	SDK Documentation
| +---html	Authoritative API HTML-based documentation. Access via ".../html/index.html"
+---extlib	External, optional libraries
| +---opencore-amr-0.1.2	Open AMR
| +---openssl-1.0.0e	OpenSSL library
+---lib	SDK Library source – to be linked with the target application
| +---android	Android Specific library instructions and precompiled libs
| +---CSM	tunneling Client State Machine
| +---EIP	Embedded TCP/UDP/IP Stack

```
|   +---include             SDK API definitions
|   +---OSAA                Operating System Application Adaptation APIs
|   +---TAPI                Tunnel Data and Control APIs
|   +---TPL                 Tunnel Control and Data Message Parsing
                            Libraries
+---tools                   Development Tools
    +---wireshark           TSCF protocol dissector
```

3 Reference Applications

Template Files

A number of small application templates are found throughout the *apps* directory. Each of these files contains a small, well-defined set of functionality that enables a software developer to easily understand its implementation via the TSCF client-side SDK.

```
SDK:
+---apps/tsc_sip/tsc_sip_client
|   +---tsc_sip_client.c            Provides a basic SIP client utilizing
                                    TCP/TLS/UDP/DTLS for tunnel
                                    transport of inner UDP sockets.
                                    Supports Linux, Windows, and Android
                                    operating systems.
+---apps/tsc_sip/tsc_sip_server
|   +---tsc_sip_server.c            Provides a basic SIP server utilizing
                                    TCP/TLS/UDP/DTLS for tunnel
                                    transport of inner UDP sockets.
                                    Supports Linux, Windows, and Android
                                    operating systems.
+---apps/tsc_sip/tsc_sip_inner_tcp
|   +---tsc_sip_inner_tcp.c         Provides a basic SIP client utilizing
                                    TCP for tunnel transport and
                                    demonstrating usage of TCP sockets
                                    for applications such as HTTP --
                                    supports Linux operating systems.
```

All of the above files contain extensive comments making it an easy task to navigate through the code. Using *tsc_sip_client.c* as an example, you can readily proceed through the file.

1. Search for *tsc_ctrl_init ()* and examine the code immediately following this function for the details of tunnel initialization.
2. Search for *Create a Tunnel* and examine the code immediately following for the details of tunnel creation and the configuration exchange between the TSCT server and client.
3. Search for *SIP SOCKET CREATION* and examine the code that creates and binds TSC sockets.
4. Search for *REGISTER TRANSACTION* and examine the code that builds a SIP REGISTRAR request and processes the REGISTRAR response.
5. Search for *INVITE TRANSACTION* and examine the code that builds a SIP INVITE request and processes the REGISTRAR response.

6. Search for *Build and send ACK* to examine ACK creation and processing code.
7. Search for *RTP Exchange* to examine RTP code.
8. Search for *BYE TRANSACTION* to find code that terminates a SIP connection.
9. Search for *TEST DONE* to find code that terminates a tunnel.

4 SDK Functions and Data Structures

SDK Documentation Reference

The latest documentation for all SDK functions and data structures is found within the SDK itself. Documentation may be accessed by opening the following html file from the TSM root directory. :

```
../tsm/sdk/docs/html/index.html
```

You may also regenerate this online documentation by navigating to the `/tsm/sdk/lib` directory and executing the `make doxygen` command.

> **Note:** Functions and data structures have been removed from this document as of version 1.3 in favor of publishing the latest version and updates of these SDK elements directly from the code.