Oracle® Communications Tunneled Session Controller

Essentials Guide

Release S-CX6.4.6F5

September 2014

ORACLE

Contents

About This Guide

Overview

Version S-CX6.4.6F5 provides support for a Tunneled Session Control Function and Message Session Relay Protocol (MSRP) B2BUA.

Audience

This guide is written for network administrators and architects, and provides information about the TSC and MSRP implementations. Supporting, related material is available in the most current version of the Oracle Communications Session Border Controller ACLI Configuration Guide. Please refer to that document as needed.

For information about system training, contact your Oracle sales representative.

Supported Platforms

Release Version S-CX6.4.6F5 is supported on the Acme Packet 4500.

Related Documentation

The following table lists the members that comprise the documentation set for this release:

Document Name	Document Description
Acme Packet 4500 System Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 4500 system.
Acme Packet 3800 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 3800 system.
Release Notes	Contains information about the current documentation set release, including new features and management changes.
ACLI Configuration Guide	Contains information about the administration and software configuration SBC.
ACLI Reference Guide	Contains explanations of how to use the ACLI, as an alphabetical listings and descriptions of all ACLI commands and configuration parameters.
Maintenance and Troubleshooting Guide	Contains information about Net-Net SBC logs, performance announcements, system management, inventory management, upgrades, working with configurations, and managing backups and archives.

Document Name	Document Description
MIB Reference Guide	Contains information about Management Information Base (MIBs), Enterprise MIBs, general trap information, including specific details about standard traps and enterprise traps, Simple Network Management Protocol (SNMP) GET query information (including standard and enterprise SNMP GET query names, object identifier names and numbers, and descriptions), examples of scalar and table objects.
Accounting Guide	Contains information about the SBC's accounting support, including details about RADIUS accounting.
HDR Resource Guide	Contains information about the SBC's Historical Data Recording (HDR) feature. This guide includes HDR configuration and system-wide statistical information.
Administrative Security Essentials	Contains information about the SBC's support for its Administrative Security license.

Document Revision History

This section contains a revision history for this document.

Date	Description
April 23, 2012	Preliminary, controlled release
May 15, 2012	Phase 1 release – removed references to currently unsupported capabilities (tscf-rekey-profiles and tscf-data-flows)
August 15, 2012	Preliminary, controlled Phase 2 release – added tscf-rekey-profiles, tscf-data-flows, tunnel redundancy, DoS protection, and new tscf **show** commands
November 9, 2012	Preliminary, controlled F2 release – augmented Rev. 1.0 with additional conceptual and operational detail • removed references to currently unsupported capabilities (tscf-rekey-profiles) • Added support for Dynamic Datagram Tunnels • Added support for Server-Initiated-Keepalives
December 4, 2012	F2 release – added miscellaneous enhancements • a new TSCF global parameter (element-id) guarantees the network-wide assignment of unique tunnel identifiers • verify-config detects the presence of TSCF and SIP interfaces on the same network interface • verify-config confirms the installation of a an ETC NUI with a minimum 8GB of DRAM

Date	Description
December 21, 2012	F2 release – added miscellaneous enhancements • new address pool design • changed default red-port • enhanced verify-config consistency checking • added clear-tunnel tscf ... commands
November 2013	F3 release – maintenance, bug-fixes
March 2014	F4 release • added user controls for Nagle algorithm • enhanced HA failover procedures to unsure survival of DTLS and DDT tunnels in the event of HA failover
September 2014	F5 release - added miscellaneous enhancements • removed the requirement for an advanced Secure Services Module (SSM2) when TSC is used • added support for the Online Certificate Status Protocol (OCSP) • added OCSP policy support

Tunneled Services Control Function

Platform Requirements

The Tunneled Services Control Function (TSCF) is supported on Oracle Communications 4500 Session Border Controllers (SBC) equipped with one or more Enhanced Traffic Control (ETC) Network Interface Units (NIU) with a minimum of 8GB on installed DRAM.

> **Note:** Prior releases required the presence of an advanced Secure Services Module (SSM2) when using TSCF. This release removes that requirement.

License Requirements

TSCF availability is dependant on the acquisition and installation of a TSCF license.

The TSCF license supports distinct tunnel tiers (counts) as follows: 1K, 5K 10K, 20K, 30K, 40K, 50K, 60K, 70K, 80K, 90K, 100K, 110K, 120K, 130K, 140K, 150K, 160K, 170K, 180K, 190K, 200K.

TSCF also requires the TLS license, and standard protocol-based licenses (for example, SIP) based on network traffic requirements. The Denial of Service detection/prevention capability described in Denial of Service, requires a DoS license.

Use the **show features** ACLI command to display a list of installed licenses. The resulting display will resemble the following example that displays a superset of available licenses.

```
ACMEPACKET# show features
Total session capacity: 32000
TSCF tunnel capacity : 200000
Enabled features:
        SIP, MGCP, H323, IWF, QOS, ACP, Routing,
        Load Balancing, Accounting, High Availability,
        PAC, LI, External BW Mgmt, TLS, Software TLS,
        External CLF Mgmt, External Policy Services, ENUM, H248,
        H248 SCF, H248 BGF, NSEP RPH, LI Debug,
        Session Replication for Recording, Transcode Codec AMR,
        Transcode Codec EVRC, DoS, IKE, IPv4-v6 Interworking, RTSP,
        IDS, Transcode Codec EVRCB, Software PCOM, Security Gateway,
        SIP Authorization/Authentication,
        Database Registrar (0 contacts), IDS Advanced,
        SLB (2000000 endpoints), Allow Unsigned SPL files,
        Session Recording, Policy Director,
        TSCF (200000 TSCF tunnels), Transcode Codec AMR-WB, CX
ACMEPACKET#
```

TSCF Overview

Tunnel Session Management (TSM) is based upon an existing 3rd Generation Partnership Project (3GPP) Technical Requirement, TR 33.8de V0.1.3 (2012-05) that seeks to define a standardized approach for overcoming non-IMS aware firewall issues. Within the 3GPP, TSM is referred to as Tunneled Services Control Function or TSCF.

TSM improves firewall traversal for over-the-top (OTT) Voice-over-IP (VoIP) applications, and reduces the dependency on SIP/TLS and SRTP by encrypting access-side VoIP within standardized Virtual Private Network (VPN) tunnels. As calls or sessions traverse a TSM tunnel, an Oracle Communications Tunneled Session Controller forwards all SIP and RTP traffic from within the TSM tunnel to appropriate servers or gateways within the secure network core. Operating in a TSM topology, the SD provides exceptional tunnel performance and capacity, as well as optional high availability (HA), DoS protection and tunnel redundancy that improves audio quality in lossy networks.

As implemented by Oracle Communications, TSM consists of two parts:

- a TSM client, referred to as a Tunneled Service Element (TSE)
- a TSM server, referred to as a Tunneled Services Control Function (TSCF)

A TSE runs within client applications residing on network elements — for example, workstations, laptops, tablets and mobile devices (Android, iPhone, Windows, or iPad). A TSE provides a software development kit (SDK) that facilitates tunnel creation and management. Refer to the companion document, the *TSE SDK Guide*, or to the SDK on-line help system for information on client-side programming, and to access reference applications that provide client-specific coding examples for supported operating systems.

To deploy TSEs, customers and 3rd party ISVs (Independent Software Vendors) need to link the open source TSE libraries with their applications, which can then initiate and establish SSL VPNs (TLS or DTLS) to the TSCF.

In contrast to a TSE, a TSCF resides on the Oracle Communications Tunneled Session Controller, and provides server-side functions that accept, establish, and manage tunnel connections between remote TSE-enabled applications and core P-CSCFs/SIP proxies. The following illustration shows various ways in which TSE-enabled clients can be deployed across different network segments (WiFi, LAN, WAN, LTE) and interface with common network elements (firewalls, HTTP proxies, and radio access network equipment).

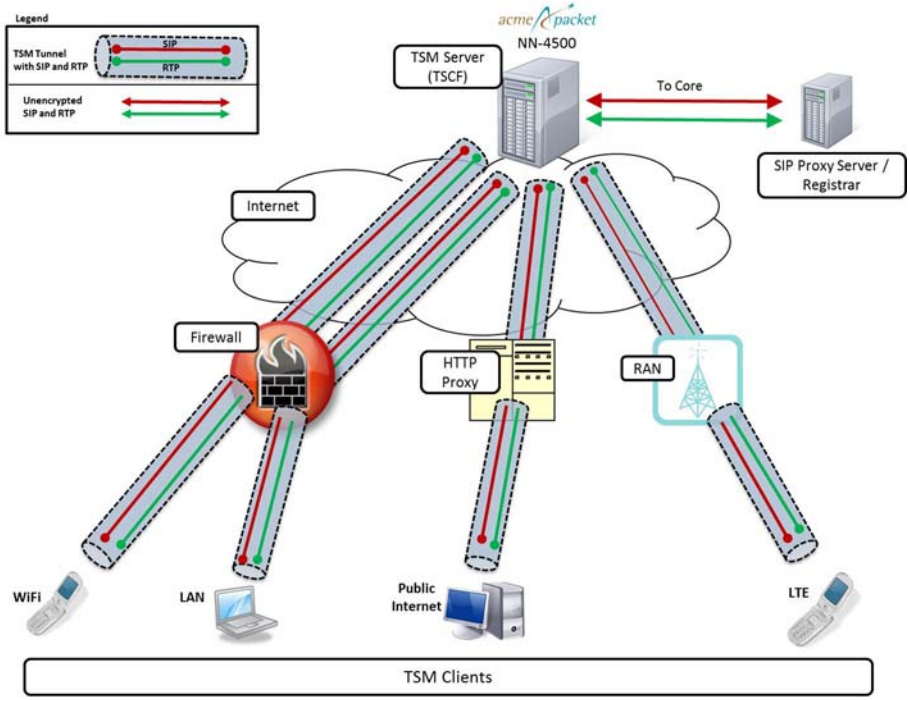


Figure 1: TSM Topologies

Deployment Models

Based on local network topology and requirements, TSM service can be deployed in two ways:

- Decomposed model
- Combined model

Decomposed Model

The decomposed model provides TSCF functionality on an Acme Packet server (for example, a Multiprotocol Security Gateway, or a generic NN-SD. Regardless of the supporting platform, the TSCF acts as a pass through server, using a data-flow (refer to TSCF Data Flow Configuration) to forward all tunnel traffic to a pre-determined IP address. In the simplest topology, this address would access a gateway providing the protocol support required to route the previously tunneled traffic to its ultimate destination. Alternatively, the data-flow could direct tunneled traffic to a another Acme Packet SD that provides P-CSCF (Proxy-Call Session Control Function) services as shown in the following illustration.

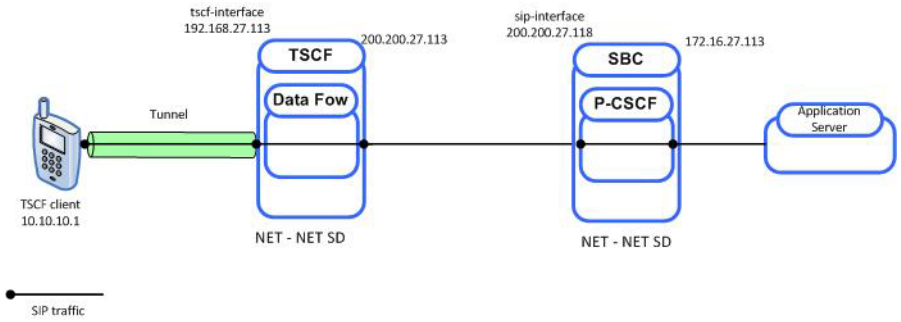


Figure 2: Decomposed Model

Within this decomposed TSCF server:

- 192.168.27.113 provides a TSCF interface to the *access* realm (refer to TSCF Interface Configuration for procedural details)
- The TSCF interface has an assigned address pool (refer to TSCF Address Pool Configuration for procedural details) that provides a contiguous range of tunnel addresses (for example, 10.10.10.1available to TSC clients
- The TSCF address pool has an assigned data-flow (refer to TSCF Data Flow Configuration for procedural details) that specifies a static route to the *core* realm
- 200.200.27.113 provides a SIP interface that provides transit to the *core* realm

Within the P-CSCF server:

- 200.200.27.18 provides a SIP interface serving the *core* realm
- 172.16.27.113 provides an interface to a SIP application server

Decomposed packet flow can be summarized as follows:

1. The TSE client connects to the TSCF server at 192.168.27.113.
2. Assuming client authentication succeeds, the TSCF server accepts connection and assigns the client a tunnel address (10.10.0.1).
3. Using the tunnel, the TSE client sends a SIP INVITE from 10.10.0.1 to a remote peer.
4. Using the data-flow, the TSCF server forwards the INVITE to the core realm (200.200.27.118).
5. The P-CSCF forwards the INVITE to the SIP application server.

Combined Model

The combined model involves deploying both the TSCF and P-CSCF on the same Acme Packet NN-SD. With this model, the TSCF can be configured to pass SIP traffic to the co-located P-CSCF (thus providing standard SIP processing), and to use a data-flow to pass all non-SIP traffic to a designated destination, most often a gateway that provides the protocol support required to route the previously tunneled data traffic to its ultimate destination.

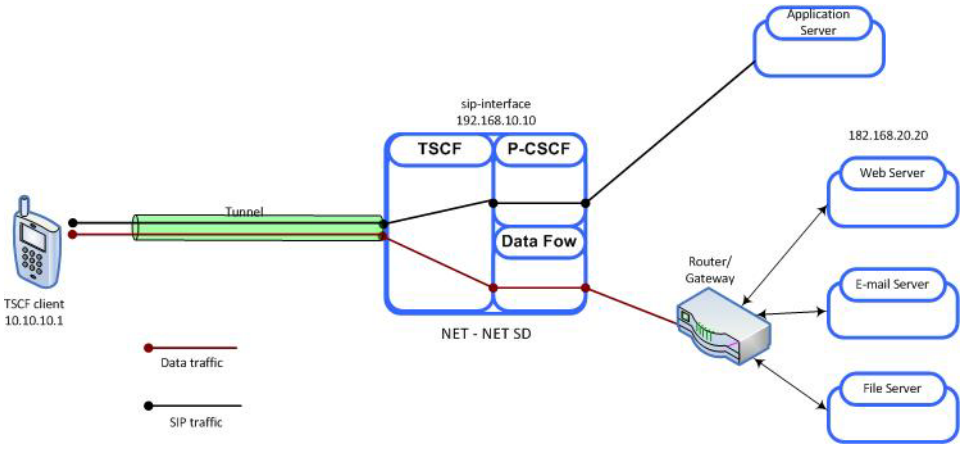


Figure 3: Combined Model - Standard Processing

Within this combined TSCF server:

- 192.168.10.10 (a SIP interface) provides a TSCF interface to the *access* realm (refer to TSCF Interface Configuration for procedural details)
- SIP signalling traffic received from a tunnel endpoint (and explicitly addressed to 192.168.10.10) is directed to an appropriate P-CSCF process for forwarding to a generic or specialized SIP application server
- The TSCF interface has an assigned address pool (refer to TSCF Address Pool Configuration for procedural details) that provides a contiguous range of tunnel addresses (for example, 10.10.1) available to TSC clients
- The TSCF address pool has an assigned data-flow (refer to TSCF Data Flow Configuration for procedural details) that specifies a static route to a router/gateway; any data traffic received from a tunnel endpoint is passed through to the gateway

The following figure illustrates traffic pass-through within the Combined model.

Within this combined TSCF server:

- 192.168.10.10 (a SIP interface) provides a TSCF interface to the *access* realm (refer to TSCF Interface Configuration for procedural details)
- The tunnel client sends SIP messages to 182.168.55.55 (a SIP interface not resident on the Acme Packet SD)
- In this case, all traffic is subject to the data-flow, which directs both SIP signalling and media streams to a gateway for routing to the destination IP address

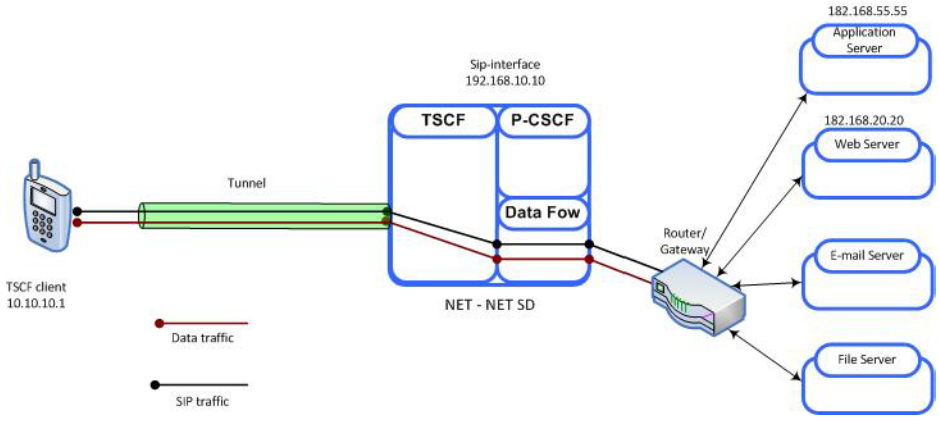


Figure 4: Combined Model - Pass-Through Processing

Tunnel Establishment

Tunnel creation is initiated by the client application that first establishes a TLS and/or DTLS tunnel between the client and the TSCF server. Tunnel creation is completed with a single exchange of configuration data accomplished by a client request and server response. Refer to the *TSE SDK Guide* for detailed information on tunnel set-up procedures.

The following illustration briefly explains the IP addresses used during tunnel establishment and operation.

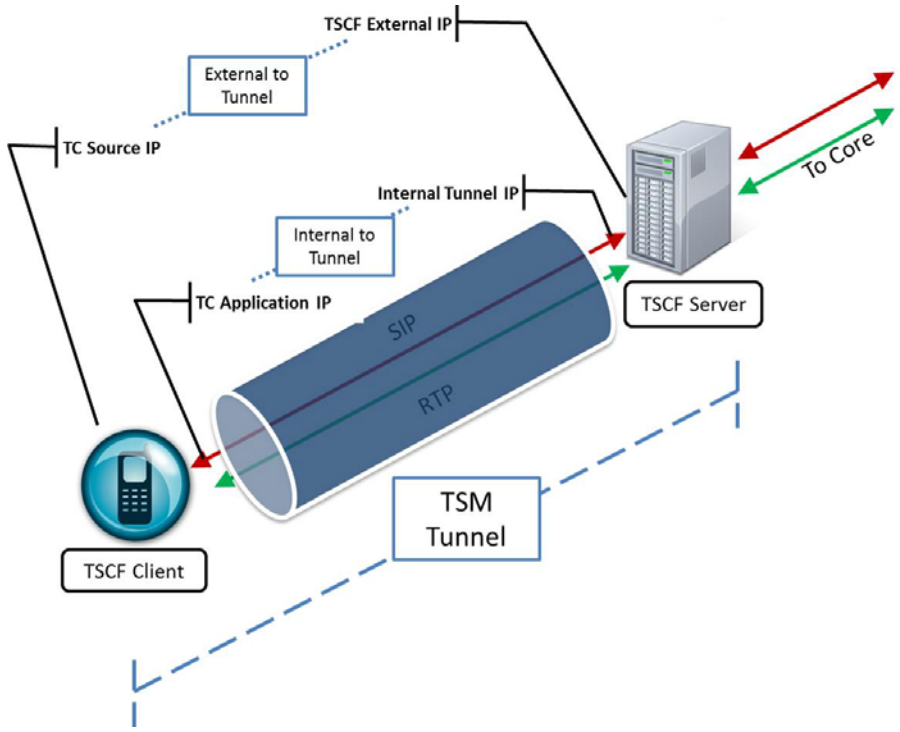


Figure 5: TSM Tunnel Address Structure

- TSCF External IP

  This public IP address is visible to any endpoint; TSE requests for tunnel establishment are directed to this address (refer to TSCF Interface Configuration for procedural details).

- TC source IP

  This public IP address identifies either the source address of the TSE client in its respective access network, or the IP address of an intervening proxy, firewall, or NAT device.

- Internal Tunnel IP

  This private address will be assigned to the TSE client (assuming authentication is successful) from a configured pool of IP addresses on the TSCF. Refer to TSCF Address Pool Configuration for procedural details.

- TC Application IP

  This private IP address identifies a specific application (SIP/RTP/etc.) at the TSE client site.

All packets between the client application and the TSCF server are comprised on *inner* and *outer* parts separated by a TLS header. All data after the TLS header is encrypted.

- outer headers contain tunnel client and TSCF server addresses
- inner headers contain application and P-CSCF addresses
- payload packets carry application data between the TSCF client and the TSCF serve
- control payloads support tunnel management and configuration activities such as the assignment of a client inner IP address, and the exchange of keepalive messages

The following two illustrations depict simplified packet formats.

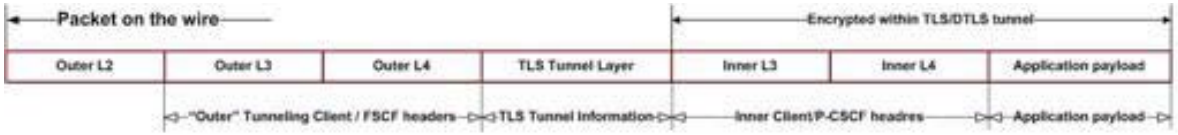


Figure 6: Data Traffic Packet Structure

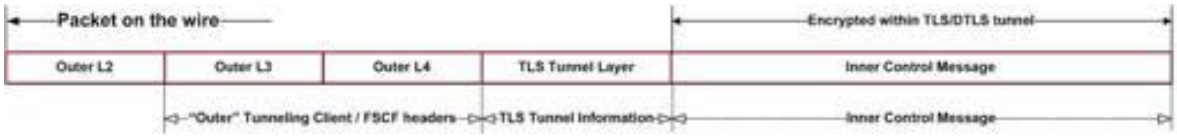


Figure 7: Control Traffic Packet Structure

The following illustrations provide a more detailed view of packet structure and addressing. The first figure provides a network reference model.

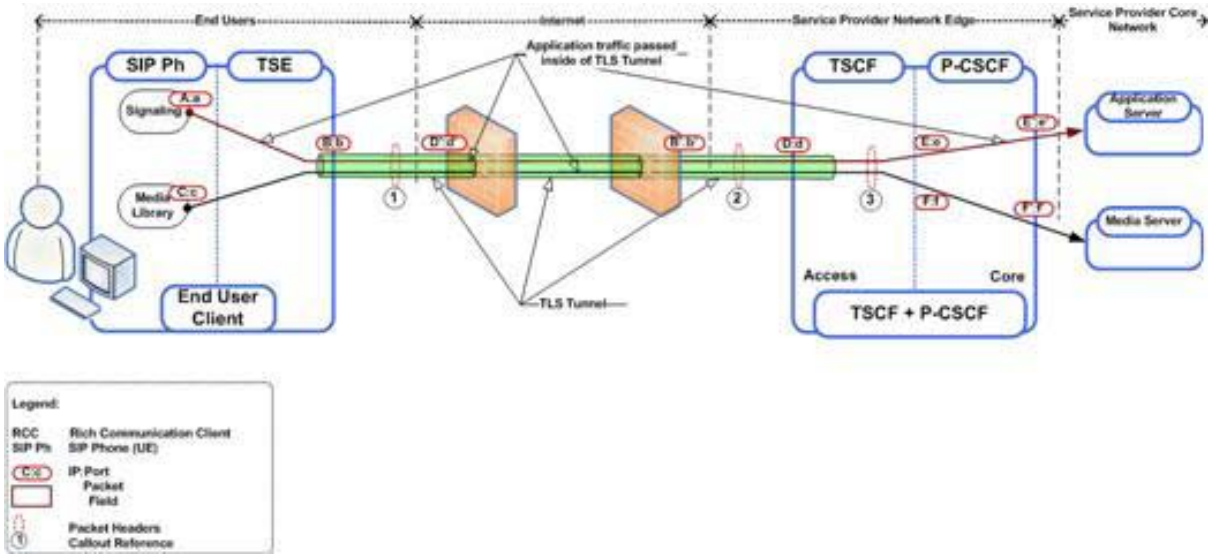


Figure 8: Reference Model

These two figures show a client-initiated and a server-initiated packet with sample addresses.

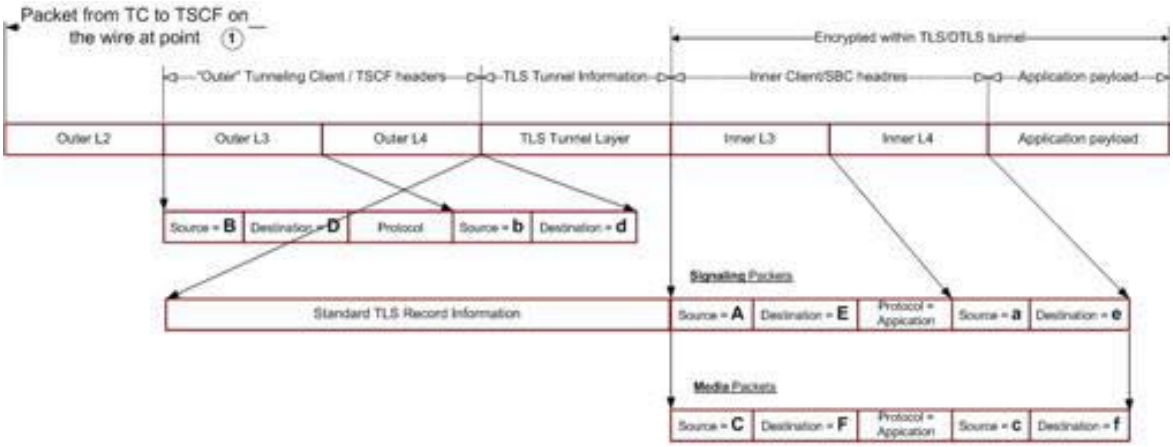


Figure 9: TSE Client to TSCF Server

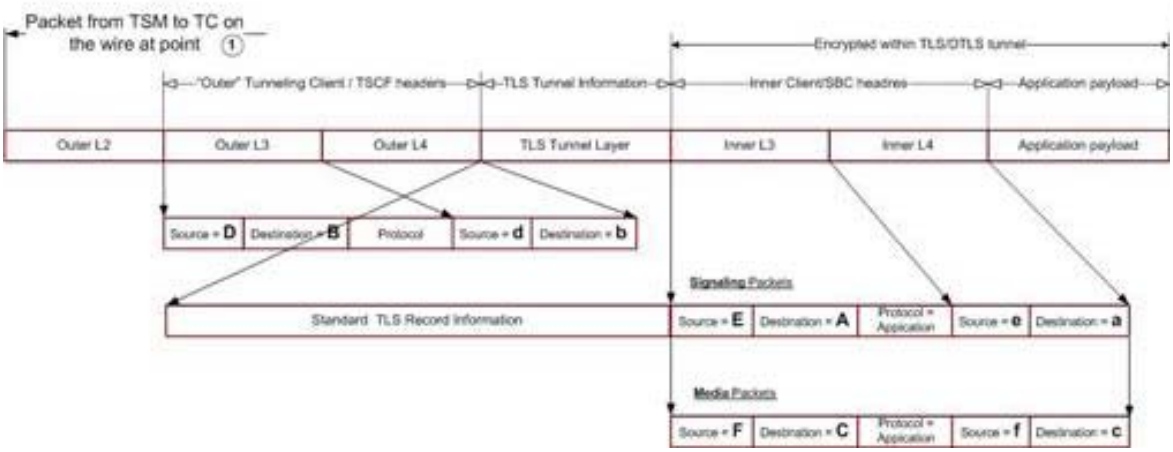


Figure 10: TSCF Server to TSE Client

Tunnel Redundancy

The ability to establish parallel, redundant tunnels improves media quality under adverse network packet loss conditions (defined as packet loss within the 5% to 25% range). As the name implies, tunnel redundancy creates secondary TSCF tunnels that replicate signaling and media transport between the TSE and the TSCF.

Tunnel redundancy is one of the assigned services, along with *ddt*, *server-keepalive*, and *sip*, that can be enabled or disabled on a TSCF interface; by default, tunnel redundancy is disabled. This service is enabled by including the string *redundancy* in the comma-separated assigned-services list (refer to TSCF Interface Configuration for procedural details).

With tunnel redundancy enabled, as packets traverse networks and encounter packet loss and drop, the TSE client and TSCF server choose the most available and timely packet from the duplicative tunnels. Within redundant tunnels, packets are slightly offset (as shown in the following figure), so as a gateway or other intervening network elements drops packets, the same packet that was dropped in one tunnel remains in another tunnel. When the packets reach their destination in the core, or on the client, the media stream is reformed as if packet loss never happened.

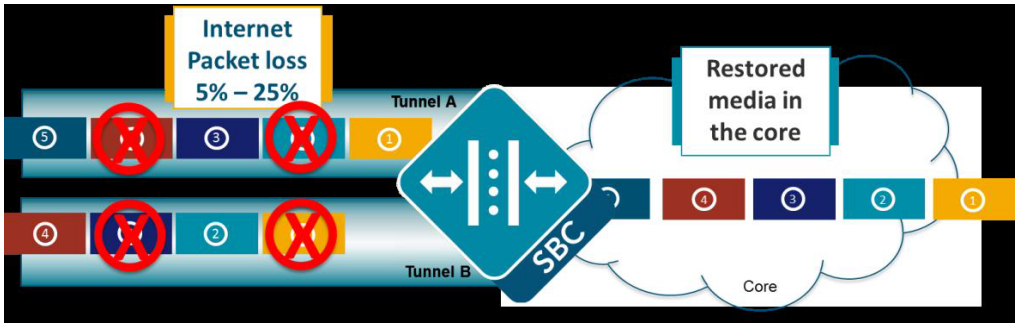


Figure 11: Tunnel Redundancy

Tunnel redundancy works in concert with RTP redundancy and packet loss concealment algorithms that are often implemented by advanced variable bit rate codecs such as SPEEX, SILK or iSAC. Like RTP redundancy, defined in RFC 6354, *Forward Shifted RTP Redundancy Payload Support*, tunnel redundancy *does increase* bandwidth usage by replicating signaling and media but also maximizes the chances of packet delivery. In addition to packet loss, tunnel redundancy can also mitigate the effects of jitter (variable packet delay), a common network impairment that can result in significant speech quality degradation.

A TSE can initiate tunnel redundancy at call establishment or, in response to current network conditions, at any time within an established call session. Depending on the transport protocol (TCP or UDP) a TSE can request one of three available tunnel redundancy modes:

- TLS/TCP Load Balancing
- TLS Fan-Out
- DTLS/UDP Redundancy

TLS/TCP Load Balancing

TLS/TCP load balancing mode is based on parallel, redundant tunnels. Both the TSE and the TSCF send RTC encapsulated packets on one tunnel at a time. Each subsequent packet is sent on the next tunnel in a circular fashion. The following illustration shows packet transmission over time with packet transmission balanced over a main tunnel and two redundant tunnels.

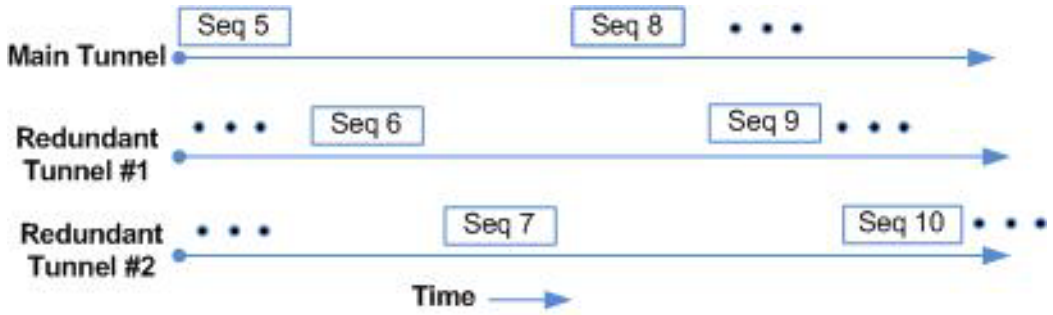


Figure 12: Redundancy — TLS/TCP Load Balancing

This illustration provides a sample message flow for TLS/TCP load balancing negotiation.

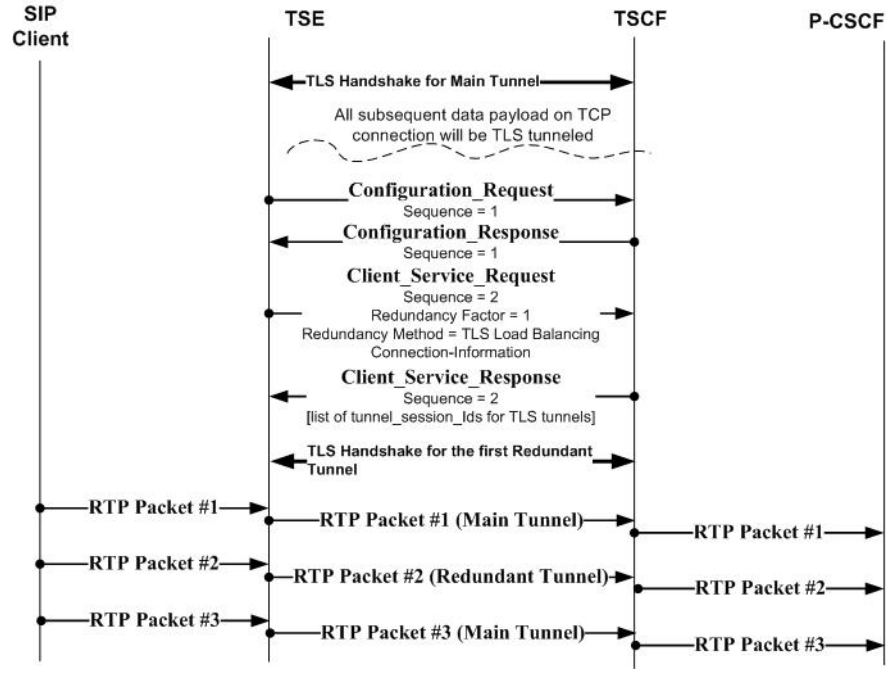


Figure 13: TLS/TCP Load Balancing Message Flow

TLS/TCP Fan-Out

TLS/TCP fan-out mode is also based on redundant parallel tunnels. Both the TSE and the TSCF send RTC encapsulated packets on the main tunnel. Additionally both the client and server send the same packet on each redundant tunnel in a time-staggered fashion. The following illustration shows packet fan-out over time with a main tunnel and two redundant tunnels.

Figure 14: Redundancy — TLS/TCP Fan-Out Load Balancing

This illustration provides a sample message flow for TLS/TCP fan-out.

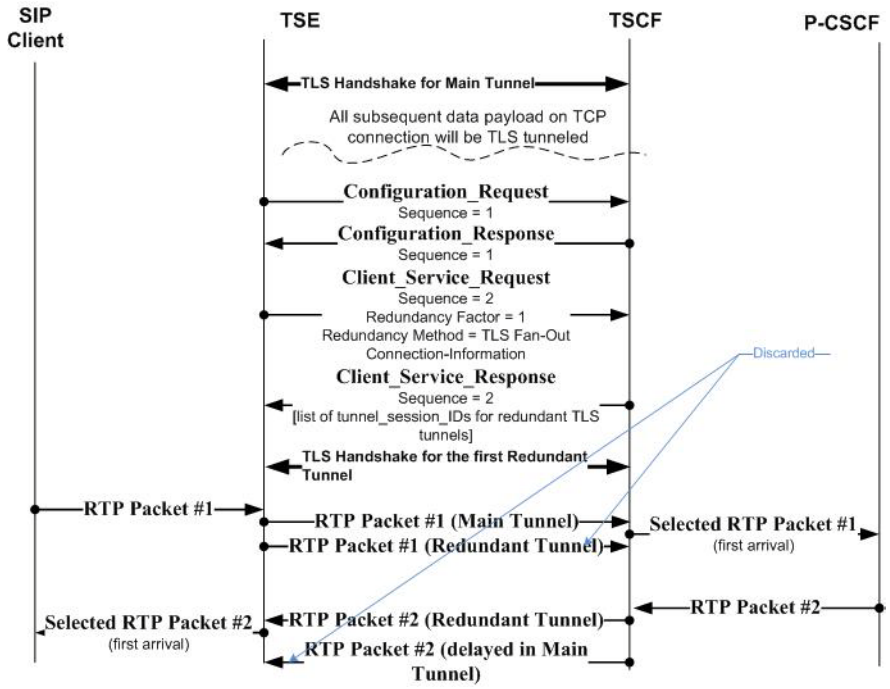


Figure 15: TLS/TCP Fan-Out Load Balancing Message Flow

DTLS/UDP Redundancy

DTLS/UDP redundancy mode, unlike TLS/TCP load balancing and TLS/TCP redundancy modes, does not establish redundant parallel tunnels between the TSE and TSCF. Instead, both the client and server exchange frames of RTC encapsulated packets. Each frame contains a new packet and some number of sequential, previously sent packets. The following illustration shows DTLS/UDP redundancy over time.

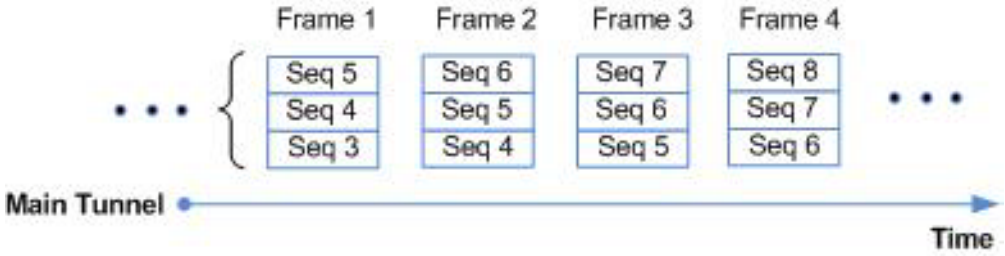


Figure 16: Redundancy — DTLS/UDP Redundancy

This illustration provides a sample message flow for DTLS/UDP redundancy.

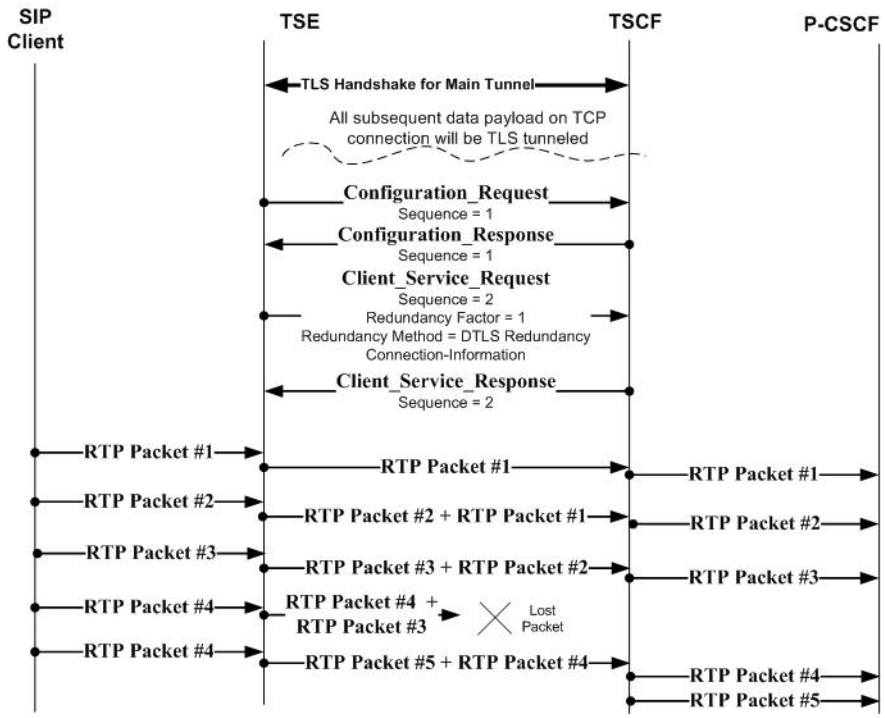


Figure 17: DTLS/IDP Load Balancing Message Flow

Denial of Service

Denial of Service (DoS) protection is a licensed capability that adds preliminary access control and bandwidth control for tunneled traffic.

Because tunneled traffic is confined to a small number of well-defined ETC-based ports, traffic processing can be streamlined. Incoming traffic is directed to one of two pipes, each supported by dedicated queues, for transmission toward the network core. A larger pipe provides maximal bandwidth for trusted tunnel endpoints, while a smaller pipe provides minimal bandwidth for untrusted tunnel endpoints.

Tunnels can be promoted to the *trusted* level, or demoted to the *untrusted* level.

Initially all incoming traffic is treated as untrusted and directed to the untrusted pipe. When the TLS handshake is successfully completed (meaning that the two peers have mutually authenticated) the tunnel transitions to the trusted state. The tunnel remains in the trusted state until the expiration of the tunnel persistence timer, as described in TSCF Global Configuration.

Server-Initiated Keepalive

For some TSE client devices, such as IPhones or iPads, operating system constraints can prevent the transmission of timely keepalive messages required to refresh an existing TSCF tunnel connection or to maintain current NAT bindings. To address this problem the TSCF server can be configured to issue periodic server-initiated keepalive (SIK) messages to clients who require such service.

SIK is one of the assigned services, along with *ddt*, *redundancy*, and *sip*, that can be enabled or disabled on a TSCF interface; by default, SIK is disabled. SIK is enabled by including the string *server-keepalive* in the comma-separated assigned-services list (refer to TSCF Interface Configuration for procedural details).

SIK services are provided only in response to a TSE client request. The client requests SIK services during the configuration process. Upon receiving such a request, the TSCF server checks the status of SIK on the interface on which the service request was received. If SIK is enabled, the request is accepted; if SIK is disabled (the default state), the request is denied.

Dynamic Datagram Tunneling

Dynamic datagram tunneling (DDT) provides parallel stream-based (TCP/TLS) and datagram-based (UDP/DTLS) tunnels to address problems/challenges in delivering Real-Time Protocol (RTP media over a connection-oriented, reliable TCP transport. With DDT enabled, the TSE client and TSCF server negotiate a single secure, persistent, stream-based tunnel, used for SIP signaling and tunnel maintenance, and a series of dynamic datagram-based tunnels used for RTP media transport.

With DDT enabled the TSCF server assigns the same inner tunnel address to both the stream-based and datagram-based tunnels. The initial datagram-based tunnel provides a template for subsequent tunnels created to service pending RTP packets. Each tunnel is configured in exactly the same manner as the initial tunnel, save for the tunnel ID, which is unique for each tunnel instance.

Dynamic datagram tunneling is one of the assigned services, along with *redundancy*, *server-keepalive*, and *sip*, that can be enabled or disabled on a TSCF interface; by default, dynamic datagram tunneling is disabled. The service is enabled by including the string *ddt* in the comma-separated assigned-services list (refer to TSCF Interface Configuration for procedural details). DDT configuration has some unique requirements as listed below:

- **assigned-services** MUST be set to sip,ddt
- a TLS/TCP port must be configured on the TSCF interface
- a DTLS/UDP port, using the same IP address and port number as assigned to the TLS/TCP port must be configured on the same TSCF interface

Dynamic datagram tunneling service is provided only in response to a TSE client request. The client requests this service during the configuration process. Upon receiving such a request, the TSCF server checks the status of dynamic datagram tunneling on the interface on which the service request was received. If the service is enabled, the request is accepted; if the service is disabled (the default state), the request is denied.

Tunnel Restoration and High Availability

Release S-CX6.4.6F4, and later releases, support disruption-free traffic flow through DTLS and Dynamic Datagram Tunnels (DDT) during HA failover.

In previous releases, the active member of the HA pair synchronized certain tunnel details with its stand-by partner to include internal IP addresses, tunnel identifiers, and so on. Upon synchronization, the stand-by maintained the tunnels in a persistent state, meaning that the tunnels were inactive and lacking any Layer 3 context.

When an HA pair experiences a failover (or in the event of any disconnect), TSC clients have a grace period (defined by the **tunnel-persistence-time** parameter) to recognize the failure and re-connect using the same tunnel. If clients react in a timely fashion, the newly active system allocates the same TSCF tunnel identifier and IP address. This procedure, however, did allow for the disruption of concurrent real-time traffic.

Release S-CX6.4.6F4 addresses such disruption by augmenting the information provided by the active member to its stand-by partner. In addition to the previously mentioned internal IP addresses and tunnel identifiers, the active member also provides DTLS context information to include cryptographic materials. Upon synchronization, the stand-by member populates the DTLS stack with the received contextual data, and creates the DTLS tunnels in a quasi-active state — such tunnels do not actually process packets. If and when HA handover occurs, the presence of the traffic-ready DTLS tunnels mitigates against disruption of the media stream.

A similar, although not identical approach, is used in DDT environments.

In pre-S-CX6.4.6F4 releases, neither the TLS tunnel, which carried control traffic and signalling, nor the DTLS tunnel, which carried the media stream, survived an HA failover. It was left to the client device to initiate new DDT service.

With release S-CX6.4.6F4 and later releases, DDT tunnel restoration proceeds as follows.

1. The active member updates the DTLS tunnels with augmented contextual information as described above.
2. In the event of HA handover, the TLS tunnel continues in the persistent state, and remains in that state until:
   - the stand-by transitions to the active role, and
   - the client re-connects

   Only after these two conditions are met does the TLS tunnel transition to the active state.
3. If the TSC client requests restoration of the TLS tunnel before the persistent timer expires, the server updates the newly established tunnel with DDT-related information.
4. If the TSC client fails to request restoration of the TLS tunnel before the persistent timer expires, the server deletes both the TLS tunnel and the associated DTLS tunnel.

The **show tscf statistics**, **show tscf tunnel all**, and **show tscf tunnel detailed** ACLI commands all provide various counts of HA tunnel restoration operations.

Nagle Algorithm Control

Like DDT, Nagle algorithm control can provide an alternative method of improving delivery of RTP media over a TCP/TLS transport.

The Nagle algorithm is defined in RFC 896, *Congestion Control in IP/TCP Networks.* The algorithm is designed to improve the efficiency of IP/TCP networks by reducing the number of transmitted packets. With the Nagle algorithm enabled, the transmitting device buffers any outgoing data until all previously sent data has been acknowledged, or until the size of the buffered data exceeds a full packet of output.

Prior to Release SCX6.4.6F4, the Nagle algorithm was enabled by default, and was not subject to user control. With SCX6.4.6F4, and later releases, user control of the algorithm is provided. Users can:

(1) specify a TSCF-Interface-specific default value (enable | disable) that determines the state of the Nagle algorithm on that TSCF interface

(2) override the default algorithm state on a specified tunnel

Override requests are included in the Application Payload as shown in Figure 9, “TSE Client to TSCF Server,” on page 16.

The **show tscf statistics**, **show tscf tunnel all**, **show tscf tunnel detailed**, and **show tscf tunnel** <*tunnel-id*> **verbose** ACLI commands all provide various counts of algorithm operations.

Refer to TSCF Interface Configuration for configuration details.

Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) is defined in RFC 2560, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. The protocol enables users to determine the revocation state of a specific certificate.

RFC 2560 specifies the data exchanged between an OCSP client, in the current implementation the TSCF, and an OCSP responder, the Certification Authority (CA), or its delegate that issued the certificate to be verified. An OCSP client issues a request to an OCSP responder and suspends acceptance of the certificate in question until the responder replies with a certificate status.

If the OCSP responder returns a status of **good**, the certificate is accepted and authentication succeeds. If the OCSP responder returns a status other than **good**, the certificate is rejected and authentication fails.

Certificate status is reported as:

- **Good**—Indicates a positive response to the status inquiry. At a minimum, a positive response indicates that the certificate is not revoked, but does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate’s validity period.
- **Revoked**—Indicates a negative response to the status inquiry. The certificate has been revoked, either permanently or temporarily.
- **Unknown**—Indicates a negative response to the status inquiry. The responder cannot identify the certificate.

When authentication of remote TSEs is certificate-based, you can enable OCSP on TSCF ports to verify certificate status. The TSCF/OCSP responder connection must be secured by the TLS protocol.

OCSP Request Processing

With OCSP enabled, the TSCF checks certificate revocation during the TLS handshake. After receiving the client certificate, the TSCF sends an OCSP request to a responder, and pauses the TLS handshake. Assuming a **good** response, the TSCF resumes the handshake, and establishes a TLS tunnel once the handshake is successfully ended. If the response is **revoked** or **unknown**, the handshake fails, and TSCF does not establish a tunnel.

The TSCF places OCSP requests in an outbound queue, and maintains a list of outstanding requests, which is updated upon the receipt of OCSP responses. When the TSCF receives a new request, it checks the certificate serial number, and possibly the identity of the issuing CA, against the list. If the request matches a list entry, TSCF ignores the request, and awaits the response to the outstanding request; if the request does not match a list entry, TSCF sends a new OCSP request to a responder.

OCSP Certificate Verification

Providing OCSP services requires the creation of a secure TLS connection between a TSCF port and one or more OCSP responders. This configuration is a three-step process:

1. Create one or more certificate-status-profiles. Each certificate-status-profile provides the information and cryptographic resources required to access a single OCSP responder.
2. Assign one or more certificate-status-profiles to a tls-profile. This tls-profile enables OCSP services and provides a list of one or more OCSP responders.
3. Assign the tls-profile to a TSCF port to enable OCSP service on that port.

For details on using the ACLI to configure OCSP-based certificate verification services, see TSCF OCSP Configuration.

TSCF Server Configuration

TSCF server configuration consists of the following steps; each step is identified as *required* or *optional*.

1. Configure TSCF global behavior (optional)

   Configuration of the *tscf-config* object specifies the handling of idle tunnel connections. Default values can be retained to enable standard tunnel processing.

2. Configure TSCF OCSP policy-based forwarding services (optional)

   3.<<The TSCF protocol policy configuration enables the forwarding of both tunneled and detunneled traffic depending on certain parameters in the packet headers. When packets are to be forwarded from one realm to another, or to a specific destination IP address as they are passing through the TSCF server, create **TSCF OCSP Protocol Policies**. Packets may also be directed to a LDAP server for user authentication, or detunneled traffic may be sent to an XMPP server.

4. Configure one or more TSCF address pools (required)

   A *tscf-address-pool* specifies one or more contiguous ranges of IPv4 addresses that are available for assignment to client applications. Later in the configuration process, you will assign an existing address pool to a TSCF interface. If the *tscf-address-pool* is used in conjunction with an optional *tscf-data-flow*, the combination provides a static egress route for tunneled traffic.

5. Configure one or more TSCF data flows (optional)

   This configuration while optional is almost always performed A *tscf-data-flow* operates in conjunction with a *tscf-address-pool* to provide a static egress route for tunneled traffic, usually to a gateway or application server.

6. Configure TLS or DTLS profiles (required for production environments)

   This configuration encrypts tunneled traffic using either TLS to encrypt streamed TCP packets or DTLS to encrypt UDP datagrams. In test/debug environments it can be advantageous to disable encryption. A *tls-profile* identifies the cryptographic resources available to ensure privacy using TLS or DTLS.

7. Configure one or more TSCF interfaces and ports (required)

   A tscf-interface and its associated *tscf-ports* provide server-side tunnel connectivity,

8. Enable DoS protection on TSCF ports (optional)

TSCF Global Configuration

TSCF global configuration specifies the handling of idle tunnel connections. By default, the TSCF server transitions an idle tunnel client from the *active* to the *persistent* state after an idle period of 300 seconds. Assuming the tunnel remains idle for an additional 330 seconds, the TSCF then transitions the tunnel from the *persistent* to the *closed* state — tearing the tunnel down and releasing its resources. If this behavior is consistent with your deployment, no changes are required or encouraged at the TSCF global configuration level. If local network conditions require modification, adjust the **keepalive-timer**, **keepalive-timer-datagram**, and **tunnel-persistence-time** parameters as described below.

TSCF global configuration also enables and manages *high-availability* (HA) topologies — topologies in which a pair of SDs operate in tandem, one in the *active* and the other in the *standby* role to provide reliable redundant operational availability. By default, HA is disabled. If your SD operates in standalone mode (not part of an HA pair), you can safely ignore all HA parameters and simply retain default values. If operating as part of an HA pair, use the **red-port** parameter to enable HA as described in the following procedure.

After enabling HA, Oracle Communications recommends the retention of default values for other HA parameters (**red-max-trans**, **red-sync-start-time**, and **red-sync-comp-time**) unless unusual local conditions require otherwise. Prior to modifying HA parameters, you should refer to your *Net-Net 4500 ACLI Configuration Guide* for more detailed HA information.

Use the following procedure to perform TSCF global configuration.

1. From superuser mode, use the following command sequence to access *tscf-config* configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# tscf
ACMEPACKET(tscf)# tscf-config
ACMESYSTEM(tscf-config)# ?

keepalive-timer            keepalive interval (in seconds)
keepalive-timer-datagram   keepalive interval (in seconds) for tunnels
                           with datagram transport
tunnel-persistence-time    tunnel persistence time after disconnection
                           (should be greater than keep-alive time)
red-port                   tscf redundancy sync port: 0 to disable and
                           2004 to enable
red-max-trans              maximum redundancy transactions to keep on
                           active
red-sync-start-time        timeout for transitioning from standby to
                           active
red-sync-comp-time         sync request timeout after initial sync
                           compeletion
element-id                 TSCF element Id, default : 0
options                    optional features/parameters
select                     select tscf config to edit
no                         delete tscf config
show                       show tscf config
done                       write tscf config information
exit                       return to previous menu

ACMESYSTEM(tscf-config)#
```

2. Use the **keepalive-timer**, **keepalive-timer-datagram**, and **tunnel-persistence-time** parameters to specify handling of idle tunnels.

   The timing of SIK transmissions is defined by the TSCF global parameters, **keepalive-timer-datagram** and **keepalive-timer**. **keepalive-timer-datagram** is used for datagram-based (UDP) tunnels, while **keepalive-timer** is used for stream-based (TCP) tunnels. In the absence of a user-supplied value for **keepalive-timer-datagram**, **keepalive-timer** provides a default timer value for all tunnels regardless of the underlying transport protocol.

   With SIK enabled, the server sends a keepalive message when no data is received from the client within the keepalive interval. The keepalive is sent approximately 15 seconds before timer expiration to ensure that it reaches the client before the client closes the tunnel for not receiving a keepalive message. For a datagram-based tunnel the TSCF server waits 5 seconds for a keepalive response or other data from the client. If no response is received, the server sends another keepalive message. This sequence is repeated until the server receives a keepalive response or other tunnel data, or a total of 3 keepalive messages has been sent, whichever occurs first. If no keepalive response or other data is received from a client (either connection-based or

datagram-based) within 15 seconds from the time the first keepalive message is sent, the server will tear down the tunnel, while saving the tunnel configuration if the tunnel-persistence-timer registers a non-zero value.

keepalive-timer specifies the maximum idle time (defined as no transmission activity within the tunnel) before the TSCF server transitions a stream-based (TCP) tunnel from the *active* to the *persistent* state. Supported values are 0 and integers within the range 30 - 550 (seconds), with a default value of 300. The special value 0 specifies that a keepalive mechanism is not employed. The integer values 30 through 550 specify the maximum size of the tolerated idle period for stream-based tunnels.

When **keepalive-timer-datagram** is set to its default value (0), keepalive-timer provides the default timer value for all tunnels, without regard to the transport protocol (TCP or UDP).

When **keepalive-timer** is set to a non-zero value, the TSCF client must initiate a keepalive message exchange before an idle period exceeds the assigned non-zero value. Failure of the client to transmit a keepalive prior to the timer expiration results in the server placing the tunnel in the *persistent* state.

The complete keepalive message exchange consists of a client-generated *Keep_Alive_Request* and a server-generated *Keep_Alive_Response* — essentially empty control messages consisting only of address fields and a common message header.

The periodic exchange of keepalive messages not only verifies the presence of the remote tunnel peer, but also facilitates the maintenance of NAT bindings through strict firewalls.

keepalive-timer-datagram specifies the maximum idle time (defined as no transmission activity within the tunnel) before the TSCF server transitions a datagram-based (UDP) tunnel from the *active* to the *persistent* state. Supported values are 0 and integers within the range 30 - 550 (seconds), with a default value of 0. The special value 0 specifies that a keepalive mechanism is not employed. The integer values 30 through 550 specify the maximum size of the tolerated idle period for datagram-based tunnels.

tunnel-persistence-time specifies the additional idle time tolerated before the TSCF server transitions an idle tunnel from the *persistent* to the *closed* state and tears down the tunnel.

Allowed values are 0 and integers within the range 10 - 600 (seconds), with a default value of 330.

The value 0 specifies that the TSCF server tears down the tunnel immediately upon expiration of the keep-alive timer. The integer values 10 through 600 specify the idle period, between transition from the *persistent* to the *closed* state.

During this interval, TSCF clients, currently in the persistent state, can initiate a keepalive message exchange which serves to reset the keepalive timer and return the client to the *active* state. During this same interval, recently disconnected clients, who are still in the *persistent* state, can re-connect and request their previous tunnel configuration.

For efficient and predictable operation, ensure that the value assigned to **tunnel-persistence-time** exceeds the value assigned to **keepalive-timer**.

```
ACMEPACKET(tscf-config)# keepalive-timer 30
ACMEPACKET(tscf-config)# tunnel-persistence-time 40
ACMEPACKET(tscf-config)#
```

3. Use the **element-id** parameter to assign a unique identifier to this TSCF server.

   Allowed values are 0 (the default) or an integer within the range 1 - 1023.

element-id can be safely ignored within network topologies that contain a single TSCF server. However, in topologies that contain multiple TSCF servers, each server must be assigned a unique network-wide identifier, provided by this parameter.

The assignment of a unique TSCF server guarantees unique tunnel identifiers regardless of the number of TSCF servers within the network.

```
ACMEPACKET(tscf-config)# element-id 10
ACMEPACKET(tscf-config)#
```

4. In HA topologies use the **red-port**, **red-max-trans**, **red-sync-start-time**, and **red-sync-comp-time** parameters to configure reliable redundant operations.

   Use the **red-port** parameter to specify the UDP port number that supports TSCF synchronization message exchanges.

   The default value (0) disables redundant HA configurations. Use a port value of *2004* to enable HA operations.

   Use the **red-max-trans** parameter to specify the maximum number of retained TSCF synchronization messages.

   Allowed values are integers within the range 0 through 999999999 (messages) with a default of *10000*.

   Use the **red-sync-start-time** parameter to set the timer value (in milliseconds) for transition from the standby to the active role — that is the maximum period of time that the standby SD waits for a heartbeat signal from the active SD before assuming the active role.

   Allowed values are integers within the range 0 through 999999999 (milliseconds) with a default of *5000*.

   Use the **red-sync-comp-time** attribute to specify the interval between synchronization attempts after the completion of a TSCF redundancy check.

   Allowed values are integers within the range 0 through 999999999 (milliseconds) with a default of *1000*.

   ```
   ACMEPACKET(tscf-config)# red-port 2004
   ACMEPACKET(tscf-config)# red-trans 15000
   ACMEPACKET(tscf-config)# red-sync-start-time 2500
   ACMEPACKET(tscf-config)# red-sync-comp-time 750
   ACMEPACKET(tscf-config)#
   ```

5. Use **done**, **exit**, and **verify-config** to complete TSCF global configuration.

TSCF Protocol Policy Configuration

Use the following procedure to configure TSCF policy-based forwarding services.

Policy-based forwarding requires the creation of a tscf-protocol-policy and the assignment of that policy to a tscf-address-pool.

1. From superuser mode, use the following command sequence to access cert-status-profile configuration mode. While in this mode, you configure a certificate-status-profile, a container for the information required to access a single, specific OCSP responder.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# tscf
   ACMEPACKET(tscf)# tscf-protocol-policy
   ACMEPACKET(tscf-protocol-policy)#
   ```

2. The required **name** parameter differentiates tscf-protocol-policy instances from one another. Each tscf-protocol-policy specifies filtering/match criteria applied to incoming packets.
3. The optional **ip-address** parameter works in conjunction with the required **port** and **transport-type** parameters to identify incoming packets subject to this tscf-protocol-policy.

   For client-side, tunneled packets **ip-address** is matched against the inner packet's destination address. For non-tunneled packets, **ip-address** is matched against the source IP-address.

   For client-side tunneled packets, **port** is matched against the inner packet's destination port. For non-tunneled packets, **port** is matched against the source port.

   Retain the default value (0.0.0.0) or specify a specific IP address.
4. The required **transport-type** parameter works in conjunction with the optional **ip-address** and required **port** parameters to identify incoming packets subject to this tscf-protocol-policy.

   For client-side, tunneled packets, **transport-type** is matched against the inner packet's transport protocol (UDP or TCP). For non-tunneled packets, **transport-type** is matched against the packet's transport protocol.

   Retain the default value (UDP), or select TCP.
5. For all tunneled packets that meet the matching criteria specified by the **ip-address**, **port**, and **transport-type** parameters, the required **realm-id** parameter specifies the egress realm. Matching packets are forwarded to the gateway that services the interface associated with the named realm.
6. Optionally, for all packets that meet the matching criteria specified by the **ip-address**, **port**, and **transport-type** parameters, the tunneled packets will be forwarded to the configured **remote-ip-address**. The destination IP address of the detunneled packet will be changed to this value.

   When the packets are coming from non-tunneled side to tunneled side, the source IP address must match the **remote-ip-address**. The source IP address will be changed back to the original destination IP address within the tunneled packet.
7. Use **done**, **exit**, and **verify-config** to complete configuration of this tscf-protocol-policy.
8. Repeat Steps 1 through 7 to configure additional tscf-protocol-policies if necessary.

For instructions on how to assign a TSCF protocol policy to a TSCF address pool, see the section TSCF Address Pool Configuration in this document. The parameter is called **protocol-policy**, and appears in Step 1 and Step

TSCF Address Pool Configuration

During the configuration stage as described in TSCF Overview, the TSCF server assigns a *tunnel* IP address to the client application. These assigned addresses are obtained by the TSCF server from a *tscf-address-pool*, a configuration object that contains an IPv4 address list. The IPv4 address list contains one or more IPv4 address ranges. Each address range consists of contiguous IPv4 addresses, and can contain a minimum of 1, or a maximum of 262,144 list entries.

The address range size, the list size, and the size of the *tscf-address-pool* are constrained by the same maximum value. Consequently, while the IPv4 address list can contain one or several ranges, the total number of IPv4 addresses contained in the individual address ranges cannot exceed 262,144.

1. Use the following required procedure to configure a *tscf-address-pool* configuration object. Later, you will assign the address pool to a specific TSCF interface.From superuser mode, use the following command sequence to access *tscf-address-pool* configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# tscf
ACMEPACKET(tscf)# tscf-address-pool
ACMEPACKET(tscf-address-pool)# ?

name                 name for the tscf address pool
address-range        address ranges for the local address pool
dns-realm-id         dns realm identifier for local-pool
data-flow            data flow name for managing data traffic with the
                     pool
protocol-policy      comma separated protocol policy name/s for
                     managing specific protocol traffic with the pool
batch                enter all arguments on one line
select               select a local address pool to edit
no                   delete selected local address pool
show                 show selected local address pool
done                 write local address pool information
exit                 return to previous menu

ACMEPACKET(tscf-address-pool)#
```

2. Use the **name** parameter to provide a unique identifier for this *tscf-address-pool*.

```
ACMEPACKET(tscf-address-pool)# name tscf-lap-1
ACMEPACKET(tscf-address-pool)#
```

3. Retain the default value (an empty string) for the **dns-realm-id** parameter.
4. Use the **address-range** parameter to access *address-range* sub-configuration mode. While in this mode, you compile an IPv4 address list, and assign contiguous ranges of IPv4 addresses to the list.

```
ACMEPACKET(tscf-address-pool)# address-range
ACMEPACKET(tscf-address-range)# ?

network-address      base network address for this range
subnet-mask          subnet mask for address range
batch                enter all arguments on one line
select               select a local address pool address range to edit
no                   delete selected local address pool range
show                 show selected local address pool range
done                 write local address pool range information
exit                 return to previous menu

ACMEPACKET(tscf-address-range)#
```

5. Use the **network-address** and **subnet-mask** ALCI parameters to specify a contiguous range of IPv4 addresses assigned to this *tscf-address-pool*.

   network-address can take any valid IPv4 address value.

subnet-mask values should take into consideration the projected number of tunnels to be supported by TSCF ports. To ensure that IPv4 address lists do not exceed the maximum supported value (262,144), mask values must begin with the initial 14 bits set to 1. The following table lists supported subnet-mask values along with the size of the resulting address pool.

Supported subnet-mask Values	Address Pool Size
255.252.0.0	262,144
255.254.0.0	131,072
255.255.0.0	65,536
255.255.128.0	32,768
255.255.192.0	16,384
255.255.224.0	8,192
255.255.240.0	4,096
255.255.248.0	2,048
255.255.252.0	1,024
255.255.254.0	512
255.255.255.0	256
255.255.255.128	128
255.255.255.192	64
255.255.255.224	32
255.255.255.240	16
255.255.255.248	8
255.255.255.252	4
255.255.255.254	2
255.255.255.255	1

This command example allocates the maximum allowed address range to current the IPv4 address list.

```
ACMEPACKET(tscf-address-pool)# network-address 10.244.0.0
ACMEPACKET(tscf-address-pool)# subnet-mask 255.252.0.0
ACMEPACKET(tscf-address-pool)#
```

This command example allocates 4096 IP addresses to the IPv4 address list.

```
ACMEPACKET(tscf-address-pool)# network-address 10.244.0.0
ACMEPACKET(tscf-address-pool)# subnet-mask 255.255.240.0
ACMEPACKET(tscf-address-pool)#
```

6. Use **done** and **exit** to add the address range to the IPv4 address list, and to return to *tscf-address-pool* configuration mode.
7. If necessary, repeat Steps 5 and 6 to add additional contiguous address ranges to the IPv4 address list.

   This command example allocates a second range of 16,384 IPv4 addresses, to the IPv4 address list.

   ```
   ACMEPACKET(tscf-address-pool)# network-address 172.16.240.0
   ACMEPACKET(tscf-address-pool)# subnet-mask 255.255.192.0
   ACMEPACKET(tscf-address-pool)#
   ```

8. Use the **data-flow** parameter to assign a specific *tscf-data-flow* to this *tscf-address-pool*.

   Refer to TSCF Data Flow Configuration for configuration details.

   ```
   ACMEPACKET(tscf-address-pool)# data-flow tscf-df-1
   ACMEPACKET(tscf-address-pool)#
   ```

9. Use the **protocol-policy** parameter to assign specific protocol traffic to the pool. Use the name of the policy that was assigned to it when it was created. If assigning multiple policies, use a comma separated list.

   ```
   ACMEPACKET(tscf-address-pool)# protocol-policy <tscf-address-policy>
   ```

10. Use **done**, **exit**, **save**, and **verify-config** to complete configuration of the *tscf-address-pool* instance.

    verify-config confirms that a valid IPv4 address and subnet mask are present in each IPv4 address list, and that the total number of IPv4 addresses contained in the list does not exceed 261,144. If the verification process finds errors, **verify-config** issues appropriate explanatory error messages.

11. Repeat Steps 1 through 9 to configure additional *tscf-address-pool* instances.

TSCF Data Flow Configuration

Use the following procedure to configure an optional *tscf-data-flow* object. If you are not using *tscf-data-flows* to provide to provide static egress routes, this procedure can be safely ignored.

1. From superuser mode, use the following command sequence to access *tscf-data-flow* configuration mode. In this mode you configure *tscf-data-flows* that enable static pass-thru of tunneled data via a specified realm.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# tscf
   ACMEPACKET(tscf)# tscf-data-flow
   ACMEPACKET(tscf-data-flow)# ?

   name                name for a data flow
   realm-id            realm-id to route the upstream data flow
   group-size          the number of UEs to be managed by the data flow
   upstream-rate       Upstream bandwidth (KB/s) for the data flow
   downstream-rate     Downstream bandwidth (KB/s) for the data flow
   batch               enter all arguments on one line
   select              select a data flow to edit
   no                  delete selected data flow
   show                show selected data flow
   done                write data flow information
   exit                return to previous menu

   ACMEPACKET(tscf-data-flow)#
   ```

2. Use the **name** parameter to provide a unique identifier for this *tscf-data-flow* instance.

   ```
   ACMEPACKET(tscf-data-flow)# name tscf-dt-01
   ACMEPACKET(tscf-data-flow)#
   ```

3. Use the **realm-id** parameter to identify the egress realm.

   ```
   ACMEPACKET(tscf-data-flow)# realm-id core-1
   ACMEPACKET(tscf-data-flow)#
   ```

4. Use the **group-size** parameter to specify the size and number of address blocks supported by this **data-flow** instance.

   The size of the associated *tscf-address-pool* (defined as the number or IPv4 addresses contained in the pool), is divided by the **group-size** value to segment the address pool

into smaller address blocks. After determining the start address for each block, the Session Director establishes two static flows for each of the address blocks — a *downstream* flow, in the access direction, and an *upstream* flow generally toward a core application server gateway/router that provides forwarding service for pass-thru data-flow.

As a result of the static-flow establishment, each address block consumes 4 NAT entries, 2 for the *downstream* flow and 2 for the *upstream* flow. Because the current software release imposes a maximum restriction of 4096 NAT entries per *data-flow* instance, dividing 4096 by 4 provides the largest number of supported address blocks, 1024.

Use the following table to associate supported **group-size** values with address pool size.

For example:

The largest supported address pool (262,144 IPv4 addresses) requires a group size of 256 addresses, which produces the maximum 1,024 address blocks with each block containing 256 addresses.

An address pool containing 131,072 addresses can be segmented as 512 address blocks, with each block containing 256 addresses, or as 256 blocks, with each block containing the maximum 1,024 addresses.

An address pool containing 1,024 addresses can be segmented as follows.

- 1 block containing 1,024 addresses
- 2 blocks containing 512 addresses
- 4 blocks containing 256 addresses
- 8 blocks containing 128 addresses
- 16 blocks containing 64 addresses
- 32 blocks containing 32 addresses
- 64 blocks containing 16 addresses
- 128 blocks containing 8 addresses
- 256 blocks containing 4 addresses

Address Pool Size	Supported group-size Values
262,144	256
131,072	128, 256
65,536	64, 128, 256
32,768	32, 64, 128, 256
16,384	16, 32, 64, 128, 256
8,192	8, 16, 32, 64, 128, 256
4,096	4, 8, 16, 32, 64, 128, 256
2,048	2, 4, 8, 16, 32, 64, 128, 256
1,024	1, 2, 4, 8, 16, 32, 64, 128, 256
512	1, 2, 4, 8, 16, 32, 64, 128, 256
256	1, 2, 4, 8, 16, 32, 64, 128, 256
128	1, 2, 4, 8, 16, 32, 64, 128
64	1, 2, 4, 8, 16, 32, 64
32	1, 2, 4, 8, 16, 32
16	1, 2, 4, 8, 16
8	1, 2, 4, 8
4	1, 2, 4
2	1, 2

```
1                                                   1

ACMEPACKET(tscf-data-flow)# group-size 128
ACMEPACKET(tscf-data-flow)#
```

5. Use the **upstream-rate** parameter to specify the allocated upstream (core-side) bandwidth.

   Allowable values are integers within the range 0 (the default) through 122070 (KB/s).

   The 0 value allocates all available upstream bandwidth.

   ```
   ACMEPACKET(tscf-data-flow)# upstream-rate 100
   ACMEPACKET(tscf-data-flow)#
   ```

6. Use the **downstream-rate** parameter to specify the allocated downstream (access-side) bandwidth.

   Allowable values are integers within the range 0 (the default) through 122070 (KB/s).

   The 0 value allocates all available downstream bandwidth.

   ```
   ACMEPACKET(tscf-data-flow)# downstream-rate 100
   ACMEPACKET(tscf-data-flow)#
   ```

7. Use **done**, **exit**, and **verify-config** to complete configuration of this *tscf-data-flow* instance.

   verify-config confirms that the total number of IPv4 addresses assigned to the *data-flow* instance does not exceed 262,144 and that the total number of NAT entries consumed by the *data-flow* instance does not exceed 4,096. In the event of discrepancies, **verify-config** issues an error message recording the actual number of IPv4 addresses and/or the number of NAT entries.

Repeat Steps 1 through 7 to configure additional *tscf-data-flow* instances.

TLS Profile Configuration

Use the following required procedure to configure a *tls-profile* configuration object that identifies the cryptographic resources, specifically certificates and protocols, required for tunnel creation. Later, you will assign the *tls-profile* to a specific TSCF port.

1. From superuser mode, use the following command sequence to access *tls-profile* configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# tls-profile
   ACMEPACKET(tls-profile)# ?

   name                      tls profile name
   end-entity-certificate    end entity certificate for the TLS
                             connection
   trusted-ca-certificates   list of trusted certificate records
   cipher-list               list of ciphers
   verify-depth              maximum length of the certificate chain
                             [default:10, valid range: 0-10].
   mutual-authenticate       mutually authenticate [default :"disabled"].
   tls-version               supported versions are "tlsv1", "sslv3" and
                             "compatibility" [default]
   options                   supported options
   cert-status-check         cert-status-check state [enabled|disabled]
   cert-status-profile-list  list of cert-status-profiles for status
                             requests
   ignore-dead-responder     Ignore dead cert status responder
                             [default:disabled]
   allow-self-signed-cert    allow self-signed certificate
                             [enabled|disabled]
   ```

```
select                    select a tls profile to edit
no                        delete selected tls profile
show                      show tls profile information
done                      write tls profile information
exit                      return to previous menu

ACMEPACKET(tls-profile)#
```

2. Use the **name** parameter to provide a unique identifier for this *tls-profile*.

```
ACMEPACKET(tls-profile)# name tscfTls01
ACMEPACKET(tls-profile)#
```

3. Use the required **end-entity-certificate** parameter to supply the unique name of a *certificate-record* configuration element referencing the identification credential (specifically, an X509.v3 certificate) offered by the TSCF server in support of its asserted identity.

```
ACMEPACKET(tls-profile)# end-entity-certificate tscfServerCert01
ACMEPACKET(tls-profile)#
```

4. Use the required **trusted-ca-certificates** parameter to compile a list or one or more *certificate-record* configuration elements referencing trusted Certification Authority (CA) certificates used to authenticate a client application.

   Provide a comma separated list of existing CA **certificate-record** configuration elements.

```
ACMEPACKET(tls-profile)# trusted-ca-certificates
                         verisignClass3-a,verisignClass3-
                         b,baltimore,thawtePremium
ACMEPACKET(tls-profile)#
```

5. Retain the default value, **all**, for the **cipher-list** parameter.

6. Use the **verify-depth** parameter to specify the maximum number of chained certificates that will be processed while authenticating the client application.

   Provide an integer within the range 1 through 10 (the default).

   The TSCF server supports the processing of certificate chains when X.509v3 certificate-based authentication is used during tunnel establishment. The following process validates a received TLS certificate chain.

   a. Check the validity dates (*Not Before* and *Not After* fields) of the end certificate. If either date is invalid, authentication fails; otherwise, continue chain validation

   b. Check the maximum length of the certificate chain (specified by **verify-depth**). If the current chain exceeds this value, authentication fails; otherwise, continue chain validation.

   c. Verify that the *Issuer* field of the current certificate is identical to the *Subject* field of the next certificate in the chain. If values are not identical, authentication fails; otherwise, continue chain validation.

   d. Check the validity dates (*Not Before* and *Not After* fields*)* of the next certificate. If either date is invalid, authentication fails; otherwise, continue chain validation.

   e. Check the *X509v3 Extensions* field to verify that the current certificate identifies a CA. If not so, authentication fails; otherwise, continue chain validation.

f. Extract the *Public Key* from the current CA certificate. Use it to decode the *Signature* field of the prior certificate in the chain. The decoded *Signature* field yields an MD5 hash value for the contents of that certificate (minus the *Signature* field).

g. Compute the same MD5 hash. If the results are not identical, authentication fails; otherwise, continue chain validation.

h. If the hashes are identical, determine if the CA identified by the current certificate is a trust anchor by referring to the *trusted-ca-certificates* attribute of the associated TLS-profile configuration object. If the CA is trusted, authentication succeeds. If not, return to Step 2.

```
ACMEPACKET(tls-profile)# verify-depth 8
ACMEPACKET(tls-profile)#
```

7. Use the **mutual-authenticate** parameter to **enable** or **disable** (the default) mutual authentication.

   Protocol requirements mandate that the server present its certificate to the client application. Optionally, the server can implement mutual authentication by requesting a certificate from the client application, and authenticating the certificate offered by the client.

   Upon receiving a server certificate request, the client application must respond with a certificate; failure to do so results in authentication failure.

```
ACMEPACKET(tls-profile)# mutual-authenticate disabled
ACMEPACKET(tls-profile)#
```

8. Retain the default value, **compatibility**, for the **tls-version** parameter.
9. Retain default values for all other parameters.
10. Use **done**, **exit**, and **verify-config** to complete *tls-profile* configuration.
11. Repeat Steps 1 through 10 to configure additional *tls-profiles* as required.

TSCF OCSP Configuration

The following sections provide instruction on using the ACLI to configure OCSP-based certificate revocation services.

Providing OCSP services requires the creation of a secure TLS connection between a TSCF port and one or more OCSP responders. This configuration is a three-step process.

1. Create one or more certificate-status-profiles. Each certificate-status-profile provides the information and cryptographic resources required to access a single OCSP responder.
2. Assign one or more certificate-status-profiles to a tls-profile. This tls-profile enables OCSP services and provides a list of one or more OCSP responders.
3. Assign the tls-profile to a TSCF port to enable OCSP service on that port.

Configure a certificate-status-profile.

1. From superuser mode, use the following command sequence to access cert-status-profile configuration mode. While in this mode, you configure a certificate-status-profile, a container for the information required to access a single, specific OCSP responder.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# cert-status-profile
ACMEPACKET(cert-status-profile)#
```

2. Therequired **name** parameter differentiates certificate-status-profile instances from one another. Each certificate-status-profile provides configuration information for a single, specific OCSP responder.
3. The **type** parameter selects the certificate revocation check methodology; the only currently supported value is OCSP.
4. Retain the default value (http) for the **trans-protocol** parameter, which identifies the transport method used to access the OCSP responder.
5. The **ip-address** parameter works in conjunction with the **port** parameter to locate the OCSP responder.

   ip-address identifies the OCSP responder by its IP address. **port** identifies the port monitored by the responder for incoming OCSP requests.

   Allowable port values are integers within the range 1025 through 65535. In the absence of an explicitly configured value, a default value of 80, the well-known HTTP port, is provided.
6. The **realm-id** parameter specifies the realm used to transmit OCSP requests and receive OCSP responses.

   In the absence of an explicitly configured value, the default value of *wancom0*, specifying OCSP protocol transmissions across the wancom0 management interface, is used.
7. The **requester-cert** parameter is meaningful only if OCSP requests are signed; ignore this parameter if requests are not signed.

   RFC 2560 does not require the digital signature of OCSP requests. OCSP responders, however, can impose signature requirements.

   If a signed request is required by the OCSP responder, provide the name of the certificate configuration element that references the certificate used to sign OCSP requests.
8. The **responder-cert** parameter identifies the certificate used to validate signed OCSP response — a public key of the OCSP responder.

   RFC 2560 requires that all OCSP responders digitally sign OCSP responses, and that OCSP requesters validate incoming signatures.

   Provide the name of the certificate configuration element that references the certificate used to validate the signed OCSP response.
9. The **retry-count** parameter specifies the maximum number of times to retry an OCSP responder in the event of connection failure.

   If the retry counter specified by this parameter is exceeded, the OCSP requester contacts another responder (if multiple responders have been configured) and quarantines the unavailable responder for a period defined by the dead-time parameter.

   In the absence of an explicitly configured value (an integer within the range 0 through 10), the default value of 1 (connection retries) is used.
10. The **dead-time** parameter specifies the quarantine period imposed on an unavailable OCSP responder. During the quarantine period, the TSCF will not send OCSP requests to the OSCP responder.

    In the absence of an explicitly configured value (an integer within the range 0 through 3600 seconds), the default value of *0* (no quarantine period) is used.

    Note: Customer implementations utilizing a single OCSP responder are encouraged to retain the default value of the **dead-time** parameter, or to specify a brief quarantine period to prevent lengthy service outages.

11. The **hostname** and **trusted-ca** parameters can be safely ignored.
12. Use **done**, **exit**, and **verify-config** to complete configuration of this certificate-status-profile.
13. Repeat Steps 1 through 12 to configure additional certificate-status-profiles if necessary.

Assign the certificate-status-profile to a TLS profile.

1. From superuser mode, use the following command sequence to access tls-profile configuration mode. While in this mode, you can alter an existing TLS profile to support OCSP transactions.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# tls-profile
ACMEPACKET(tls-profile)#
```

2. Use the **select** command to identify a specific tls-profile that will support OCSP requests and responses.
3. Set the **mutual-authenticate** parameter to enabled.
4. Use the **cert-status-check** parameter to enable certificate status checking on this tls- profile.

   Set this parameter to enabled.
5. Use the **cert-status-profile-list** parameter to assign an OCSP responder or responders to this TLS profile.

   For example, this ACLI sequence assigns a single OCSP responder.

```
ACMEPACKET(tls-profile)#  cert-status-profile-list OCSP_Symantic
ACMEPACKET(tls-profile)#
```

   Use quotation marks to assign multiple OCSP responders. For example, this ACLI sequence assigns three certificate-status-profiles, OCSP_Symantic, OCSP_Thawte, and OCSP_Entrust to this TLS profile.

```
ACMEPACKET(tls-profile)#  cert-status-profile-list “OCSP_Symantic
OCSP_Thawte OCSP_Entrust”
ACMEPACKET(tls-profile)#
```

   Each assigned certificate-status-profile contains the data needed to access a single OCSP responder. Responders are accessed in list order, from first to last. In the above example, the TSCF first contacts the Symantic OCSP responder. If the responder cannot be contacted within the limits specified by the **retry-count** parameter of the certificate-status-profile, the responder is quarantined for the time specified by the dead-time parameter, and the TSCF moves to the next list entry.
6. Use **done**, **exit**, and **verify-config** to complete this configuration.
7. If necessary, repeat this procedure to prepare other TLS profiles for OCSP-based certificate checking support.

Assign the tls-profile to a TSCF port.

1. From superuser mode, use the following command sequence to access tscf-port configuration mode. While in this mode, you assign an existing TLS profile to a TSCF port.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# tscf
ACMEPACKET(tscf)# tscf-interface
ACMEPACKET(tscf-interface)# tscf-ports
ACMEPACKET(tscf-port)#
```

2. Use the **select** command to identify a specific tscf-port that will support OCSP requests and responses.
3. Use the **tls-profile** parameter to assign an OCSP-enabled tls-profile to the current TSCF port enabled.
4. Use **done**, **exit**, and **verify-config** to complete this configuration.
5. If necessary, repeat this procedure to prepare other TSCF ports for OCSP-based certificate checking support.

Sample OCSP Configurations

A sample certificate-status-profile configuration follows:

```
ACMEPACKET# show running-config cert-status-profile
cert-status-profile
        name                            OCSP_Symantic
        ip-address                      192.0.2.100
        hostname
        port                            8080
        type                            OCSP
        trans-proto                     HTTP
        requestor-cert                  signOCSP
        responder-cert                  SymanticPublic-1
        trusted-cas
        realm-id                        wancom0
        retry-count                     1
        dead-time                       60
        last-modified-by                admin@console
        last-modified-date              2014-07-24 18:25:25
task done
ACMEPACKET#
```

This configuration creates a certificate-status-profile named OCSP_Symantic. The profile identifies an OCSP responder located at 192.0.2.100:8080. The required **responder-cert** parameter specifies the CA public certificate used by the TSCF to verify the signed OCSP response. The optional **requester-cert** parameter indicates that the OCSP responder requires signed requests, and identifies the certificate used by the TSCF to digitally sign OCSP requests. The optional **dead-time** parameter imposes a 60 second quarantine if the OCSP responder is unreachable. Retention of default values for the **realm-id** and **retry-count** parameters specify OCSP responder access via the wancom0 management interface and a retry count of *1*.

A sample tls-profile configuration follows:

```
ACMEPACKET# show running-config tls-profile
tls-profile
        name                            TLS_OCSP
        end-entity-certificate          TSCFCert_1
        trusted-ca-certificates         CA_Symantic
                                        CA_Thawte
                                        CA_Entrust
                                        CA_DigiSign
        cipher-list                     All
        verify-depth                    10
        mutual-authenticate             enabled
        tls-version                     compatibility
        cert-status-check               enabled
```

```
        cert-status-profile-list       OCSP_Symantic
                                       OCSP_Thawte
                                       OCSP_Entrust
        ignore-dead-responder          disabled
        allow-self-signed-cert         disabled
        last-modified-by               admin@console
        last-modified-date             2014-07-24 19:40:37
task done
ACMEPACKET#
```

This configuration creates a tls-profile named TLS_OCSP. The profile uses the **mutual-authenticate** parameter to enable mutual authentication between the TSCF and the OCSP responders, the **cert-status-check** parameter to enable OCSP services, and the **cert-status-profile-list** parameter to identify three OCSP responders.

A sample portion of a tscf-interface/tscf-port configuration follows:

```
ACMEPACKET# show running-config tscf-interface
tscf-interface
     realm-id                      access
     state                         enabled
     max-tunnels                   200000
     local-address-pools            pool1
     nagle-state                   enabled
     assigned-services             SIP,redundancy,DDT,
                                   server-keepalive
     tscf-port
       address                     172.16.21.2
       port                        443
       transport-protocol          TLS
       tls-profile                 TLS_OCSP

...
...
...

        last-modified-by              admin@console
        last-modified-date            2014-07-24 19:51:03
task done
ACMEPACKET#
```

This configuration enables OCSP support on the TSCF port 172.16.21.2:443.

Monitoring OCSP Operations

The TSCF generates an SNMP trap when a configured OSCP responder becomes unreachable. It generates second trap when connectivity is re-established with a previously unreachable OCSP responder.

The **show security ocsp stats** ACLI command provides OCSP operational counts.

```
ACMEPACKET# show security ocsp stats
18:49:34-147
CERTD OCSP Statistics              ---- Lifetime ----
                                 Recent      Total  PerMax
Req Rcvd from Cavium                0          0       0
Resp Sent to Cavium                 0          0       0
OCSP Req Sent                       0          0       0
```

```
OCSP Resp Rcvd                    0          0       0
OCSP Timeout                      0          0       0
Duplicate OCSP Request            0          0       0
OCSP Status-Good                  0          0       0
OCSP Status-Revoked               0          0       0
OCSP Status-Unknown               0          0       0
```

Key to OCSP Operational Counts:

- **Req RCVD from Cavium**—Number of certificates presented to the OCSP task for verification.
- **Resp Sent to Cavium**—Number of valid responses (***good***, ***revoked***, ***unknown***) received from OCSP responders.
- **OCSP Req Sent**—Number of requests sent to OCSP responders.
- **OCSP RespRcvd**—Number of responses received from OCSP responders.
- **OCSP Timeout**—Number of unanswered OCSP requests plus the number of internal timeouts.
- **OCSP Status-Good**—Number of **good** status responses received from OCSP responders.
- **OCSP Status-Revoked**—Number of **revoked** status responses received from OCSP responders.
- **OCSP Status-Unknown**—Number of **unknown** status responses received from OCSP responder.

TSCF Interface Configuration

TSCF interface configuration specifies the SD IP address that is accessed by TSCF clients to initiate tunnel creation, assigns resources that facilitate tunnel creation, identifies specific TSCF services offered by the interface, and limits the number of supported tunnels.

Use the following procedure to configure a TSCF interface — keeping in mind that

- a TSCF interface must be physically supported by an ETC NIU with a minimum of 8GB of installed DRAM
- a TSCF interface and SIP interface cannot coexist on the same network interface

1. From superuser mode, use the following command sequence to access *tscf-interface* configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# tscf
ACMEPACKET(tscf)# tscf-interface
ACMEPACKET(tscf-interface)# ?

realm-id            realm identifier
state               state
max-tunnels         maximum number of tunnels
local-address-pools address pool name
assigned-services   comma-separated list of services
options             optional features/parameters
tscf-ports          list of TSCF ports
select              select tscf config to edit
no                  delete tscf config
show                show tscf config
done                write tscf config information
exit                return to previous menu

ACMEPACKET(tscf-interface)#
```

2. Use the optional **state** parameter to select the TSCF interface operational state.

   Supported values are **enabled** (the default) | **disabled**.

   Retain the default value to place the TSCF interface in an active operational state. Use **disabled**. to place the TSCF interface in a non-operational state.

```
ACMEPACKET(tscf-interface)# state enabled
ACMEPACKET(tscf-interface)#
```

3. Use the required **realm-id** parameter to identify the realm that supports TSCF client access to the SD.

   Enter the realm name.

```
ACMEPACKET(tscf-interface)# realm-id tsmAccess
ACMEPACKET(tscf-interface)#
```

4. Use the optional **nagle-state** parameter to select the operational mode of the Nagle algorithm on this TSCF interface.

   Retain the default value (enabled) to support Nagle algorithm operations on this interface; use the alternate value (disabled) to disable algorithm operations.

   ```
   ACMEPACKET(tscf-interface)# nagle-state disabled
   ACMEPACKET(tscf-interface)#
   ```

5. Use the optional **assigned-services** parameter to enable one of more services supported by this TSCF interface. Currently supported services are **ddt** (refer to Dynamic Datagram Tunneling), **nagle** (refer to The show tscf statistics, show tscf tunnel all, and show tscf tunnel detailed ACLI commands all provide various counts of HA tunnel restoration operations.), **redundancy** (refer to Tunnel Redundancy), **server-keepalive** (refer to Server-Initiated Keepalive), and **sip**.

   Retain the default value, **sip**, to enable tunneling of SIP traffic.

   Use the keyword, **ddt**, to enable the establishment of a linked tunnel pair — a persistent TCP/TLS tunnel to carry signalling data, and an on-demand UDP/DTLS tunnel for datagram traffic.

   Use the keyword, **nagle**, to support the override of the TSCF-interface-specific Nagle algorithm state (enabled | disabled) on an individual tunnel.

   If **nagle** is specific as an assigned service, the ACLI **verify-config** command issues a warning if it fails to find an existing TCP/TLS port.

   Use the keyword, **redundancy**, to enable tunnel redundancy for improved call quality under adverse packet loss.

   Use the keyword, **server-keepalive**, to enable the periodic generation of TSCF-initiated keepalive messages on an idle tunnel — defined as no transmission activity within the tunnel.

   To enable multiple services, for example SIP and tunnel redundancy, use **sip,redundancy**. The required comma (,) serves as an argument delimiter.

   ```
   ACMEPACKET(tscf-interface)# assigned-services sip,redundancy
   ACMEPACKET(tscf-interface)#
   ```

   To enable dynamic datagram tunneling, use **sip,ddt**.

   ```
   ACMEPACKET(tscf-interface)# assigned-services sip,ddt
   ACMEPACKET(tscf-interface)#
   ```

6. Use the optional **max-tunnels** parameter to specify the maximum number of concurrent tunnels supported by this TSCF interface.

   This parameter allows available tunnel capacity (specified by the installed TSCF license) to be allocated across multiple realms. Allowable values are any integer from 0 (the default) to the maximum license capacity.

   The special value, 0, indicates that the interface-specific limit is bounded only the licensed value.

   ```
   ACMEPACKET(tscf-interface)# max-tunnels 20000
   ACMEPACKET(tscf-interface)#
   ```

7. Use the required **local-address-pools** parameter to identify the local address pool that provides tunnel addresses to TSCF clients.

   A specific address from the address pool will be transmitted to the TSCF client as part of the tunnel creation process. This address provides the inner source address for tunneled packets originated by the TSCF client application.

   Refer to TSCF Address Pool Configuration for address pool details.

   ```
   ACMEPACKET(tscf-interface)# local-address-pools tscfPool172
   ```

```
ACMEPACKET(tscf-interface)#
```

8. Use the **tscf-ports** parameter to move to *tscf-port* configuration mode.

```
ACMEPACKET(tscf-interface)# tscf-port
ACMEPACKET(tscf-port)# ?

address             IP Address
port                port
transport-protocol  transport protocol
tls-profile         tls profile name
select              select a tscf port to edit
no                  delete selected tscf port
show                show tscf port information
done                write tscf port information
exit                return to previous menu

ACMEPACKET(tscf-port)#
```

9. Use the required **address** and **port** parameters to specify the IP address and port number monitored for incoming tunnel client messages.

   This address and port number will be transmitted to the TSCF client as part of the tunnel creation process. This address provides the outer destination address for tunneled packets originated by the TSCF client application.

   The TSCF port IP address must match the address range of the realm designated by the **realm-id** parameter. As previously stated, the physical interface must be provided by an ETC NIU.

   Enter a currently configured IP address and a port number with the range 1025 through 65535, with a default value of 5060.

```
ACMEPACKET(tscf-port)# address 192.168.11.22
ACMEPACKET(tscf-port)# port 443
ACMEPACKET(tscf-port)#
```

10. Use the optional **transport-protocol** parameter to identify the tunnel transport protocol.

    Supported values are **TLS** (the default, TCP/TLS) | **DTLS** (UDP/DTLS) | **TCP** | **UDP**. Note that the unsecured TCP and UDP protocols are only used for debug purposes, and are not intended for operational environments.

```
ACMEPACKET(tscf-port)# transport-protocol TLS
ACMEPACKET(tscf-port)#
```

11. If the **transport-protocol** parameter is either **TLS** or **DTLS**, use the required **tls-profile** parameter to identify the TLS profile that specifies the cryptographic resources available to support TLS or DTLS operations.

    If the specified tunnel transport protocol is either TCP or UDP (non-operational environments only), this parameter can be safely ignored.

    Refer to TSCF Interface Configuration for details.

```
ACMEPACKET(tscf-port)# tls-profile tscfTLS01
ACMEPACKET(tscf-port)#
```

12. Use **done** and **exit** to return to *tscf-interface* configuration mode.

13. If required, repeat Steps 7 through 11 to configure additional TSCF ports.
14. Use **done**, **exit**, and **verify-config** to complete *tscf-interface* configuration.

    When validating the *tscf-interface* configuration **verify-config** confirms the following

    - the presence of an ETC NIU equipped with a minimum of 8GB of installed DRAM, and issues an ERROR: MINOR ALARM message if a compliant NIU is absent
    - the realm specified by the **realm-id** parameter actually exists, and issues an error message in the case of a non-existent realm
    - the TSCF port address matches the address specified by the **realm-id** parameter
    - the presence of an associated SIP Interface, and issues an error message in the case of a non-existent realm
    - the TSCF address pool specified by the **local-address-pool** parameter actually exists, and issues an error message in the case of a non-existent address pool
    - the TSCF address pool is not mis-configured, and issues an error message if it is
    - the presence of one or more configured *tscf-ports*, and issues a warning message in the event of a missing port
    - if **assigned-services** is *ddt,sip*, the configuration of matching (same **address** and **port** values) stream-based and datagram-based tunnels — issues an error message in the case of a missing tunnel
    - if **transport-protocol** is *TLS* or *DTLS*, the assignment of a TLS profile to the port — issues an error message in the case of a no assignment or the assignment of a non-existent profile

    verify-config confirms that TSCF and SIP interfaces have not been configured on the same network interface. In the event of such a conflict, **verify-config** issues an ERROR message identifying the TSCF interface, the SIP interface, and the network interface by name. Additionally, verify-config places the TSCF interface in an inactive state, where it remains until the configuration has been repaired.
15. Repeat Steps 1 through 13 to configure additional *tscf-interfaces* as required.

TSCF DoS Protection Configuration

Use the following procedure to configure DoS protection as described in Denial of Service. DoS protection is assigned via the realm that supports the TSCF port.

1. From superuser mode, use the following command sequence to access *realm* configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# media-manager
   ACMEPACKET(media-manager)# realm-config
   ACMEPACKET(realm-config)#
   ```
2. Use the **select** command to identify the target realm, the realm supporting the TSCF port.

   ```
   ACMEPACKET(realm-config)# select
   ...
   ...
   3. tsmAccess
   ...
   ...
   ```

```
ACMEPACKET(realm-config)# 3
ACMEPACKET(realm-config)#
```

3. Use the **access-control-trust-level** parameter to enable DoS protection.

   Set the parameter to **high** to enable DoS protection.

   ```
   ACMEPACKET(realm-config)# access-control-trust-level high
   ACMEPACKET(realm-config)#
   ```

4. Use **done**, **exit**, and **verify-config** to complete DoS configuration for the current realm.
5. Repeat Steps 1 through 4 to enable DoS on other TSCF ports.

TSCF Management/Monitoring

A series of **show** commands provides data on:

- TSCF global statistics
- address pool usage
- tunnel creation and state

TSCF Global Statistics

The **show tscf statistics brief** command displays summary TSCF global data.

```
ACMEPACKET# show tscf statistics brief
22:08:16-133
Tunnel Capacity=200000
                                -- Period -- -------- Lifetime --------
                       Active    High   Total       Total  PerMax     High
Established Tunnels         0       0       0           0       0        0

22:08:16-133
TSCF Statistics          ---- Lifetime ----
                     Recent      Total  PerMax
Tunnel Adds               0          0       0
Tunnel Rejects            0          0       0
Tunnel Deletes            0          0       0
Tunnel Errors             0          0       0
ACMEPACKET#
```

The **show tscf statistics** command displays verbose TSCF global data.

```
ACMEPACKET# show tscf statistics
22:08:02-119
Tunnel Capacity=200000
                                -- Period -- -------- Lifetime --------
                       Active    High   Total       Total  PerMax     High
Established Tunnels         0       0       0           0       0        0

22:08:02-119
TSCF Statistics          ---- Lifetime ----
                     Recent      Total  PerMax
Tunnel Adds               0          0       0
Tunnel Rejects            0          0       0
Tunnel Deletes            0          0       0
Tunnel Errors             0          0       0

Active Tunnels                              : 1500
Established Tunnels                         : 1500
Finished Tunnels                            : 0
Released Tunnels                            : 0
Max Active Tunnels                          : 1500
Failed Tunnels - Malformed Request          : 0
Failed Tunnels - Non Existing Tunnel Id     : 0
Failed Tunnels - Out of Resources           : 0
Failed Tunnels - Server Failure             : 0
Failed Tunnels - Version Not Supported      : 0
ACMEPACKET#
```

Address Pool Statistics

The **show tscf address-pool all** command displays address pool data for all configured address pools.

```
ACMEPACKET# show tscf address-pool all

================================================================================
#   |Addr Pool Name                 |Total    |In Use   |Free     |Invalid  |
================================================================================
1   |pool1                          |65279    |1500     |63779    |0        |
================================================================================
ACMEPACKET#
```

The **show tscf address-pool <addressPoolName>** command displays data for a single specified address pool.

```
ACMEPACKET# show tscf address-pool pool1

================================================================================
#   |Addr Pool Name                 |Total    |In Use   |Free     |Invalid  |
================================================================================
1   |pool1                          |65279    |1500     |63779    |0        |
================================================================================
ACMEPACKET#
```

Tunnel Statistics

The **show tscf tunnel all** command displays basic summary information for each tunnel.

- tunnel ID
- tunnel inner IP address
- tunnel outer IP address
- tunnel state

```
ACMEPACKET# show tscf tunnel all

=====================================================================
#      |Tunnel Id         |Inner IP          |Outer IP          |St |
=====================================================================
1      |FEEDBEEF00000001 |128.1.1.1         |192.168.95.1      |A  |
2      |FEEDBEEF00000002 |128.1.1.2         |192.168.95.1      |A  |
3      |FEEDBEEF00000003 |128.1.1.3         |192.168.95.1      |A  |
4      |FEEDBEEF00000004 |128.1.1.4         |192.168.95.1      |A  |
5      |FEEDBEEF00000005 |128.1.1.5         |192.168.95.1      |A  |
6      |FEEDBEEF00000006 |128.1.1.6         |192.168.95.1      |A  |
7      |FEEDBEEF00000007 |128.1.1.7         |192.168.95.1      |A  |
8      |FEEDBEEF00000008 |128.1.1.8         |192.168.95.1      |A  |
9      |FEEDBEEF00000009 |128.1.1.9         |192.168.95.1      |A  |
10     |FEEDBEEF0000000A |128.1.1.10        |192.168.95.1      |A  |
11     |FEEDBEEF0000000B |128.1.1.11        |192.168.95.1      |A  |
...
...
...
ACMEPACKET#
```

The **show tscf tunnel all detailed** command displays verbose information for each tunnel.

```
ACMEPACKET# show tscf tunnel all detailed

Prot = Protocol     Red = Redundancy [Off/Redundancy factor]
Creation time date format : mm/dd/yy
T1 = Time (seconds) since last data packet received from TSC
T2 = Time (seconds)since last data packet sent on TSC
T3 = Time (seconds) since last control message received from TSC
T4 = Time (seconds) since last control message sent on TSC
St = State [S=Start  No=NoSession  I=Init  Ne=Negotiating A=Active P=Persistent R=Released]
RC = Reconnect count
=============================================================================================================================================
#    |Tunnel Id         |Inner IP       |Outer IP       :Port  |Prot|St |Red|Creation Time      |  T1  |  T2  |  T3  |  T4  |RC |
=============================================================================================================================================
1    |FEED8EEF00000001 |128.1.1.1      |192.168.95.1   : 59728|DTLS|A  |Off|08-07-12 20:24:23|    0 |  983 | 1016 | 1016 |0  |
2    |FEED8EEF00000002 |128.1.1.2      |192.168.95.1   : 37458|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
3    |FEED8EEF00000003 |128.1.1.3      |192.168.95.1   : 60606|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
4    |FEED8EEF00000004 |128.1.1.4      |192.168.95.1   : 50513|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
5    |FEED8EEF00000005 |128.1.1.5      |192.168.95.1   : 49393|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
6    |FEED8EEF00000006 |128.1.1.6      |192.168.95.1   : 48710|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
7    |FEED8EEF00000007 |128.1.1.7      |192.168.95.1   : 54811|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
8    |FEED8EEF00000008 |128.1.1.8      |192.168.95.1   : 46762|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
9    |FEED8EEF00000009 |128.1.1.9      |192.168.95.1   : 58803|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
10   |FEED8EEF0000000A |128.1.1.10     |192.168.95.1   : 51221|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
11   |FEED8EEF0000000B |128.1.1.11     |192.168.95.1   : 54133|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
12   |FEED8EEF0000000C |128.1.1.12     |192.168.95.1   : 59848|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
13   |FEED8EEF0000000D |128.1.1.13     |192.168.95.1   : 43607|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
14   |FEED8EEF0000000E |128.1.1.14     |192.168.95.1   : 60597|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
15   |FEED8EEF0000000F |128.1.1.15     |192.168.95.1   : 43952|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
16   |FEED8EEF00000010 |128.1.1.16     |192.168.95.1   : 35743|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
17   |FEED8EEF00000011 |128.1.1.17     |192.168.95.1   : 55732|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
18   |FEED8EEF00000012 |128.1.1.18     |192.168.95.1   : 45225|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
19   |FEED8EEF00000013 |128.1.1.19     |192.168.95.1   : 56454|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
20   |FEED8EEF00000014 |128.1.1.20     |192.168.95.1   : 49385|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
21   |FEED8EEF00000015 |128.1.1.21     |192.168.95.1   : 60977|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
22   |FEED8EEF00000016 |128.1.1.22     |192.168.95.1   : 54848|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
23   |FEED8EEF00000017 |128.1.1.23     |192.168.95.1   : 56677|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
24   |FEED8EEF00000018 |128.1.1.24     |192.168.95.1   : 46506|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
25   |FEED8EEF00000019 |128.1.1.25     |192.168.95.1   : 44595|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
26   |FEED8EEF0000001A |128.1.1.26     |192.168.95.1   : 48193|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
27   |FEED8EEF0000001B |128.1.1.27     |192.168.95.1   : 37941|DTLS|A  |Off|08-07-12 20:24:23|    0 |    0 | 1016 | 1016 |0  |
28   |FEED8EEF0000001C |128.1.1.28     |192.168.95.1   : 58508|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
29   |FEED8EEF0000001D |128.1.1.29     |192.168.95.1   : 48392|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
30   |FEED8EEF0000001E |128.1.1.30     |192.168.95.1   : 37856|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
31   |FEED8EEF0000001F |128.1.1.31     |192.168.95.1   : 52008|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
32   |FEED8EEF00000020 |128.1.1.32     |192.168.95.1   : 56968|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
33   |FEED8EEF00000021 |128.1.1.33     |192.168.95.1   : 40850|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
34   |FEED8EEF00000022 |128.1.1.34     |192.168.95.1   : 49292|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
35   |FEED8EEF00000023 |128.1.1.35     |192.168.95.1   : 59491|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
36   |FEED8EEF00000024 |128.1.1.36     |192.168.95.1   : 34017|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
37   |FEED8EEF00000025 |128.1.1.37     |192.168.95.1   : 35687|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
38   |FEED8EEF00000026 |128.1.1.38     |192.168.95.1   : 34994|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
39   |FEED8EEF00000027 |128.1.1.39     |192.168.95.1   : 54479|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
40   |FEED8EEF00000028 |128.1.1.40     |192.168.95.1   : 38831|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
41   |FEED8EEF00000029 |128.1.1.41     |192.168.95.1   : 44057|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
42   |FEED8EEF0000002A |128.1.1.42     |192.168.95.1   : 48298|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
43   |FEED8EEF0000002B |128.1.1.43     |192.168.95.1   : 56459|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
44   |FEED8EEF0000002C |128.1.1.44     |192.168.95.1   : 59873|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
45   |FEED8EEF0000002D |128.1.1.45     |192.168.95.1   : 50620|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
46   |FEED8EEF0000002E |128.1.1.46     |192.168.95.1   : 50866|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
47   |FEED8EEF0000002F |128.1.1.47     |192.168.95.1   : 35776|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
48   |FEED8EEF00000030 |128.1.1.48     |192.168.95.1   : 40177|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
49   |FEED8EEF00000031 |128.1.1.49     |192.168.95.1   : 33085|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
50   |FEED8EEF00000032 |128.1.1.50     |192.168.95.1   : 55147|DTLS|A  |Off|08-07-12 20:24:24|    0 |    0 | 1015 | 1015 |0  |
...
...
...
ACMEPACKET#
```

The **show tscf tunnel peer-inner-ip <ipAddress>** command displays summary information for a specific tunnel identified by its IP address as assigned from an address pool.

- tunnel ID
- tunnel inner IP address
- tunnel outer IP address
- tunnel state

```
ACMEPACKET# show tscf tunnel peer-inner-ip 128.1.1.1

=====================================================================
#      |Tunnel Id         |Inner IP         |Outer IP         |St |
=====================================================================
1      |FEEDBEEF00000001 |128.1.1.1        |192.168.95.1     |A  |
=====================================================================
ACMEPACKET#
```

The **show tscf tunnel peer-outer-ip <ipAddress>** command displays summary information for a set of tunnels emanating from the specified client interface address.

- tunnel ID
- tunnel inner IP address
- tunnel outer IP address
- tunnel state

```
ACMEPACKET# show tscf tunnel peer-outer-ip 195 2.168.95.1

=====================================================================
#      |Tunnel Id         |Inner IP         |Outer IP         |St |
=====================================================================
1      |FEEDBEEF00000001 |128.1.1.1        |192.168.95.1     |A  |
2      |FEEDBEEF00000002 |128.1.1.2        |192.168.95.1     |A  |
3      |FEEDBEEF00000003 |128.1.1.3        |192.168.95.1     |A  |
4      |FEEDBEEF00000004 |128.1.1.4        |192.168.95.1     |A  |
5      |FEEDBEEF00000005 |128.1.1.5        |192.168.95.1     |A  |
6      |FEEDBEEF00000006 |128.1.1.6        |192.168.95.1     |A  |
7      |FEEDBEEF00000007 |128.1.1.7        |192.168.95.1     |A  |
8      |FEEDBEEF00000008 |128.1.1.8        |192.168.95.1     |A  |
9      |FEEDBEEF00000009 |128.1.1.9        |192.168.95.1     |A  |
10     |FEEDBEEF0000000A |128.1.1.10       |192.168.95.1     |A  |
11     |FEEDBEEF0000000B |128.1.1.11       |192.168.95.1     |A  |
12     |FEEDBEEF0000000C |128.1.1.12       |192.168.95.1     |A  |
13     |FEEDBEEF0000000D |128.1.1.13       |192.168.95.1     |A  |
14     |FEEDBEEF0000000E |128.1.1.14       |192.168.95.1     |A  |
15     |FEEDBEEF0000000F |128.1.1.15       |192.168.95.1     |A  |
...
...
...
ACMEPACKET#
```

The **show tscf tunnel state <state>** command displays basic tunnel data for all tunnels in the specified state.

- tunnel ID
- tunnel inner IP address
- tunnel outer IP address
- tunnel state

<state> can take one of the following values: *Established, Finished, Negotiating, NoSession, Released, Start, Unassigned*.

```
ACMEPACKET# show tscf tunnel state Established

=====================================================================
#      |Tunnel Id         |Inner IP         |Outer IP          |St |
=====================================================================
...
...
...
53     |FEEDBEEF00000035 |128.1.1.53       |192.168.95.1      |A  |
54     |FEEDBEEF00000036 |128.1.1.54       |192.168.95.1      |A  |
55     |FEEDBEEF00000037 |128.1.1.55       |192.168.95.1      |A  |
56     |FEEDBEEF00000038 |128.1.1.56       |192.168.95.1      |A  |
57     |FEEDBEEF00000039 |128.1.1.57       |192.168.95.1      |A  |
58     |FEEDBEEF0000003A |128.1.1.58       |192.168.95.1      |A  |
59     |FEEDBEEF0000003B |128.1.1.59       |192.168.95.1      |A  |
60     |FEEDBEEF0000003C |128.1.1.60       |192.168.95.1      |A  |
61     |FEEDBEEF0000003D |128.1.1.61       |192.168.95.1      |A  |
62     |FEEDBEEF0000003E |128.1.1.62       |192.168.95.1      |A  |
63     |FEEDBEEF0000003F |128.1.1.63       |192.168.95.1      |A  |
64     |FEEDBEEF00000040 |128.1.1.64       |192.168.95.1      |A  |
65     |FEEDBEEF00000041 |128.1.1.65       |192.168.95.1      |A  |
66     |FEEDBEEF00000042 |128.1.1.66       |192.168.95.1      |A  |
67     |FEEDBEEF00000043 |128.1.1.67       |192.168.95.1      |A  |
68     |FEEDBEEF00000044 |128.1.1.68       |192.168.95.1      |A  |
69     |FEEDBEEF00000045 |128.1.1.69       |192.168.95.1      |A  |
...
...
...
ACMEPACKET#
```

The **show tscf tunnel tscf-interface <realmID>** command displays basic tunnel data for all tunnels supported by a TSCF port. The TSCF port is identified by the name of its supporting realm.

- tunnel ID
- tunnel inner IP address
- tunnel outer IP address
- tunnel state

```
ACMEPACKET# show tscf tunnel tscf-interface access

=================================================================
#      |Tunnel Id         |Inner IP          |Outer IP          |St |
=================================================================
1      |FEEDBEEF00000001 |128.1.1.1         |192.168.95.1      |A  |
2      |FEEDBEEF00000002 |128.1.1.2         |192.168.95.1      |A  |
3      |FEEDBEEF00000003 |128.1.1.3         |192.168.95.1      |A  |
4      |FEEDBEEF00000004 |128.1.1.4         |192.168.95.1      |A  |
5      |FEEDBEEF00000005 |128.1.1.5         |192.168.95.1      |A  |
6      |FEEDBEEF00000006 |128.1.1.6         |192.168.95.1      |A  |
7      |FEEDBEEF00000007 |128.1.1.7         |192.168.95.1      |A  |
8      |FEEDBEEF00000008 |128.1.1.8         |192.168.95.1      |A  |
9      |FEEDBEEF00000009 |128.1.1.9         |192.168.95.1      |A  |
10     |FEEDBEEF0000000A |128.1.1.10        |192.168.95.1      |A  |
11     |FEEDBEEF0000000B |128.1.1.11        |192.168.95.1      |A  |
12     |FEEDBEEF0000000C |128.1.1.12        |192.168.95.1      |A  |
13     |FEEDBEEF0000000D |128.1.1.13        |192.168.95.1      |A  |
14     |FEEDBEEF0000000E |128.1.1.14        |192.168.95.1      |A  |
15     |FEEDBEEF0000000F |128.1.1.15        |192.168.95.1      |A  |
...
...
...
ACMEPACKET#
```

The **show tscf tunnel tunnel-id <tunnelID>** command displays basic tunnel data for a specific tunnel identified by its tunnel ID.

- tunnel ID
- tunnel inner IP address
- tunnel outer IP address
- tunnel state

A tunnel ID is assigned by the TSCF server during the configuration exchange, and can be most easily obtained with the **show tscf tunnel all** command.

.

```
ACMEPACKET# show tscf tunnel tunnel-id FEEDBEEF00000000

=================================================================
No TSCF tunnels to display
weld# 054   show tscf tunnel tunnel-id FEEDBEEF00000000   033

=================================================================
#      |Tunnel Id         |Inner IP          |Outer IP          |St |
=================================================================
1      |FEEDBEEF00000033 |128.1.1.51        |192.168.95.1      |A  |
=================================================================
```

Tunnel Deletion

ACLI CLI commands provide the ability to delete an individual tunnel, a group of tunnels, or all tunnels.

Use the **clear-tunnel tscf all** command to delete all tunnels.

```
ACMEPACKET# clear-tunnel tscf all

WARNING: This will clear all tunnels and may cause service outage!!

Clear all TSCF tunnels ? [y/n]?: y

Command was successful. Deleted 1502 TSCF tunnels.

ACMEPACKET#
```

Use the **clear-tunnel tscf interface ...** command to delete a group of tunnels.

This command deletes all tunnels on a specified interface

```
ACMEPACKET# clear-tunnel tscf M1:1

WARNING: This will clear all tunnels and may cause service outage!!

Clear all TSCF tunnels ? [y/n]?: y

Command was successful. Deleted 753 TSCF tunnels.

ACMEPACKET#
```

This command deletes all TLS tunnels on a specified TSCF port

```
ACMEPACKET# clear-tunnel tscf M1:1 tls 192.168.11.22:5060

WARNING: This will clear all tunnels on this port and may cause
service outage!!

Clear all TSCF tunnels ? [y/n]?: y

Command was successful. Deleted 753 TSCF tunnels.

ACMEPACKET#
```

This command deletes all DTLS tunnels on a specified TSCF port

```
ACMEPACKET# clear-tunnel tscf M1:1 dtls 192.168.11.22:5060

WARNING: This will clear all tunnels on this port and may cause
service outage!!

Clear all TSCF tunnels ? [y/n]?: y

Command was successful. Deleted 226 TSCF tunnels.

ACMEPACKET#
```

This command deletes all TCP tunnels on a specified TSCF port

```
ACMEPACKET# clear-tunnel tscf M1:1 tcp 192.168.11.22:5060

WARNING: This will clear all tunnels on this port and may cause
service outage!!

Clear all TSCF tunnels ? [y/n]?: y

Command was successful. Deleted 10 TSCF tunnels.

ACMEPACKET#
```

This command deletes all UDP tunnels on a specified TSCF port

```
ACMEPACKET# clear-tunnel tscf M1:1 udp 192.168.11.22:5060

WARNING: This will clear all tunnels on this port and may cause
service outage!!

Clear all TSCF tunnels ? [y/n]?: y

Command was successful. Deleted 10 TSCF tunnels.

ACMEPACKET#
```

Use the **clear-tunnel tscf tunnel-id ...** command to delete a single tunnel.

```
ACMEPACKET# clear-tunnel tscf tunnel-id feed3eef000000cf

WARNING: This will clear this tunnel and may cause service outage!!

Clear this TSCF tunnels ? [y/n]?: y

Command was successful. Deleted 1 TSCF tunnels.

ACMEPACKET#
```

Message Session Relay Protocol

The Acme Packet Session Director supports Message Relay Protocol (MSRP) sessions initiated by Session Description Protocol (SDP) messages exchanged via the Session Initiation Protocol (SIP) offer/answer model. MSRP usage with SDP and SIP is described in Section 8 of RFC 4975, *The Message Relay Protocol*. The Session Director functions as a Back-to-Back User Agent (B2BUA) for MSRP sessions, terminating incoming MSRP, proxying for the MSRP session originator, initiating outgoing MSRP to the endpoint peer, and providing Network Address Translation (NAT) services.

Platform Requirements

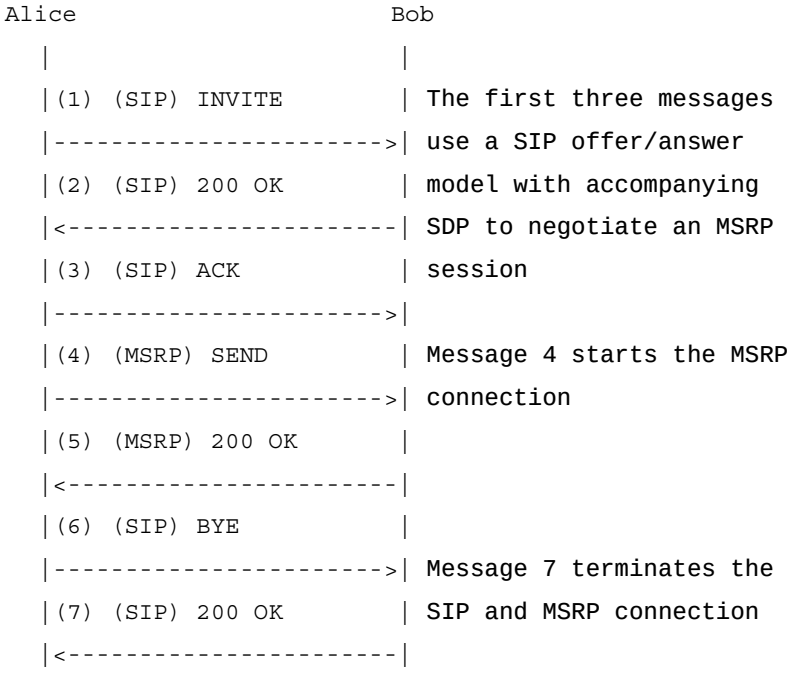
MSRP is supported on NN3800 and NN4500 platforms equipped with Enhanced Traffic Control (ETC) Network Interface Units (NIU).

License Requirements

MSRP functionality is enabled with the standard SIP license. Note that a TLS license is required to encrypt MSRP connections.

MSRP Operational Description

A sample RFC 4975-compliant Offer/Answer SDP exchange for an MSRP session is shown below.



1. Alice sends an INVITE request with accompanying SDP to Bob.

   The SDP media (M) line is defined in RFC 4975, and adheres to the format

   m=<media> <port> <protocol> <format-list>

   MSRP operations require the following values:

   media = *message*

   protocol = *TCP/MSRP* (for an unencrypted connection)

   format-list =*

   port = the TCP port (7777 in the SDP example, although any valid port number can be specified) monitored by the message originator for a response to the SDP offer

   The required SDP attributes, *accept-types* and *path*, are also defined in RFC 4975.

   accept-types contains a list of media types that the message originator is willing to receive. It may contain zero or more registered media-types, or an * wildcard character in a space-delimited string.

   path contains the MSRP URI of the message originator. An MSRP URI is constructed as shown below.

   scheme = *msrp* (for an unencrypted connection), or

   = *msrps* (for an encrypted connection)

   //

   address = IP address of the message originator, or

   = FQDN of the message originator

   :

   port = the port (7777 in the SDP example, although any valid port number can be specified) monitored by the message originator for MSRP responses

   session-id = a random local value generated by the message originator used to produce an ephemeral MSRP URI lasting only for the duration of the current MSRP session

   ;

   protocol = *tcp*

```
Alice->Bob (SIP):

INVITE sip:bob@example.com
SDP:
v=0
o=alice 2890844557 2890844559 IN IP4 alicepc.example.com
s=
c=IN IP4 alicepc.example.com
m=message 7777 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://alicepc.example.com:7777/iau39soe2843z;tcp
```

2. Bob accepts the SDP offer, generates a local session-id (contained in his MSRP URI specified by the *path* attribute), and issues a 200 OK response to Alice.

   The port parameter in the Media line indicates that Bob listens for MSRP messages on TCP port 8888

```
Bob->Alice (SIP):

SIP/2.0 200 OK
SDP:
v=0
o=bob 2890844612 2890844616 IN IP4 bob.example.com
s=
c=IN IP4 bob.example.com
m=message 8888 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://bob.example.com:8888/9di4eae923wzd;tcp
```

3. Alice ACKs Bob's answer, establishing a SIP session between the two.

```
Alice->Bob (SIP):

ACK sip:bob@example.com
```

4. Alice initiates an MSRP session with an MSRP SEND request to Bob.

   All MSRP requests begin with the MSRP start line, which contains three elements.

protocol-id	=	*MSRP*
transaction-id	=	an ephemeral transaction identifier (*d93kswow* in the following MSRP example) used to correlate MSRP requests and responses, and to frame the contents of the MSRP message
method	=	*SEND* (MSRP method that supports data transfer)

   The MSRP start line is followed by the To-Path and From-Path headers, which contain destination and source addresses — the MSRP URIs exchanged during the MSRP negotiation.

   The Message-ID header contains a random string generated by the message originator. This ephemeral value whose lifetime is measured from message start to message end, is used to correlate MSRP status reports with a specific message, and to re-assemble MSRP message fragments (chunks in MSRP terminology).

   The Byte-Range header contains the message length, in bytes, and the specific byte range carried by this message. Contents of this header are generally of interest only if the message has been fragmented.

   The Content-Type header describes the message type, and must conform to the results of the MSRP negotiation.

   The actual message follows the Content-Type header.

   Finally, the SEND request is closed with an end-line of seven hyphens, the transaction-id, and a

   $ to indicate that this request contains the end of a complete message, or

   + to indicate that this request does not contain the message end

```
Alice->Bob (MSRP):

MSRP d93kswow SEND
To-Path: msrp://bob.example.com:8888/9di4eae923wzd;tcp
From-Path: msrp://alicepc.example.com:7777/iau39soe2843z;tcp
Message-ID: 12339sdqwer
Byte-Range: 1-16/16
Content-Type: text/plain
Hi, I'm Alice!
-------d93kswow$
```

5. Bob acknowledges receipt with an MSRP 200 OK response to Alice.

   Note that the response includes the initiator-originated transaction-id, *d93kswow*.

```
Bob->Alice (MSRP):

MSRP d93kswow 200 OK
To-Path: msrp://alicepc.example.com:7777/iau39soe2843z;tcp
From-Path: msrp://bob.example.com:8888/9di4eae923wzd;tcp
-------d93kswow$
```

6. Alice sends a BYE request to Bob.

   Alice sends a BYE request to terminate the SIP session and MSRP sessions. Alice can send more SEND requests to Bob before sending the BYE.

```
Alice->Bob (SIP):

BYE sip:bob@example.com
```

7. Bob sends a 200 OK response to Alice.

```
Bob->Alice (SIP):

SIP/2.0 200 OK
```

   The SIP session and the MSRP session are terminated.

Secure MSRP Session Negotiation

An Offer/Answer SDP exchange for a TLS (secure) MSRP session is specified in RFC 4572, *Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)*. RFC 4572 defines the syntax and symantics for an SDP *fingerprint* attribute that identifies the certificate (most likely a self-signed certificate) that will be presented during the TLS negotiation. A sample SDP exchange is shown below.

The protocol field (TCP/TLS/MSRP) of the media (M) line designates a TLS encrypted connection.

The *fingerprint* attribute is constructed as follows:

protocol	identifies the hashing method used to produce the certificate fingerprint, SHA-1 in the following sample SDP
hash value	a sequence of uppercase hexadecimal bytes separated by colons with the sequence length determined by the hash function

Offer SDP: (Alice to Bob)

```
v=0
o=usera 2890844526 2890844527 IN IP4 alice.example.com
s=
c=IN IP4 1.1.1.1
m=message 7394 TCP/TLS/MSRP *
a=accept-types:text/plain
a=path:msrps://alice.example.com:7394/2s93i9ek2a;tcp
a=fingerprint:SHA-1
4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
```

Answer SDP: (Bob to Alice)

```
v=0
o=userb 2890844530 2890844532 IN IP4 bob.example.com
s= -
c=IN IP4 2.2.2.2
t=0 0
m=message 8493 TCP/TLS/MSRP *
a=accept-types:text/plain
a=path:msrps://bob.example.com:8493/si438dsaodes;tcp
a=fingerprint:SHA-1
DA:39:A3:EE:5E:6B:4B:0D:32:55:BF:EF:95:60:18:90:AF:D8:07:09
```

MSRP Session Setup

The B2BUA processes MSRP media descriptions in offer/answer SDPs to negotiate and establish MSRP sessions and then constructs internal data flows for the actual MSRP sessions. After establishing an MSRP session, the B2BUA forwards MSRP requests and responses from and to the session participants.

Initiating MSRP Sessions

After accepting an offer SDP with MSRP session initiation, the B2BUA constructs an egress offer SDP as follows.

1. The B2BUA sets the transport protocol of the *m=* line to the transport protocol of the selected egress profile.
2. If the value of *listen-port* of the selected egress profile is not zero, the B2BUA sets the port of the *m=* line to the value of listen-port. If the value of listen-port is zero, the port in the *m=* line is chosen from a steering port of the egress realm.
3. The B2BUA adds an *a=setup* attribute to the SDP. The value of this attribute is determined by the value of the **preferred-setup-role** ACLI command. For example, if the value of **preferred-setup-role** is passive (the default value) the B2BUA adds the attribute *a=setup:passive*.
4. The B2BUA performs NAT on the MSRP Universal Resource Identifier (URI) in the *a=path* attribute.

The B2BUA does not include an *a=fingerprint* in the offer SDP if the selected egress profile has TCP/TLS/MSRP transport protocol. However, if the egress profile specifies both TCP/MSRP and TCP/TLS/MSRP the B2BUA selects the TCP/TLS/MSRP transport protocol, resulting in an egress offer containing an *m=* line with TCP/TLS/MSRP transport protocol. The B2BUA offers only a single media line, TCP/MSRP or TCP/TLS/MSRP. The B2BUA does not perform recursion (that is, first initiation attempt with TCP/TLS/MSRP and re-attempt with TCP/MSRP if first attempt is rejected).

Connection Negotiation

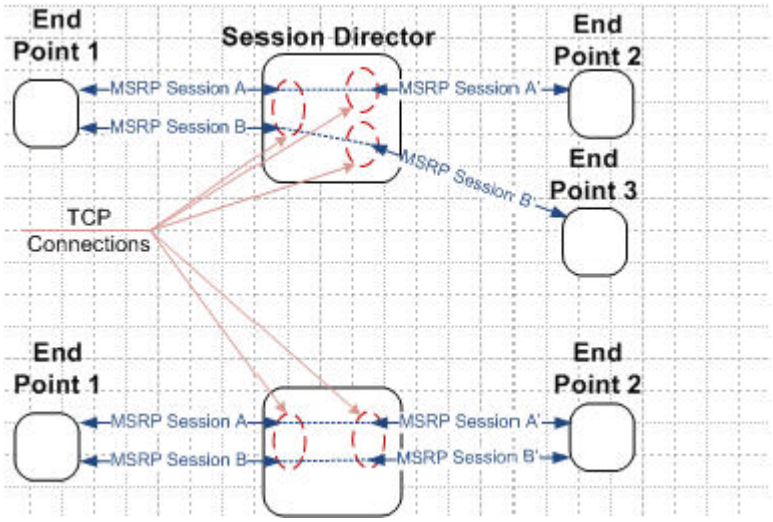
When making the offer MSRP session, a user agent client can follow RFC 4145, *TCP-Based Media Transport in the Session Description Protocol (SDP*), and include an *a=setup* attribute in a MSRP media line, or it may assume the default connection direction as specified in section 8.5 of RFC 4975, *The Message Session Relay Protocol*, which specifies that the endpoint that sent the original offer is responsible for connecting to its remote peer. The B2BUA, in contrast, always includes an *a=setup* attribute in its offer SDP, the attribute value set by the **preferred-setup-role** ACLI command.

If the B2BUA receives an answer SDP that does not include an *a=setup* attribute, it assumes that the user agent server does not support connection negotiation per RFC 4145. In that case, the B2BUA assumes the active role as specified in RFC 4975, and makes an outgoing connection.

If the B2BUA receives an offer or answer SDP that does include an *a=setup* attribute, it follows the connection negotiation as specified by RFC 4145. When the B2BUA receives an offer SDP with the attribute *a=setup:actpass* attribute, it will set the *a=setup* attribute of the answer SDP to the value set by the **preferred-setup-role** ACLI command.

Multiple MSRP Connections

The MSRP B2BUA supports sharing of a single TCP connection by multiple MSRP sessions. In such a topology, each TCP connection maintains a list active MSRP sessions.



With regard to the above figure, the upper Session Director's TCP connection between Endpoint 1 and the MSRS B2BUA is shared by 2 MSRP sessions. At bottom, each MSRP session uses a separate TCP connection. When the B2BUA assumes the *active* role, it always initiate a separate connection to the MSRP endpoint peers, endpoints 2 and 3 as shown above.

When the list of active MSRP sessions for a shared TCP connection becomes empty as a result of SIP session terminations or disconnections of peer TCP connections, B2BUA disconnects the shared TCP connection. If the shared connection is disconnected by the peer, the B2BUA disconnects all the separate TCP connections it initiated for the MSRP sessions that use the shared connection.

Accepting Connections

When the B2BUA is in passive mode, it listens for incoming connection from the active party, monitoring the port specified by the **listen-port** ACLI command.

Making Connections

When the B2BUA is in active mode, it makes the connection to the passive party, selecting a port allocated from the steering-pool of the applicable realm.

MSRP Session Termination

An MSRP session is terminated by sending or receiving a BYE request in the *parent* SIP session, that is the session that set up the MSRP exchange, followed by the disconnect of the TCP connection that supports MSRP message exchange.

Some SIP endpoints close their MSRP TCP connections upon receiving a BYE possibly before its peer finishes sending all the MSRP messages and closes the connection. To facilitate graceful session completion, the B2BUA offers a configurable time delay between the receipt of a BYE request from an MSRP endpoint and the transmission of the BYE to the recipient endpoint peer.

With the configurable BYE delay enabled, the MSRP B2BUA upon receiving a BYE request acknowledges the request with a 200 OK response. The B2BUA, however, does not immediately forward the BYE to the other MSRP endpoint.

Rather, the B2BUA triggers two user-configurable timers that monitor the specific MSRP session initiated by the current SIP exchange. The first timer measures inactivity intervals on media flows (*calling-to-called* and *called-to-calling*) associated with the MSRP session to be closed. The second timer sets an unconditional outer limit at which point the delayed BYE is transmitted to the MSRP endpoint and the MSRP is terminated.

Expiration of either timer generates an internal stop event which generates transmission of the delayed BYE to the MSRP endpoint and tem ina ti on of the underlying SIP connection.

Note that the MSRP-specific timers, the session inactivity timer and the MSRP delayed-BYE timer, are roughly analogous to two existing TCP timers, the TCP subsequent guard timer and the TCP flow time limit timer. The following chart summarizes timer operations.

MSRP interval timer

Purpose:

Measures inactivity periods within MSRP data sessions. The timer is triggered in the absence of MSRP data. If new MSRP is not detected prior to timer expiration, a stop event is generated resulting in delayed BYE transmission and MSRP connection termination. If new MSRP traffic is detected prior to timer expiration, the timer is reset.

Associated ACLI Command:

session-inactivity-timer

Allowable command values: 0 | 5 to 10 (seconds)

Default value: 5

When set to 0, session monitoring is disabled. No MSRP session monitoring is done when the corresponding SIP session receives a BYE request.

Timer Expiration:

Initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the SIP connection.

TCP subsequent guard timer

Purpose:

Measures the maximum interval allowed between media-over-TCP packets.

Associated ACLI Command:

tcp-subsq-guard-timer

Allowable command values: 0 to 999999999 (seconds)

Default value: 300

Timer Expiration:

Initiates tear down of the TCP connection. In the possible, but unlikely event that the value assigned to this timer is less than the value assigned to the MSRP interval timer, expiration initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the TCP connection.

MSRP delayed -BYE timer

Purpose:

Measures inactivity periods within MSRP traffic sessions. The timer is triggered in the absence of MSRP data. If new MSRP is not detected prior to timer expiration, a stop event is generated resulting in delayed BYE transmission and MSRP connection termination. If new MSRP traffic is detected prior to timer expiration, the timer is reset.

Associated ACLI Command:

msrp-delayed-bye-timer

Allowable command values: 0 to 60 (seconds)

Default value: 15

Timer Expiration:

Initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the SIP connection.

TCP flow time limit timer

Purpose:

Measures the maximum allowed lifetime of a media-over-TCP connection.

Associated ACLI Command:

tcp-flow-limit-timer

Allowable command values: 0 to 999999999 (seconds)

Default value: 86400 (1 day)

Timer Expiration:

Initiates tear down of the TCP connection. In the possible, but unlikely event that the value assigned to this timer is less than the value assigned to the MSRP delayed-BYE timer, expiration initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the TCP connection.

Network Address Translation

If the value of the **uri-translation** ACLI command is enabled, the B2BUA performs network address translation on the MSRP URI in the *a=path* attribute and in the *From-Path* and *To-Path* of MSRP requests and response. The host part of the URI will be the IP address of the steering pool of the realm. The port will be chosen as follows:

- If the B2BUA role is in passive mode, and the **listen-port** ACLI command is non-zero, the B2BUA monitors that specified port.
- If the B2BUA role is in passive mode, and the **listen-port** ACLI command is zero, the B2BUA selects a port from the steering pool of the applicable realm and monitors that chosen port.
- If the B2BUA role is in active mode, the port is chosen from the steering pool of the applicable realm. Since B2BUA is active, that port is used only to support the outgoing connection.

Certificate Fingerprint

If the value of the field *require-fingerprint* in the ingress and/or egress *tcp-media-profile* is enabled, and the transport protocol is TCP/TLS/MSRP the B2BUA requires the offer and/or the answer SDPs, respectively, to have an *a=fingerprint* attribute as specified in RFC 4572, *Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)*.

If the B2BUA receives an offer SDP without *a=fingerprint* attribute, it rejects the offer SDP. If the rejected SDP is in an INVITE request, the B2BUA issues a *488 Not Acceptable Here* response. If the rejected SDP is in a 200 OK response, the B2BUA ACKs that 200 OK, sends a BYE to the server user agent, and a *503 Service Unavailable* response to the client user agent.

If the B2BUA receives an answer SDP without a *a=fingerprint* attribute, it terminates the related SIP session. If the rejected SDP is in a 200 OK response, the B2BUA ACKs that 200 OK, sends a BYE to the server user agent, and a *503 Service Unavailable* response to the client user agent. If the rejected SDP is in an ACK, the B2BUA simply sends a BYE to both the server and client user agent.

If the value of the field *require-fingerprint* in the profile is disabled, the B2BUA accepts offer and answer SDP that do not include an *a=fingerprint* attribute.

Because the B2BUA certificate is expected to be signed by a trusted Certification Authority, an *a=fingerprint* attribute is not included in offer and answer SDPs sent by the B2BUA.

The B2BUA maintains a fingerprint mismatch counter for each MSRP session. This counter is incremented when the B2BUA cannot match the certificate it receives during the TLS handshake with the fingerprint it receives in the SDP.

MSRP Configuration

MSRP configuration consists of the following steps.

1. Configure the *msrp-config* configuration object that governs MSRP global behavior.
2. Configure one or more *tcp-media-profile* configuration objects that define MSRP operations within a realm.
3. Assign a *tcp-media-profile* to a target realm.
4. If MSRP sessions are secured with TLS, create and assign *tls-profile* configuration objects to the *tcp-media-profile* of the target realm.
5. Create and assign *steering-pools* configuration objects to target realms. Refer to the *Net-Net 4000 ACLI Configuration Guide* for information on Steering Pool Configuration.

msrp-config Configuration

Use the following procedure to perform MSRP global configuration.

1. From superuser mode, use the following command sequence to access *msrp-config* configuration mode. While in *msrp-config* mode, you configure global MSRP behavior.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# msrp-config
ACMEPACKET(msrp-config)# ?

state                     state
uri-translation           perform translation of MSRP URI
session-inactivity-timer  timer value (seconds) for session inactivity
                          monitoring period
select                    select msrp config to edit
no                        delete msrp config
show                      show msrp config
done                      write msrp config information
exit                      return to previous menu

ACMEPACKET(msrp-config)#
```

2. Use the **state** parameter to enable MSRP operations.

   Retain the default value, **enabled**, to enable MSRP operations.

   If necessary, you can use **disabled** to temporarily suspend all MSRP operations.

```
ACMEPACKET(msrp-config)# state enabled
ACMEPACKET(msrp-config)#
```

3. Use the **uri-translation** parameter to enable or disable NAT of URIs found in the *From-Path* and *To-Path* headers of MSRP requests and responses, and in *a=path* attributes found in SDP offers.

   Refer to Network Address Translation for procedural details.

   NAT is enabled by default.

   Retain the default value (**enabled**) to enable NAT; use **disabled** to disable NAT.

   ```
   ACMEPACKET(msrp-config)# uri-translation enabled
   ACMEPACKET(msrp-config)#
   ```

4. Use the **session-inactivity-timer** parameter in connection with the **msrp-delayed-bye-timer** parameter to implement the delayed transmission of SIP BYE requests, thus establishing a configurable transition interval allowing for the completion of active MSRP sessions.

   The **session-inactivity-timer** parameter specifies the maximum inactivity interval (defined as the absence of transmitted data) tolerated before the MSRP connection is terminated.

   Retain the default value (**5**), or specify another inactivity interval within the range 5 to 10 seconds.

   ```
   ACMEPACKET(msrp-config)# session-inactivity-timer 7
   ACMEPACKET(msrp-config)#
   ```

5. Use **done**, **exit**, and **verify-config** to complete MSRP global configuration.

6. If you wish to implement the delayed transmission of SIP BYE requests, use the following command sequence to access *sip-config* configuration model

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)# sip-config
   ACMEPACKET(sip-config)#
   ```

7. Use the **msrp-delayed-bye-timer** parameter to enable the delayed transmission of SIP BYE requests, thus establishing a configurable transition interval allowing for the completion of active MSRP sessions.

   With the delayed transmission of SIP BYE requests enabled (the default state), the original BYE request, which can contain non-standard/proprietary headers, is buffered. When the *msrp-delayed-bye-timer* expires, the MSRP task searches its buffers for the original BYE request. If the request is found, it is transmitted. If not found, the MSRP process constructs a substitute BYE request containing only standards-based headers.

   The **msrp-delayed-bye-timer** parameter specifies the maximum delay period allowed before transmitting the delayed BYE request.

   Retain the default value (1**5**), or specify another delay period within the range **1** to **60** seconds.

   Delayed transmission of BYE requests is enabled by default. Use the special value of **0** to disable delay, and transmit BYE requests immediately upon receipt.

   ```
   ACMEPACKET(sip-config)# msrp-delayed-bye-timer 20
   ACMEPACKET(sip-config)#
   ```

tcp-media-profile Configuration

Use the following procedure to construct a TCP Media Profile that defines MSRP operations within a realm.

1. From superuser mode, use the following command sequence to access *tcp-media-profile* configuration mode. While in *tcp-media-profile* mode, you begin construction of a TCP Media Profile.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# tcp-media-profile
ACMEPACKET(tcp-media-profile)# ?

name                name
profile-list        list of TCP media profiles
select              select profile to edit
no                  delete profile
show                show profile
done                write profile information
exit                return to previous menu

ACMEPACKET(tcp-media-profile)#
```

2. Use the **name** parameter to provide a unique identifier for this TCP Media Profile instance.

```
ACMEPACKET(tcp-media-profile)# name tlsMSRP
ACMEPACKET(tcp-media-profile)#
```

3. Use the **profile-list** command to move to *tcp-media-profile-entry* configuration mode. While in this mode, you complete configuration of the named *tls-media-profile*.

```
ACMEPACKET(tcp-media-profile)# profile-list
ACMEPACKET(tcp-media-profile-entry)# ?

media-type            media type
transport-protocol    transport protocol
listen-port           listening port
preferred-setup-role  preferred setup role
tls-profile           tls profile name
require-fingerprint   always require TLS certificate fingerprint
select                select a profile entry to edit
no                    delete selected profile entry
show                  show profile entry information
done                  write profile entry information
exit                  return to previous menu

ACMEPACKET(tcp-media-profile-entry)#
```

4. Use the **listen-port** parameter to identify the TCP port monitored by the B2BUA for incoming MSRP connections.

   Supported values are **0** (the default) | **1025** through **65535**.

   The **0** default value indicates that the listening port will be chosen by the B2BUA from the steering pool of the realm identified by the **name** parameter.

   ```
   ACMEPACKET(tcp-media-profile-entry)# listen-port 43000
   ACMEPACKET(tcp-media-profile-entry)#
   ```

5. Use the **media-type** parameter in conjunction with the **transport-protocol** parameter to identify the media-types and transport protocols (found in the SDP media description, *m=*, field as described in RFC 4566, *SDP: Session Description Protocol*) subject to this TCP Media Profile.

   media-type identifies the media subject to this TCP Media Profile. Use the value, **message**, for MSRP operations.

   ```
   ACMEPACKET(tcp-media-profile-entry)# media-type message
   ACMEPACKET(tcp-media-profile-entry)#
   ```

   transport-protocol identifies the transport layer protocols subject to this TCP Media Profile. Use either **TCP/MSRP** to specify unsecure TCP traffic or **TCP/TLS/MSRP** to specify secure TLS traffic.

   ```
   ACMEPACKET(tcp-media-profile-entry)# transport-protocol TCP/TLS/MSRP
   ACMEPACKET(tcp-media-profile-entry)#
   ```

6. If **transport-protocol** is **TCP/TLS/MSRP**, use the **tls-profile** parameter to identify the TLS profile that specifies the cryptographic resources available to support TLS operations.

   This parameter can be safely ignored if **transport-protocol** is **TCP/MSRP**.

   Refer to tls-profile Configuration for procedural details of TLS configuration.

   ```
   ACMEPACKET(tcp-media-profile-entry)# tls-profile msrp1
   ACMEPACKET(tcp-media-profile-entry)#
   ```

7. If **transport-protocol** is **TCP/TLS/MSRP**, use the **require-fingerprint** parameter to enable or disable endpoint authentication using the certificate fingerprint methodology defined in RFC 4572, *Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP).*

   By default, mutual authentication is **disabled**.

   This parameter can be safely ignored if **transport-protocol** is **TCP/MSRP**.

   ```
   ACMEPACKET(tcp-media-profile-entry)# require-fingerprint enabled
   ACMEPACKET(tcp-media-profile-entry)#
   ```

8. Use the **preferred-setup-role** parameter to specify the value the B2BUA uses for the *a=setup* attribute when negotiating the setup up role, regardless of the role (offerer or answerer) assumed by the B2BUA in the SDP offer/answer exchange.

   The value of **preferred-setup-role** is used for the value of the *a=setup* attribute when the B2BUA makes an offer SDP and when the B2BUA replies to an offer SDP that has *a=setup:actpass*. It is not used when the B2BUA is forced into a role by the offerer, that is, if the offerer sends *a=setup:active*, the B2BUA must answer with *a=setup:passive* (and vice versa).

Allowable values are **passive** (the default) and **active**.

active indicates that the B2BUA creates an outgoing connection.

passive indicates that the B2BUA accepts an incoming connection.

Acme Packet strongly recommends that users retain the default value, *passive*.

```
ACMEPACKET(tcp-media-profile-entry)# preferred-setup-role passive
ACMEPACKET(tcp-media-profile-entry)#
```

9. Use **done**, **exit**, and **verify-config** to complete *tcp-media-profile* configuration.
10. Repeat Steps 1 through 9 to configure additional *tcp-media-profiles* as required.

realm Configuration

Use the following procedure to assign a single, specific *tcp-media-profile* to a target realm.

1. From superuser mode, use the following command sequence to access *realm-config* configuration mode. While in *realm-config* mode, you assign a *tcp-media-profile* to a realm.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# media-manager
   ACMEPACKET(media-manager)# realm-config
   ACMEPACKET(realm-config)#
   ```

2. Use the **select** command to identify the target realm.
3. Use the **tcp-media-profile** parameter to assign a specific, named *tcp-media-profile* to the target realm.

   ```
   ACMEPACKET(realm-config)# tcp-media-profile tlsMutualAuth
   ACMEPACKET(realm-config)#
   ```

4. Use **done**, **exit**, and **verify-config** to complete *tcp-media-profile* assignment.

tls-profile Configuration

Use the following procedure to create a *tls-profile* configuration object, which specifies cryptographic resources available in support of TLS operations.

1. From superuser mode, use the following command sequence to access *tls-profile* configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# tls-profile
   ACMEPACKET(tls-profile)# ?
   ```

2. Use the **name** parameter to provide a unique identifier for this TLS Profile instance.

   ```
   ACMEPACKET(tls-profile)# name tlsMutualAuth
   ACMEPACKET(tls-profile)#
   ```

3. If the *require-fingerprint* attribute of the *tcp-media-profile* is set to **enabled**, use the **mutual-authenticate** parameter to enable mutual authentication.

   ```
   ACMEPACKET(tls-profile)# mutual-authenticate enabled
   ACMEPACKET(tls-profile)#
   ```

4. Retain default values for other parameters.
5. Use **done**, **exit**, and **verify-config** to complete *tls-profile* configuration.
6. Repeat Steps 1 through 5 to configure additional *tls-profiles* as required.

The **show mbcd** statistics command now displays MSRP flow counts for the current statistics window.

```
ACMEPACKET# show mbcd statistics
16:38:49-145
MBCD Status                -- Period -- -------- Lifetime --------
                  Active     High   Total      Total   PerMax     High
Client Sessions        1        1       0          1        1        1
Client Trans           0        0       0          3        3        1
Contexts               2        2       0          3        2        3
Flows                  5        5       0          7        4        7
Flow-Port              0        0       0          0        0        0
Flow-NAT               5        5       0         11        6        7
Flow-RTCP              0        0       0          0        0        0
Flow-Hairpin           0        0       0          0        0        0
Flow-Released          0        0       0          0        0        0
MSM-Release            0        0       0          0        0        0
Rel-Port               0        0       0          0        0        0
Rel-Hairpin            0        0       0          0        0        0
NAT Entries            5        5       0         11        6        7
Free Ports            12       12       0         14       12       12
Used Ports             0        0       0          2        2        2
Port Sorts             -        -       0          0        0
Queued Notify          0        0       0          0        0        0
MBC Trans              0        0       0          3        3        3
MBC Ignored            -        -       0          0        0
ARP Trans              0        0       0          0        0        0
Relatch NAT            0        0       0          0        0        0
Relatch RTCP           0        0       0          0        0        0
Flow-MSRP              2        2       0          2        2        2
SRTP Only Flows        0        0       0          0        0        0
SRTCP Only Flow        0        0       0          0        0        0
SRTP Collapsed         0        0       0          0        0        0
SRTP Sessions          0        0       0          0        0        0

Flow Rate = 0.0
Load Rate = 0.0

ACMEPACKET#
```

The **show msrp stats** command displays the following cumulative counts:

```
ACMEPACKET# show msrp statistics
PPM_ID_MSRP:
msrpStatsActiveSessions                     : 0
msrpStatsMaxActiveSessions                  : 2
msrpStatsEstablishedSessions                : 2
msrpStatsProvisionedSessions                : 2
msrpStatsFinishedSessions                   : 2
msrpStatsReleasedSessions                   : 2
msrpStatsFailedSessionsCannotRoute          : 0
msrpStatsFailedSessionsCannotConnect        : 0
msrpStatsFailedSessionsFingerprintMismatch  : 0
msrpStatsFailedMessagesCannotBeSent         : 0
msrpStatsFailedMessagesMalformed            : 0
msrpStatsSendQFullEvents                    : 0
msrpStatsSendQCongestedEvents               : 0
msrpStatsSendQCongestionRelievedEvents      : 0
msrpStatsStreamRequestsReceived             : 4
msrpStatsStreamRequestSent                  : 4
msrpStatsStreamResponsesReceived            : 4
msrpStatsStreamResponseSent                 : 4
msrpStatsStreamErrorNoTransId               : 0
msrpStatsStreamErrorNoMsgType               : 0
msrpStatsStreamErrorNoByteLength            : 0

ACMEPACKET#
```