Oracle® Communications Security Gateway

Essentials Guide

Version M-CX2.0.0

Formerly Net-Net Security Gateway

October 2013

ORACLE

Contents

About This Guide

Overview

Version M-CX2.0.0 provides support for a Security Gateway as described in *3rd Generation partnership Project, Technical Specification Group Services and System Aspects3GPP system to Wireless Local Area Network (WLAN) internetworking, System Description,* commonly referred to as 3GPP TS23.234. This guide describes that Security Gateway implementation and provides background material on Security Gateway components.

Audience

This guide is written for network administrators and architects, and provides information specific to the Security Gateway implementation. Supporting, related material is available in the latest iteration of the *Oracle Communications Session Border Controller ACLI Configuration Guide*. Please refer to that document as needed.

Supported Platforms

Release Version S-CX6.3.0 is supported on the Net-Net 4500 platform.

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.
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.

Document Revision History

This section contains a revision history for this document.

Date	Revision Number	Description
May 29, 2009		Initial release
August 13, 2010	Rev. 1	MC-X1.1.0 release provides the following new elements/features/capabilities. • Adds task-based ACLI procedural descriptions for all supported functionality. • Adds configuration details for IPsec Security Policies that open designated ports for IKEv2 protocol traffic and enforce IPsec encryption on other ports. • Adds support for the EAP-SIM Protocol as defined in RFC 4186, E*xtensible Authentication Profile Method for Global System for Mobile Communications (GSM) Subscriber identity Modules (EAP-SIM)*. • Adds support for the creation of IMSI (International Mobile Subscriber Identity) white lists. Such white lists consist of approved IMSI MCC (Mobile Country Code) and MNC (Mobile Network Code) prefixes and provide a pre-screening mechanism before received IMSIs are passed to a RADIUS server for final authentication. • Adds enhanced IKEv2 interface management functions to include new performance counters, new error counters, and new ACLI show commands to display aggregated data • Adds support for the creation of threshold crossing alerts that monitor failed IKEv2 authentications, IPsec tunnel removals, and dead peer detections. The alerts trigger SNMP traps as user-defined threshold levels are attained.

Date	Revision Number	Description
March 7, 2011	Rev. 2	MC-X1.2.0F1 release adds the following new capabilities. • Support for a required new Security Gateway license which is required to enable IKEv2 operations on media ports, data-flows, and local-address-pools. The new license also enforces revised licensing tiers. • Support for Diffie-Helman groups 5 and 14, which are available during IKEv2 Security Association negotiation. With this new support DH Groups 1 (768-bit), 2 (1048-bit), 5 (1536-bit), and 14 (2048-bit) are available. • Support for the Advanced Encryption Standard-Extended Cipher Block Chaining Algorithm (defined in RFC 3566), which is available in support of IKEv2 SA authentication operations. • Support for the Advanced Encryption Counter Mode Algorithm (defined in RFC 5930), which is available in support of IKEv2 SA encryption operations. • Support for the creation of a second child SA after the completion of the initial IKE_SA_INIT and IKE_AUTH exchanges. The CREATE_CHILD_SA exchange must be requested by the IKEv2 initiator; the Security Gateway will never initiate creation of the child SA. • Support for multiple authentication as specified in RFC 4739. Multiple authentication enables the Security Gateway to engage in an initial shared-secret-based or certificate-based authentication with a remote IKEv2 peer, a femtocell for example. followed by a subsequent EAP-based authentication exchange, which authenticates a remote mobile subscriber. • Support for obtaining RADIUS authorization information from a RADIUS-compatible femtocell gateway. With support enabled, the Security Gateway, after authenticating a remote femtocell, engages in a RADIUS Access-Request/Access-Accept transaction with a pre-configured femtocell gateway. • Support for Certificate Revocation Lists (as specified in RFC 5280) as a source of certificate status information. With this release, certificate validity can be verified by either a CRL or an OCSP responder. • Support for an enhanced suite of management tools to detect and protect against distributed denial of service attacks • Support for both RADIUS and DIAMETER reporting of IPsec tunnel metrics.

Date	Revision Number	Description
		• Support for RADIUS *early start* reporting of IPsec tunnel metrics. • Support for Disconnect Request (RADIUS code 40) Messages when received from a trusted RADIUS server. Disconnect Requests are specified in RFC 5176.
August 18, 2011	Rev. 3	MC-X1.2.0F2 release adds the following new capabilities. • Support for the assignment of multiple traffic selectors to a specified IKEv2 interface. • Support for the transport of DIAMETER accounting data via IKEv2 media interfaces. • Support for DIAMETER Device-Watchdog-Request (DWR) and Device-Watchdog-Answer (DWA) messages.
December 2, 2011		MC-X1.2.0F3 release adds the following new capabilities. • Support for pre-populating the ARP table with a list of frequently accessed core-side hosts • Support for the processing of received INITIAL_CONTACT notifications which enables persistent IP addresses for IPsec tunnels • Support for a new processing algorithm when RADIUS-based Authentication and Authorization are used in tandem • Support for configurable sources used in response to Configuration payload requests for DNS server information • Support for the retransmission of unacknowledged requests to re-key an IKEv2 or IPsec Security Association • Support for an SNMP trap and associated alarm issued upon failure to retrieve a CRL – a second trap is thrown (and the alarm state cleared) when the CRL is successfully retrieved • Support for enhanced IKEv2 interface monitoring/management • Support for enhanced processing of CFG_REQUEST messages that contain an INTERNAL_IP4_SUBNET attribute
February 9, 2012		MC-X1.2.0F3 release modification that updates descriptions of IKEv2 error and performance counters.

Date	Revision Number	Description
June 15, 2012	Rev. 1	MC-X2.0.0 release adds the following new capabilities. • Support for a new state parameter that allows an IKEv2 interface to be placed in either an enabled or inactive state • Support for additional IKEv2 interface statistics (available via SNMP, ACLI show command, and HDR) • Support for new RADIUS server statistics (available via SNMP, ACLI show command, and HDR) • Support for new DIAMETER server statistics (available via SNMP, ACLI show command, and HDR)
October 12. 2012	Rev. 2	• Documents an options-based method for suppressing log entries and SNMP traps generated as a result of an authentication failure during IPsec tunnel establishment • Reformats presentation of various counters and related statistics
June 6, 2013	Rev. 3	• Adds a new Appendix, MIB/SNMP Quick Reference that a provides a baseline description of MSG-specific MIB constructs and SNMP traps. Subsequent versions of this document wil note MIB and SNMP changes from this baseline.

Oracle Communications Security Gateway

This chapter describes the Security Gateway services provided by Version M-CX2.0.0. The offered functionality allows the Security Gateway to support both data and multimedia applications at the service provider access border, and is of particular interest to both mobile operator and Mobile Virtual Network Operator (MVNO) networks.

Product Overview

The initial Security Gateway implementation operates as shown in Figure 1.

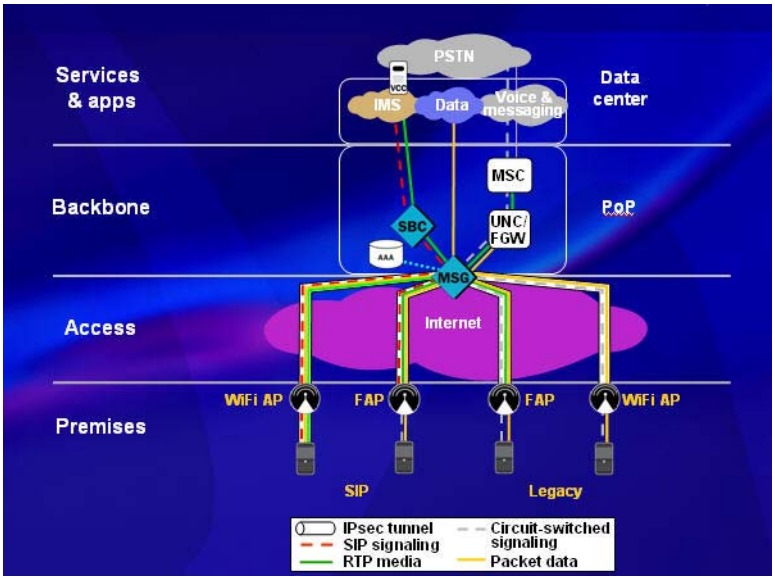


Figure 1: Net-Net Security Gateway for Femtocells and Dual-Mode Handsets

The Security Gateway provides for termination of IPsec tunnels between the Security Gateway and user endpoints. The following interfaces are supported:

Wu (described in section 6.3.12 of 3GPP TS 23.234) which terminates IPsec tunnels from user endpoints or mobile systems network devices.

Gi/*Wi* interface to the IMS (IP Multimedia Subsystem) core.

Wm (described in section 6.3.10 of 3GPP TS 23.234) interface to a RADIUS server

Additionally, the Security Gateway provides the following standards-compliant functionality.

- IPsec IKEv2 tunnel termination with support of EAP-MSCHAPV2, EAP-MD5, EAP-SIM, and EAP-AKA
- UDP encapsulation for NAT transversal
- AES/3DES support for both IKEv2 and IPsec Security Associations
- Rapid IPsec tunnel resumption when operating in High Availability (HA) Mode

- RADIUS support on the Wm interface
- Support for IP address assignment via either locally configured address pools, or via a RADIUS server, as described in RFC 3580, *IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines*.
- Tunnel Management to include IKEv2 SA re-keying
- Protection against tunnel-target DoS/DDos attacks (IKE-SA-INIT flooding)
- IKEv2 cookie support

Security Gateway Protocol Support

The following sections describe major Security Gateway components

Internet Key Exchange (IKEv2) Protocol

Security Gateway functionality is provided by the IKE Protocol as defined by RFC 4306, *Internet Key Exchange (IKEv2) Protocol*. Standalone tunnel operations closely resemble the usage scenario described in section 1.1.3, *Endpoint to Security Gateway Tunnel*, of RFC 4306.

IKE performs mutual authentication between two parties, for instance a remote subscriber and an Security Gateway, and establishes an IKE Security Association (SA) that includes shared secret information that can be used to establish SAs for Encapsulating Security Payload (ESP), defined in RFC 4303, and/or Authentication Header (AH), defined in RFC 4302, and a set of cryptographic algorithms to be used by the SAs to protect the traffic that they carry.

An IKE *initiator* (usually the remote subscriber) proposes one or more cryptographic choices by listing supported cryptographic suites. An IKE *responder* (usually the Security Gateway) selects one of the proposed suites, and signals acceptance of the suite in a return response.

All IKE communications consist of pairs of messages: a request and a response. The request/response pair is referred to as an IKE *exchange*.

The IKEv2 implementation conforms to these RFCs,

- RFC 3580, *IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines*
- RFC 3706, *A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers*
- RFC 3748, *Extensible Authentication Protocol*
- RFC 3948, *UDP Encapsulation of IPsec ESP Packets*
- RFC 4306, *Internet Key Exchange (IKEv2) Protocol*

and provides the following specific functions.

- IKEv2 pre-shared key support, which enables the passing of shared secrets — including the usage of a single shared secret that can be used by UEs to authenticate the Security Gateway as the VPN gateway
- *Wu* interface for UEs to the Security Gateway using IPsec (refer to Figure 1)
- 3DES, AES-CBC (128 or 256 bit), or AES-CTR (128 or 256 bit)-based encryption on the *Wu* interface; also provides Null cipher for debug purpose (refer to Figure 1)
- AES-CBC and AES-CTR-based encryption on IKEv2 SA's

- HMAC-MD5 or HMAC-SHA1-based data integrity checks on the *Wu* interface (refer to Figure 1)
- UDP and non-UDP encapsulation with tunnel mode for the *Wu* interface (refer to Figure 1)
- Support for *Wm* authentication interface — to include EAP-MSCHAPV2, EAP-MD5, EAP-SIM, and EAP-AKA
- Support Diffie Hellman groups 1 (768-bit), 2 (1024-bit), 5 (1536-bit), and 14 (2048-bit)
- Configurable dead peer detection with a resulting trap
- Obtain IP address for the user endpoint via *Wm* interface using RADIUS VSAs (refer to Figure 1)
- Mitigate against DDoS attacks through the use of cookies — referred to as the IKE SPI for IKEv2
- Support IPsec tunnel establishment with PFS (perfect forward security)
- Support IKEv2 High Availability (stateful SA Failover for established IKE and IPsec SA's)

UDP Encapsulation of IKE and ESP (NAT Support)

The Security Gateway supports the UDP encapsulation of IKE and ESP (Encapsulation Security Protocol) packets. Such encapsulation is required for packets originated by UEs behind a NAT (Network Address Translation) device.

In order to support UDP encapsulation the IKE process monitors port 4500 (the *well known* NAT port) in addition to the port monitored for incoming IKE traffic (port 500 by default).

IKE checks notify payloads of type NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP contained within IKE_SA_INIT packets. Examination of these payloads can detect the presence of a NAT device between the user endpoint and the Security Gateway.

If none of the received NAT_DETECTION_SOURCE_IP payloads matches the hash of the source IP address/port pair found in the IP header of the packet containing the payload, the packet source is behind a NAT device, which has changed the source address of the original packet to match the address of the NAT device. The IKE initiator on detecting a NAT device in the path, tunnels all future IKE and ESP packets via UDP port 4500

EAP/RADIUS Support

The Security Gateway supports new RADIUS attributes defined in RFC 3579, *RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)*.

EAP-Message Attribute

This attribute (Type 79) encapsulates EAP packets so as to allow the Security Gateway to authenticate peers without having to understand the EAP method it is passing through to the RADIUS server.

Message-Authenticator Attribute

This attribute (Type 80), when present in an Access-Request packet, contains an encrypted MD5 hash of the entire Access Request Packet including Type, ID, Length, and Authenticator. The computed MD5 hash value is encrypted using the shared secret.

Dead Peer Detection

The Security Gateway supports dead peer detection as described in RFC 3706, *A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers*.

As a responder, the Security Gateway replies to INFORMATIONAL requests that contain no payloads (other than the empty encrypted payload required by the syntax). Such requests are commonly used as a check for liveliness. Similarly, the Security Gateway responds to the receipt of a Notify (R U THERE) message, with a Notify (R U THERE ACK) message

As an initiator, the Security Gateway transmits INFORMATIONAL requests described above on the basis of a global timer

Local Address Pool

In many cases a remote endpoint requires an internal (or local) IP address to access network resources protected by the Security Gateway. With IKEv2, an endpoint request for such an internal address from the Security Gateway is communicated with a Configuration Payload request.

The Configuration Payload may contain a request for a specific address, or for any available address. A specific request indicates that a previous connection had been established with this address, and the requester is seeking to re-use that address. In response to a specific address request, the Security Gateway checks the availability of that address, and responds affirmatively if the address is available. If the address in not available, another available address is provided to the endpoint.

The Security Gateway fulfills the request by obtaining the address from a pre-configured local address pool that is assigned to a specific IKEv2 interface.

After address assignment from the local address pool, the endpoint retains rights to that IP address for the tunnel lifetime. Tunnels are terminated either by an INFORMATIONAL exchange, defined in Section 1.4 of RFC4306, or by expiration of the tunnel SAs as specified by the **v2-ike-life-seconds** and **v2-ipsec-life-seconds** configuration parameters. In either case, a subsequent request for an assigned IP address may, or may not result, in the assignment of the previous IP address. However, the Security Gateway can be configured to ensure that a prematurely terminated tunnel can be restored with that previous address. Refer to Persistent Tunnel Addressing in IKEv2 Global Configuration for configuration details.

Immediately after an address is released, network routers and endpoints may contain routing tables or ARP caches containing the now released address. To allow for proper routing data convergence, the Security Gateway does not re-issue that address for a period of 30 seconds.

Consequently, addresses contained in the local pool are marked as being in one of three available states.

- allocated

  meaning that the address has been assigned to a subscriber.

- available

  meaning that the address is available for assignment

- inactive

  meaning that the address has been returned to the pool and time-stamped with the DTG of its return; an inactive address is not available for assignment until the expiration of the 30-second grace period.

IKEv2 Global Configuration

The Security Gateway provides support for Version 2 of the Internet Key Exchange Protocol as defined in RFC 4306, *Internet Key Exchange (IKEv2) Protocol*, and for the Dead Peer Detection (DPD) protocol as defined in RFC 3706, *A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers*.

IKEv2 provides an initial handshake in which IKE peers negotiate cryptographic algorithms, mutually authenticate, and establish session keys, ultimately creating an IKEv2 Security Association (SA) and an IPsec SA.

All IKEv2 messages are request/response pairs. It is the responsibility of the side sending the request to retransmit if it does not receive a timely response.

The initial exchange usually consists of two request/response pairs between an IKEv2 initiator and an IKEv2 responder, a role performed by the Security Gateway.

This first request/response pair negotiates cryptographic algorithms and does a Diffie-Hellman exchange. The second request/response pair is encrypted and integrity protected with keys generated from the cryptographic material provided by the Diffie-Hellman exchange. In this second exchange the initiator and responder divulge their identities, and *prove* the purported identity using a mutual integrity check based on the secret associated with their identity (private key or shared secret key).

After the initial handshake, the IKEv2 initiator can request the establishment of a additional child SA. The exchange to establish a child SA consists of an optional Diffie-Hellman exchange (if perfect forward secrecy for that child-SA is desired), nonces (so that a unique key for that child-SA will be established), and negotiation of traffic selector values which indicate what addresses, ports, and protocol types are to be transmitted over that child-SA.

Refer to IKE_SA_INIT and IKE_AUTH in Configuring IKEv2 Interfaces for details of the initial handshake exchange.

Subsequent informational messages to include null messages for detecting peer aliveness, can be originated by either IKEv2 peer.

IKEv2 global configuration consists of the following steps.

1. Configure IKEv2 global parameters.
2. Configure a default certificate profile.
3. Configure one or more RADIUS servers (optional).
4. Configure the default address pool (optional).
5. Configure pre-shared-keys if authentication is based on the contents of the IKEv2 Identification payload (optional).

IKEv2 Global Configuration

Use the following procedure to perform IKEv2 global configuration.

Note that the following commands are relevant only to IKEv1 and are ignored for IKEv2 operations: **negotiation-timeout**, **event-timeout**, **phase1-mode**, **phase1-dh-mode**, **phase1-life-seconds**, **phase1-life-secs-max**, **phase2-life-seconds**, **phase2-life-secs-max**, **phase2-exchange-mode**.

1. From superuser mode, use the following command sequence to access *ike-config* configuration mode. While in *ike-config* mode, you configure global IKEv2 configuration parameters.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-config
   ACMEPACKET(ike-config)#
   ```

2. Use the **state** attribute to enable IKE operations.

   ```
   ACMEPACKET(ike-config)# state enable
   ACMEPACKET(ike-config)#
   ```

3. Use the **ike-version** attribute to select the IKE protocol version.

   Security Gateway operations require IKEv2.

   ```
   ACMEPACKET(ike-config)# ike-version 2
   ACMEPACKET(ike-config)#
   ```

4. Use the **log-level** attribute to determine the contents of the IKEv2 log.

   Log messages are listed below in descending order of severity.

emergency	— the most severe
critical	
major	
minor	
warning	
notice	
info	— (default) the least severe
trace	— (test/debug, not used in production environments)
debug	— (test/debug, not used in production environments)
detail	— (test/debug, not used in production environments)

   In the absence of an explicitly configured value, **log-level** defaults to *info*, specifying that log messages with a severity of info or greater are written to the IKEv2 log.

   ```
   ACMEPACKET(ike-config)# log-level warning
   ACMEPACKET(ike-config)#
   ```

5. Use the **udp-port** attribute to specify the UDP port monitored for IKEv2 protocol traffic.

   Allowable values are within the range 1024 through 65535 (ports).

   In the absence of an explicitly configured value, **udp-port** defaults to *500*, the port designated by the IANA for ISAKMP (Internet Security Association and Key Management Protocol) exchanges.

   ```
   ACMEPACKET(ike-config)# udp-port 5000
   ACMEPACKET(ike-config)#
   ```

6. Use the **v2-ike-life-seconds** attribute to specify the default lifetime (in seconds) of the IKEv2 SA.

   Allowable values are within the range 1800 through 999999999 (seconds).

   In the absence of an explicitly configured value, **v2-ike-life-seconds** defaults to *86400* seconds (1 day).

   To optimize performance, assigned attribute values should be 30-minute intervals — for example, 1800 seconds for 30 minutes, 3600 seconds for 1 hour, 7200 seconds for 2 hours, 14400 seconds for 40hourss, 73800 seconds for 20.5 hours, and so on.

   ```
   ACMEPACKET(ike-config)# v2-ike-life-seconds 3600
   ACMEPACKET(ike-config)#
   ```

7. Use the **v2-ipsec-life-seconds** attribute to specify the default lifetime (in seconds) for the IPsec SA.

   Allowable values are within the range 1 through 999999999 (seconds).

   In the absence of an explicitly configured value, **v2-ipsec-life-seconds** defaults to *28800* seconds (8 hours).

   ```
   ACMEPACKET(ike-config)# v2-ipsec-life-seconds 7200
   ACMEPACKET(ike-config)#
   ```

8. Use the **v2-rekey** attribute to enable or disable the re-keying of expired IKEv2 or IPsec SAs.

   Allowable values are *disabled* (the default) and *enabled*.

   This global default can be over-ridden at the IKE interface level.

   With this attribute enabled, the Security Gateway initiates a new negotiation to restore an expired IKEv2 or IPsec SA. Expiration of the **v2-ike-life-secs** timer triggers an IKE SA negotiation, while expiration of the **v2-ipsec-life-secs** triggers an IPsec SA negotiation.

   With automatic re-keying enabled, and with the global **dpd-time-interval** parameter set to a non-zero value, the Security Gateway retransmits the re-keying request if it does not receive a response from the remote IPsec peer within the interval specified by the **dpd-time-interval** parameter. The Security Gateway makes a maximum of three retransmission attempts before abandoning the re-keying effort.

   ```
   ACMEPACKET(ike-config)# v2-rekey enabled
   ACMEPACKET(ike-config)#
   ```

9. Use the **sd-authentication-method** attribute to select the method used for local authentication of the IKEv2 peer.

   Local authentication is performed by the Security Gateway. Two standard authentication methods are supported.

   shared-password — (the default) uses a pre-shared key (PSK) to authenticate the IKEv2 peer.

   certificate — uses a Public Key Infrastructure (PKI) X.509 certificate to authenticate the IKEv2 peer.

   This global default method can be over-ridden at the IKEv2 interface level.

   Other authentication methods that make use of a RADIUS authentication server are supported. Refer to RADIUS-Based Authentication for descriptions of these methods and configuration details.

   ```
   ACMEPACKET(ike-config)# sd-authentication-method certificate
   ACMEPACKET(ike-config)#
   ```

10. If **sd-authentication-method** is *shared-password*, use the **shared-password** attribute to provide the PSK used while authenticating the remote IKEv2 peer. You must ensure that the remote peer is configured with the same PSK.

    The PSK is a string of ACSII printable text with a maximum length of 255 characters.

    This global default PSK can be over-ridden by an interface-specific PSK.

    ```
    ACMEPACKET(ike-config)# shared-password!yetAnotherPaSSword1of87354
    ACMEPACKET(ike-config)#
    ```

11. If **sd-authentication-method** is *shared-password*, use the **id-auth-type** attribute to specify that the PSK used while authenticating the remote IKEv2 peer is associated with the asserted identity contained within an IKEv2 Identification payload. This attribute can be safely ignored if the PSK is defined globally, or at the IKEv2 Interface level.

    With PSK-based authentication, the actual secret can be provided (1) at the global IKEv2 configuration level, (2) at the IKEv2 interface level, or (3) by an *ike-keyid* configuration element, which associates a PSK with an asserted identity contained within an IKEv2 Identification payload. The Identification payload is passed by the IKEv2 initiator in the first message of the IKE_AUTH exchange.

    Allowable values are *idi* (the default), indicating that the asserted identity is contained within the required Initiator Identification payload, or *idr*, indicating that the asserted identity is contained within an optional Responder Identification payload. Refer to IKE_AUTH in Configuring IKEv2 Interfaces for Identification payload exchange details.

    The PSK selection algorithm works as follows:

    If **id-auth-type** is set to *idi* (the default value), the Security Gateway checks the value of the *ID_TYPE* field of the required Initiator Identification payload. If the value of that field is *ID_KEY_ID* (essentially indicating a request for vendor-specific proprietary processing), the Security Gateway examines existing *ike-keyid* configuration elements for the asserted identity extracted from the *Identification Data* field of the required Initiator Identification payload. If a match is found, the PSK associated with the asserted identity is used. If a match is not found, or if no *ike-keyid* elements are configured, the Security Gateway selects the IKEv2-interface-specific PSK (if one has been configured). In the absence of an interface key, the Security Gateway selects the global default PSK.

    It is possible, although unlikely, that the IKEv2 initiator passes a responder identity within the optional Responder Identification payload. If this is the expected behavior, set the **id-auth-type** attribute to *idr*. The Security Gateway checks the value of the *ID_TYPE* field of the optional Responder Identification payload. If the value of that field is *ID_KEY_ID*, the Security Gateway examines existing *ike-keyid* configuration elements for the asserted identity extracted from the *Identification Data* field of the optional Responder payload. If a match is found, the PSK associated with the asserted identity is used. If a match is not found, or if no *ike-keyid* elements are configured, the Security Gateway selects the IKEv2-interface-specific PSK (if one has been configured). In the absence of an interface key, the Security Gateway selects the global default PSK.

    Refer to key-id Configuration in this chapter for information on configuring asserted identity and PSK associations.

    ```
    ACMEPACKET(ike-config)# id-auth-type idi
    ACMEPACKET(ike-config)#
    ```

12. If **sd-authentication-method** is *certificate*, use the **certificate-profile-id** attribute to specify the default *ike-certificate-profile* configuration element that contains identification and validation credentials required for PKI certificate-based IKEv2 authentication.

    Provide the name of an existing *ike-certificate-profile* configuration element.

    This default *ike-certificate-profile* can be over-ridden at the IKEv2 interface level.

    Refer to Certificate Profile Configuration in this chapter for information on configuring Certificate Profiles.

    ```
    ACMEPACKET(ike-config)# certificate-profile-id valCred-IKEv2
    ACMEPACKET(ike-config)#
    ```

13. Use the optional **addr-assignment** attribute to select the method used to assign a local address in response to an IKEv2 Configuration payload request.

    The Configuration payload supports the exchange of configuration information between IKEv2 peers. Typically, the remote IKEv2 peer initiates the exchange by requesting an IP address on the carriers's protected network. In response, the Security Gateway returns a local address for the peer's temporary use.

    This parameter specifies the source of the returned IP address, and can be over-ridden at the IKEv2 interface level.

    local — (the default) use local address pool

    radius-only — obtain local address from RADIUS server

    radius-local — try RADIUS server first, then local address pool

    Refer to Local Address Pool Configuration in this chapter for information on configuring Local Address Pools.

    ```
    ACMEPACKET(ike-config)# addr-assignment radius-only
    ACMEPACKET(ike-config)#
    ```

14. Retain the default value for the **eap-protocol** attribute.

    The default, and only currently-supported value, *eap-radius-passthru*, specifies the use of a RADIUS server for all Extensible Authentication Protocol (EAP) processing. The Security Gateway shuttles incoming and outgoing EAP protocol messages (EAP-AKS, EAP-MD5, EAP-MSCHAPv2, and EAP-SIM) between the remote IKEv2 peer and the RADIUS server.

    Refer to RADIUS Authentication in this chapter for information on configuring RADIUS authentication servers.

    ```
    ACMEPACKET(ike-config)# eap-protocol eap-radius-passthru
    ACMEPACKET(ike-config)#
    ```

15. Use the optional **eap-bypass-identity** attribute to specify whether or not to bypass the EAP identity phase.

    EAP, defined in RFC 3748, provides an authentication framework that supports numerous authentication methods.

    An Identity exchange is optional within the EAP protocol exchange. Therefore, it is possible to omit the Identity exchange entirely, or to use a method-specific identity exchange once a secure channel has been established.

However, where roaming is supported, it may be necessary to locate the appropriate backend authentication server before the authentication process can proceed. The realm portion of the Network Access Identifier (NAI) is typically included within the EAP-Response/Identity to enable the routing of the authentication exchange to the appropriate authentication server. Therefore, while the peer-name portion of the NAI may be omitted in the EAP identity phase where proxies or relays are present, the realm portion may be required.

EAP Identify bypass is disabled by default — thus requiring an identity exchange.

```
ACMEPACKET(ike-config)# eap-bypass-identity enabled
ACMEPACKET(ike-config)#
```

16. Use the optional **dpd-time-interval** attribute to specify the maximum period of inactivity before the DPD protocol is initiated on a specific endpoint.

    Allowable values are within the range 1 through 999999999 (seconds) with a default of *0*.

    The default value, *0*, disables the DPD protocol; setting this parameter to a non-zero value globally enables the protocol and sets the inactivity timer.

    ```
    ACMEPACKET(ike-config)# dpd-time-interval 20
    ACMEPACKET(ike-config)#
    ```

17. Use the optional **anti-replay** attribute to enable or disable anti-replay protection on IPsec SAs.

    The default value, *enable*, enables anti-replay detection services.

    IPsec authentication provides anti-replay protection against an attacker duplicating encrypted packets by assigning a unique incrementally increasing 32-bit sequence number to each encrypted packet. Packets received with previously processed sequence numbers are rejected.

    With anti-replay protection enabled, roll-over of the sequence number triggers re-negotiation of the IPsec Security Association

    RFC 2402, *IP Authentication Header*, and RFC 2406, *IP Encapsulating Security Payload*, specified the use of a 32-bit monotonically increasing sequence number which could be used to enable an anti-replay service. With anti-replay enabled, peers were required to re-key when the sequence number turned over.

    Both original RFCs have since been updated; RFC 2402 by RFC 4302, and RFC 2406 by RFC 4303. The updated versions allow a larger 64-bit sequence number, referred to as an Extended Sequence Number (ESN), that provides faster and more efficient traffic transfer by delaying the required re-keying.

    The use of ESNs is negotiated during the IPsec data SA establishment process. The IKEv2 initiator's proposal for ESN support is contained in Transform Type 5 proposals. Transform IDs take the values

    0 — extended sequence numbers not supported

    1 — extended sequence numbers not supported

    The Security Gateway supports both standard (32-bit) and extended (64-bit) sequence numbers, and generally defers to the sequence number type proposed by the IKEv2 initiator.

- Should the initiator propose the use of 32-bit sequence numbers, the Security Gateway accepts.
- Should the initiator propose the use of 64-bit sequence numbers, the Security Gateway accepts.
- Should the initiator propose the use of either 32-bit or 64-bit sequence numbers, the Security Gateway opts for 64-bit sequence numbers.

```
ACMEPACKET(ike-config)# replay-detection enable
ACMEPACKET(ike-config)#
```

18. Use the optional **account-group-list** parameter to designate one or two existing IPsec accounting groups as available to support IPsec accounting transactions.

    Default accounting groups can be over-ridden at the IKEv2 interface level.

    Use double quotes to bracket parameter arguments if two IPsec accounting groups are being made available; leave a space between the group names.

    Refer to IPsec Accounting for detailed information on IPsec accounting capabilities and configuration.

    ```
    ACMEPACKET(ike-config)# account-group-list "IPsecRADIUSaccounting
    IPsecDIAMETERaccounting"
    ACMEPACKET(ike-config)#
    ```

19. Use the **overload-threshold**, **overload-interval**, **overload-action**, **overload-critical-threshold**, and **overload-critical-interval** attributes to configure system response to an overload state.

    Use the **overload-threshold** attribute to specify the percentage of CPU usage that triggers an overload state.

    Values are within the range 1 through 100 (percent) with a default of *100*, which effectively disables overload processing.

    ```
    ACMEPACKET(ike-config)# overload-threshold 60
    ACMEPACKET(ike-config)#
    ```

    Use the **overload-action** attribute to select the response to an overload state. The overload state is reached when CPU usage exceeds the percentage threshold specified by the **overload-threshold** parameter.

    By default, no preventive action is taken in response to an overload, You can, however, use this attribute to implement a call rejection algorithm in response to the overload. With the algorithm enabled, the CPU uses the following calculation to reject/drop some number of incoming calls:

    DropRate = (currentLoad - overloadThreshold) / (100 -overloadThreshold)

    Thus, assuming a current CPU load of 70% and an overload threshold of 60%, the Security Gateway drops 1 of out every 4 incoming calls until the load falls below the threshold value.

    Use **none** to retain default behavior (no action); use **drop-new-connection** to implement call rejection.

    ```
    ACMEPACKET(ike-config)# overload-action drop-new-connection
    ACMEPACKET(ike-config)#
    ```

Use the optional **overload-critical-threshold** attribute to specify the percentage of CPU usage that triggers a critical overload state.

When this threshold is exceeded, the Security Gateway drops all incoming calls until the load drops below the critical threshold level, at which point it may drop selective calls depending on the value of the **overload-threshold** parameter.

Values are within the range 1 through 100 (percent) with a default of 100, which effectively disables overload processing.

Ensure that this threshold value is greater than the value assigned to **overload-threshold**.

```
ACMEPACKET(ike-config)# overload-critical-threshold 75
ACMEPACKET(ike-config)#
```

Use the **overload-interval** attribute to specify the interval (in seconds) between CPU load measurements when in the overload state.

Values are within the range 1 through 60 (seconds) with a default of *1*.

```
ACMEPACKET(ike-config)# overload-critical-interval 3
ACMEPACKET(ike-config)#
```

Use the **overload-critical-interval** attribute to specify the interval (in seconds) between CPU load measurements when in the critical overload state.

Values are within the range 1 through 60 (seconds) with a default of *1*.

```
ACMEPACKET(ike-config)# overload-criticalinterval 2
ACMEPACKET(ike-config)#
```

20. In high-availability topologies (a topology with two Security Gateways, one acting in the active role, and the other acting in a stand-by role), use the **red-port**, **red-max-trans**, **red-sync-start-time**, and **red-sync-comp-time** attributes to configure redundancy.

    Use the **red-port** attribute to specify the UDP port number that supports IKEv2 synchronization message exchanges.

    The default value (*0*) effectively disables redundant high-availability configurations. Specify a port value other than 0 (for example, 1995) to enable high-availability operations.

    ```
    ACMEPACKET(ike-config)# red-port 1995
    ACMEPACKET(ike-config)#
    ```

    Use the **red-max-trans** attribute to specify the maximum number of retained IKEv2 synchronization messages.

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

    ```
    ACMEPACKET(ike-config)# red-trans 35000
    ACMEPACKET(ike-config)#
    ```

    Use the **red-sync-start-time** attribute 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 Security Gateway waits for a heartbeat signal from the active Security Gateway before assuming the active role.

    Values are within the range 0 through 999999999 (milliseconds) with a default of *500*.

    ```
    ACMEPACKET(ike-config)# red-sync-start-time 2500
    ACMEPACKET(ike-config)#
    ```

Use the **red-sync-comp-time** attribute to specify the interval between synchronization attempts after the completion of an IKEv2 redundancy check.

Values are within the range 0 through 999999999 (milliseconds) with a default of *500*.

```
ACMEPACKET(ike-config)# red-sync-comp-time 750
ACMEPACKET(ike-config)#
```

21. Use **done**, **exit**, and **verify-config** to complete global IKEv2 configuration.

Certificate Profile Configuration

If authentication between IKEv2 peers is certificate based, use the following procedure to create one or more certificate profiles that provide identification and validation credentials for a specific IKEv2 identity.

1. From superuser mode, use the following command sequence to access *ike-certificate-profile* configuration mode. While in this mode, you configure certificate profiles.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-certificate-profile
   ACMEPACKET(ike-certificate-profile)#
   ```

2. Use the required **identity** parameter to specify the IKEv2 entity that uses the authentication and validation credentials provided by this *ike-certificate-profile* instance.

   Identify the subject of this *ike-certificate-profile* by either an IP address or fully-qualified domain name (FQDN).

   identity enables the creation of multiple *ike-certificate-profile* instances.

   ```
   ACMEPACKET(ike-certificate-profile)# identity jojo.net
   ACMEPACKET(ike-certificate-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 a local IKEv2 entity in support of its asserted identity.

   ```
   ACMEPACKET(ike-certificate-profile)# end-entity-certificate ACME-1a
   ACMEPACKET(ike-certificate-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 remote IKEv2 peer

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

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

5. Use the optional **verify-depth** parameter to specify the maximum number of chained certificates that will be processed while authenticating the IKEv2 peer.

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

   ```
   ACMEPACKET(ike-certificate-profile)# verify-depth 10
   ACMEPACKET(ike-certificate-profile)#
   ```

6. Use **done**, **exit**, and **verify-config** to complete configuration of the *ike-certificate-profile* instance.
7. Repeat Steps 1 through 6 to configure additional *ike-certificate-profile* instances.

Certificate Chain Validation

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

1. 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.
2. Check the maximum length of the certificate chain. If the current chain exceeds this value, authentication fails; otherwise, continue chain validation.
3. 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.
4. 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.
5. Check the *X509v3 Extensions* field to verify that the current certificate identifies a CA. If not so, authentication fails; otherwise, continue chain validation.
6. 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).
7. Compute the same MD5 hash. If the results are not identical, authentication fails; otherwise, continue chain validation.
8. 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.

The ACLI **verify-config** command confirms that the entity (end) certificate specified by the *end-entity-certificate* attribute can to chained back to a trusted certificate (specified by **trusted-ca-certificates** attribute) within the chain length constraints imposed by the **verify-depth** attribute.

RADIUS Authentication

All EAP-based authentication is performed by RADIUS servers. When such authentication is specified, the Security Gateway operates as a relay between the remote IKVv2 peer and a RADIUS authentication server.

Configuring RADIUS Authentication

RADIUS authentication support requires:

- configuration of one or more RADIUS authentication servers, with each server configuration record providing all values required for server access
- configuration of a RADIUS Authentication Servers List containing the IP address of one or more previously configured RADIUS authentication servers
- assignment of a RADIUS Authentication Servers List to the authentication configuration object designating these RADIUS authentication servers as globally available

RADIUS Authentication Servers

Use the following procedure to configure a RADIUS authentication server.

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

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# authentication
ACMEPACKET(authentication)# radius-servers
ACMEPACKET(radius-servers)#
```

2. Use the **state** attribute to set the operational state of this RADIUS authentication server.

   Retain the default value, *enabled*, to identify this RADIUS authentication server as operational. Use *disabled* to place this RADIUS authentication server in a non-operational mode.

```
ACMEPACKET(radius-servers)# state enabled
ACMEPACKET(radius-servers)#
```

3. Use the **authentication-methods** attribute to specify the authentication methods supported by this RADIUS authentication server.

   Retain the default value, *all*.

```
ACMEPACKET(radius-servers)# authentication-methods all
ACMEPACKET(radius-servers)#
```

4. Use the **address** attribute to specify the IP address of this RADIUS authentication server.

```
ACMEPACKET(radius-servers)# address 172.30.0.10
ACMEPACKET(radius-servers)#
```

5. Use the **port** attribute to specify the remote port monitored for RADIUS authentication requests.

   Retain the default value, *1812*, or specify an integer value within the ranges1025 through 65535.

```
ACMEPACKET(radius-servers)# port 6512
ACMEPACKET(radius-servers)#
```

6. Use the **realm-id** attribute to identify the realm that provides transport services to this RADIUS authentication server.

```
ACMEPACKET(radius-servers)# realm-id radiusServers
ACMEPACKET(radius-servers)#
```

7. Use the **secret** attribute to specify the shared secret between the Security Gateway and this RADIUS authentication server.

   The shared secret is used for mutual authentication between the Security Gateway and this RADIUS server.

```
ACMEPACKET(radius-servers)# secret 27softSiftInAnHourGlass
ACMEPACKET(radius-servers)#
```

8. Use the **nas-id** attribute to provide a string that uniquely identifies the Security Gateway to this RADIUS authentication server.

```
ACMEPACKET(radius-servers)# nas-id nas-id-170-30-0-1
ACMEPACKET(radius-servers)#
```

9. Use the **retry-limit** attribute to specify the number of times the Security Gateway retransmits an unacknowledged authentication request to this RADIUS authentication server.

   If the RADIUS authentication server does not respond within the retry limit, the Security Gateway marks it as quarantined.

   Retain the default value, *3*, or provide an alternate integer value within the range 1 through 5.

```
ACMEPACKET(radius-servers)# retry-limit 1
ACMEPACKET(radius-servers)#
```

10. Use the **retry-time** attribute to specify the interval (expressed in seconds) between unacknowledged authentication requests.

    Retain the default value, *5*, or provide an alternate integer value within the range 5 through 10.

```
ACMEPACKET(radius-servers)# retry-time 5
ACMEPACKET(radius-servers)#
```

11. Use the **dead-time** attribute to specify the length (expressed in seconds) of the quarantine period imposed an unresponsive RADIUS authentication server.

    Retain the default value, *10*, or provide an alternate integer value within the range 10 through 1000.

```
ACMEPACKET(radius-servers)# dead-time 20
ACMEPACKET(radius-servers)#
```

12. Use the **maximum-sessions** attribute to specify the maximum number of outstanding sessions for this RADIUS authentication server.

    Retain the default value, *255*, or provide an alternate integer value within the range 1 through 255.

```
ACMEPACKET(radius-servers)# maximum-sessions 50
ACMEPACKET(radius-servers)#
```

13. Use the **class** attribute to select the RADIUS authentication server class, either *primary* or *secondary*.

    The Security Gateway tries to initiate contact with primary RADIUS authentication servers first, and only turns to secondary RADIUS authentication servers if no primaries are available.

    If more than one RADIUS authentication server is designated as *primary*, the Security Gateway uses a round-robin strategy to distribute authentication requests among available primaries.

    ```
    ACMEPACKET(radius-servers)# class primary
    ACMEPACKET(radius-servers)#
    ```

14. Use **done**, **exit**, and **verify-config** to complete configuration of this RADIUS authentication server.
15. If necessary, complete Steps 1 through 14 to configure additional RADIUS authentication servers.

RADIUS Authentication Servers List

Use the following procedure to configure a RADIUS Authentication Servers List, a list of RADIUS servers available to provide authentication services.

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

    ```
    ACMEPACKET# configure terminal
    ACMEPACKET(configure)# security
    ACMEPACKET(security)# auth-params
    ACMEPACKET(auth-params)#
    ```

2. Use the **name** attribute to provide a unique name for this RADIUS Authentication Servers List.

    ```
    ACMEPACKET(auth-params)# name radiusauthenticationList
    ACMEPACKET(auth-params)#
    ```

3. Use the **servers** attribute to compile a RADIUS Authentication Servers List.

    Provide the IP addresses of a previously configured RADIUS authentication server to add that server to this list.

    Use double quotes (“) to bracket lists which contain multiple entries.

    ```
    ACMEPACKET(auth-params)# servers 172.30.0.1
    ACMEPACKET(auth-params)#

    ACMEPACKET(auth-params)# servers “172.30.0.1 172.30.0.15”
    ACMEPACKET(auth-params)#
    ```

4. Retain default values for all other parameters.
5. Use **done**, **exit**, and **verify-config** to complete configuration of this RADIUS Authentication Servers List.

    verify-config ensures that each IP address included in the Authentication Servers List is supported by a *radius-servers* configuration element with the identical IP address.

6. If necessary, complete Steps 1 through 5 to configure additional RADIUS Authentication Servers Lists.

Complete RADIUS authentication configuration by assigning an existing RADIUS Authentication Servers List to the *authentication* configuration element.

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

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# authentication
   ACMEPACKET(authentication)#
   ```

2. Use the **ike-radius-params-name** attribute to assign a previously configured Authentication Servers List to the *authentication* configuration element.

   ```
   ACMEPACKET(authentication)# ike-radius-params-name
   radiusAuthenticationList
   ACMEPACKET(authentication)#
   ```

3. Retain default values for all other parameters.
4. Use **done**, **exit**, and **verify-config** to complete list assignment.

RFC 5176 Support

If EAP-based authentication is used in conjunction with RADIUS-based assignment of requested local addresses, the Security Gateway responds to a *Disconnect-Request* message (as defined in RFC 5176, *Dynamic Authorization Extensions to Remote Authentication Dial-In User Service*) received from a configured RADIUS server.

The Security Gateway parses the received *Disconnect-Request* for *User-Name* and *Framed-IP-address* attribute values. If the *User-Name* value matches the authenticated EAP identity, and the *Framed-IP-address* value matches the inner IP address assigned to the authenticated endpoint, the Security Gateway deletes the IPsec tunnel described by the received values. Tunnel deletion is reported to the RADIUS server with a *Disconnect-ACK* message, which, in conformity to Section 3.5 of RFC 5176, contains an Error Cause of 201 indicating *Residual Session Context Removed*.

If the IPsec tunnel cannot be deleted because of faulty/incorrect *User-Name* and/or *Framed-IP-address* values, the Security Gateway returns a *Disconnect-NACK* message, which, in conformity to Section 3.5 of RFC 5176, contains an Error Cause of 404 indicating *Invalid Request*.

RADIUS Authorization

Some mobile architectures contain femtocell gateways used primarily to aggregate traffic from multiple femtocells and forward the aggregated traffic toward the network core. Some of these femtocell gateways also provide provided additional services to include

- RADIUS-compatible authorization services, which can be employed after a remote femtocell has successfully authenticated with the Security Gateway
- provision of local IP addresses thus ensuring that return traffic will transit the femtocell gateway and then be forwarded to the destination femtocell

How It Works

If RADIUS-based authorization is enabled, the Security Gateway first authenticates the endpoint femtocell. Authentication can be performed locally, using either a password or an X.509 certificate, or, if authentication is EAP-based, by a pre-configured RADIUS AAA server. RADIUS authentication is accomplished by a RADIUS *Access-Request* (originated by the Security Gateway)/*Access-Accept* (originated by the RADIUS AAA server) exchange. Assuming that the initiating femtocell is authenticated, using any of the supported methods, the Security Gateway constructs another RADIUS *Access-Request* message consisting of the following RADIUS attributes:

- *User-Name*, which contains the femtocell identity received in the initiator IDi payload
- *Calling-Station-ID*, which contains the femtocell IP address
- State, which contains an RFC 2865, *Remote Authentication Dial-In User Service*, compliant value
- *Framed-IP-address*, which contains the internal IP address assigned by the Security Gateway (sent only if address assignment is locally performed — if address assignment is performed by a RADIUS server, this attribute is empty)

The Security Gateway suspends IPsec tunnel establishment, and uses a UDP transport to send the *Access-Request* message to a femtocell gateway that provides RADIUS-based authorization services.

If the femtocell gateway authorizes the transaction, it returns an *Access-Accept* message. If address assignment is provided by a RADIUS server, the *Framed-IP-address* attribute contains an address on the femtocell gateway network, thus ensuring that return traffic will transit the femtocell gateway and then be forwarded to the destination femtocell. IPsec tunnel establishment proceeds upon receipt of the RADIUS *Access-Accept*.

If the femtocell gateway does not authorize the transaction, it returns an *Access-Reject* message. Upon receiving an *Access-Reject*, the Security Gateway terminates the IPsec tunnel by deleting the existing SAs and associated resources.

In a smaller number of architectures, configuration restraints can prevent the RADIUS AAA server (the authenticating server) from providing a *Framed-IP-address* in the Access-Accept message. The Security Gateway considers the absence of the *Framed-IP-address* as an authentication failure and terminates the IPsec tunnel. This approach does not take into account that address assignment may be performed by the femtocell gateway.

As a consequence the following processing is used in Release M-CX1.20F3 and subsequent releases.

Authentication is performed as previously described with one exception. Previously the Security Gateway required the return of both a *Framed-IP-address* and *Framed-IP-Netmask* in an Access-Accept message. Receipt of a *Framed-IP-Netmask* is no longer mandatory. In the absence of a specified mask value, the Security Gateway provides a default value of 255.255.255.255 assuming that the Framed-IP-address is that of a host and not a router or network. The *Framed-IP-address* (if any) returned by the RADIUS AAA server is stored in memory. Failure of the RADIUS AAA server to return a *Framed-IP-address* is not deemed a failure. The receipt of an Access-Accept message is considered a provisional authentication, and the Security Gateway continues to the authorization phase.

Authorization is also performed as previously described. with the exception that an explicit Framed-IP-Netmask is no longer required. After receipt of an *Access-Accept* from the authorization server, the received *Framed-IP-addresses* (if any) are compared.

- If neither the authentication server nor the authentication server return a *Framed-IP-address*, the Security Gateway terminates the IPsec tunnel establishment.
- If both the authentication server and the authentication server return the same *Framed-IP-address*, the Security Gateway completes the IPsec tunnel establishment using this common address.
- If the authentication server and the authentication server return different *Framed-IP-addresses*, the Security Gateway completes the IPsec tunnel establishment using the *Framed-IP-address* provided by the authorization server.

Configuring RADIUS Authorization

RADIUS authorization using femtocell gateways requires:

- configuration of one or more RADIUS authorization servers, with each server configuration record providing all values required for server access
- configuration of a RADIUS Authorization Servers List containing the IP address of one or more previously configured RADIUS authorization servers

After completing required configuring elements, you enable RADIUS-based authorization on individual IKEv2 interfaces.

RADIUS Authorization Servers

Use the following procedure to configure a femto-cell gateway or similar device that provides RADIUS-compatible authorization service. Within this procedure, such a device is referred to generically as a *RADIUS-compatible authorization server*.

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

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# authentication
ACMEPACKET(authentication)# radius-servers
ACMEPACKET(radius-servers)#
```

2. Use the **state** attribute to set the operational state of this RADIUS-compatible authorization server.

   Retain the default value, *enabled*, to identify this RADIUS-compatible authorization server as operational. Use *disabled* to place this RADIUS-compatible authorization server in a non-operational mode.

   ```
   ACMEPACKET(radius-servers)# state enabled
   ACMEPACKET(radius-servers)#
   ```

3. Use the **authentication-methods** attribute to specify the authentication methods supported by this RADIUS-compatible authorization server.

   Retain the default value, *all*.

   ```
   ACMEPACKET(radius-servers)# authentication-methods all
   ACMEPACKET(radius-servers)#
   ```

4. Use the **address** attribute to specify the IP address of this RADIUS-compatible authorization server.

   ```
   ACMEPACKET(radius-servers)# address 172.30.0.10
   ACMEPACKET(radius-servers)#
   ```

5. Use the **port** attribute to specify the remote port monitored by this RADIUS-compatible authorization server.

   Retain the default value, *1812*, or specify an integer value within the ranges1025 through 65535.

   ```
   ACMEPACKET(radius-servers)# port 6512
   ACMEPACKET(radius-servers)#
   ```

6. Use the **realm-id** attribute to identify the realm that provides transport services to this RADIUS-compatible authorization server.

   ```
   ACMEPACKET(radius-servers)# realm-id radiusServers
   ACMEPACKET(radius-servers)#
   ```

7. Use the **secret** attribute to specify the shared secret between the Security Gateway and this RADIUS-compatible authorization server.

   The shared secret is used for mutual authentication between the Security Gateway and this RADIUS-compatible authorization server.

   ```
   ACMEPACKET(radius-servers)# secret 27IAmSoftSiftInAnHourGlass!
   ACMEPACKET(radius-servers)#
   ```

8. Use the **nas-id** attribute to provide a string that uniquely identifies the Security Gateway to this RADIUS-compatible authorization server.

   ```
   ACMEPACKET(radius-servers)# nas-id nas-id-170-30-0-1
   ACMEPACKET(radius-servers)#
   ```

9. Use the **retry-limit** attribute to specify the number of times the Security Gateway retransmits an unacknowledged authorization request to this RADIUS-compatible authorization server.

   If the RADIUS-compatible authorization server does not respond within the retry limit, the Security Gateway marks it as quarantined.

   Retain the default value, *3*, or provide an alternate integer value within the range 1 through 5.

   ```
   ACMEPACKET(radius-servers)# retry-limit 1
   ACMEPACKET(radius-servers)#
   ```

10. Use the **dead-time** attribute to specify the length (expressed in seconds) of the quarantine period imposed on an unresponsive RADIUS-compatible authorization server.

    Retain the default value, *10*, or provide an alternate integer value within the range 10 through 1000.

    ```
    ACMEPACKET(radius-servers)# dead-time 20
    ACMEPACKET(radius-servers)#
    ```

11. Use the **retry-time** attribute to specify the interval (expressed in seconds) between unacknowledged authorization requests.

    Retain the default value, *5*, or provide an alternate integer value within the range 5 through 10.

    ```
    ACMEPACKET(radius-servers)# retry-time 5
    ACMEPACKET(radius-servers)#
    ```

12. Use the **maximum-sessions** attribute to specify the maximum number of outstanding sessions for this RADIUS-compatible authorization server.

    Retain the default value, *255*, or provide an alternate integer value within the range 1 through 255.

    ```
    ACMEPACKET(radius-servers)# maximum-sessions 50
    ACMEPACKET(radius-servers)#
    ```

13. Use the **class** attribute to select the RADIUS-compatible authorization server class, either *primary* or *secondary*.

    The Security Gateway tries to initiate contact with primary RADIUS-compatible authorization servers first, and only turns to secondary servers if no primaries are available.

    If more than one RADIUS-compatible authorization server is designated as *primary*, the Security Gateway uses a round-robin strategy to distribute authorization requests among available primaries.

    ```
    ACMEPACKET(radius-servers)# class primary
    ACMEPACKET(radius-servers)#
    ```

14. Use **done**, **exit**, and **verify-config** to complete configuration of this RADIUS-compatible authorization server.

    verify-config ensures that each IP address included in the Authorization Servers List is supported by a *radius-servers* configuration element with the identical IP address.

15. If necessary, complete Steps 1 through 14 to configure additional RADIUS-compatible authorization servers.

RADIUS Authorization Servers List

Use the following procedure to configure a RADIUS Authorization Servers List, a list of RADIUS servers available to provide authorization services.

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

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# auth-params
ACMEPACKET(auth-params)#
```

2. Use the **name** attribute to provide a unique name for this RADIUS Authorization Servers List.

```
ACMEPACKET(auth-params)# name radiusauthorizationList
ACMEPACKET(auth-params)#
```

3. Use the **authorization-servers** attribute to construct a RADIUS Authorization Servers List.

   Provide IP addresses to identify each RADIUS-compatible authorization server.

   Use double quotes (") to bracket lists which contain multiple entries.

```
ACMEPACKET(auth-params)# authorization-servers 172.30.0.30
ACMEPACKET(auth-params)#

ACMEPACKET(auth-params)# authorization-servers "172.30.0.30
172.30.0.50"
ACMEPACKET(auth-params)#
```

4. Retain default values for all other parameters.
5. Use **done**, **exit**, and **verify-config** to complete configuration of this RADIUS Authorization Servers List.

   verify-config ensures that each IP address included in the Authorization Servers List is supported by a *radius-servers* configuration element with the identical IP address.

6. If necessary, complete Steps 1 through 5 to configure additional RADIUS Authorization Servers Lists.

Complete RADIUS authorization configuration by enabling RADIUS authorization on an IKEv2 interface.

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

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-interface
ACMEPACKET(ike-interface)#
```

2. Use the **select** command to specify the target interface.
3. Use the **authorization** attribute to enable RADIUS authorization on the selected interface.

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

```
ACMEPACKET(ike-interface)# authorization enabled
ACMEPACKET(ike-interface)#
```

4. Use **done**, **exit**, and **verify-config** to complete enabling RADIUS authorization.
5. If necessary, repeat steps 1 through 4 to enable RADIUS authorization on additional IKEv2 interfaces.

Local Address Pool Configuration

If your network environment requires local address pools that serve as a source of IPv4 addresses temporarily leased for use by remote IKEv2 peers, use the procedures in the following two sections to configure such pools.

During the IKE_AUTH exchange, the IPsec initiator (the remote endpoint) often requests an internal IP address from an IPsec responder (the Security Gateway). Refer to Section 2.19 of RFC 4306, *Internet Key Exchange (IKEv2) Protocol*, for a description of the request process. Procuring such a *local* IP address ensures that traffic returning to the endpoint is routed to the Security Gateway, and then tunneled back to the endpoint. Local address pools provide the source of these addresses available for temporary endpoint lease.

After address assignment from the local address pool, the endpoint retains rights to that IP address for the tunnel lifetime. Tunnels are terminated either by an INFORMATIONAL exchange, defined in Section 1.4 of RFC4306, or by expiration of the tunnel SAs as specified by the **v2-ike-life-seconds** and **v2-ipsec-life-seconds** configuration parameters. In either case, a subsequent request for an assigned IP address may, or may not result, in the assignment of the previous IP address. However, the Security Gateway can be configured to ensure that a prematurely terminated tunnel, resulting for example from the reset or re-boot of the remote IP peer, can be restored with that previous address. Refer to Persistent Tunnel Addressing in this chapter for operational and configuration details.

During the IKE_AUTH request phase, the IKEv2 initiator can use the Configuration payload in conjunction with either the INTERNAL_IP4_DNS or INTERNAL_IP6_DNS attribute to request the addresses of DNS providers from the Security Gateway. In environments where authorization is performed by a RADIUS AAA server, there are two potential sources of DNS information: local Security Gateway DNS configuration elements, and external RADIUS servers that may provide DNS information in the Access-Accept packet that concludes a successful authentication effort. The source of DNS information provided by the Security Gateway to an IKEv2 peer is subject to user configuration.

A RADIUS source of DNS information is enabled by support for certain Microsoft vendor-specific RADIUS attributes specified in RFC2548, *Microsoft Vendor-Specific RADIUS Attributes*. Operationally, the Security Gateway extracts the values of the *MS-Primary-DNS-Server* and *MS-Secondary-DNS-Server* attributes from an Access-Accept packet and returns these values to the IKEv2 initiator.

When the DNS information is from external source, the Security Gateway installs a NAT flow (a static traffic path) that provides access to the DNS server. The NAT flow is calculated based on the location of the DNS server IP returned from RADIUS AAA server and configured realm information.

Configuration of DNS information services takes place at the local address pool and IKEv2 interface levels.

Data Flow Configuration

If you will be configuring address pools, you initially configure data flows that you subsequently assign to a specific local address pool. In practice, a data flow establishes a static route between a remote IKEv2 peer and a core gateway/router which provides routing services after the associated traffic exits the Security Gateway.

1. From superuser mode, use the following command sequence to access *data-flow* configuration mode. While in this mode, you configure static traffic relays.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# data-flow
   ACMEPACKET(data-flow)#
   ```

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

   name enables the creation of multiple *data-flow* instances.

   ```
   ACMEPACKET(data-flow)# name omar
   ACMEPACKET(data-flow)#
   ```

3. Use the required **realm-id** parameter to identify the realm that supports *data-flow* upstream traffic, that is traffic toward the network core.

   ```
   ACMEPACKET(data-flow)# realm-id access-1
   ACMEPACKET(data-flow)#
   ```

4. Use the optional **group-size** parameter to specify the maximum number of user elements grouped together by this **data-flow** instance.

   The size of the associated *local-address-pool* is divided by this value to segment the address pool into smaller groups. After determining the start address for each of the smaller address groups, the Security Gateway uses the **data-flow** configuration to establish two static flows for each of the address groups — a downstream data-flow, in the access direction, and an upstream data-flow (via the realm specified by the **realm-id** parameter) toward a core gateway/router which provides forwarding service for the pass-thru data-flow.

   Allowable values are integers within the range 1 through 255.

   For maximum efficiency, this value should be set to a power of 2.

   ```
   ACMEPACKET(data-flow)# group-size 32
   ACMEPACKET(data-flow)#
   ```

5. Use the optional **upstream-rate** parameter to specify the allocated upstream bandwidth.

   Allowable values are integers within the range 0 (the default) through 999,999,999.

   The default value (0) allocates all available bandwidth.

   ```
   ACMEPACKET(data-flow)# upstream-rate 560000000
   ACMEPACKET(data-flow)#
   ```

6. Use the optional **downstream-rate** parameter to specify the allocated downstream bandwidth.

   Allowable values are integers within the range 0 (the default) through 999,999,999.

   The default value (0) allocates all available bandwidth.

   ```
   ACMEPACKET(data-flow)# downstream-rate 280000000
   ACMEPACKET(data-flow)#
   ```

7. Use **done**, **exit**, and **verify-config** to complete configuration of the *data-flow* instance.
8. Repeat Steps 1 through 7 to configure additional *data-flow* instances.

Local Address Pool Configuration

You configure an address pool by associating a contiguous range of IPv4 addresses with an existing *data-flow*.

1. From superuser mode, use the following command sequence to access *local-address-pool* configuration mode. While in this mode, you configure local address pools.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# local-address-pool
   ACMEPACKET(local-address-pool)#
   ```

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

   name enables the creation of multiple *local-address-pool* instances.

   ```
   ACMEPACKET(local-address-pool)# name ikePool
   ACMEPACKET(local-address-pool)#
   ```

3. Use the **dns-assignment** parameter to identify the DNS source used to respond to incoming IKE_AUTH requests for DNS information.

   Select **local** to use locally configured configuration data as the source of DNS information.

   Select **radius** to use a remote RADIUS AAA server as the source of DNS information.

   Select **radius-local** to use a remote RADIUS AAA server as the preferred source of DNS information; if no DNS data is available from the RADIUS server, use locally configured configuration data as the secondary source of DNS information.

   Retain the default value (an empty string) to disable responses to incoming IKE_AUTH requests for DNS information.

   ```
   ACMEPACKET(local-address-pool)# dns-assignment radius-local
   ACMEPACKET(local-address-pool)#
   ```

4. Use the **dns-realm-id** parameter when **radius** or **radius-local** is identified as the DNS source to specify the realm that provides first hop transit service to the DNS provider returned by the RADIUS AAA server. This parameter can be safely ignored if either **local** or the default value is specified as the DNS source.

   ```
   ACMEPACKET(local-address-pool)# dns-realm-id defaultRouter
   ACMEPACKET(local-address-pool)#
   ```

5. Use the required **data-flow** parameter to identify the *data-flow* configuration element assigned to this *local-address-pool* instance.

```
ACMEPACKET(local-address-pool)# data-flow dFlow-1
ACMEPACKET(local-address-pool)#
```

6. Use **address-range** to move to address-range configuration mode (refer to *address-range Configuration Element*).

```
ACMEPACKET(local-address-pool)# address-range
ACMEPACKET(address-range)#
```

7. Use **network-address** in conjunction with **subnet-mask** to define a contiguous pool of IPv4 addresses.

   The following sequence defines a range of 62 addresses from 192.168.0.1 through 192.168.0.62.

```
ACMEPACKET(address-range)# network-address 192.168.0.0
ACMEPACKET(address-range)# subnet-mask 255.255.255.96
```

8. Use **done** and **exit** to complete configuration of the *address-range* instance.
9. Use **done**, **exit**, and **verify-config** to complete configuration of the *local-address-pool* instance.
10. Repeat Steps 1 through 9 to configure additional *local-address-pool* instances.

Persistent Tunnel Addressing

After address assignment from the local address pool, the endpoint retains rights to that IP address for the tunnel lifetime. Tunnels can be terminated either by an INFORMATIONAL exchange, defined in Section 1.4 of RFC4306, or by expiration of the tunnel SAs as specified by the **v2-ike-life-seconds** and **v2-ipsec-life-seconds** parameters. In either case, a subsequent request for an assigned IP address may, or may not result, in the assignment of the previous IP address. However, the Security Gateway can be configured to ensure that a prematurely terminated tunnel can be restored with that previous address.

Tunnels are usually prematurely terminated because of re-boot or reset of the remote endpoint. In either case, the endpoint's IKEv2 and IPsec SAs are lost and the tunnel no longer exists. From the point of view of the Security Gateway, however, the tunnel remains *live*. The local IKEv2 and IPsec SAs still exist, and the tunnel remains available in an active state until the expiration of the lifetime timers. Similarly, the IP address assignment from the local address poll remains in effect until timer expiration.

When a crashed endpoint attempts to re-establish a tunnel, it can insert a Notify payload in the initial IKE_AUTH request. The Notify payload contains an INITIAL_CONTACT message that asserts a prior connection between the endpoint and the Security Gateway. When receiving an INITIAL_CONTACT message, the Security Gateway checks for the existence of a live tunnel with the requesting endpoint. If such a tunnel is found, the Security Gateway stores the assigned IP address, tears down the tunnel by removing the supporting IKEv2 and IPsec SAs, and authenticates the endpoint. Assuming authentication succeeds, the Security Gateway, retrieves the previously assigned IP address and returns it to the endpoint.

If a live tunnel is not found (meaning that the tunnel has timed out during the interval between the endpoint reset/re-boot and the new IKE_AUTH), the assertion of a prior connection is ignored, and address assignment is made from the local address pool.

You can use a global configuration option (**assume-initial-contact**) to enable persistent address processing with or without the reception of an INITIAL_CONTACT message. With this option enabled, all IKE_AUTH requests are processed as if they contained an INITIAL_CONTACT message.

Use the following command sequence to enable persistent tunnel addressing.

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

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-config
   ACMEPACKET(ike-config)#
   ```

2. Use the **options** command to enable address persistence.

   The following example adds address persistence to the list of currently supported options.

   ```
   ACMEPACKET(ike-config)# options +assume-initial-contact
   ACMEPACKET(local-address-pool)#
   ```

3. Use **done** and **exit** to complete configuration of address persistence.

key-id Configuration

If authentication between IKEv2 peers is based on a PSK associated with an identity asserted in the IKE Identification Payload, use the following procedures to associate received asserted identities with a specified PSK.

1. From superuser mode, use the following command sequence to access *ike-keyid* configuration mode. While in this mode, you configure associates between an IKE asserted identities and a PSK.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-keyid
   ACMEPACKET(ike-keyid)#
   ```

2. Use the required **name** parameter to provide a unique identifier for this *ike-keyid* instance.

   name enables the creation of multiple *ike-keyid* instances.

   ```
   ACMEPACKET(ike-keyid)# name defaultIKEid
   ACMEPACKET(ike-keyid)#
   ```

3. Use the required **keyid** and **presharedkey** parameters to associate an asserted identity with a PSK.

   ```
   ACMEPACKET(ike-keyid)# keyid 172.16.20.20
   ACMEPACKET(ike-keyid)# presharedkey **********
   ```

4. Use **done**, **exit**, and **verify-config** to complete configuration of the *ike-keyid* instance.

5. Repeat Steps 1 through 4 to configure additional *ike-keyid* instances.

ike-config ACLI Reference

ike-config

The **ike-config** command moves from *ike* configuration mode to *ike-config* configuration mode where you define global IKEv2 parameters.

Syntax	`ike-config`
Mode	Superuser
Path	Type **ike-config** at the *ike* prompt.
Release	First appearance: M-C1.0.0
License Requirements	IKE License
RTC Status	Supported
Notes	This is a single instance configuration element.

Access Path

```
ACMEPACKET> enable
password: xxxxxxx
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-config
ACMEPACKET(ike-config)#
```

ike-config Configuration Element

This configuration element defines a single, global Internet Key Exchange (IKEv2) configuration object.

Syntax

```
ike-config < addr-assignment | anti-replay |
certificate-profile-id | dpd-time-interval | eap-bypass-identity |
eap-protocol | id-auth-type | ike-version |
log-level | overload-action | overload-critical-interval |
overload-critical-threshold | overload-interval |
overload-threshold | red-max-trans | red-port |
red-sync-comp-time | red-sync-start-time |
sd-authentication-method | shared-password | state | udp-port |
v2-ike-life-secs | v2-ipsec-life-secs | v2-rekey | done | exit |
no | select | show>
```

Parameters

addr-assignment — selects the method used to assign addresses in response to an IKEv2 Configuration Payload request.

The Configuration Payload supports the exchange of configuration information between IKEv2 peers. Typically, an IRAC (IPsec Remote Access Client) initiates the exchange by requesting an IP address on the gateway's protected network. In response, the gateway, referred to as an IRAS (IPsec Remote Access Server), returns a local address for the IRAC's use.

This parameter specifies the source of the returned IP address, and can be over-ridden at the IKE interface level.

Default local

Values
- local — use local address pool
- radius-only — obtain local address from RADIUS server
- radius-local — try RADIUS server first, then local address pool

anti-replay — enables/disables anti-replay protection on IPsec Security Associations.

IPsec authentication provides anti-replay protection against an attacker duplicating encrypted packets by assigning a unique incrementally increasing 32-bit sequence number to each encrypted packet. Packets received with previously processed sequence numbers are rejected.

With anti-replay protection enabled, roll-over of the 32-bit sequence number triggers re-negotiation of the IPsec Security Association

Default enabled

Values disabled | enabled

certificate-profile-id — used when **sd-authentication-method** is *certificate*, identifies the default *ike-certificate-profile* configuration element that contains identification and validation credentials required for certificate-based IKEv2 authentication.

This parameter can be over-ridden at the IKE interface level.

Default none

Values the name of an existing *ike-certificate-profile* configuration element

dpd-time-interval — specifies the maximum period of inactivity before the Dead Peer Detection (DPD) protocol is initiated on a specific endpoint.

The default value, 0, disables the DPD protocol; setting this parameter to a non-zero value globally enables the protocol and sets the inactivity timer.

Default 0 (DPD disabled)

Values 1 through 999999999 (seconds)

eap-bypass-identity — specifies whether or not to bypass the EAP (Extensible Authentication Protocol) identity phase.

EAP, defined in RFC 3748, provides an authentication framework widely used in wireless networks.

An Identity exchange is optional within the EAP conversation. Therefore, it is possible to omit the Identity exchange entirely, or to use a method-specific identity exchange once a protected channel has been established.

However, where roaming is supported, it may be necessary to locate the appropriate backend authentication server before the authentication conversation can proceed. The realm portion of the Network Access Identifier (NAI) is typically included within the EAP-Response/Identity to enable the routing of the authentication exchange to the appropriate authentication server. Therefore, while the peer-name portion of the NAI may be omitted in the EAP-Response/Identity where proxies or relays are present, the realm portion may be required.

Default disabled (requires an identity exchange)

Values disabled | enabled

eap-protocol — identifies the EAP protocol used with IKEv2.

With the current release, EAP operations are performed by a designated RADIUS server or server group; retain the default value.

Default eap-radius-passthru

Values eap-radius-passthru (only currently supported value)

id-auth-type — is relevant only when **sd-authentication-method** is *shared-password*.

If authentication is based on a pre-shared secret key (PSK), the actual secret can be specified (1) at the global IKE configuration level, (2) at the IKEv2 interface level, or (3) by an *ike-keyid* configuration element, which associates a PSK with an IKEv2 asserted identity (*IDi* for the IKE initiator or *IDr* for the IKE responder) carried within an IKE Identification Payload.

The PSK selection algorithm works as follows:

If **id-auth-type** is set to *idi* (the default value), the Security Gateway checks the value of the *ID_TYPE* field of the IDi Payload. If the value of that field is *ID_KEY_ID* (essentially indicating a request for vendor-specific proprietary processing), the Security Gateway examines existing *ike-keyid* configuration elements for the asserted identity extracted from the *Identification Data* field of the IDi payload. If a match is found, the PSK associated with the asserted identity is used. If a match is not found, or if no *ike-keyid* elements are configured, the Security Gateway selects the IKEv2-interface-specific PSK (if one has been configured). In the absence of an interface key, the Security Gateway selects the global default PSK.

If **id-auth-type** is set to *idr*, the Security Gateway operates in the same manner checking the *ID_TYPE* and *Identification Dat*a fields of the IDr Payload.

Default idi

Values idi | idr

ike-version — selects IKEv1 or IKEv2 as the key exchange protocol.

Select IKEv2 for Security Gateway operations.

Default 2

Values 1 | 2

log-level — selects the IKE log level; events of the specified level and other events deemed more critical are written to the system log.

Events are listed below in descending order of criticality.

Default info

Values

• emergency	(most critical)
• critical	
• major	
• minor	
• warning	
• notice	
• info	(least critical)
• trace	(test/debug, not used in production environments)
• debug	(test/debug, not used in production environments)
• detail	(test/debug, not used in production environments)

options +no-cookie — (a parameter specific to IKEv2) disables the use of COOKIE in IKE-SA-INIT messages.

Interoperability testing revealed that some femtocells do not properly respond to IKE-SA-INIT messages that contain the cookie NOTIFY payload.

With the no-cookie option, messages containing the cookie payload will not be sent.

Use this option only to address interoperability issues.

Default N/A

Values N/A

overload-action — specifies the action taken when the Security Gateway CPU enters an overload state. The overload state is reached when CPU usage exceeds the percentage threshold specified by the **overload-threshold** parameter.

By default, no preventive action is taken in response to an overload, You can, however, use this parameter to implement a call rejection algorithm in response to the overload. With the algorithm enabled, the Security Gateway uses the following calculation to reject/drop some number of incoming calls:

DropRate = (currentLoad - overloadThreshold) / (100 -overloadThreshold)

Thus, assuming a current CPU load of 70% and an overload threshold of 60%, the Security Gateway drops 1 of out every 4 incoming calls until the load falls below the threshold value.

Use *none* to retain default behavior (no action); use *drop-new-connection* to implement call rejection.

Default none

Values drop-new-connection | none

overload-critical-interval — specifies the interval (in seconds) between CPU load measurements while in the critical overload state.

Default 1

Values an integer within the range 1 through 60

overload-critical-threshold — specifies the percentage of CPU usage that triggers a critical overload state.

When this threshold is exceeded, the Security Gateway drops all incoming calls until the load drops below the critical threshold level, at which point it may drop selective calls depending on the value of the **overload-threshold** parameter.

Default 100 (disabling overload processing)

Values an integer from 1 to 100, and greater than the value of the **overload-threshold** parameter

overload-interval — specifies the interval (in seconds) between CPU load measurements while in the overload state.

Default 1

Values an integer within the range 1 through 60

overload-threshold — specifies the percentage of CPU usage that triggers an overload state.

Default 100 (disabling overload processing)

Values an integer from 1 to 100, and less than the value specified by the **overload-critical-threshold** parameter

red-max-trans —(used in high-availability environments) specifies the maximum number of retained IKEv2 synchronization messages.

Default 10000 (messages)

Values 0 through 999999999 (messages)

red-port — (used in high-availability environments) specifies the port number monitored for IKEv2 synchronization messages.

The default value (0) effectively disables redundant high-availability configurations. Select a port value other than 0 (for example, 1995) to enable high-availability operations.

Default 0

Values 1024 - 65535

red-sync-comp-time — (used in high-availability environments) specifies the interval between synchronization attempts after the completion of an IKEv2 redundancy check

Default 1000 (milliseconds)

Values 0 through 999999999 (milliseconds)

red-sync-start-time — (used in high-availability environments) sets the timer value for transitioning from the standby to the active role — that is the amount of time (specified in milliseconds) that a standby device waits for a heartbeat signal from the active device, before transitioning to the active role.

Default 5000 (milliseconds)

Values 0 through 999999999 (milliseconds)

sd-authentication-method — selects the authentication method used to authenticate the IKE SA.

Two authentication methods are supported.

shared-password — (the default) uses a PSK that is used to calculate a hash over a block of data.

certificate — uses an X.509 certificate to digitally sign a block of data.

This global default can be over-ridden at the IKE interface level.

Default shared-password

Values certificate | shared-password

shared-password — used when **sd-authentication-method** is *shared-password*, and otherwise ignored, provides the default PSK used during IKE SA authentication.

This global default can be over-ridden at the IKE interface level, or if IDi/IDr-based PSKs are used. Refer to the **id-auth-type** property.

Default none

Values a string of ACSII printable characters no longer than 255 characters (not displayed by the ACLI)

state — specifies the state (enabled or disabled) of the *ike-config* configuration element.

Default enabled

Values disabled | disabled

udp-port — specifies the UDP port used for IKE protocol traffic.

Default 500

Values 1025 - 65535

v2-ike-life-secs — specifies the default IKEv2 SA lifetime in seconds.

This global default can be over-ridden at the IKE interface level.

Default 86400 (24 hours)

Values 1 through 999999999 (seconds)

v2-ipsec-life-secs — specifies the default IPsec SA lifetime in seconds.

This global default can be over-ridden at the IKE interface level.

Default 28800 (8 hours)

Values 1 through 999999999 (seconds)

v2-rekey — enables or disables the re-keying of expired IKE or IPsec SAs.

This global default can be over-ridden at the IKE interface level.

With this parameter enabled, the Security Gateway initiates a new negotiation to restore an expired IKE or IPsec SA. Expiration of the **v2-ike-life-secs** timer triggers an IKE SA negotiation, while expiration of the **v2-ipsec-life-secs** triggers an IPsec SA negotiation.

Default disabled

Values disabled | enabled

ike-certificate-profile ACLI Reference

ike-certificate-profile

The **ike-certificate-profile** command moves from *ike* configuration mode to *ike-certificate-profile* configuration mode where you configure certificate profiles that provide identification and validation credentials to a specific IKEv2 identity.

Syntax	`ike-certificate-profile`
Mode	Superuser
Path	Type **ike-certificate-profile** at the *ike* prompt.
Release	First appearance: M-C1.0.0
Hardware Requirements	SSM2
License Requirements	IKE License
RTC Status	Supported
Notes	This is a multiple instance configuration element.

Access Path

```
ACMEPACKET> enable
password: xxxxxxx
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-certificate-profile
ACMEPACKET(ike-certificate-profile)#
```

ike-certificate-profile Configuration Element

The *ike-certificate-profile* configuration element references a public certificate which authenticates a specific IKEv2 identity, as well as one of more CA certificates used to validate a certificate offered by a remote peer.

Syntax

```
ike-certificate-profile <end-entity-certificate | identity |
trusted-ca-certificates | verify-depth | batch | done | exit |
no | select | show>
```

Parameters

end-entity-certificate — specifies the *certificate-record* configuration element referencing the identification credential (specifically, an X509.v3 certificate) offered by a local IKEv2 entity in support of its asserted identity.

Default none

Values the name of an existing *certificate-record* configuration element

identity — specifies the local IKEv2 entity that uses the authentication and validation credentials provided by this *ike-certificate-profile* instance.

Default none

Values an IP address or fully-qualified domain name (FQDN) that uniquely identifies the user of resources provided by this *ike-certificate-profile* instance

trusted-ca-certificates — specifies one or more certificate-record configuration elements referencing CA certificates used to authenticate a remote IKEv2 peer.

Default none

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

verify-depth — specifies the maximum number of chained certificates that will be processed while authenticating the IKEv2 peer.

Default 10

Values an integer within the range 1 through 10

ike-certificate-profile

The **radius-servers** command moves from *authentication* configuration mode to *radius-servers* configuration mode where you configure one or more RADIUS servers that can provide address assignment, authorization, and/or authentication services within a Security Gateway environment.

Syntax `radius-servers`

Mode Superuser

Path Type **radius-servers** at the *authenticatiom* prompt.

Hardware Requirements SSM2

RTC Status Supported

Notes This is a multiple instance configuration element.

Access Path

```
ACMEPACKET> enable
password: xxxxxxx
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# authentication
ACMEPACKET(authentication)# radius-servers
ACMEPACKET(radius-servers)#
```

radius-servers Configuration Element

The *radius-servers* configuration element provides the data needed by the Security Gateway to locate and access one or more RADIUS servers.

Syntax

```
radius-servers <address | authentication-methods | class |
dead-time | maximum-sessions | nas-id | port | realm-id |
retry-limit | retry-time | secret | state | done | exit |
no | select | show>
```

Parameters

address — specifies the IP address of the RADIUS authentication server.

Default none

Values the IP address of a RADIUS authentication server

authentication-methods — specifies the authentication methods supported by the RADIUS authorization server

Default all

Values Retain the default

class — specifies the server class, *primary* or *secondary*

Default primary

Values primary | secondary

dead-time — specifies the quarantine period imposed upon an unresponsive RADIUS authentication server.

Default 10 (seconds)

Values 10 - 1000

maximum-sessions — specifies the server-specific maximum number of tolerated outstanding RADIUS request messages

Default 255 (messages)

Values 1 - 255

nas-id — specifies the string that identifies the Security Gateway to the RADIUS authentication server.

Default none

Values a string

port — specifies the port monitored by the RADIUS authentication server for incoming RADIUS request messages.

Default 1812

Values an integer within the range 1025 - 65535

realm-id — specifies the name of the existing realm that supports media exchange between the Security Gateway and the RADIUS authentication server.

Default none

Values the name of an existing realm

retry-limit — specifies the maximum number of times that the Security Gateway re-sends an unacknowledged RADIUS request message before moving to another RADIUS authentication server (if one is available)

Default 3

Values 1 - 5

retry-time — specifies the number of seconds that the Security Gateway waits to receive an acknowledgement of a RADIUS request message.

Default 5

Values 5 - 10

secret — specifies the shared secret between the Security Gateway and the RADIUS authentication server.

Default none

Values a string

state — enables or disables the RADIUS authentication server.

Default enabled

Values disabled | enabled

data-flow ACLI Reference

data-flow

The **data-flow** command moves from *security* configuration mode to *data-flow* configuration mode where you configure Security Gateway *data-flow* configuration elements, which enable the static pass-thru of data-traffic (non-SIP, non-RTP, non-RTCP, and non-DNS packets) to neighboring gateways/routers.

After defining one or more *data-flow* configuration elements, you assign a specific *data-flow* instance to a *local-address-pool* configuration element.

Syntax

```
data-flow
```

Mode — Superuser

Path — Type **data-flow** at the *ike* prompt.

Release — First appearance: M-C1.0.0

License Requirements — IKE License

RTC Status — Supported

Notes — This is a multiple instance configuration element.

Access Path

```
ACMEPACKET> enable
password: xxxxxxx
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(security)# data-flow
ACMEPACKET(data-flow)#
```

data-flow Configuration Element

The *date-flow* configuration element specifies pass-thru data-traffic processing.

Syntax

```
data-flow <downstream-rate | group-size | name | realm-id |
upstream-rate | batch | done | exit | no | select | show>
```

Parameters

downstream-rate — specifies the allocated downstream (access side) bandwidth.

Default 0 (unlimited, no bandwidth restrictions)

Values an integer within the range 0 through 999,999,999 that specifies the downstream bandwidth in kbs.

group-size — specifies the maximum number of user elements grouped together by this *data-flow* instance.

The size of the associated *local-address-pool* is divided by this value to segment the address pool into smaller groups. After determining the start address for each of the smaller address groups, the Security Gateway uses the *data-flow* configuration to establish two static flows for each of the address groups — a downstream data-flow, in the access direction, and an upstream data-flow (via the realm specified by the **realm-id** parameter) toward a core gateway/router which provides forwarding service for the pass-thru data-flow.

For maximum efficiency, this value should be set to a power of 2.

Default 128

Values 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256

name — specifies the name of this **data-flow** instance.

Default none

Values a valid configuration element name, that is unique within the **data-flow** namespace

realm-id — specifies the realm that supports the upstream (core side) data-flow.

Default none

Values the name of an existing **realm** configuration element

upstream-rate — specifies the allocated upstream bandwidth.

Default 0 (allocates all available bandwidth)

Values an integer within the range 0 through 999,999,999 that specifies the upstream (core side) bandwidth in kbs.

local-address-pool ACLI Reference

local-address-pool

The **local-address-pool** command moves from *ike* configuration mode to *local-address-pool* configuration mode where you configure an IP address range that is made available in response to Configuration Payload requests for local addresses.

Syntax	`local-address-pool`
Mode	Superuser
Path	Type **local-address-pool** at the *ike* prompt.
Release	First appearance: M-C1.0.0
License Requirements	IKE License
RTC Status	Supported
Notes	This is a multiple instance configuration element.

Access Path

```
ACMEPACKET> enable
password: xxxxxxx
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# local-address-pool
ACMEPACKET(local-address-pool)#
```

local-address-pool Configuration Element

The **local-address-pool** configuration element enables creation of local address pools, which can be used to provide a local (internal) address in response to remote requests for IP addresses.

Syntax

```
local-address-pool <address-range | data-flow | dns-assignment |
dns-realm-id | name | batch | done | exit | no | select | show>
```

Parameters

data-flow — identifies the *data-flow* configuration element assigned to this *local-address-pool* instance.

The *data-flow* configuration element specifies bandwidth available to the pool of addresses specified by this *local-address-pool* instance.

Default none

Values the name of an existing **data-flow** configuration element

dns-assignment — identifies the DNS source used to respond to incoming IKE_AUTH requests for DNS information.

Default an empty string

Values local | radius | radius-local

dns-realm-id — when **dns-assignment** is either **radius** or **radius-local**, identifies the realm that provides first hop transit service to the DNS provider returned by the RADIUS AAA server.

Default none

Values the name of an existing *realm* configuration element

name — uniquely identifies this *local-address-pool* instance.

Default none

Values a valid configuration element name, that is unique within the *local-address-pool* namespace

address-range — enters address-range configuration mode.

address-range Configuration Element

The *address-range* configuration element specifies a single range of contiguous IPv4 addresses that are available to fulfill remote requests for a local address.

Syntax

```
address-range <network-address | subnet-mask | | batch | done |
exit | no | select | show>
```

Parameters

network-address — (in conjunction with **subnet-mask**) defines a range of IPv4 addresses available for dynamic assignment.

Default none

Values a valid IPv4 network address

subnet-mask — (in conjunction with **network-address**) defines a range of IPv4 addresses available for dynamic assignment.

Default none

Values a valid IPv4 subnet mask

ike-keyid ACLI Reference

ike-keyid

The **ike-keyid** command moves from *ike* configuration mode to *ike-keyid* configuration mode where you configure associations between asserted identities carried in IKEv2 Identification Payloads with a pre-shared secret key (PSK).

Syntax	`ike-keyid`
Mode	Superuser
Path	Type **ike-keyid** at the *ike* prompt.
Release	First appearance: M-C1.0.0
Hardware Requirements	SSM2
License Requirements	IKE License
RTC Status	Supported
Notes	This is a multiple instance configuration element.

Access Path

```
ACMEPACKET> enable
password: xxxxxxx
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-keyid
ACMEPACKET(ike-keyid)#
```

ike-keyid Configuration Element

The *ike-keyid* configuration element associates a PSK with and IKE asserted identity.

Syntax

```
ike-keyid <keyid | name | presharedkey | batch | done | exit |
no | select | show>
```

Parameters

name — uniquely identifies this instance of the *ike-keyid* configuration element.

Default none

Values a valid configuration element name, that is unique within the *ike-keyid* namespace

keyid — specifies the asserted identity (derived from the *Identification Data* field of the IKEv2 Information Payload) associated with *presharedkey*.

Default none

Values a string containing an IKEv2 asserted identity

Configuring IKEv2 Interfaces

After configuring global IKE parameters, use the procedures described in this chapter to configure and monitor IKEv2 interfaces.

IKEv2 interface configuration consists of the following steps.

1. Configure IKE interface attributes
2. Configure Security Associations
3. Configure Security Policies
4. Configure the Dead Peer Detection Protocol (optional)
5. Configure the Online Certificate Status Protocol or Certificate Revocation List Support (optional)
6. Configure Threshold Crossing Alerts (optional)
7. Configure Mobile Network Codes white lists (optional)

IKEv2 Negotiation, Authentication, and IPsec Tunnel Establishment

As described in RFC 4306, Internet Key Exchange (IKEv2) Protocol, IKEv2 protocol negotiation, to include peer identification and authentication, and establishment of an IPsec tunnel generally requires only four messages between IKEv2 peers (for example, a femtocell and the Security Gateway).

All IKEv2 communications consist of request/response pairs. The initial request/response pair is referred to as the IKE_SA_INIT exchange,

IKE_SA_INIT

During IKE_SA_INIT, the IKEv2 initiator (the remote IKEv2 peer) and the IKEv2 responder (the Security Gateway) negotiate cryptographic algorithms and exchange Diffie-Hellman public values and nonces (randomly generated values of at least 128-bits that are input into cryptological functions).

Initiator (remote IKEv2 peer) ----------> HDR, SAi1, KEi, Ni

HDR

is the common IKEv2 header

SAi1

is the Security Association payload (defined in Section 3.3 of RFC 4306) proposed by the initiator for creation of the IKEv2 SA. This payload contains the initiator's ranked preferences for encryption algorithms, pseudo-random functions, authentication algorithms, Diffie-Hellman groups 1, 2, 5, and 14 are supported), and Extended Sequence Numbers (ESN) support.

KEi

is the initiator's Key Exchange payload (defined in Section 3.3 of RFC 4306). It contains the initiator's public Diffie-Hellman value.

Ni

is the initiator's Nonce payload (defined in Section 3.9 of RFC 4306). It contains the initiator's nonce, a randomly generated value at least 128-bits in length; the nonce value is input into cryptological functions to increase randomness.

HDR, SAr1, KEr, Nr [CertReq] <---------- (Security Gateway) Responder

HDR

is the common IKEv2 header

SAr1

is the Security Gateway's Security Association payload that specifies cryptological material used for creation of the IKEv2 SA. It selects from the proposals offered in the initiator's corresponding payload.

KEr

is the Security Gateway's Key Exchange payload. It contains the gateway's public Diffie-Hellman value.

Nr

is the Security Gateway's Nonce payload (defined in Section 3.9 of RFC 4306), a randomly generated value at least 128-bits in length; the nonce value is input into cryptological functions to increase randomness.

[CertReq]

is the Security Gateway's optional Certificate Request payload (defined in Section 3.7 of RFC 4306), used only when authentication is PKI-based, to request a certificate from the IKEv2 initiator. If sent, the payload contains a preferred certificate type and a hashed list of preferred CAs.

At the completion of IKE_SA_INIT, each peer can use Diffie-Hellman to derive the same cryptokey seed, which is used to generate an encryption key, an authentication key, and a third key used to produce further keying material. From this point forward, all protocol traffic, except for the common IKEv2 header, is encrypted and authenticated.

IKE_AUTH

The request/response pair following IKE_SA_INIT is referred to as the IKE_AUTH exchange, During IKE_AUTH, the IKEv2 initiator and IKEv2 responder transmit their identities, mutually authenticate, and negotiate a child SA for an IPsec tunnel.

Initiator -----> HDR { IDi, [Cert], [CertReq], [IDr], Auth, SAi2, [CP], TSi, TSr }

HDR

is the common IKEv2 header

{

indicates the beginning of encryption

IDi

is the initiator's Identification payload (defined in Section 3.5 of RFC 4306). It contains the identification type (for example, IPv4 address or FQDN) and the actual address of the initiator.

[Cert]

is the initiator's optional Certificate payload (defined in Section 3.6 of RFC 4306), used only when authentication is PKI-based. It replies to the responder's [CERTREQ] request, and contains the certificate used to authenticate the initiator to the responder.

[CertReq]

is the initiator's optional Certificate Request payload (defined in Section 3.7 of RFC 4306), used only when authentication is PKI-based to request a certificate from the IKEv2 peer. If sent, the payload contains a preferred certificate type and a hashed list of preferred CAs.

[IDr]

is the initiator's optional Identification payload. It is used to designate a specific responder identity, in cases where the responder may have multiple identifies on the same IKEv2 interface. If used, it contains the specific responder address preferred by the initiator.

Auth

is the initiator's Authentication payload (defined in Section 3.8 of RFC 4306). It is used to convey authentication data, which can be verified by the IKEv2 peer — thus confirming the peer's purported identity. It contains an *Auth Method* field that specifies the authentication method, either Message Authentication Code (MAC) when authentication is shared-secret-based, or digital signature when the authentication is PKI-based, as well as an *Authentication Data* field that contains the computed signature or hash value. Both the shared-secret and PKI-based authentication methods are fully described in Section 2.15 of RFC 4306.

SAi2

is the Security Association payload proposed by the initiator for the IPsec SA. This payload contains the initiator's ranked preferences for encryption algorithms, pseudo-random functions, authentication algorithms, Diffie-Hellman groups (groups 1, 2, 5, and 14 are supported), and ESN support.

[CP]

is the optional Configuration payload request. The Configuration payload is used to exchange configuration information between IKE peers. This payload can be used to request and exchange information of the sort that a client would acquire with the Dynamic Host Configuration Protocol (DHCP) if the client were directly connected to a LAN.

Within a Security Gateway deployment, the Configuration payload is most commonly used by an IKEv2 initiator to request a local IP address from the Security Gateway, acting as an IPsec server. Obtaining such a local IP address ensures that traffic returning to the client is routed through the Security Gateway, and then tunneled back to the client. Address requests are enabled by the inclusion of an INTERNAL_IP4_ADDRESS attribute within the Configuration payload.

The Configuration payload is also commonly used by IKEv2 initiators to (1) request the addresses of Domain Name Service (DNS) providers -- accomplished by the inclusion of the INTERNAL_IP4_DNS or INTERNAL_IP6_DNS attribute, or (2) obtain a list of networks served by the IPsec responder. This last exchange is essentially an alternate method of obtaining traffic selectors and is accomplished by the inclusion of the INTERNAL_IP4_SUBNET or INTERNAL_IP6_SUBNET attribute.

Refer to Local Address Pool Configuration in IKEv2 Global Configuration for details on the exchange of DNS information, and to IKEv2 Interface Configuration in this chapter for details on the exchange of sub-network information.

TSi

is an initiator Traffic Selector payload as defined in Section 3.12 of RFC 4306. TSi works in conjunction with the TSr payload to convey proposed selection criteria for packets that are carried over this SA. The TSi payload contains proposed source/destination addresses of traffic forwarded *from* or *to* the SA initiator. Proposed packet flows are defined in terms of IP protocol type, IP address type (IPv4/IPv6), and address:port number ranges. The payload can contain one or more ranges of contiguous IP addresses and port numbers.

TSr

is a responder Traffic Selector payload as defined in Section 3.12 of RFC 4306. TSr works in conjunction with the TSi payload to provide selection criteria for packets that are carried over this SA.The TSr payload conveys proposed destination/source addresses of traffic forwarded *to* or *from* the SA responder. Proposed packet flows are defined in terms of IP protocol type, IP address type (IPv4/IPv6), and address:port number ranges. The payload can contain one or more ranges of contiguous IP addresses and port numbers.

For example, if the initiator wants to tunnel all traffic from subnet 192.0.1.0 on the initiator's side to subnet 192.0.2.0 on the responder's side, the initiator includes a single traffic selector in each Traffic Selector payload, with TSi specifying the address range (192.0.1.0 - 192.0.1.255) and TSr specifying the address range (192.0.2.0 - 192.0.2.255).

}

indicates the end of encryption

HDR { IDr, [Cert], Auth, SAr2, [CP], TSi, TSr } <- - - - - - - - - Responder

HDR

is the common IKEv2 header

{

indicates the beginning of encryption

IDr

is the responder's Identification payload. It contains the identification type (for example, IPv4 address or FQDN) and the actual address of the responder.

[Cert]

is the responder's optional Certificate payload, used only when authentication is PKI-based. It replies to the initiator's [CERTREQ], and contains the certificate used to authenticate the responder to the initiator.

Auth

is the responder's Authentication payload (defined in Section 3.8 of RFC 4306). It is used to convey authentication data, which can be verified by the IKEv2 peer — thus confirming the peer's purported identity. It contains an *Auth Method* field that specifies the authentication method, either Message Authentication Code (MAC) when authentication is shared-secret-based, or digital signature when the authentication is PKI-based, as well as an *Authentication Data* field that contains the computed signature or hash value. Both the shared-secret and PKI-based authentication methods are fully described in Section 2.15 of RFC 4306.

SAr2

is the responder's Security Association payload that specifies cryptological material used for creation of the IPsec SA. It selects from the proposals offered in the initiator's corresponding payload.

[CP]

is the optional Configuration payload response. Assuming that the initiator has requested an IP address on the Security Gateway's network, the Configuration payload provides the required address, which can be assigned from a local address pool or provided by a RADIUS server.

TSi

is an initiator Traffic Selector payload as defined in Section 3.12 of RFC 4306. TSi works in conjunction with the TSr payload to convey negotiated selection criteria for packets that are carried over this SA. The TSi payload contains negotiated source/destination addresses of traffic forwarded from or to the SA initiator. Proposed packet flows are defined in terms of IP protocol type, IP address type (IPv4/IPv6), and address:port number ranges. The TSi Traffic Selector payload can contain one or more ranges of contiguous IP addresses and port numbers. Negotiated ranges can either replicate the contents of the initiator's TS1, or contain a subset of the original proposal. Negotiated ranges are determined by evaluating the SA initiator's proposed routes against the address ranges defined by the interface-specific security policy, and by interface-specific traffic selectors. if any have been assigned to the interface. Refer to the traffic-selectors ACLI command in IKEv2 Interface Configuration in this chapter for additional details.

TSr

is a responder Traffic Selector payload as defined in Section 3.12 of RFC 4306. TSr works in conjunction with the TSi payload to provide selection criteria for packets that are carried over this SA.The TSr payload conveys proposed destination/source addresses of traffic forwarded *to* or *from* the SA responder. Proposed packet flows are defined in terms of IP protocol type, IP address type (IPv4/IPv6), and address:port number ranges. The payload can contain one or more ranges of contiguous IP addresses and port numbers. The TSr Traffic Selector payload can contain one or more ranges of contiguous IP addresses and port numbers. Negotiated ranges can either replicate the contents of the initiator's TSr, or contain a subset of the original proposal. Negotiated ranges are determined by evaluating the SA initiator's proposed routes against the address ranges defined by the interface-specific security policy, and by interface-specific traffic selectors,

if any have been assigned to the interface. Refer to the traffic-selectors ACLI command in IKEv2 Interface Configuration in this chapter for additional details.

}

indicates the end of encryption

At the completion of IKE_AUTH, peers are connected by an established IPsec tunnel.

CREATE_CHILD_SA

After completion of the initial IKE_SA_INIT and IKE_AUTH exchanges, the Security Gateway will, at the request of the initiator, engage in a single CREATE_CHILD_SA exchange, resulting in the establishment of a second child SA. With two child SAs, one can be used for signalling and time-sensitive traffic while the second, for example, can be dedicated to billing or accounting traffic.

Initiator --------> **HDR { SA, Ni, {KEi], [CP], TSi, TSr }**

HDR

is the common IKEv2 header

{

indicates the beginning of encryption

SA

is the Security Association payload proposed by the initiator for the second child SA. This payload contains the initiator's ranked preferences for encryption algorithms, pseudo-random functions, authentication algorithms, Diffie-Hellman groups (groups 1, 2, 5, and 14 are supported), and ESN support.

Ni

is the initiator's Nonce payload (defined in Section 3.9 of RFC 4306). It contains the initiator's nonce, a randomly generated value at least 128-bits in length; the nonce value is input into cryptological functions to increase randomness.

KEi

is the initiator's optional Key Exchange payload (defined in Section 3.3 of RFC 4306). In the absence of the Key Exchange payload crypto material negotiated during the initial IKE_SA_INIT and IKE_AUTH exchanges are used for creation of the second child SA.

[CP]

is the optional Configuration payload. The initiator uses the Configuration payload to request an IP address on the Security Gateway's network. Procuring such a *local* IP address ensures that traffic returning to the initiator is routed to the Security Gateway, and then tunneled back to the initiator. In the absence of a Configuration payload, the IP address assigned during the IKE_AUTH exchange is used by the second child SA.

TSi

is an initiator Traffic Selector payload as defined in Section 3.12 of RFC 4306. TSi works in conjunction with the TSr payload to convey proposed selection criteria for packets that are carried over this child SA. The TSi payload contains proposed source/destination addresses of traffic forwarded *from* or *to* the child SA initiator. Proposed packet flows are defined in terms of IP protocol type, IP address type (IPv4/IPv6), and address:port number ranges. The payload can contain one or more ranges of contiguous IP addresses and port numbers.

TSr

is a responder Traffic Selector payload as defined in Section 3.12 of RFC 4306. TSr works in conjunction with the TSi payload to convey proposed selection criteria for packets that are carried over this child SA.The TSr payload contains proposed destination/source addresses of traffic forwarded *to* or *from* the child SA responder. Proposed packet flows are defined in terms of IP protocol type, IP address type (IPv4/IPv6), and address:port number ranges. The payload can contain one or more ranges of contiguous IP addresses and port numbers.

}

indicates the end of encryption

HDR { SA, Nr, [KEr], [CP], TSi, TSr } <- - - - - - - - - Responder

HDR

is the common IKEv2 header

{

indicates the beginning of encryption

SA

is the responder's Security Association payload that specifies cryptological material used for creation of the second child SA. It selects from the proposals offered in the initiator's corresponding payload.

Nr

is the Security Gateway's Nonce payload (defined in Section 3.9 of RFC 4306), a randomly generated value at least 128-bits in length; the nonce value is input into cryptological functions to increase randomness.

KEr

is the Security Gateway's optional Key Exchange payload (defined in Section 3.3 of RFC 4306). In the absence of the Key Exchange payload crypto material negotiated during the initial IKE_SA_INIT and IKE_AUTH exchanges are used for creation of the second child SA.

[CP]

is the optional Configuration payload. Assuming that the initiator has requested an IP address on the Security Gateway's network, the Configuration payload provides the required address, which can be assigned from a local address pool or provided by a RADIUS server.

TSi

is an initiator Traffic Selector payload as defined in Section 3.12 of RFC 4306. TSi works in conjunction with the TSr payload to convey negotiated selection criteria for packets that are carried over this child SA. The TSi payload contains negotiated source/destination addresses of traffic forwarded *from* or *to* the SA initiator. Proposed packet flows are defined in terms of IP protocol type, IP address type (IPv4/IPv6), and address:port number ranges. The TSi Traffic Selector payload can contain one or more ranges of contiguous IP addresses and port numbers. Negotiated ranges can either replicate the contents of the initiator's TS1, or contain a subset of the original proposal. Negotiated ranges are determined by evaluating the SA initiator's proposed routes against the address ranges defined by the interface-specific security policy, and by interface-specific traffic selectors. if any have been assigned to the interface. Refer to the traffic-selectors ACLI command in IKEv2 Interface Configuration in this chapter for additional details.

TSr

is a responder Traffic Selector payload as defined in Section 3.12 of RFC 4306. TSr works in conjunction with the TSi payload to provide selection criteria for packets that are carried over this child SA.The TSr payload conveys proposed destination/source addresses of traffic forwarded *to* or *from* the SA responder. Proposed packet flows are defined in terms of IP protocol type, IP address type (IPv4/IPv6), and address:port number ranges. The payload can contain one or more ranges of contiguous IP addresses and port numbers. The TSr Traffic Selector payload can contain one or more ranges of contiguous IP addresses and port numbers. Negotiated ranges can either replicate the contents of the initiator's TSr, or contain a subset of the original proposal. Negotiated ranges are determined by evaluating the SA initiator's proposed routes against the address ranges defined by the interface-specific security policy, and by interface-specific traffic selectors. if any have been assigned to the interface. Refer to the traffic-selectors ACLI command in IKEv2 Interface Configuration in this chapter for additional details.

}

indicates the end of encryption

EAP-based Authentication

RFC 3748, *Extensible Authentication Protocol (EAP)* describes a flexible and extensible framework that enables authentication services. While the RFC itself describes only a single authentication method, *MD5-Challenge*, the provided framework supports numerous authentication methods.

The Security Gateway currently supports four EAP-based authentication methods. Each supported method is described in the following sections. For all supported EAP authentication methods the actual authentication is performed by an associated RADIUS authentication server. During the EAP-based authentication exchange the Security Gateway functions as a relay between the authenticating client and the RADIUS server.

EAP-MD5 Authentication

EAP-MD5 is based on RFC 1994, *PPP Challenge Handshake Authentication Protocol (CHAP)*. This RFC describes an authentication method that uses an agreed-upon hashing algorithm, a random challenge value, and a shared secret known only to the authenticator and the EAP peer. In the case of EAP-MD5 the hashing algorithm, which produces a 128-bit message-digest or fingerprint, is described in RFC 1321, *The MD5 Message-Digest Algorithm*.

Using EAP-MD5, authentication of the EAP peer is accomplished as follows.

1. The authenticator issues a *Challenge* packet, which contains, among other fields, an *Identifier* field that serves to correlate message exchanges, and a *Data* field that contains an arbitrary challenge string.
2. The peer concatenates the contents of the *Identifier* field, the shared-secret, and the challenge string. The peer inputs the concatenated string to the MD5 hash function, computes the 128-bit fingerprint, and returns that value to the authenticator in a *Response* packet.
3. The authenticator performs the same calculation, and compares its results with those reported by the EAP peer.
4. .If the fingerprints are identical, the authenticator issues a *Success* packet; otherwise the authenticator issues a *Failure* packet.

Note that EAP-MD5 does not provide for mutual authentication; the authenticator does not authenticate to the EAP peer.

EAP-MSCHAPv2 Authentication

EAP-MSCHAPv2 is based on RFC 2759, *Microsoft PPP CHAP Extensions, Version 2*. This RFC describes an authentication method that uses a user-name and password model in conjunction with Microsoft encryption routines.

Using EAP-MSCHAPv2, mutual authentication of the EAP peer and authenticator is accomplished as follows.

1. The authenticator issues a *Challenge* packet, which contains, among other fields, an *Identifier* field that serves to correlate message exchanges, and a *Data* field that contains an arbitrary 16-octet challenge string.
2. The peer returns a *Response* packet that includes the user name, a newly-generated 16-octet challenge for the authenticator, and a one-way encryption of the received challenge string, the generated challenge string, the contents of the *Identifier* field, and the user password.
3. The authenticator performs the same calculation as was performed by the EAP peer, and compares its results with those reported by the peer. If the results are identical, the authenticator issues a *Success* packet which also contains a one-way encryption of the authenticator-originated challenge string, the peer-originated challenge string, the encrypted string received from the peer in the *Response* packet, and the user password.
4. The EAP peer performs the same calculation as was performed by the authenticator, and compares its results with those reported by the authenticator. If the results are identical, the peer uses the mutually authenticated connection; otherwise, it drops the connection.

EAP-AKA Authentication

The Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA) was devised by the 3GPP (3rd Generation Partnership Project), and made available to the Internet community in RFC 4187. EAP-AKA makes use of the Universal Subscriber Identity Module (USIM), an application resident on the smart card inserted in a 3G mobile phone. The USIM has access to authentication data stored on the smart card.

EAP-SIM Authentication

The EAP-SIM Protocol specifies an authentication method for GSM (Global System for Mobile Communication) subscribers. GSM is a second generation mobile standard, and still the most widely used. The authentication method is described in RFC 4186, Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identify Modules (EAP-SIM). Originally developed by the 3GPP, the EAP-SIM protocol specifies an EAP method for authentication and session key distribution using the GSM Subscriber Identity Module (SIM), a smart card installed in the GSM phone.

Multiple Authentication

The Security Gateway supports multiple authentication exchanges during IKEv2 negotiation. These exchanges are defined in RFC 4739, *Multiple Authentication Exchanges in the Internet Key Exchange (IKEv2) Protocol*. Multiple authentication enables the Security Gateway to engage in an initial certificate-based or shared-secret-based authentication with a remote IKEv2 peer (for example, a femtocell), followed by a subsequent EAP-AKA or EAP-SIM authentication of the remote mobile subscriber.

Multiple authentication exchanges require the use of two specific Notify payloads, MULTIPLE_AUTH_SUPPORTED and ANOTHER_AUTH_FOLLOWS (Notify message type s16404 and 16405) defined in Sections 3.1 and 3.2 of RFC 4739.

Message exchange is as follows.

Initiator (IKEv2 peer) Responder

```
1. HDR, SAi1, KEi, Ni --->
2.    <--- HDR, SAr1, KEr, Nr, CERTREQ, N (MULTIPLE_AUTH_SUPPORTED)
3. HDR, { IDi, CERT, CERTREQ, {IDr], AUTH, SAi2, TSi, TS2
   (MULTIPLE_AUTH_SUPPORTED) N (ANOTHER_AUTH_FOLLOWS) } --->
4.                                        <--- HDR, { IDr, CERT, AUTH }
5. HDR, { IDi } --->
6.                                           <--- HDR, { EAP (Request)}
7. HDR, { EAP (Response) } --->
8.                                           <--- HDR, { EAP (Request)}
9. HDR, { EAP (Response) } --->
10.                                          <--- HDR, { EAP (Success)}
11. HDR, { AUTH } --->
12.                                 <--- HDR, { AUTH, SAr2, TSi, TSr }
```

In Step 2 the responder advertises support for multiple authentication via the MULTIPLE_AUTH_SUPPORTED Notification Payload.

In Step 3 the initiator advertises support for multiple authentication and, using the ANOTHER_AUTH_FOLLOWS Notification Payload, signals its readiness for such authentication.

Step 4 completes mutual certificate authentication.

In Step 5 the initiator discloses its identity.

In Step 6 the responder initiates the EAP process

In Steps 7 and 8 the initiator and responder exchange authentication information for the remote peer.

In Steps 9 and 10 the initiator and responder exchange authentication information for the mobile subscriber.

In Steps 11 and 12 report successful authentication.

Log/Trap Suppression

Authentication failures that occur during IPsec tunnel establishment generate both a log entry and an SNMP trap. When using any of the previously described EAP authentication methods, users can suppress these entries and traps.

Note: Suppression of log entries and SNMP traps is available **only** when the authentication method is EAP based.

Only those log entries and SNMP traps generated by a authentication failure during the establishment of an IPsec tunnel are suppressed.

Authentication failures that may occur prior to IPsec tunnel establishment are not subject to suppression.

Use the following procedure to disable the generation of log entries and traps when an authentication failure occurs during IPsec tunnel establishment.

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

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# auth-params
   ACMEPACKET(auth-params)#
   ```

2. Use the **name** parameter to provide a unique name for this *auth-params* configuration object.

   ```
   ACMEPACKET(auth-params)# name suppressLogsSNMP
   ACMEPACKET(auth-params)#
   ```

3. Use the **protocol** parameter to specify EAP as the authentication protocol.

   ```
   ACMEPACKET(auth-params)# protocol eap
   ACMEPACKET(auth-params)#
   ```

4. Use the **servers** attribute to identify previously configured RADIUS authentication servers that provide EAP-based authentication.

   Use double quotes (“) to bracket lists which contain multiple entries.

   ```
   ACMEPACKET(auth-params)# servers 172.30.0.10
   ACMEPACKET(auth-params)#

   ACMEPACKET(auth-params)# servers “172.30.0.10 172.30.0.150”
   ACMEPACKET(auth-params)#
   ```

5. Based on network preferences, use the **strategy** attribute to identify the method used to access multiple RADIUS authentication servers.

   Use double quotes (") to bracket lists which contain multiple entries.

   ```
   ACMEPACKET(auth-params)# strategy hunt
   ACMEPACKET(auth-params)#
   ```

6. Use the **options** attribute to disable log entries and SNMP traps that report authentication failure during IPsec tunnel establishment.

   ```
   ACMEPACKET(auth-params)# options + suppress-auth-error-notifications
   ACMEPACKET(auth-params)#
   ```

7. Ignore the **authorization-servers** attribute.
8. Use the **done, exit, and verify-config** to complete configuration of this *auth-params* configuration object.
9. From superuser mode, use the following command sequence to access *authentication* configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# authentication
   ACMEPACKET(authentication)#
   ```

10. Use the **ike-radius-params-name** attribute to activate suppression of log entries and SNMP traps that report authentication failure during IPsec tunnel establishment.

    ```
    ACMEPACKET(authentication)# ike-radius-params-name suppressLogsSNMP
    ACMEPACKET(authentication)#
    ```

11. Use the **done, exit, and verify-config** to complete activating log entry and SNMP trap suppression.

IKEv2 Interface Configuration

Use the following procedure to configure IKEv2 interfaces.

1. From *superuser* mode, use the following command sequence to access *ike-interface* configuration mode. While in this mode, you configure IKEv2 interface parameters.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-interface
   ACMEPACKET(ike-interface)#
   ```

2. Use the **state** parameter to enable or disable the IKEv2 interface.

   Use *enabled* (the default value) to place a newly-created IKEv2 interface in immediate service, or to restore a currently disabled IKEv2 to service.

   Use *disabled* to remove an IKEv2 interface from service. When an IKEv2 interface is disabled, any tunnels associated with the interface are torn down, and new tunnels cannot be established.

   The following command removes the current IKEv2 interface from service.

   ```
   ACMEPACKET(ike-interface)# state disabled
   ACMEPACKET(ike-interface)#
   ```

The following command restores the current IKEv2 interface to service.

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

3. Use the **address** parameter to specify the IPv4 address of the interface.

```
ACMEPACKET(ike-interface)# address 192.169.204.15
ACMEPACKET(ike-interface)#
```

4. Use the **realm-id** parameter to specify the realm that contains the IP address assigned to this IKEv2 interface.

```
ACMEPACKET(ike-interface)# realm-id access-10
ACMEPACKET(ike-interface)#
```

5. Retain the default value (*responder*) for the **ike-mode** parameter, indicating that the Security Gateway always acts as an IKEv2 responder — IKEv2 negotiation is always initiated by the remote peer.

```
ACMEPACKET(ike-interface)# ike-mode responder
ACMEPACKET(ike-interface)#
```

6. Use the **sd-authentication-method** parameter to select the interface-specific method used by IKEv2 peers to mutually authenticate one to the other.

   By default, this parameter inherits the value set at the IKEv2 global level. The global level can be over-ridden at the interface level.

   Two alternative authentication methods (each described in Section 2.15 of RFC 4306, *Internet Key Exchange (IKEv2) Protocol* are supported.

   shared-password — (the default) uses a pre-shared-secret to authenticate the remote IKEv2 peer.

   certificate — uses an X.509 certificate to authenticate the remote IKEv2 peer.

   If using multiple authentication, select the method used during the initial exchange to authentication the remote IKEv2 peer (for example, a femtocell). Refer to Multiple Authentication for specific details.

   sd-authentication-method can be safely ignored, if authentication utilizes the EAP-based authentication methods as defined in RFC 3748, *Extensible Authentication Profile (EAP)*.

```
ACMEPACKET(ike-interface)# sd-authentication-method shared-password
ACMEPACKET(ike-interface)#
```

7. If **sd-authentication-method** is *shared-password*, use the optional interface-specific **shared-password** parameter to assign the shared secret.

   The shared secret is a string of ACSII printable characters no longer than 255 characters (not displayed by the ACLI).

   By default, this parameter inherits the value set at the IKEv2 global level. The global level can be over-ridden at the interface level.

```
ACMEPACKET(ike-interface)# shared-password 123ffGGH65900tnojbt=+
ACMEPACKET(ike-interface)#
```

8. If **sd-authentication-method** is *certificate*, use the optional interface-specific **certificate-profile-id-list** parameter to identify the *ike-certificate-profile* configuration element or elements that contain identification and validation credentials required for certificate-based IKEv2 authentication.

   By default, this parameter inherits the value set at the IKEv2 global level. The global level can be over-ridden at the interface level.

   The assignment of multiple *ike-certificate-profile* configuration elements supports multiple IKEv2 identities per interface.

   Use **certificate-profile-id-list**, in conjunction with the name or names of existing *ike-certificate-profiles*, to assign profiles to the current IKEv2 interface.

   ```
   ACMEPACKET(ike-interface)# certificate-profile-id jojo.net
   ACMEPACKET(ike-interface)#
   ```

9. Use the **multiple-authentication** parameter to enable multiple authentication as defined in RFC 4739 on this IKEv2 interface.

   Refer to Multiple Authentication for specific details of multiple authentication processing and restrictions.

   By default, **multiple-authentication** is disabled.

   ```
   ACMEPACKET(ike-interface)# multiple-authentication enabled
   ACMEPACKET(ike-interface)#
   ```

10. Use the optional interface-specific **v2-ike-life-seconds** parameter to specify the lifetime (in seconds) for the IKEv2 SAs supported by this IKEv2 interface.

    By default, this parameter inherits the value set at the IKEv2 global level. The global level can be over-ridden at the interface level.

    Allowable values are within the range 1 through 999999999 (seconds) with a default of 86400 (24 hours).

    ```
    ACMEPACKET(ike-interface)# v2-ike-life-seconds 21600
    ACMEPACKET(ike-interface)#
    ```

11. Use the optional interface-specific **v2-ipsec-life-seconds** parameter to specify the lifetime (in seconds) for the IPsec SAs supported by this IKEv2 interface.

    By default, this parameter inherits the value set at the IKEv2 global level. The global level can be over-ridden at the interface level.

    Allowable values are within the range 1 through 999999999 (seconds) with a default of 28800 (8 hours).

    ```
    ACMEPACKET(ike-interface)# v2-ipsec-life-seconds 7200
    ACMEPACKET(ike-interface)#
    ```

12. Use the optional interface-specific **v2-rekey** parameter to enable or disable the automatic re-keying of expired IKEv2 or IPsec SAs on this IKEv2 interface.

    By default, this parameter inherits the value set at the IKEv2 global level. The global level can be over-ridden at the interface level.

    Allowable values are *disabled* (the default) or *enabled*.

    In the default state, the expiration of the **v2-ike-life-secs** or **v2-ipsec-life-secs** parameter results in the destruction of the IKEv2 or IPsec SA. With re-keying enabled, expiration of the **v2-ike-life-secs** or **v2-ipsec-life-secs** parameter triggers an IKEv2 or IPsec re-negotiation.

With automatic re-keying enabled, and with the global **dpd-time-interval** parameter set to a non-zero value, the Security Gateway retransmits the re-keying request if it does not receive a response from the remote IPsec peer within the interval specified by the *ike-config* **dpd-time-interval** parameter. The Security Gateway makes a maximum of three retransmission attempts before abandoning the re-keying effort.

```
ACMEPACKET(ike-interface)# v2-rekey enabled
ACMEPACKET(ike-interface)#
```

13. Use the optional **dpd-params-name** parameter to enable the Dead Peer Detection Protocol on this IKEv2 interface.

    The protocol is initially enabled by setting a non-zero value to the **dpd-time-interval** parameter during IKEv2 global configuration process. The protocol is enabled at the local level by assigning an existing *dpd-params* configuration element to this IKEv2 interface.

    Refer to Dead Peer Detection Protocol Configuration in this chapter for information on configuring *dpd-params* configuration elements.

    ```
    ACMEPACKET(ike-interface)# dpd-params-name ikeDPD
    ACMEPACKET(ike-interface)#
    ```

14. Use the optional **cert-status-check** parameter to enable certificate status checking using either the Online Certificate Status Profile (OCSP) or a local copy of a Certificate Revocation List.

    By default, certificate status checking is disabled.

    ```
    ACMEPACKET(ike-interface)# cert-status-check enabled
    ACMEPACKET(ike-interface)#
    ```

15. Use the optional **cert-status-profile-list** parameter to assign one or more *cert-status-profile* configuration elements to this IKEv2 interface.

    Each assigned *cert-status-profile* provides the information needed to access either an OCSP responder, or a CRL source

    Use **cert-status-profile-list**, in conjunction with the name or name of existing *cert-status-profiles*, to assign profiles to the current IKEv2 interface.

    Use quotation marks to assign multiple OCSP responders. The following sequence assigns three *cert-status-profiles*, *VerisignClass3Designate*, *Verisign-1*, and *Thawte-1* to the current IKEv2 interface.

    ```
    ACMEPACKET(ike-interface)# cert-status-profile-list
    “VerisignClass3Designate Verisign-1 Thawte-1”
    ACMEPACKET(ike-interface)#
    ```

16. Use the optional **threshold-crossing-alert-group-name** parameter to assign an existing Threshold Crossing Alert Group (TCA) to this IKEv2 interface.

    TCAs monitor MIB variables or counters, and generate SNMP traps when object values cross defined thresholds. TCAs specifically monitor:

    - IKEv2 failed Authentications
    - IPsec tunnel removals
    - Dead peer detections

    Refer to Threshold Crossing Alert Configuration in this chapter for information on configuring Threshold Crossing Alert Groups.

Use **threshold-crossing-alert-group-name**, in conjunction with the name of an existing Threshold Crossing Alert Group, to assign that group to the current IKEv2 interface.

```
ACMEPACKET(ike-interface)# threshold-crossing-alert-group-name ikeTCA
ACMEPACKET(ike-interface)#
```

17. Use the optional **access-control-name** parameter to assign an existing IMSI White List to this IKEv2 interface.

    This parameter is meaningful only when authentication uses a RADIUS server to implement the EAP-SIM method, and can otherwise be safely ignored. If authentication is performed using EAP-SIM, **access-control-name** enables the assignment of an IMSI White List to the current IKEv2 interface.

    The assigned IMSI list filters GSM subscriber identities through an allowed list of country and network codes, thus restricting RADIUS server access to specified IMSI prefixes. Refer to Configuring IMSI White Lists for information on configuring IMSI lists.

    Use **access-control-name**, in conjunction with the name of an existing IMSI White List, to assign that list to the current IKEv2 interface.

    ```
    ACMEPACKET(ike-interface)# access-control-name domestic-1
    ACMEPACKET(ike-interface)#
    ```

18. Use the optional interface-specific **addr-assignment** parameter to specify the method used to assign addresses in response to an IKEv2 Configuration Payload request.

    The Configuration payload supports the exchange of configuration information between IKEv2 peers. Typically, an IRAC (IPsec Remote Access Client) initiates the exchange by requesting an IP address on the gateway's protected network. In response, the gateway, referred to as an IRAS (IPsec Remote Access Server), returns a local address for the IRAC's use.

    By default, this parameter inherits the value set at the IKEv2 global level. The global level can be over-ridden at the interface level.

    Supported values are:

    local—(the default) use local address pool

    radius-only—obtain local address from RADIUS server

    radius-local —try RADIUS server first, then local address pool

    ```
    ACMEPACKET(ike-interface)# addr-assignment local
    ACMEPACKET(ike-interface)#
    ```

19. If **addr-assignment** is *local* or *radius-local*, you can use the interface-specific **local-address-pool-id-list** parameter to assign one or more address pools to the current interface.

    Local address pools provide a group of IP address that can be temporarily leased to remote endpoints who request an IP address on a Security Gateway subnet, and also specify DNS information sources made available to remote endpoints.

    Refer to Local Address Pool Configuration in IKEv2 Global Configuration for information on configuring Local Address Pools.

    During the IKE_AUTH exchange, the IKEv2 initiator (the remote endpoint) often requests an internal IP address from an IPsec responder (the Security Gateway). Refer to Section 2.19 of RFC 4306, *Internet Key Exchange (IKEv2) Protocol*, for a description of the request process. Procuring such a *local* IP address ensures that traffic returning to the endpoint is routed to the Security

Gateway, and then tunneled back to the endpoint. Local address pools provide the source of these addresses available for temporary endpoint assignment.

After address assignment from the local address pool, the endpoint retains rights to that address for the tunnel lifetime, which is terminated either by an INFORMATIONAL exchange as defined in Section 1.4 of RFC4306, or by expiration of the tunnel SAs as specified by the **v2-ike-life-seconds** and **v2-ipsec-life-seconds** parameters. In either case, a subsequent request for an assigned IP address results, in all likelihood, with the assignment of a new IP address. However, if the remote endpoint is prematurely terminated by, for example an unscheduled reset or re-boot, a subsequent request for an assigned IP address (assuming that SA timers have not expired) results in the assignment of the previously issued IP address.

Use **local-address-pool-id-list**, in conjunction with the name or names of existing address pools, to assign those address pools to the current IKEv2 interface.

```
ACMEPACKET(ike-interface)# local-address-pool-id-list ikePool
ACMEPACKET(ike-interface)#
```

20. Use the **ip-subnets** parameter to specify the response to a CFG_REQUEST message that contains an INTERNAL_IP4_SUBNET attribute.

    In certain Security Gateway environments, Unlicensed Mobile Access (UMA) handsets issue a CFG_REQUEST message that contains an INTERNAL_IP4_SUBNET attribute to request information regarding the networks served/protected by the Security Gateway. In response to such a request, the Security Gateway can be configured to return a list of networks that can be reached through the gateway.

    Note that any attribute value contained in the CFG_REQUEST is ignored by the Security Gateway.

    By default, **ip-subnets** is set to an empty string. With the default value in place, the Security Gateway responds to CFG_REQUEST/INTERNAL_IP4_SUBNET with a CFG-REPLY containing three INTERNAL_IP4_SUBNET variables providing the IP address and subnet mask for

    - the primary DNS server
    - the secondary DNS server
    - this IKEv2 interface

    The default behavior reflects pre-M-CX1.2.0F3 operations and ensures backwards compatibility with prior configuration images.

    Alternatively, you can assign a list of up to 12 subnets to this IKEv2 interface. Each list item consists of a network address and a bit mask in the form 192.168.2.0 2 55.255.255.0 or 192.168.2.0/24. Overlapping IP subnets are not allowed in the IP subnets list. The list, however, can contain the 0.0.0.0/0.0.0.0 empty address prefix that in practice matches all addresses. Note that **verify-config** issues a Warning message that notes the presence of the empty address prefix in the configured list.

    Because arbitrary subnets an be configured for **ip-subnets**, there are potential conflicts between the list contents and the Traffic Selectors conveyed in the Traffic Selector payload of the same message. The Security Gateway does not check the consistency of the Configuration and Traffic Selector payloads, under the assumption that the endpoint devices will use one or the other, but not both payloads to pass traffic to the Security Gateway.

Use the **ip-subnets** command to construct the ip-subnets list.

The list can contain a maximum of 12 entries, with each list entry specifying an INTERNAL_IP4_SUBNET attribute in a CFG_REPLY. Use double quotes (“) to delimit the list

```
ACMEPACKET(ike-interface)# ip-subnets “10.11.12.13 255.255.255.0
192.168.2.0/27”
ACMEPACKET(ike-interface)#
```

After defining the list you can use the **add-ip-subnet** and **remove-ip-subnet** parameters to edit the list. For example, this ACLI sequence adds a single list entry:

```
ACMEPACKET(ike-interface)# add-ip-subnet 9.100.120.0/24
ACMEPACKET(ike-interface)#
```

This ACLI sequence removes a single list entry:

```
ACMEPACKET(ike-interface)# remove-ip-subnet 10.11.12.13255.255.255.0
ACMEPACKET(ike-interface)#
```

21. Use the interface-specific **traffic-selectors** parameter to assign one or traffic selectors to this IKEv2 interface. These traffic selectors are used to determine the contents of responder TSi and TSr payloads sent by the Security Gateway to the IKE initiator during the IKE_AUTH and CREATE_CHILD_SA exchanges as described earlier in this chapter.

    You can configure a list of up to 10 traffic selectors which are assigned to this interface. Traffic selectors are used in conjunction with the security policy assigned to this interface to restrict the networks or hosts that can be reached by the IPsec endpoint peer.

    Each of the assigned traffic selectors must be a subset of the IP address ranges defined by the security policy. Often, although not always, security policies are configured with default addressing (0.0.0.0/0) which matches all valid IP addresses.

    In the absence of interface-specific traffic selectors (the default state), TSi and TSr payloads proposed by the IKE initiator are evaluated against the security policy only. After this evaluation, proposed network/host addresses can be rejected (in which case a TS_UNACCEPTABLE is returned to the IKE initiator), accepted (in which case the proposed values are returned to the IKE initiator), or narrowed (in which case, a subset of the returned values are returned to the IKE initiator).

    With interface-specific traffic selectors in place, processing proceeds as follows. Assume (1) the associated security policy uses default addressing, which allows all valid IP addresses, (2) the TSr proposed by the IKE initiator contains 192.168.0.0/16, and (3) configured interface-specific traffic selectors are 192.168.20.34 and 192.168.16.0/24.

    Evaluation of the IKE initiator’s proposal against the security policy allows the proposed subnet address range.

    Evaluation of the IKE initiator’s proposal against the interface-specific traffic selectors produces two matches.

    The Security Gateway returns a TSr payload contains 2 traffic selectors: a host address 192.168.20.34. and a subnet address 192.168.0.0/16.

    Use the **traffic-selectors** command to construct the traffic selectors list.

The list can contain a maximum of 10 entries, with each list entry specifying a supported address range. Each list entry identifies a supported host or subnet, and takes the form of an IP address followed by an optional subnet mask. The absence of a mask indicates that the address is to be interpreted as a host address. Use double quotes (“) to delimit the list

```
ACMEPACKET(ike-interface)# traffic-selectors “192.168.14.90
192.168.10.0/24”
ACMEPACKET(ike-interface)#
```

After defining the list you can use the **add-traffic-selectors** and **remove-traffic-selectors** parameters to edit the list. For example, this ACLI sequence adds a single list entry:

```
ACMEPACKET(ike-interface)# add-traffic-selector 192.168.15.0/24
ACMEPACKET(ike-interface)#
```

This ACLI sequence removes a single list entry:

```
ACMEPACKET(ike-interface)# remove-traffic-selector 192.168.0.0/24
ACMEPACKET(ike-interface)#
```

22. Retain the default value for the optional **eap-protocol** parameter.

    The default, the only currently-supported value, *eap-radius-passthru*, specifies the use of a RADIUS server for all EAP-based authentication. The Security Gateway shuttles EAP requests (issued by the authenticator, in this case the RADIUS server) and EAP responses (issued by the remote IKEv2 peer) between the remote peer and the RADIUS server.

    Authentication is established between the IKEv2 remote peer and the RADIUS server.

    The following EAP authentication methods are supported:

    MD5 — EAP Type 5

    EAP-SIM — EAP Type 18

    EAP-AKA — EAP Type 23

    EAP-MSCHAPv2 — EAP Type 29

    ```
    ACMEPACKET(ike-interface)# eap-protocol eap-radius-passthru
    ACMEPACKET(ike-interface)#
    ```

23. Use **done**, **exit**, and **verify-config** to complete configuration of IKEv2 interface.

24. Repeat Steps 1 through 23 to configure additional IKEv2 interfaces.

After configuring all IKEv2 interfaces, you next configure IPsec Security Policies for each interface. You will configure policies that open a single port for IKE protocol traffic, and that require IPsec encryption on all other ports.

Refer to the following section for procedural instructions.

You first define *ike-sainfo* elements that identify cryptographic material available for Security Association negotiation, and then define interface-specific IPsec Security Policies.

IPsec SA Configuration

During the IKE_AUTH exchange, cooperating peers use the secure channel previously established by the IKE_SA_INIT exchange to negotiate child IPsec SAs to construct secure end-to-end IPsec tunnels between the peers. IKE_SA_INIT negotiations use the values provided by the *ike-sainfo* configuration element.

Use the following procedure to create an *ike-sainfo* configuration element that specifies cryptographic material used for IPsec tunnel establishment. You will later assign this *ike-sainfo* configuration element to an IPsec Security Policy which defines IPsec services for a specified IKEv2 interface.

1. From superuser mode, use the following command sequence to access *ike-sainfo* configuration mode. While in this mode, you configure IPsec data SAs.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-sainfo
   ACMEPACKET(ike-sainfo)#
   ```

2. Use the required **name** parameter to provide a unique identifier for this *ike-sainfo* configuration element.

   name enables the creation and reuse of multiple *ike-sainfo* instances.

   ```
   ACMEPACKET(ike-sainfo)# name SA-1
   ACMEPACKET(ike-sainfo)#
   ```

3. Use the **security-protocol** parameter to specify the IPsec security (authentication and encryption) protocols supported by this SA.

   The following security protocols are available.

   Authentication Header (AH) — the default value — as defined by RFC 4302, *IP Authentication Header*, which provides authentication integrity to include the mutual identification of remote peers, non-repudiation of received traffic, detection of data that has been altered in transit, and detection of data that has been replayed, that is copied and then re-injected into the data stream at a later time. Authentication services utilize the authentication algorithm specified by the **auth-algo** parameter.

   Encapsulating Security Payload (ESP) as defined by RFC 4303, *IP Encapsulating Security Payload*, which provides both authentication and privacy services. Privacy services utilize the encryption algorithm

   ESP-AUTH (also RFC 4303-based), which supports ESP's optional authentication.

   ESP-NULL (also RFC 4303-based) which proves NULL encryption as described in RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*. This option provides no privacy services, and is not recommended for production environments.

   ```
   ACMEPACKET(ike-sainfo)# security-protocol esp
   ACMEPACKET(ike-sainfo)#
   ```

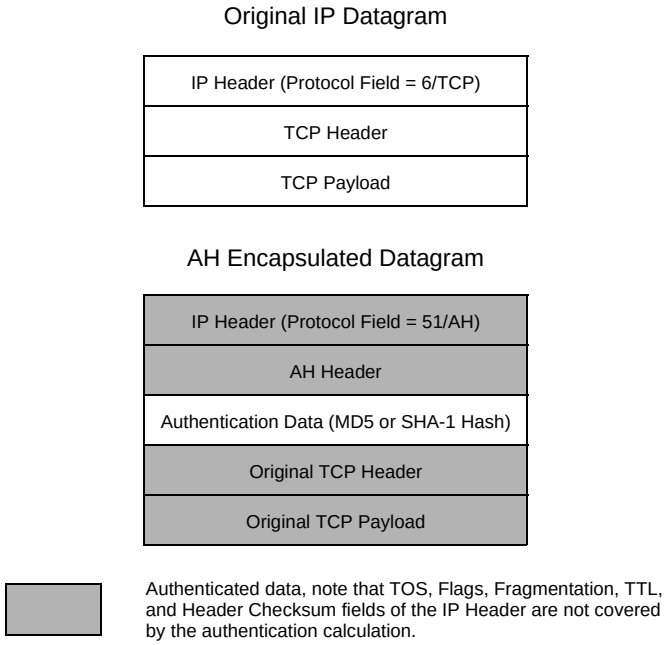


Figure 2: AH Transport Mode

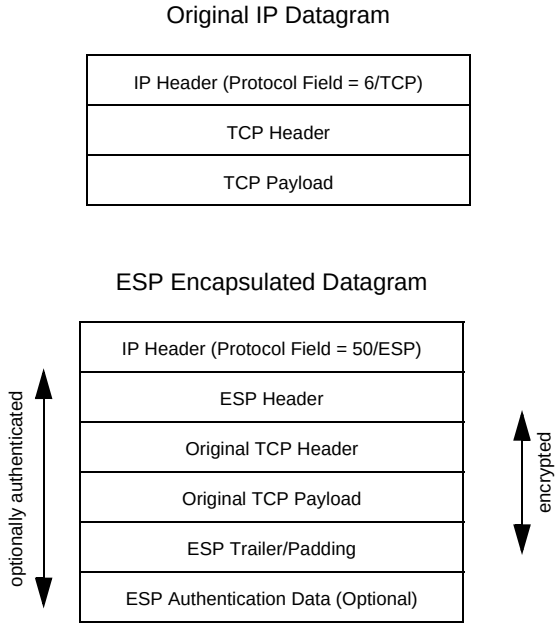


Figure 3: ESP Transport Mode

4. Use the **auth-algo** parameter to specify the authentication algorithms supported by this SA.

   The following authentication protocols are available:

   Advanced Encryption Standard-Extended Cipher Block Chaining (**aes-xcbc**) — as defined by RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec*.

   Message Digest Algorithm 5 (**md5**) — as defined by RFC 1321, *The MD5 Message-Digest Algorithm*.

   Secure Hash Algorithm (**sha**) — as defined by FIPS PUB 180-1, *Secure Hash Standard*.

   any (the default) — AES-XCBC, MD5 and SHA-1.

   ```
   ACMEPACKET(ike-sainfo)# auth-algo md5
   ACMEPACKET(ike-sainfo)#
   ```

5. Use the **encryption-algo** parameter to specify the encryption algorithms supported by this SA.

   The following encryption protocols are available:

   Advanced Encryption Standard (**aes**) — as defined by FIPS PUB 197, *Advanced Encryption Standard*.

   Advanced Encryption Standard (**aes-ctr**) — as defined by RFC 5930, Using *Advanced Encryption Standard Counter Mode (AES-CTR) with the Internet Key Exchange Version 2 (IKEv2)*.

   Triple DES (**3des**) — as defined by ANSI X.9.52 1998, *Triple Data Encryption Algorithm Modes of Operation*.

   NULL Encryption (null) — as described in RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*. This option provides no privacy services, and is not recommended for production environments.

   any (the default) — supports all listed encryption protocols.

```
ACMEPACKET(ike-sainfo)# encryption-algo aes
ACMEPACKET(ike-sainfo)#
```

6. Use the **ipsec-mode** parameter to specify the IPsec operational mode.

   Transport mode (the default) provides a secure end-to-end connection between two IP hosts. Transport mode encapsulates the IP payload.

   Tunnel mode provides VPN service where entire IP packets are encapsulated within an outer IP envelope and delivered from source (an IP host) to destination (generally a secure gateway) across an untrusted internet.

   ```
   ACMEPACKET(ike-sainfo)# ipsec-mode tunnel
   ACMEPACKET(ike-sainfo)#
   ```

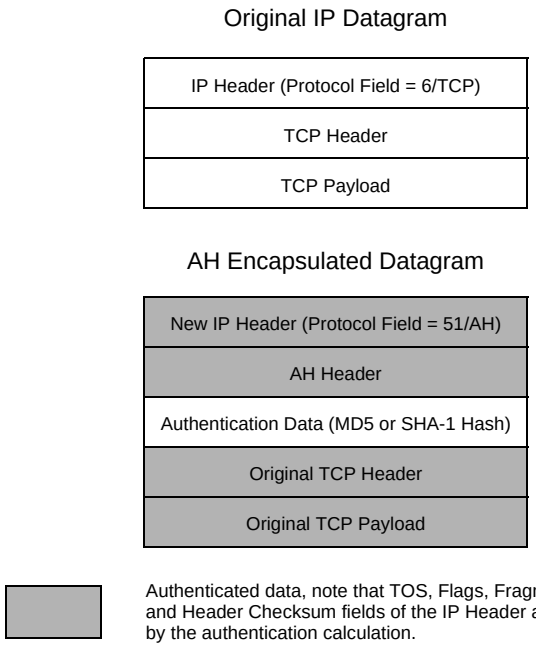


Figure 4: AH Tunnel Mode

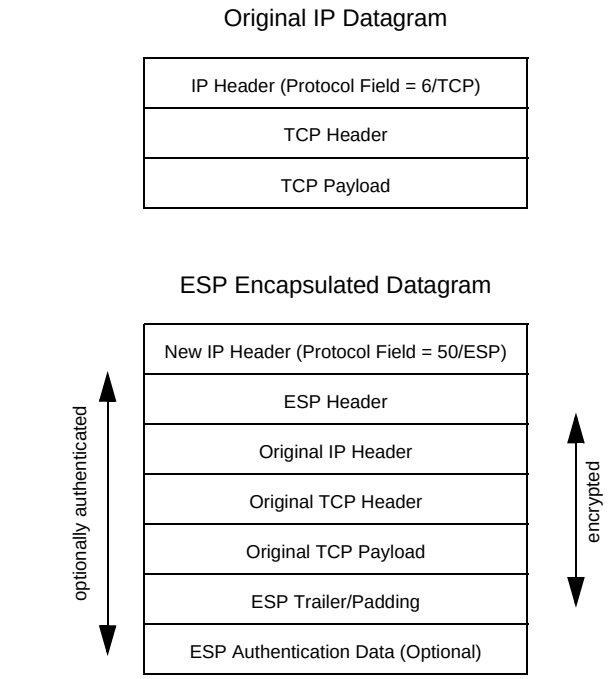


Figure 5: ESP Tunnel Mode

7. If **ipsec-mode** is *tunnel*, use the required **tunnel-local-addr** parameter to specify the IP address of the local IKEv2 interface that terminates the IPsec tunnel.

   This parameter can safely be ignored if **ipsec-mode** is *transport*.

   ```
   ACMEPACKET(ike-sainfo)# tunnel-local-addr 172.30.89.10
   ACMEPACKET(ike-sainfo)#
   ```

8. If **ipsec-mode** is *tunnel*, use the **tunnel-remote-addr** parameter to specify the IP address of the remote IKEv2 peer that terminates the IPsec tunnel.

   Provide the remote IP address, or use the default wild-card value (*) to match all IP addresses.

   This parameter can safely be ignored if **ipsec-mode** is *transport*.

   ```
   ACMEPACKET(ike-sainfo)# tunnel-remote-addr *
   ACMEPACKET(ike-sainfo)#
   ```

9. Use **done**, **exit**, and **verify-config** to complete configuration of the IPsec data SA.
10. If necessary, repeat Steps 1 through 9 to configure additional IPsec data SAs.

Security Policy Configuration

Use the following procedure to define an IPsec Security Policy.

1. From superuser mode, use the following command sequence to access *security-policy* configuration mode. While in this mode, you configure new security policies or modify existing policies.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ipsec
   ACMEPACKET(ipsec)# security-policy
   ACMEPACKET(security-policy)#
   ```

2. \Use the required **name** parameter to identify this IPsec Security Policy.

   ```
   ACMEPACKET(security-policy)# name requireIPsec
   ACMEPACKET(security-policy)#
   ```

3. Use the required **network-interface** parameter to provide the network interface name of the IKEv2 interface to which this security policy is applied.

   ```
   ACMEPACKET(security-policy)# network-interface M00:0
   ACMEPACKET(security-policy)#
   ```

4. Use the optional **priority** parameter to assign a priority to this IPsec Security Policy.

   Supported values are integers within the range *0* (the highest priority) through *254* (the lowest priority).

   You can assign more than one IPsec Security Policy to a specific interface. With multiple IPsec Security Policy assignments, each policy is applied in order of its priority (highest to lowest).

   ```
   ACMEPACKET(security-policy)# priority 0
   ACMEPACKET(security-policy)#
   ```

5. Use the optional **action** parameter to specify the processing of IPsec and non-IPsec traffic streams.

   Use **ipsec** to reject all traffic that is not IPsec-encrypted.

   Use **allow** to enable the processing of non-IPsec traffic.

   ```
   ACMEPACKET(security-policy)# action ip-sec
   ACMEPACKET(security-policy)#
   ```

6. Use the optional **direction** parameter to identity the traffic streams subject to the processing specified by the **action** parameter.

   Use **both** to apply the specified processing to both inbound and outbound traffic.

   ```
   ACMEPACKET(security-policy)# direction both
   ACMEPACKET(security-policy)#
   ```

7. Use the optional **local-ip-addr-match** to specify the local IP address of the network interface.

   Provide the local IP address or retain the default value, *0.0.0.0*, which matches all local IP addresses.

   ```
   ACMEPACKET(security-policy)# local-ip-addr-match 172.30.89.10
   ACMEPACKET(security-policy)#
   ```

8. Use the optional **remote-ip-addr-match** to specify the IP address of the remote IKEv2 peer.

   Provide the remote IP address or retain the default value, *0.0.0.0*, which matches all remote IP addresses.

   ```
   ACMEPACKET(security-policy)# remote-ip-addr-match 0.0.0.0
   ACMEPACKET(security-policy)#
   ```

9. Use the optional **local-port-match** to specify the local ports to which this IPsec Security applies.

   Use an integer within the range 1 through 65535 to specify a particular local port; use *0* to specify all local ports.

   ```
   ACMEPACKET(security-policy)# local-port-match 0
   ACMEPACKET(security-policy)#
   ```

10. Use the optional **remote-port-match** to specify the remote ports to which IPsec Security Policy applies.

    Use an integer within the range 1 through 65535 to specify a particular remote port; use *0* to specify all remote ports.

    ```
    ACMEPACKET(security-policy)# remote-port-match 0
    ACMEPACKET(security-policy)#
    ```

11. Retain the default value, *all*, for the **trans-protocol-match** parameter.
12. Retain the default value, *255.255.255.255*, for the **local-ip-mask** parameter.
13. Retain the default value, *255.255.255.255*, for the **remote-ip-mask** parameter.
14. Use the optional **ike-sainfo-name** parameter to assign an IPsec data SA to this Security Policy.

    ```
    ACMEPACKET(security-policy)# ike-sainfo-name SA-1
    ACMEPACKET(security-policy)#
    ```

15. Use **done**, **exit**, and **verify-config** to complete configuration of the IPsec Security Policy.
16. Repeat Steps 1 through 15 to configure additional IPsec Security Policies.

The following sample security policies support IKEv2 over the M00:0 network interface. The first policy (*ikev2Protocol*) opens port 500 for IKEv2 traffic, while the second policy (*requireIPsec*) mandates IPsec on all other ports, and assigns the *ike-sainfo* element, *star*, to that Security Policy in support of IPsec operations.

```
ACMEPACKET# show running-config security-policy
ACMEPACKET#
name                    ikev2Protocol
network-interface       M00:0
priority                0
local-ip-addr-match     172.30.55.127
remote-ip-addr-match    0.0.0.0
local-port-match        500
remote-port-match       500
trans-protocol-match    ALL
direction               both
local-ip-mask           255.255.255.255
remote-ip-mask          255.255.255.255
action                  allow
ike-sainfo-name
outbound-sa-fine-grained-mask
local-ip-mask           255.255.255.255
remote-ip-mask          255.255.255.255
local-port-mask         0
remote-port-mask        0
trans-protocol-mask     0
valid                   enabled
vlan-mask               0xFFF
last-modified-by        admin@console
last-modified-date      2010-08-08 19:06:32

security-policy
name                    requireIPsec
network-interface       M00:0
priority                1
local-ip-addr-match     172.30.55.127
remote-ip-addr-match    0.0.0.0
local-port-match        0
remote-port-match       0
trans-protocol-match    ALL
direction               both
local-ip-mask           255.255.255.255
remote-ip-mask          255.255.255.255
action                  ipsec
ike-sainfo-name         SA-1
outbound-sa-fine-grained-mask
local-ip-mask           255.255.255.255
remote-ip-mask          255.255.255.255
local-port-mask         0
remote-port-mask        0
trans-protocol-mask     0
valid                   enabled
vlan-mask               0xFFF
last-modified-by        admin@console
last-modified-date      2010-08-08 19:07:03
```

Dead Peer Detection Protocol Configuration

Ignore this section if you did not enable the Dead Peer Detection Protocol.

IKEv2 peers can lose connectivity unexpectedly, perhaps as a result of routing problems, or reboot of one of the peers. Neither IKEv2 nor IPsec offers an efficient and scalable method to respond to connectivity loss. As a result established SAs can remain in place until their configured lifetimes eventually expire. Such behavior results in mis-management of system resources and the presence of black holes where packets are tunneled to oblivion.

With the Dead Peer Detection Protocol (DPD) enabled, each peer's state is largely independent of the other's. A peer is free to request proof of connectivity when needed — there are no mandatory, periodic exchanges as would be required by a detection method based on keepalive or heartbeat messages. DPD asynchronous exchanges, consisting of a simple R-U-THERE and ACK, require fewer messages and achieve greater scalability.

If there is ongoing valid IPSec traffic between peers (Peer A and Peer B), there is little need to check connectivity. After a period of inactivity, however, connectivity is questionable. Verification of connectivity is only urgently necessary if there is traffic to be sent. For example, if Peer A has IPsec traffic to send after the period of idleness, it needs to know if Peer B is still alive. At this point, peer A can initiate the DPD exchange. Conversely, if Peer B has traffic to send, it to can initiate the DPD exchange

DPD is enabled during IKEv2 global configuration by setting the dpd-time-interval parameter to a non-zero value. After globally enabling DPD, you configure one or more dpd-params configuration elements. Each dpd-params configuration element consists of a set of parameter values that specify DPD operational behavior. You then complete DPD configuration by assigning a dpd-params configuration element to an IKEv2 interface.

Use the following procedure to configure DPD.

1. From *superuser* mode, use the following command sequence to access *dpd-params* configuration mode. While in this mode, you configure *dpd-params* configuration elements.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# dpd-params
   ACMEPACKET(dpd-params)#
   ```

2. Use the required **name** parameter to provide a unique identifier for this *dpd-params* instance.

   name enables the creation of multiple *dpd-params* instances.

   ```
   ACMEPACKET(dpd-params)# name ikeDPD
   ACMEPACKET(dpd-params)#
   ```

3. Use the **max-loop** parameter to specify the maximum number DPD peers whose liveliness is examined every **dpd-interval** period — the interval value is established during IKE global configuration.

   Periodic liveliness is tested by the Security Gateway issuing an R-U-THERE message to each peer in the current group. If the peer acknowledges receipt of the message, it is confirmed as alive. If the peer fails to respond, its status is determined by the max-retrans and max-attempts parameter values.

   If CPU workload surpasses the threshold set by max-cpu-limit, this value is over-ridden by load-max-loop.

   Allowable values are within the range 1 through 999999999 (endpoints) with a default of 100.

   ```
   ACMEPACKET(dpd-params)# max-loop 80
   ACMEPACKET(dpd-params)#
   ```

4. Use the **max-retrans** parameter to specify the maximum number of times that the Security Gateway, acting as a DPD initiator, retransmits an unacknowledged R-U-THERE message while performing periodic liveliness tests.

   Allowable values are within the range 0 through 4 (re-transmissions) with a default of 3.

   Assuming the default value, the Security Gateway sends 4 unacknowledged R-U-THERE messages before incrementing a peer-specific failure counter, and then moving to the next peer in the test group.

   ```
   ACMEPACKET(dpd-params)# max-retrans 2
   ACMEPACKET(dpd-params)#
   ```

5. Use the **max-attempts** parameter to specify the number of failed liveliness tests required to declare a peer as dead — thus taking down the IKE tunnel.

   Allowable values are within the range 1 through 999999999 (failed tests) with a default of 1.

   The default value specifies that a single failed liveliness test results in the peer being declared dead.

   Values greater than the default, specify that multiple liveliness failures are required to declare a peer's death. Users should keep in mind that with a large number of tunnels, the interval between liveliness tests can be significant.

```
ACMEPACKET(dpd-params)# max-attempts 1
ACMEPACKET(dpd-params)#
```

6. Use the **max-endpoints** parameter to specify the maximum number of simultaneous DPD protocol negotiations supported when the CPU is not under load (as specified by the **max-cpu-limit** property).

   If CPU workload surpasses the threshold set by max-cpu-limit, this value is over-ridden by load-max-endpoints.

   Allowable values are within the range 1 through 999999999 (endpoints) with a default of 25.

   ```
   ACMEPACKET(dpd-params)# max-endpoints 20
   ACMEPACKET(dpd-params)#
   ```

7. Use the **max-cpu-limit** parameter to specify a threshold value (expressed as a percentage of CPU capacity) at which DPD protocol operations are minimized to conserve CPU resources.

   Allowable values are within the range 0, which effectively disables DPD operations, through 100 (percent) with a default of 60.

   ```
   ACMEPACKET(dpd-params)# max-cpu-limit 50
   ACMEPACKET(dpd-params)#
   ```

8. Use the **load-max-loop** parameter to specify the maximum number of endpoints examined every **dpd-time-interval** when the CPU is under load, as specified by the **max-cpu-limit** parameter.

   Allowable values are within the range 1 through 999999999 (endpoints) with a default of *40*. Ensure that the configured value is less than the value assigned to **max-loop**.

   ```
   ACMEPACKET(dpd-params)# load-max-loop 30
   ACMEPACKET(dpd-params)#
   ```

9. Use the **load-max-endpoints** parameter to specify the maximum number of simultaneous DPD Protocol negotiations supported when the CPU is under load, as specified by the **max-cpu-limit** property.

   Allowable values are within the range 1 through 999999999 (endpoints) with a default of *5*. Ensure that the configured value is less than the value assigned to **max-endpoints**.

   ```
   ACMEPACKET(dpd-params)# load-max-endpoints 3
   ACMEPACKET(dpd-params)#
   ```

10. Use **done**, **exit**, and **verify-config** to complete configuration of the *dpd-params* configuration element.

11. If necessary, repeat Steps 1 through 10 to configure additional *dpd-params* configuration elements.

12. From *ike* mode, use the following command sequence to access *ike-interface* configuration mode. While in this mode, you configure IKEv2 interface parameters.

    ```
    ACMEPACKET(security)# ike
    ACMEPACKET(ike)# ike-interface
    ACMEPACKET(ike-interface)#
    ```

13. Use the optional **dpd-params-name** parameter to enable the Dead Peer Detection Protocol on this IKEv2 interface.

    The protocol is initially globally enabled by setting a non-zero value to the **dpd-time-interval** parameter during IKEv2 global configuration process. The protocol is enabled at the local level by assigning an existing *dpd-params* configuration element to this IKEv2 interface.

    ```
    ACMEPACKET(ike-interface)# dpd-params-name ikeDPD
    ACMEPACKET(ike-interface)#
    ```

14. Use **done**, **exit**, and **verify-config** to complete local configuration of the DPD protocol.

Certificate Revocation Lists

The IETF defines Certificate Revocation List (CRL) usage in RFC 3280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. A CRL contains a list of the serial numbers of certificates that have been revoked by the issuing Certification Authority (CA). Such issuing authorities update CRLs periodically, and make the updates lists available to subscribers.

RFC 3280 specifies the data exchanged between an HTTP client, in the current implementation the Security Gateway, and an HTTP server that provides a CRL source.

CRLs offer the convenience of local certificate status checking. Local status checking, however, requires periodic HTTP GETs of the freshest CRL from the CRL source, which can impose a transient but heavy CPU load.

When authentication of remote IKEv2 peers is certificate-based, you can enable CRL usage on IKEv2 interfaces to verify certificate status.

Configuring CRL-Based Certificate Verification

This section provides instruction on using the ACLI to configure periodic retrieval of CRLs.

Configuration of CRL-based certificate verification is a three-step process.

1. Specify the information and cryptological resources required to access one or more CRL sources.
2. If not already done, enable CRL usage on an IKEv2 interface.
3. Associate one or more CRLs with an IKEv2 interface.

ACLI Instructions

To configure CRL-based certificate status checking:

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

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

2. The **name** attribute differentiates *cert-status-profile* configuration elements one from another. Each *cert-status-profile* provides configuration information for a single, specific CRL source.
3. The **type** attribute selects the certificate revocation check methodology, either OCSP or Certificate Revocation List.

   Choose *crl* for CRL-based certificate status checking.
4. Retain the default value (*http*) for **trans-protocol** attribute, which identifies the transport method used to access the CRL source.

5. The **ip-address** attribute works in conjunction with the **port** attribute to locate the CRL source.

   ip-address identifies the CRL source by its IPv4 address. **port** identifies the port monitored by the HTTP server for incoming CRL requests.

   The **port** attribute can be safely ignored if the CRL source is specified as a FQDN by the **host-name** attribute, but is required if the CRL source is identified by the **ip-address** attribute.

   Allowable **port** values are integers within the range 1025 through 65535. In the absence of an explicitly configured value, the Security Gateway provides a default value of *80*, the well-known HTTP port.

6. Alternatively, use the **host-name** attribute to identify the CRL source.

   ip-address identifies the CRL source by its IPv4 address. **host-name** identifies the source by its fully-qualified domain name (FQDN).

   If you provide an IPv4 address, the Security Gateway uses that address and the value of the **port** attribute to access the CRL source. It you provide the FQDN, the Security Gateway uses DNS to resolve the source's address.

   If values are provided for both attributes, the Security Gateway uses the IP address and ignores the **host-name** value.

7. The **crl-list** attribute specifies the source filepath(s) to one or more requested CRLs.

   For example, assume the CRLs are stored in a top-level directory named *crl* at the source location.

   ```
   ACMEPACKET(cert-status-profile)# crl-list
   /crl/v2/tc_class_3_ca_II.crl
   ACMEPACKET#
   ```

   specifies a CRL at the source location */crl/v2/tc_class_3_ca_II.crl*.

   ```
   ACMEPACKET(cert-status-profile)# crl-list
   "/crl/v2/tc_class_3_ca_II.crl /crl/v2/tc_class_3_ca_II.crl"
   ACMEPACKET#
   ```

   specifies two CRLs at the specified locations.

   Note that the data provided by the **trans-protocol**, **host-name**, and **crl-list** attributes provides for the construction of an unambiguous URL for a requested CRL.

   For example, assuming a **host-id** of *www.trustcenter.de* and a source filepath of */crl/v2/tc_class_3_ca_II.crl*

   http://www.trustcenter.de/crl/v2/tc_class_3_ca_II.crl

8. The **realm-id** attribute specifies the realm used to request and receive CRLs.

   In the absence of an explicitly configured value, the Security Gateway provides a default value of *wancom0*, specifying CRL-related transmissions across the wancom0 management interface.

   If the CRL source is identified by its FQDN, the realm identified by **realm-id** must be DNS-enabled.

9. Ignore the **requester-cert** attribute.

   CRLs are requested with an HTTP GET method — this method does not support certificate-based authentication.

10. The **responder-cert** attribute identifies the certificate used to validate the received CRL — a public key of the CRL source.

    RFC 3280 requires the digital signature of all CRLs.

    Provide the name of the *certificate* configuration element that contains the certificate used to validate the signed CRL.

11. The **retry-count** attribute specifies the maximum number of times to retry an CRL source in the event of connection failure.

    If the retry counter specified by this attribute is exceeded, the Security Gateway contacts another CRL source (if multiple sources have been configured) and quarantines the unavailable source for a period defined the **dead-time** attribute.

    In the absence of an explicitly configured value (an integer within the range 0 through 10), the Security Gateway provides a default value of 1 (connection retries).

12. The **dead-time** attribute specifies the quarantine period imposed on an unavailable CRL source.

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

    Customer implementations utilizing a single CRL source are encouraged to retain the default value, or to specify a brief quarantine period to prevent lengthy service outages.

13. The **crl-update-interval** attribute specifies the interim in seconds, between CRL updates.

    In the absence of an explicitly configured value (an integer within the range 600 through 2600000), the Security Gateway provides a default value of *86400* (24 hours).

    CRLs are stored in the */code/crls* directory. Outdated, invalid CRLs are over-written with the each newly-obtained current CRL.

14. Use **done**, **exit**, and **verify-config** to complete configuration of this *cert-status-profile* instance.

    verify-config confirms (1) that the CRL source is identified by the **host-name** attribute, or by the combination of the **ip-address** and **port** attributes; (2) that the **responder-cert** attribute references an existing certificate configuration element; and, (3) that the **crl-list** attribute contains at least one entry.

15. Repeat Steps 1 through 14 to configure additional *cert-status-profile* configuration elements.

To enable CRL usage on an IKEv2 interface:

1. From *security* mode, use the following command sequence to access *ike-interface* configuration mode. While in this mode, you enable CRL-based certificate status checking on an IKEv2 interface. By default, certificate status checking is disabled on IKEv2 interfaces.

```
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-interface
ACMEPACKET(ike-interface)#
```

2. Use the **select** command to specify the specific IKEv2 interface on which CRL-based certificate status checking will be enabled.
3. Use the **cert-status-check** attribute to enable certificate status checking on this IKEv2 interface.

   Certificate status checking usage is enabled at the local level by setting this attribute to *enabled*.

To associate a CRL source with an IKEv2 interface:

4. Use the **cert-status-profile-list** attribute to assign a CRL source or sources to the IKEv2 interface.

   Use quotation marks to assign multiple CRL sources. The following sequence assigns three *cert-status-profiles*, *CRL-1VS*, *CRL2-VS*, and *CRL3-VS* to the current IKEv2 interface — each profile contains the data needed to identity a single CRL source.

```
ACMEPACKET(ike-interface)# cert-status-profile-list “CRL1-VS CRL2-VS
CRL3-VS”
ACMEPACKET(ike-interface)#
```

5. Use **done**, **exit**, and **verify-config** to complete CRL configuration on the currentIKEv2 interface.
6. Repeat Steps 1 through 20 to configure additional IKEv2 interfaces for CRL usage.
7. Use **done**, **exit**, and **verify-config** to complete CRL configuration.

Example CRL Configuration

A sample CRL configuration follows:

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# cert-status-profile
ACMEPACKET(cert-status-profile)# type crl
ACMEPACKET(cert-status-profile)# name CRL1-VS
ACMEPACKET(cert-status-profile)# host-name www.trustcenter.de
ACMEPACKET(cert-status-profile)# crl-list
/crl/v2/tc_class_3_ca_II.crl
ACMEPACKET(cert-status-profile)# responder-cert VS-CRL
ACMEPACKET(cert-status-profile)#
```

This configuration creates a *cert-status-profile* configuration object named *CRL1-VA*. The c*ert-status-profile* identifies an CRL source located at *www,trustcenter.com.de*. The source filepath to the requested CRL is */crl/v2/tc_class_3_ca_II.crl*. The required **responder-cert** attribute specifies the CA certificate used to verify the CRL obtained from the source. Retention of default values for remaining attributes specify default service via the wancom0 management interface.

```
ACMEPACKET(ike-interface)# cert-status-check enabled
ACMEPACKET(ike-interface)# cert-status-profile-list “CRL1-VS”
ACMEPACKET(ike-interface)#
```

This configuration uses the **cert-status-check** attribute to enable CRL usage on the current IKEv2 interface. The **cert-status-profile list** attribute associates a single *cert-status-profile*, *CRL1-VS*, to the IKEv2 interface, thus associating a single CRL source with the interface.

SNMP Traps

An SNMP trap is thrown, and a major alarm generated, if the Security Gateway is unable to retrieve a CRL from the server. This trap includes the server's FQDN, assuming that the FQDN has been identified during the configuration process, the server's IP address, the reason for the failure, and the time of the last successful CRL retrieval, with the time expressed as the number of seconds since midnight January 1, 1970.

A second SNMP trap is thrown when the Security Gateway successfully retrieves a CRL. This trap includes the server's FQDN, assuming that the FQDN has been identified during the configuration process, and the server's IP address. The issue of this trap also clears any associated major alarm.

Configuring Manual CRL Updates

The ACLI provides the ability to perform an immediate manual refresh of one or more CRLs.

Use the following command to refresh a single CRL.

```
ACMEPACKET# load-crl local-file <fileName>
```

where <*fileName*> is a remote filepath specified by the **crl-list** attribute.

Use the following command to refresh all CRLs.

```
ACMEPACKET# load-crl local-file all
```

Use the following command to refresh all CRLs from a specific CRL source.

```
ACMEPACKET# load-crl cert-status-profile <certStatusProfileName>
```

where <*certStatusProfileName*> references the certificate-status-profile configuration element that contains the CRL source IP address or FQDN.

Use the following command to refresh all CRLs.

```
ACMEPACKET# load-crl cert-status-profile all
```

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. Because OCSP ensures access to the freshest CRL, it can provide a more timely source of revocation information than is possible with dynamically or manually loaded CRLs. Guaranteed access to the most recent CRL, however, comes at the expense of increased protocol traffic — a single request/response exchange for each revocation check.

RFC 2560 specifies the data exchanged between an OCSP client, in the current implementation the Security Gateway, 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, which 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 interval.

revoked, which indicates a negative response to the status inquiry. The certificate has been revoked, either permanently or temporarily.

unknown, which indicates a negative response to the status inquiry. The responder cannot identify the certificate.

When authentication of remote IKEv2 peers is certificate-based, you can enable OCSP on IKEv2 interfaces to verify certificate status.

Configuring OCSP-Based Certificate Verification

The following sections provides instruction on using the ACLI to configure OCSP-based certificate verification.

Configuration of OCSP-based certificate verification is a three-step process.

1. Specify the information and cryptological resources required to access one or more OSCP responders.
2. Enable OCSP on an IKEv2 interface.
3. Associate one or more OCSP responders with an IKEv2 interface.

ACLI Instructions

To configure OCSP-based certificate status checking.

1. From *superuser* mode, use the following command sequence to access *cert-status-profile* configuration mode. While in this mode, you configure a *cert-status-profile* configuration element, 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. The **name** attribute differentiates *cert-status-profile* configuration elements one from another. Each *cert-status-profile* provides configuration information for a single, specific OCSP responder.

3. The **type** attribute selects the certificate revocation check methodology, either OCSP or Certificate Revocation List.

   Choose *ocsp* for OCSP-based certificate status checking.

4. Retain the default value (*http*) for trans-protocol attribute, which identifies the transport method used to access the OCSP responder.

5. The **ip-address** attribute works in conjunction with the **port** attribute to locate the OCSP responder.

   ip-address identifies the OCSP responder by its IPv4 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, the Security Gateway provides a default value of *80*, the well-known HTTP port.

6. The **realm-id** attribute specifies the realm used to transmit OCSP requests and receive OCSP responses.

   In the absence of an explicitly configured value, the Security Gateway provides a default value of *wancom0*, specifying OCSP protocol transmissions across the wancom0 management interface.

7. The **requester-cert** attribute is meaningful only if OCSP requests are signed; ignore this attribute 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 contains the certificate used to sign OCSP requests.

8. The **responder-cert** attribute 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 contains the certificate used to validate the signed OCSP response.

9. The **retry-count** attribute specifies the maximum number of times to retry an OCSP responder in the event of connection failure.

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

   In the absence of an explicitly configured value (an integer within the range 0 through 10), the Security Gateway provides a default value of 1 (connection retries).

10. The **dead-time** attribute specifies the quarantine period imposed on an unavailable OCSP responder.

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

    Customer implementations utilizing a single OCSP responder are encouraged to retain the default value, or to specify a brief quarantine period to prevent lengthy service outages.

11. Use **done**, **exit**, and **verify-config** to complete configuration of this *cert-status-profile* instance.

12. Repeat Steps 1 through 11 to configure additional *cert-status-profile* configuration elements.

To enable OCSP on an IKEv2 interface:

1. From *security* mode, use the following command sequence to access *ike-interface* configuration mode. While in this mode, you enable OCSP-based certificate status checking on an IKEv2 interface. By default, certificate status checking is disabled on IKEv2 interfaces.

   ```
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-interface
   ACMEPACKET(ike-interface)#
   ```

2. Use the **select** command to specify the specific IKEv2 interface on which OCSP-based certificate status checking will be enabled.

3. Use the **cert-status-check** attribute to enable certificate status checking on this IKEv2 interface.

   Certificate status checking usage is enabled at the local level by setting this attribute to *enabled*.

To associate an OCSP responder with an IKEv2 interface:

4. Use the **cert-status-profile-list** attribute to assign an OCSP responder of responders to the IKEv2 interface.

   Use quotation marks to assign multiple OCSP responders. The following sequence assigns three *cert-status-profiles*, *VerisignClass3Designate*, *Verisign-1*, and *Thawte-1* to the IKEv2 interface — each profile contains the data needed to identity a single OCSP responder.

   ```
   ACMEPACKET(ike-interface)# cert-status-profile-list
   “VerisignClass3Designate Verisign-1 Thawte-1”
   ACMEPACKET(ike-interface)#
   ```

5. Use **done**, **exit**, and **verify-config** to complete OCSP configuration on an IKEv2 interface.
6. Repeat Steps 1 through 5 to configure additional IKEv2 interfaces for OCSP operations.
7. Use **done**, **exit**, and **verify-config** to complete OCSP configuration.

Example OCSP Configuration

A sample OCSP configuration follows:

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# cert-status-profile
ACMEPACKET(cert-status-profile)# type ocsp
ACMEPACKET(cert-status-profile)# name VerisignClass3Designate
ACMEPACKET(cert-status-profile)# ip-address 192.168.7.100
ACMEPACKET(cert-status-profile)# port 8080
ACMEPACKET(cert-status-profile)# requester-cert VerisignClass3SignOCSP
ACMEPACKET(cert-status-profile)# responder-cert VerisignClass3ValOCSP
ACMEPACKET(cert-status-profile)# dead-time 60
ACMEPACKET(cert-status-profile)#
```

This configuration creates a *cert-status-profile* configuration object named *VerisignClass3Designate*. The c*ert-status-profile* identifies an OCSP responder located at 192.168.7.100:8080. The required **responder-cert** attribute specifies the CA certificate used to verify the signed OCSP response. The optional **requester-cert** attribute indicates that the OCSP responder requires signed responses and identifies the certificate used for digital signature. The optional **dead-time** attribute imposes a 60 second quarantine if the OCSP responder is unreachable. Retention of default values for the **realm-id** and **retry-count** attributes specify OCSP responder access via the wancom0 management interface and a retry count of 1.

```
ACMEPACKET(ike-interface)# cert-status-check enabled
ACMEPACKET(ike-interface)# cert-status-profile-list
“VerisignClass3Designate Thawte-1”
ACMEPACKET(ike-interface)#
```

This configuration uses the *cert-status-check* attribute to enable OCSP operations on the current IKEv2 interface. The cert-status-profile list attribute associates two *cert-status-profiles*, *VerisignClass3Designate* and *Thawte-1*, to the interface, with each profile identifying a specific OCSP responder.

SNMP Traps

An SNMP trap is thrown if a configured OSCP responder becomes unreachable.

A second SNMP trap is thrown when connectivity is re-established with a previously unreachable OCSP responder.

Configuring IMSI White Lists

Use the procedures described in this section only when authentication is performed by the EAP-SIM protocol. This section can be ignored when the Security Gateway employs any other authentication method.

EAP-SIM Protocol Overview

The EAP-SIM Protocol is described in RFC 4186, Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identify Modules (EAP-SIM). Originally developed by the 3GPP (3rd Generation Partnership Project), the EAP-SIM protocol provides for mutual authentication between the authenticator (a RADIUS server) and a GSM subscriber.

Within the EAP-SIM framework the GSM subscriber identifies itself with its International Mobile Subscriber Identity (IMSI), a digit string providing a globally unique identity for the subscriber's device. The IMSI is stored on a Subscriber Identity Module (SIM) installed in the GSM phone.

The IMSI is usually a 15-digit string that takes the following form:

<MCC><MNC><MSIN>

MCC (Mobile Country Code) prefix — 3 digits that uniquely identify the carrier's residence, not the subscriber's current location

MNC (Mobile Network Code) prefix — 2 or 3 digits that identify the carrier (the concatenation of the MCC and MNC prefixes provide unambiguous identification of the carrier network)

MSIN (Mobile Station Identification Number) — the remaining digits identify the specific device within the carrier's network

IMSI Prefix Filtering

With EAP-SIM protocol in use, authentication is accomplished by a RADIUS server. Using the Wm interface, the Security Gateway passes the received IMSI identity to the RADIUS server. In order to minimize server processing, the Security Gateway provides users with the optional ability to compile IMSI prefix white lists that filter identities presented for RADIUS authentication. White lists are inclusive in that only those identities matching list contents are granted RADIUS access; non-matching identities are summarily rejected by the Security Gateway. The white lists contain numeric strings or simple regular expressions that identify blocks of subscribers eligible for access to the RADIUS server.

IMSI prefix white lists consist of one or more identifiers. Identifiers are constructed using the digits 0 through 9 and the ^ wild-card character, which specifies any single digit. Each identifier specifies one or more blocks or subscribers eligible for authentication.

Sample identifiers are as follows:

744	matches the country of Paraguay
74401	matches a specific Paraguayan carrier (Hola Paraguay S.A.)
7440^	matches all current Paraguayan carriers (74401, 74402, 74404, and 74405)

IMSI Prefix White List Configuration

Use the following procedure to create an ***ike-access-control*** configuration element. This configuration element provides an IMSI prefix white list that filters IMSI identities presented by remote peers during the authentication process. Only those identities matching the literal or regular expressions contained within the IMSI prefix white list are forwarded via the Wm interface to a RADIUS server for authentication.

1. From superuser mode, use the following command sequence to access *ike-access-control* configuration mode. While in this mode, you configure IMSI prefix white lists.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-access-control
   ACMEPACKET(ike-access-control)#
   ```

2. Use the required **name** parameter to provide a unique identifier for this *ike-access-control* configuration element.

   name enables the creation and reuse of multiple *ike-access-control* instances.

   ```
   ACMEPACKET(ike-access-control)# name domestic-1
   ACMEPACKET(ike-access-control)#
   ```

3. Use the optional **state** parameter to specify the operational state of this *ike-access-control* configuration element.

   enabled — (the default) creates the configuration element in the enabled state.

   disabled — creates the configuration element in the disabled state.

   ```
   ACMEPACKET(ike-access-control)# state disabled
   ACMEPACKET(ike-access-control)#
   ```

4. Use the optional **identifier** parameter to provide one or more MCC or MCC/MNC match patterns.

   This identifier, a literal string, matches the Russian Federation.

   ```
   ACMEPACKET(ike-access-control)# identifier 250
   ACMEPACKET(ike-access-control)#
   ```

   This identifier, which uses the wildcard symbol (^) signifying any single digit within the range 0 through 9, matches the continental United States.

   ```
   ACMEPACKET(ike-access-control)# identifier 31^
   ACMEPACKET(ike-access-control)#
   ```

   This identifier, a double-quote delimited list of prefixes separated by spaces, matches T-Mobile United States networks.

   ```
   ACMEPACKET(ike-access-control)# identifier “26201 26206”
   ACMEPACKET(ike-access-control)#
   ```

   This identifier, a double-quote delimited list of prefixes separated by spaces, matches Verizon Wireless United States networks.

   ```
   ACMEPACKET(ike-access-control)# identifier “310004 310012”
   ACMEPACKET(ike-access-control)#
   ```

   Note: Do not configure an empty IMSI prefix white list— that is a list that lacks any identifier parameters. Assigning an empty white list to an IKEv2 interface results in authentication failure for all presented identities.

5. Use **done**, **exit**, and **verify-config** to complete configuration of the *ike-access-control* configuration element.
6. If necessary, repeat Steps 1 through 5 to configure additional *ike-access-control* configuration elements.

After completing the prefix list, use the following procedure to assign it to an IKEv2 interface.

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

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-interface
ACMEPACKET(ike-interface)#
```

2. Use the **select** command to identify the target IKEv2 interface, the interface to which the IMSI prefix white list will be assigned.
3. Use the optional **access-control-name** parameter to identify the IMSI prefix white list assigned to the current interface.

```
ACMEPACKET(ike-interface)# access-control-name domestic-1
ACMEPACKET(ike-interface)#
```

4. Use **done**, **exit**, and **verify-config** to complete assignment of the IMSI prefix white list to the current IKEv2 interface.

Threshold Crossing Alert Configuration

Threshold Crossing Alerts (TCAs) monitor specific MIB variables or counters, and generate SNMP traps when object values cross defined thresholds. Three types of TCAs are supported:

- IKE Failed Authentication (monitors IKE negotiation counters)
- IPsec Tunnel Removal (monitors IPsec tunnel counters)
- Dead Peer Detections (monitors DPD protocol counters)

Threshold levels, listed in order of increasing importance are clear, minor, major, and critical. Each threshold level is user-configurable and is accompanied by a associated reset-counter, also user-configurable, which prevents the issue of extraneous SNMP traps when a counter is bouncing across threshold values.

A threshold crossing event occurs when the associated counter value rises above the next-highest threshold value, or when the associated counter value falls below the next-lowest reset-threshold value. An SNMP trap, raising the alert level, is generated as soon as the counter value exceeds the next-highest threshold. An SNMP trap, lowering the alert level, occurs only during a check period when the TCA examines all counter values. Such check periods occur at 100 second intervals.

The following scenario illustrates TCA operations. The sample TCA, ike-tca-group, monitors the count of dead IKEv2 peers. Threshold and reset values are shown. A minor alarm threshold and its associated reset threshold have not been configured.

```
nameike-tca-group
tca-typeike-dpd
critical100
reset-critical90
major80
reset-major50
minor0
reset-minor0
```

t=time

t=0	ike-dpd counter= 30	ike-dpd alert level=clear	
t=1	ike-dpd counter= 60	ike-dpd alert level=clear	
t=2	ike-dpd counter= 80	ike-dpd alert level=major	trap sent
t=3	ike-dpd counter= 95	ike-dpd alert level=major	
t=4	ike-dpd counter=100	ike-dpd alert level=critical	trap sent
t=5	ike-dpd counter=120	ike-dpd alert level=critical	
t=6	ike-dpd counter= 99	ike-dpd alert level=critical	
t=7	ike-dpd counter= 90	ike-dpd alert level=major	trap sent
t=8	ike-dpd counter= 60	ike-dpd alert level=major	
t=9	ike-dpd counter= 0	ike-dpd alert level=clear	trap sent

Use the following procedure to configure TCAs.

1. From *superuser* mode, use the following command sequence to access *threshold-crossing-alert-group* configuration mode. While in this mode, you configure *threshold-crossing-alert-group* configuration elements.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# system
   ACMEPACKET(system)# threshold-crossing-alert-group
   ACMEPACKET(threshold-crossing-alert-group)#
   ```

2. Use the **name** parameter to provide a unique identifier for this *threshold-crossing-alert-group* instance.

   name enables the creation of multiple *threshold-crossing-alert-group* instances.

   ```
   ACMEPACKET(threshold-crossing-alert-group)# name ikeTCA
   ACMEPACKET(threshold-crossing-alert-group)#
   ```

3. Use the **threshold-crossing-alert** parameter to enter *threshold-crossing-alert* configuration mode. While in this mode, you create specific TCA types and associated values.

   ```
   ACMEPACKET(threshold-crossing-alert-group)# threshold-crossing-alert
   ACMEPACKET(threshold-crossing-alert)#
   ```

4. Use the **type** parameter to specify the TCA type.

   Supported values are:

   ike-failed-auth — (the default) tracks authentication failures

   ipsec-tunnel-removal — tracks the destruction of IPsec tunnels

   ike-dpd — tracks the detection of dead DPD peers

   ```
   ACMEPACKET(threshold-crossing-alert)# type ike-dpd
   ACMEPACKET(threshold-crossing-alert)#
   ```

5. Use the **critical** parameter to specify the critical threshold level.

   The default value (*0*) indicates that the threshold is not configured.

   ```
   ACMEPACKET(threshold-crossing-alert)# critical 100
   ACMEPACKET(threshold-crossing-alert)#
   ```

6. Use the **reset-critical** parameter to specify the value at which the critical level is replaced with the next lowest configured threshold level (major, minor, or clear, depending on configuration values).

   The default value (*0*) indicates that the threshold is not configured.

   ```
   ACMEPACKET(threshold-crossing-alert)# reset-critical 90
   ACMEPACKET(threshold-crossing-alert)#
   ```

7. Use the **major** parameter to specify the major threshold level.

   The default value (*0*) indicates that the threshold is not configured.

   ```
   ACMEPACKET(threshold-crossing-alert)# major 80
   ACMEPACKET(threshold-crossing-alert)#
   ```

8. Use the **reset-major** parameter to specify the value at which the major level is replaced with the next lowest configured threshold level (minor or clear, depending on configuration values).

   The default value (*0*) indicates that the threshold is not configured.

   ```
   ACMEPACKET(threshold-crossing-alert)# reset-major 50
   ACMEPACKET(threshold-crossing-alert)#
   ```

9. Use the **minor** parameter to specify the minor threshold level.

   The default value (*0*) indicates that the threshold is not configured.

   ```
   ACMEPACKET(threshold-crossing-alert)# minor 0
   ACMEPACKET(threshold-crossing-alert)#
   ```

10. Use the **reset-minor** parameter to specify the value at which the minor level is replaced with the next lowest configured threshold level (clear).

    The default value (*0*) indicates that the threshold is not configured.

    ```
    ACMEPACKET(threshold-crossing-alert)# reset-minor 0
    ACMEPACKET(threshold-crossing-alert)#
    ```

11. If required, repeat Steps 4 through 10 to add other TCA types to the current *threshold-crossing-alert-group* configuration element.

    The *threshold-crossing-alert-group* configuration element can contain a maximum of three individual threshold-crossing-alerts, one of each supported type.

12. Use **done**, **exit**, and **verify-config** to complete configuration of the *threshold-crossing-alert-group* configuration element.

13. If necessary, repeat Steps 1 through 12 to configure additional *threshold-crossing-alert-group* configuration elements.

14. From *superuser* mode, use the following command sequence to access *ike-config* configuration mode. While in this mode, you configure IKEv2 interface parameters.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-interface
ACMEPACKET(ike-interface)#
```

15. Use the optional **threshold-crossing-alert-group-name** parameter to assign an existing *threshold-crossing-alert-group* configuration element to this IKEv2 interface.

```
ACMEPACKET(ike-interface)# threshold-crossing-alert-group-name ikeTCA
ACMEPACKET(ike-interface)#
```

16. Use **done**, **exit**, and **verify-config** to complete configuration of the TCA.

The following four sections provide details on available counters that gather usage and error data related to IKEv2/IPsec operations on the Security Gateway.

The first section, *IKEv2 Protocol Operations*, describes a series of 32-bit counters that report interface-specific data on various protocol transactions. Protocol operations counter values are available with SNMP, through the ACLI **show security ike statistics** command, and can also be obtained by subscription to the *ike_stats* HDR group.

The second section, *IKEv2 Negotiation Errors*, describes a series of 32-bit counters that report interface-specific errors encountered during IKEv2 negotiations. Negotiation errors counter values are also available with SNMP, through the ACLI **show security ike statistics** command, and can also be obtained by subscription to the *ike-stats* HDR group.

The third section, *RADIUS Protocol Operations*, describes a series of 32-bit counters that report RADIUS-server-specific data. RADIUS protocol operations counter values are also available with SNMP, through the ACLI **show radius** command, and can also be obtained by subscription to the *radius-stats* HDR group.

The final section, *Diameter Protocol Operations*, describes a series of 32-bit counters that report Diameter-server-specific data. Diameter protocol operations counter values are also available with SNMP, and can also be obtained by subscription to the *diameter-stats* HDR group.

IKEv2 Protocol Operations

Current Child SA Pairs

Description	The number of current child IPsec SA pairs on the interface. As each IPsec tunnel requires two unidirectional SAs, this number equals the current number of tunnels on the interface.
Type	gauge
Range	0 to $2^{32} - 1$
ACJI "show' command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.33 (X specifies the interface index)

Maximum Child SA Pairs

Description	The largest number of child IPsec SA pairs on the interface since this counter was last reset. As each IPsec tunnel requires a single SA pair, this value equates to the largest number of tunnels on the interface.
Type	gauge
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics

Last Reset Timestamp

Description	The time that this interface was last reset – expressed as a UNIX timestamp containing the number of seconds since January 1, 1970.
Type	UNIX timestamp
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics

Child SA Request

Description	The number of requests to add a child SA pair that were received on the interface. These requests include IPsec SA rekey requests.
Type	counter
Range	0 to $2^{32} - 1$.
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.1 (X specifies the interface index)

Child SA Success

Description	The number of requests to add a child SA pair that were successfully completed on the interface. These successes include new child elements created by IPsec SA rekeys.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.2 (X specifies the interface index)

Child SA Failure

Description	The number of requests to add a child SA pair that were not successfully completed on the interface. These failures include unsuccessful IPsec SA rekeys.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.3 (X specifies the interface index)

Child SA Delete Requests

Description	The number of requests to delete a child SA pair that were received on the interface. These requests include deletion requests associated with IPsec SA rekeys.
Type	counter
Range	0 to $2^{32} - 1$.
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.4 (X specifies the interface index)

Child SA Delete Success

Description	The number of requests to delete a child SA pair that were successfully completed on the interface. These successes include children deleted by IPsec SA rekeys.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.5 (X specifies the interface index)

Child SA Delete Failure

Description	The number of requests to delete a child SA pair that were not successfully completed on the interface. These failures include unsuccessful deletions associated IPsec SA rekeys.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.6 (X specifies the interface index)

Child SA Rekey

Description	The number of child IPsec rekey exchanges transacted on the interface.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.7 (X specifies the interface index)

Initial Child SA Establishment

Description	The number of initial child SA pair establishments, in other words, the number of successful IKE_AUTH exchanges transacted on the interface.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.8 (X specifies the interface index)

DPD Received Port Change

Description	The number of DPD messages received on the interface that contained a port change from the previously received message. The port change indicates that the IKEv2 has moved to another port, or that an intervening NAT device has changed port mapping. These actions do not impact SA functions.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.9 (X specifies the interface index)

DPD Received IP Change

Description	The number of DPD messages received on the interface that contained an IP address change from the previously received message.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.10 (X specifies the interface index)

DPD Response Received

Description	The number of DPD ACK responses received on the interface. An ACK is sent by an IKEv2 peer in response to an R-U-THERE issued by the SG. A successful R-U-THERE/ACK exchange establishes availability on the remote IKEv2 peer.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.11 (X specifies the interface index)

DPD Response Not Received

Description	The number of R-U-THERE messages transmitted on the interface that were not acknowledged within the DPD allowed interval.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.12 (X specifies the interface index)

DPD Received

Description	The number of all DPD protocol messages received on the interface.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.13 (X specifies the interface index)

DPD Retransmitted

Description	The number of R-U-THERE messages that were re-transmitted because the original R-U-THERE message was not acknowledged.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.14 (X specifies the interface index)

DPD Sent

Description	The number of R-U-THERE messages that were sent across the interface, to include retransmitals.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.15 (X specifies the interface index)

IKE SA Packets Sent

Description	The number of IKEv2 SA packets sent across the interface.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.16 (X specifies the interface index)

IKE SA Packets Received

Description	The number of IKEv2 SA packets received across the interface.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.17 (X specifies the interface index)

IKE SA Packets Dropped

Description	The number of IKEv2 SA packets dropped by the interface.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.18 (X specifies the interface index)

Authentication Failures

Description	The number of authentication failures that occurred after the purported identity of the remote IKEv2 peer was ascertained.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.19 (X specifies the interface index)

IKE Message Errors

Description	The number of otherwise uncharacterized IKEv2 message errors.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.20 (X specifies the interface index)

Authentication ID Errors

Description	The number of errors that occurred during the identification stage of the authentication process.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.21 (X specifies the interface index)

Certificate Status Requests

Description	The number of certificate status requests sent across the interface to an OCSP responder.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.22 (X specifies the interface index)

Certificate Status Success

Description	The total number of OCSP successes, that is the number of OCSP requests that generated a good status from an OCSP responder.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.23 (X specifies the interface index)

Certificate Status Fail

Description	The total number of OCSP failures, to include unacknowledged OCSP requests and those requests that generated a revoked or unknown response from an OCSP responder.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.24 (X specifies the interface index)

DDoS Sent

Description	The number of suspicious, and possibly malicious, endpoints reported by the interface-specific DDoS process (if configured as described in the IKEv2 DDoS Protection section of the Security Gateway Essentials guide).
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.25 (X specifies the interface index)

DDoS Received

Description	The number of suspicious, and possibly malicious, endpoints reported by statically provisioned deny lists (as described in SIP Signaling Services and Security chapters of the ACLI Configuration Guide).
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.26 (X specifies the interface index)

IKE Message Retransmissions

Description	The total number of IKEv2 message re-transmissions.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.27 (X specifies the interface index)

SA Init Messages Received

Description	The total number of IKEv2 message re-transmissions.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.28 (X specifies the interface index)

SA Init Message Sent

Description	The total number of IKEv2 message re-transmissions.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.29 (X specifies the interface index)

SA Establishment Attempts

Description	The total number of IKEv2 message re-transmissions.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.30 (X specifies the interface index)

SA Establishment Success

Description	The total number of IKEv2 SA successfully established on the IKEv2 interface.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.31 (X specifies the interface index)

Tunnel Rate

Description	Specifies the tunnel establishment rate, in terms of tunnels created per second. Note that this count is available through both an ACLI show command and an SNMP GET operation.
Type	gauge
Range	0 to $2^{32} - 1$
ACJI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.9 – apSecurityIkeInterfaceInfoTable 1.3.6.1.4.1.9148.3.9.1.9.X.32 (X specifies the interface index)

IKEv2 Negotiation Errors

CPU Overload Errors

Description	The number of IKEv2 requests that were rejected because of CPU load constraints.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.3 (X specifies the interface index)

Init Cookie Errors

Description	The number of all IKEv2 exchanges that failed because of faulty Security Parameter Index (SPI) values. SPIs provide a local SA identifier and are exchanged between IKEv2 peers in the common IKEv2 header and in Notify Payloads.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.4 (X specifies the interface index)

Auth Errors

Description	The number of failed IKE_AUTH exchanges, regardless of the specific reason for failure.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.5 (X specifies the interface index)

EAP Access Request Errors

Description	The number of authentication failures that occur ed during the EAP access phase.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.6 (X specifies the interface index)

EAP Access Challenge Errors

Description	The number of authentication failures that occur ed during the EAP challenge phase.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.7 (X specifies the interface index)

TS Errors

Description	The number of CREATE_CHILD_SA exchanges that failed because of faulty TS payload contents, or failure on the part of the remote peers to negotiate the offered traffic selectors.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.8 (X specifies the interface index)

CP Errors

Description	The number of IKE_AUTH and/or CREATE_CHILD_SA exchanges that failed because of faulty, unsupported, or unknown Configuration Payload contents.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.9 (X specifies the interface index)

KE Errors

Description	The number of IKE_SA_INIT and/or CREATE_CHILD_SA exchanges that failed because of faulty, unsupported, or unknown Key Exchange Payload contents.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.10 (X specifies the interface index)

Proposal Errors

Description	The number of failed negotiations that resulted from the inability to reconcile crytographic proposals contained in the Security Association Payloads exchanged by IKEv2 peers. Security Association Payloads are exchanged during the IKE_SA_INIT, IKE_AUTH, and CREATE_CHILD_SA stages.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.11 (X specifies the interface index)

Syntax Errors

Description	The number of failed negotiations, of any type, resulting from otherwise uncharacterized errors.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.12 (X specifies the interface index)

Critical Payload Errors

Description	The number of failed negotiations that resulted from the presence of a Critical flag in a payload that could not be parsed, or was not supported. IKEv2 adds a critical flag to each payload header for further flexibility for forward compatibility. If the critical flag is set and the payload type is unrecognized, the message must be rejected and the response to the IKE request containing that payload MUST include a Notify payload UNSUPPORTED_CRITICAL_PAYLOAD, indicating an unsupported critical payload was included. If the critical flag is not set and the payload type is unsupported, that payload must be ignored.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show security ike statistics
SNMP MIB	1.3.6.1.4.1.9148.3.9.1.3 – apSecurityIkeInterfaceStatsEntry 1.3.6.1.4.1.9148.3.9.1.3.X.13 (X specifies the interface index)

Server Roundtrip Time

Description	Contains the average round trip time for a response from this RADIUS server.
Type	integer
Range	0 to 2^{32} - 1
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.3

Server Malformed Access Response

Description	Contains the number of malformed access responses received on this RADIUS server.
Type	counter
Range	0 to 2^{32} - 1
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.4

Server Access Requests

Description	Contains the number of access requests received on this RADIUS server.
Type	counter
Range	0 to 2^{32} - 1
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.5

Server Disconnect Requests

Description	Contains the number of disconnect requests received on this RADIUS server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.6

Server Disconnect ACKS

Description	Contains the number of acknowledged disconnects on this RADIUS server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.7

Server Disconnect NACKS

Description	Contains the number of unacknowledged disconnects on this RADIUS server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.8

Server Bad Authenticators

Description	Contains the number of authentication rejections on this RADIUS server.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.9

Server Access Retransmissions

Description	Contains the number of access retransmitals on this RADIUS server.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.10

Server Access Accepts

Description	Contains the number of successful authentications on this RADIUS server.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.11

Server Timeouts

Description	Contains the number of Response timeouts on this RADIUS server.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.12

Server Access Rejects

Description	Contains the number of unsuccessful authentications on this RADIUS server.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.14

Server Unknown PDUTypes

Description	Contains the number or unknown/unreadable PDUs received by this RADIUS server
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.14

Server Access Challenges

Description	Contains the number of Access Challenges on this RADIUS server.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show radius
SNMP MIB	1.3.6.1.4.1.9148.3.18.1.1.1 – apRadiusServerStatsEntry 1.3.6.1.4.1.9148.3.18.1.1.1.15

Diameter Protocol Operations

Diameter Messages Sent

Description	Contains the number of messages sent by this Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.3 (X specifies the Diameter Server index)

Diameter Messages Sent Failed

Description	Contains the number of unacknowledged messages sent by this Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.4 (X specifies the Diameter Se rev er index)

Diameter Messages Resent

Description	Contains the number of messages re-transmitted to this Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.5 (X specifies the Diameter Se rev er index)

Diameter Messages Received

Description	Contains the number of messages received by this Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.6 (X specifies the Diameter Se rev er index)

Diameter Messages Processed

Description	Contains the number of messages processed by this Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.7 (X specifies the Diameter Se rev er index)

Diameter Connection Timeouts

Description	Contains the number of connection timeouts on the Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.8 (X specifies the Diameter Server index)

Diameter BadState Drops

Description	Contains the number of packets dropped because of faulty state on the Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.9 (X specifies the Diameter Server index)

Diameter BadType Drops

Description	Contains the number of packets dropped because of faulty type on the Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.10 (X specifies the Diameter Server index)

Diameter BadID Drops

Description	Contains the number of packets dropped because of faulty ID on the Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACJI "show' command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.11 (X specifies the Diameter Server index)

Diameter AuthFail Drops

Description	Contains the number of failed authentications on the Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.12 (X specifies the Diameter Server index)

Diameter Invalid Peer Messages

Description	Contains the number of client messages that could not be parsed on the Diameter server.
Type	counter
Range	0 to $2^{32} - 1$
ACLI "show" command	show diameter
SNMP MIB	1.3.6.1.4.1.9148.3.13.1.1.2.2 – apDiamInterfaceStatsTable 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.13 (X specifies the Diameter Server index)

ACLI Show Commands

ACLI **show** commands

- display and reset IKEv2 performance and error counters
- display IKEv2 SA data
- display IKEv2 TCA data

Performance and Error Counters

Three ACLI commands display and reset IKEv2 performance and error counters.

Use the **show security** command to display performance and error counters for a specified IKEv2 interface, or for all IKEv2 interfaces.

```
ACMEPACKET# show security 192.169.204.15
```

with a specified interface, displays performance and error counters for the target interface

```
ACMEPACKET# show security all
```

with *all*, displays performance and error counters for all IKEv2 interfaces

Use the **reset ike-stats** command to reset (set to 0) performance and error counters for a specified IKEv2 interface, or for all IKEv2 interfaces.

```
ACMEPACKET# reset ike-stats 192.169.204.15
```

with a specified interface, resets performance and error counters for the target interface

```
ACMEPACKET# reset ike-stats all
```

with *all*, resets performance and error counters for all IKEv2 interfaces

Use the **reset ike-mib** command to reset (set to 0) MIB-based error counters for all IKEv2 interfaces.

```
ACMEPACKET# reset ike-mib
```

re-sets the MIB-based error counters for all IKEv2 interfaces

IKEv2 and Child SAs

Use the **show security** command with optional arguments to display IKEv2 and child SA information to include:

- IP address and port of remote end-point
- intervening NAT device (yes | no)
- local IP address
- tunnel state (up | down)
- initiator cookie
- responder cookie
- remote inner (tunnel) IP address
- incoming/outgoing Security Parameter Indexes (SPI) of the child SA

```
ACMEPACKET# show security sad ike-interface 192.169.204.15
```

with a specified interface address, displays SA information for a single IKEv2 interface

```
ACMEPACKET# show security sad ike-interface all
```

with *all*, displays SA information for all IKEv2 interfaces

```
ACMEPACKET# show security sad ike-interface all
Displaying the total (4321) number of entries may take long and could
affect system performance.
Continue? [y/n]?: y
Peer: 6.0.0.36:500 (NAT: No) Host: 172.16.101.2 State: Up
     IKE Cookies: 0x23e71b73d5a10c58[I] 0xd2017a6fb84a4fa6[R]
     Child Peer IP: 101.0.0.36:0 Child SPI: 4236760138[I] 1721373661[O]
Peer: 6.0.0.28:500 (NAT: No) Host: 172.16.101.2 State: Up
     IKE Cookies: 0xf64d031d32525730[I] 0xcea2d5ae3c91050f[R]
     Child Peer IP: 101.0.0.28:0 Child SPI: 3632387333[I] 1421117246[O]
Peer: 6.0.0.9:500 (NAT: No) Host: 172.16.101.2 State: Up
     IKE Cookies: 0x84ec95a1cd0a4c5d[I] 0x1b61b385c4e627b4[R]
     Child Peer IP: 101.0.0.9:0 Child SPI: 2432742837[I] 3872387177[O]
Peer: 6.0.0.25:500 (NAT: No) Host: 172.16.101.2 State: Up
     IKE Cookies: 0x541b2651e88c9368[I] 0xdc393a61af6dc909[R]
     Child Peer IP: 101.0.0.25:0 Child SPI: 785656546[I] 148357787[O]
Peer: 6.0.0.27:500 (NAT: No) Host: 172.16.101.2 State: Up
     IKE Cookies: 0x3ba43c5c685e37e6[I] 0x7bfa6f0781dce1a8[R]
     Child Peer IP: 101.0.0.27:0 Child SPI: 767765646[I] 3797275291[O]
Peer: 6.0.0.22:500 (NAT: No) Host: 172.16.101.2 State: Up
     IKE Cookies: 0x925e540ecbd58dbb[I] 0x7e1101371a5a5823[R]
     Child Peer IP: 101.0.0.22:0 Child SPI: 787745714[I] 876969665[O]
Peer: 6.0.0.2:500 (NAT: No) Host: 172.16.101.2 State: Up
     IKE Cookies: 0xda0f568684ba5e2c[I] 0x74c533da2fd29901[R]
     Child Peer IP: 101.0.0.2:0 Child SPI: 3884481109[I] 1862217459[O]
Peer: 6.0.0.7:500 (NAT: No) Host: 172.16.101.2 State: Up
     IKE Cookies: 0x6166bac4438f3ca7[I] 0x71d1049a0f8520f4[R]
     Child Peer IP: 101.0.0.7:0 Child SPI: 2798332266[I] 2789214337[O]
Peer: 6.0.0.15:500 (NAT: No) Host: 172.16.101.2 State: Up
     IKE Cookies: 0x0e060701115069bf[I] 0x2e69adbf15438000[R]
     Child Peer IP: 101.0.0.15:0 Child SPI: 713005957[I] 1985608540[O]
Continue? [y/n]?: y
...
...
...
```

Use **show security** with the peer address obtained by the previous command to display more detailed information regarding a specific tunnel to include:

- IKE version
- Diffie Hellman group
- the IKE SA hash algorithm
- the IKE SA message authentication code algorithm
- the IKE SA encryption algorithm
- seconds since SA creation
- SA lifetime in seconds
- remaining lifetime in seconds
- IPsec operational mode (tunnel | transport)
- IPsec security protocol (AH |ESP)
- IPsec authentication protocol (SHA1 | MD5 | any)
- IPsec encryption protocol (AES | 3DES | null| any)

```
ACMEPACKET# show security sad ike-interface <ipAddress> peer
<ipAddress>

ACMEPACKET# show security sad ike-interface 172.16.101.2 peer
6.0.0.36:500

IKE SA:

    IKE Version : 2
    Tunnel State : Up
    Last Response [Seconds] : 212
    AAA Identity :
    NAT : No

    IP Addresses [IP:Port]
        Peer : 6.0.0.36:500
        Server Instance : 172.16.101.2:500

    Cookies
        Initiator : 0x23e71b73d5a10c58
        Responder : 0xd2017a6fb84a4fa6

    Algorithms
        DH Group : 2
        Hash : HMAC-SHA1
        MAC : SHA1-96
        Cipher : 3DES

    SA Times [Seconds]
        Creation : 141
        Expiry : 86400
        Remaining : 86188

IPSec SA:

    IP Addresses [IP:Port]
        Destination : 101.0.0.36:0
        Source : 172.16.101.2:0

    SPI
        Outbound : 1721373661
        Inbound : 4236760138

    Algorithms
        Mode : TUNNEL
        Protocol : ESP
        Authentication : SHA1
        Encryption : AES

    Traffic Selectors [Start IP - End IP]
        Destination : 101.0.0.36 - 101.0.0.36
        Source : 172.16.101.2 - 172.16.101.2
```

TCA Counters

An ACLI command is provided to display TCA information.

```
ACMEPACKET# show security ike threshold-crossing-alert <ipAddress> ||
all
```

with a specified IPv4 interface address, displays TCA information for the specified IKEv2 interface, otherwise displays TCA information for all IKEv2 interfaces

```
ACMEPACKET# show security ike threshold-crossing-alert all
IKE Threshold Crossing Alerts
tca-type: ike-auth-failure
           reset                reset                reset
critical   critical   major      major      minor      minor
---------- ---------- ---------- ---------- ---------- ----------
       40         30         25         24         12          1
current value:
   Window      Total      Maximum
        0          0            0
current level: clear

tca-type: ipsec-tunnel-removal
           reset                reset                reset
critical   critical   major      major      minor      minor
---------- ---------- ---------- ---------- ---------- ----------
        0          0         10          5          0          0

current value:
   Window      Total      Maximum
        0          0            0
current level: clear
```

TCA Traps

TCAs generate the following SNMP traps to report crossing of threshold levels, or to clear threshold levels.

```
apSysMgmtTcaTrap NOTIFICATION-TYPE
    OBJECTS {
        apSysMgmtTcaOid,
        apSysMgmtTcaCurrent,
        apSysMgmtTcaMinorThreshold,
        apSysMgmtTcaMajorThreshold,
        apSysMgmtTcaCriticalThreshold
        }
        STATUS current
        DESCRIPTION
        "The trap will be generated when a Threshold Crossing Alert counter
        crosses a configured TCA threshold"
::= { apSystemManagementMonitors 72 }
```

```
apSysMgmtTcaClearTrap NOTIFICATION-TYPE
    OBJECTS {
         apSysMgmtTcaOid,
         apSysMgmtTcaCurrent,
         apSysMgmtTcaMinorThreshold,
         apSysMgmtTcaMajorThreshold,
         apSysMgmtTcaCriticalThreshold
         }
         STATUS current
         DESCRIPTION
         "The trap will be generated when a Threshold Crossing Alert counter
         has fallen below the lowest configured TCA reset-threshold value"
::= { apSystemManagementMonitors 73 }


--Objects
apSysMgmtTcaOid OBJECT-TYPE
    SYNTAX OBJECT IDENTIFIER
    MAX-ACCESS accessible-for-notify
    STATUS current
    DESCRIPTION
    "The Object ID for the Threshold Crossing Alert counter that is
    changing alert level."
::= { apSysMgmtMonitorObjects 65 }

apSysMgmtTcaCurrent OBJECT-TYPE
    SYNTAX Unsigned32
    MAX-ACCESS accessible-for-notify
    STATUS current
    DESCRIPTION
    "The current value of the counter associated with the TCA."
::= { apSysMgmtMonitorObjects 66 }

apSysMgmtTcaCriticalThreshold OBJECT-TYPE
    SYNTAX Unsigned32
    MAX-ACCESS accessible-for-notify
    STATUS current
    DESCRIPTION
    "The current configured TCA critical threshold value."
::= { apSysMgmtMonitorObjects 67 }

apSysMgmtTcaMajorThreshold OBJECT-TYPE
    SYNTAX Unsigned32
    MAX-ACCESS accessible-for-notify
    STATUS current
    DESCRIPTION
    "The current configured TCA major threshold value."
::= { apSysMgmtMonitorObjects 68 }
```

```
apSysMgmtTcaMinorThreshold OBJECT-TYPE
    SYNTAX Unsigned32
    MAX-ACCESS accessible-for-notify
    STATUS current
    DESCRIPTION
    "The current configured TCA minor threshold value."
::= { apSysMgmtMonitorObjects 69 }
```

Historical Data Records

Various statistical counts are available as comma separated values (CSV) Historical Data Record (HDR) files. HDR files are specified and pushed to an accounting server as described in the *Overview* chapter of the *Net-Net 4000 C-Series Historical Data Recording (HDR) Resource Guide*.

IKEv2 Interface HDR

CSV header fields for IKEv2 Interface HDRs are listed below.

TimeStamp	Integer
Interface	IP Address
Current Child SA Pairs	Counter
Maximum Child SA Pairs	Counter
Last Reset TimeStamp	Integer
Child SA Requests	Counter
Child SA Success	Counter
Child SA Failure	Counter
Child SA Delete Request	Counter
Child SA Delete Success	Counter
Child SA Delete Failure	Counter
Child SA Rekey	Counter
Initial Child SA Establishment	Counter
DPD Received Port Change	Counter
DPD Received IP Change	Counter
DPD Response Received	Counter
DPD Response Not Received	Counter
DPD Received	Counter
DPD Retransmitted	Counter
DPD Sent	Counter
IKE SA Packets Sent	Counter
IKE SA Packets Received	Counter
IKE SA Packets Dropped	Counter
Authentication Failures	Counter
IKE Message Errors	Counter
Authentication ID Errors	Counter
Certificate Status Requests	Counter
Certificate Status Success	Counter
Certificate Status Fail	Counter

DDoS Sent	Counter
DDoS Received	Counter
IKE Message Retransmissions	Counter
Tunnel Rate	Counter
Child SA Pair	Guage
IKE SA INIT Messages Received	Counter
IKE SA INIT Messages Sent	Counter
IKE SA Establishment Attempts	Counter
IKE SA Establishment Success	Counter
IKE CPU Overload Error	Counter
IKE Init Cookie Error	Counter
IKE EapAccessRequestError	Counter
IKE EapAccessChallengeError	Counter
IKE TS Error	Counter
IKE CP Error	Counter
IKE KE Error	Counter
IKE Proposal Error	Counter
IKE Syntax Error	Counter
IKE Critical Payload Error	Counter

RADIUS HDR

CSV header fields for RADIUS HDRs are listed below.

Time Stamp	Integer
RADIUS Sever IP Address	IP Address
RADIUS Server Port	Port Address
Round Trip Time	Number
Malformed Access Response	Counter
Access Requests	Counter
Disconnect Requests	Counter
Disconnect ACKs	Counter
Bad Authenticators	Counter
Access Retransmissions	Counter
Access Accepts	Counter
Timeouts	Counter
Access Rejects	Counter
Unknown PDU Types	Counter
Access Challenges	Counter

Diameter HDR

CSV header fields for Diameter HDRs are listed below.

Time Stamp	Integer
Diameter Sever IP Address	IP Address
Diameter Server Port	Port Address
Messages Sent	Counter
Messages Sent Failed	Counter
Messages Resent	Counter
Messages Received	Counter
Messages Processed	Counter
Connection Timeouts	Counter
Bad State Drops	Counter
Bad Type Drops	Counter
Bad ID Drops	Counter
Auth Failed Drops	Counter
Invalid Peer Messages	Counter

IPsec Accounting

IPsec Accounting enables the Security Gateway to gather and store detailed information regarding the establishment and usage of IPsec tunnels. Stored data can be used for both forensic and billing purposes.

IPsec Accounting uses either the RADIUS or DIAMETER protocols which are described in the following sections.

RADIUS Accounting

The RADIUS (Remote Authentication Dial In User Service) Protocol provides AAA (Authentication, Authorization, and Accounting) services. RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*, describes the RADIUS client/server model and specifies the authentication and authorization functions. RFC 2866, *RADIUS Accounting*, specifies the accounting function.

RADIUS accounting is based on a client/server model running over User Datagram Protocol (UDP) in which a Network Access Server (NAS) acts in the client role and a RADIUS accounting server acts in the server role. For purposes of the current IPsec Accounting implementation, the Security Gateway acts as the NAS. In its client role, the Security Gateway is responsible for accessing one or more RADIUS accounting servers and passing IPsec data to the server. The accounting server, in turn, is responsible for receiving the data and returning a response to the client indicating that it has successfully processed the data. All transactions between the client and server are authenticated through the use of a pre-configured shared secret.

Generally, the Security Gateway initiates an accounting session in response to the initiation of the IPsec Security Association (SA) negotiation, or the successful establishment of an IPsec tunnel. At one of these points, the Security Gateway generates either an Accounting Request Early-Start packet (triggered by the start of IKEv2 SA negotiation), or an Accounting Request Start packet (triggered by the completion of tunnel establishment). Each packet contains the tunnel creation data available at the time of transmission. The Security Gateway sends the packet to the RADIUS accounting server, which returns an acknowledgement when the packet has been received and processed. When the tunnel is torn down, the Security Gateway generates an Accounting Request Stop packet that contains previously reported tunnel establishment details along with traffic counts accumulated over the lifetime of the tunnel, and sends that packet to the accounting server. As before, the accounting server returns an acknowledgement when the packet has been received and processed. The RADIUS client implementation also provides the optional ability to generate interim updates, which can be asynchronously triggered by specific IPsec events, or by the expiration of a configured synchronous timer.

The RADIUS-based IPsec accounting implementation is based upon the following RFCs.

RFC2865, *Remote Authentication Dial In User Service (RADIUS)*

RFC 2866, *RADIUS Accounting*

RFC 2868, *RADIUS Attributes for Tunnel Protocol Support*

RFC 2869, R*ADIUS Extensions*

RADIUS Message Exchange

All RADIUS messages consist of a common authentication header and a sequence of RADIUS attributes.

The RADIUS-based IPsec accounting function uses the following message formats

Accounting-Request Packets

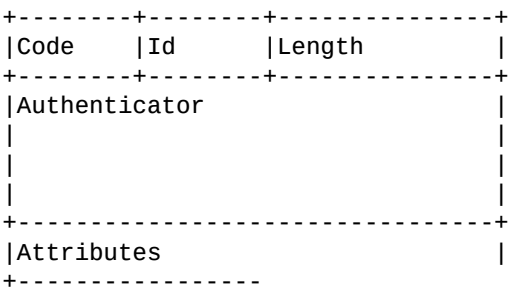
Accounting-Request packets (defined in Section 4.1 of RFC 2866 are of three types:

- Start packets
- Early-Start packets
- Interim-Update packets
- Stop packets

Accounting-Request Start Packet

The Security Gateway sends an Accounting-Request Start packet to a RADIUS accounting server immediately upon establishment of an IPsec tunnel. Upon successful processing of the Start packet, the accounting server must transmit an Accounting-Response packet (defined in Section 4.2 of RFC 2866) to the Security Gateway. The accounting server must not transmit any reply if it fails to process the Start packet.

The Accounting-Request Start packet format is as follows.



Code

This required one octet field identifies the packet type — 4 for an Accounting Request packet.

Id

This required one octet field acts as a session identifier, and contains an 8-bit integer used to correlate RADIUS requests and responses.

The Identifier field must be changed whenever the content of the Attributes field changes, and whenever a valid reply has been received for a previous request. For retransmissions where the contents are identical, the *Id* must remain unchanged.

Length

This required two octet field contains the packet length to include the entire header and any appended attributes.

Authenticator

This required four octet field contains a message-digest value used to authenticate messages between the Security Gateway and the RADIUS accounting server.

In Accounting-Request packets, the Authenticator value is a 16 octet MD5 hash, referred to as the Request Authenticator. MD5 is a hashing algorithm defined in RFC 1321, *The MD5 Message-Digest Algorithm*.

The Security Gateway and the RADIUS accounting server share a pre-configured secret used to calculate a one-way MD5 hash over a stream of octets consisting of the concatenation of:

the contents of the *Code* field
the contents of the *Id* field
the contents of the *Length* field
16 octets of zeros
the contents of the *Attributes* list
the shared secret

Attributes

This field contains a sequence of RADIUS and Vendor-Specific Attributes (VSAs).

The standards-based RADIUS attributes that report protocol data and tunnel establishment details are as follows:

User-Name	Attribute Type 1
NAS-IP-Address	Attribute Type 4
NAS-Port	Attribute Type 5
Framed-IP-Address	Attribute Type 8
Class	Attribute Type 25
Called-Station-Id	Attribute Type 30
Calling-Station-Id	Attribute Type 31
NAS-Identifier	Attribute Type 32
Acct-Status-Type	Attribute Type 40
Acct-Session-Id	Attribute Type 44
Event-Timestamp	Attribute Type 55
Calling-Station-Endpoint	Attribute Type 66
Server-Endpoint	Attribute Type 67

The *Acct-Status-Type* attribute contains a value of *1*, specifying an Accounting Request Start packet.

The Acme Packet VSAs that report tunnel establishment details are as follows:

Acme-IPSEC_IN_SPI	Acme VSA 180
Acme_IPSEC_OUT_SPI	Acme VSA 181
Acme_ATTR_IPSEC_AUTH_METHOD	Acme VSA 186
Acme_ATTR_IPSEC_ENC_METHOD	Acme VSA 187
Acme_ATTR_IKE_AUTH_METHOD	Acme VSA 188
Acme_ATTR_IKE_ENC_METHOD	Acme VSA 189
Acme_ATTR_IKE_HASH_METHOD	Acme VSA 190

Acme_ATTR_IKE_DH_GROUP	Acme VSA 191
Acme_ATTR_IKE_COOKIE_I	Acme VSA 192
Acme_ATTR_IKE_COOKIE_R	Acme VSA 193

Refer to RADIUS Attributes for lists of attributes used in IPsec accounting

Accounting-Request Early-Start Packet

If so configured, the Security Gateway can send an Accounting-Request Early-Start packet to a RADIUS accounting server. In contrast to the Start packet, which is generated at the end of the tunnel establishment process, the Early-Start packet is generated earlier in the tunnel establishment process, specifically at the initiation of the IKEv2 SA negotiation. In some network environments the Early-Start packet can provide useful information for debugging and troubleshooting purposes.

In format the Accounting-Request Early-Start packet is identical to the Start packet. Like the Start packet, the *Acct-Status-Type* attribute contains a value of *1*; the accompanying *Calling-Station-Id* attribute contains the IP address of the remote peer (for example, a femtocell) that initiated tunnel establishment.

Accounting-Request Interim-Update Packet

If so configured, the Security Gateway can send an Accounting-Request Interim-Update packet to a RADIUS accounting server upon re-keying of an IPsec tunnel or re-keying of the IKEv2 SA that supports tunnel operations. Updates can also be generated at regularly scheduled intervals. Upon successful processing of the received Interim-Update packet, the accounting server must transmit an Accounting-Response to the Security Gateway. The accounting server must not transmit any reply if it fails to process the Interim-Update packet.

The format of the Accounting-Request Interim-Update packet is similar to that of the Start packet. The *Acct-Status-Type* attribute, however, contains a value of 3, specifying an Interim-Update packet, and additional attributes are added to report usage statistics for the current lifetime of the tunnel.

Standards-based RADIUS attributes that report protocol data and tunnel establishment and usage details are as follows:

User-Name	Attribute Type 1
NAS-IP-Address	Attribute Type 4
NAS-Port	Attribute Type 5
Framed-IP-Address	Attribute Type 8
Class	Attribute Type 25
Called-Station-Id	Attribute Type 30
Calling-Station-Id	Attribute Type 31
NAS-Identifier	Attribute Type 32
Acct-Status-Type	Attribute Type 40
Acct-Input-Octets	Attribute Type 42
Acct-Output-Octets	Attribute Type 43
Acct-Session-Id	Attribute Type 44
Acct-Session-Time	Attribute Type 46
Acct-In-Packets	Attribute Type 47
Acct-Output-Packets	Attribute Type 48
Acct-Input-Gigawords	Attribute Type 52
Acct-Output-Gigawords	Attribute Type 53
Event-Timestamp	Attribute Type 55
Calling-Station-Endpoint	Attribute Type 66
Server-Endpoint	Attribute Type 67

Acme Packet VSAs that report tunnel establishment details are as follows:

Acme_ATTR_EVENT_TIME	Acme VSA 176
Acme_ATTR_IPSEC_EVENTS	Acme VSA 179
Acme-IPSEC_IN_SPI	Acme VSA 180
Acme_IPSEC_OUT_SPI	Acme VSA 181
Acme_ATTR_IPSEC_AUTH_METHOD	Acme VSA 186
Acme_ATTR_IPSEC_ENC_METHOD	Acme VSA 187
Acme_ATTR_IKE_AUTH_METHOD	Acme VSA 188
Acme_ATTR_IKE_ENC_METHOD	Acme VSA 189
Acme_ATTR_IKE_HASH_METHOD	Acme VSA 190
Acme_ATTR_IKE_DH_GROUP	Acme VSA 191
Acme_ATTR_IKE_COOKIE_I	Acme VSA 192
Acme_ATTR_IKE_COOKIE_R	Acme VSA 193
Acme_ATTR_IPSEC_REKEY_SPI_I	Acme VSA 194
Acme_ATTR_IPSEC_REKEY_SPI_O	Acme VSA 195
Acme_ATTR_IKE_REKEY_COOKIE_I	Acme VSA 196
Acme_ATTR_IKE_REKEY_COOKIE_R	Acme VSA 197

Accounting-Request Stop Packet

The Security Gateway sends an Accounting-Request Stop packet to a RADIUS accounting server immediately upon tear-down of an IPsec tunnel. Upon successful processing of the received packet, the accounting server must transmit an Accounting-Response to the Security Gateway. The accounting server must not transmit any reply if it fails to process the Stop packet.

The format of the Accounting-Request Stop packet mirrors that of the Interim-Update packet save for the value of the *Acct-Status-Type* attribute which contains a value of 2, specifying a Stop packet, and the update of usage statistics to reflect the tunnel lifetime.

Standards-based RADIUS attributes that report protocol data and tunnel establishment and final usage details are as follows:

User-Name	Attribute Type 1
NAS-IP-Address	Attribute Type 4
NAS-Port	Attribute Type 5
Framed-IP-Address	Attribute Type 8
Class	Attribute Type 25
Called-Station-Id	Attribute Type 30
Calling-Station-Id	Attribute Type 31
NAS-Identifier	Attribute Type 32
Acct-Status-Type	Attribute Type 40
Acct-Input-Octets	Attribute Type 42
Acct-Output-Octets	Attribute Type 43
Acct-Session-Id	Attribute Type 44
Acct-Session-Time	Attribute Type 46
Acct-In-Packets	Attribute Type 47
Acct-Output-Packets	Attribute Type 48
Acct-Input-Gigawords	Attribute Type 52
Acct-Output-Gigawords	Attribute Type 53
Event-Timestamp	Attribute Type 55
Calling-Station-Endpoint	Attribute Type 66
Server-Endpoint	Attribute Type 67

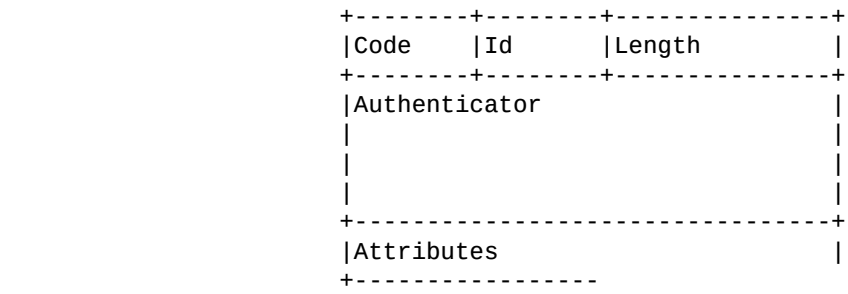
Acme Packet VSAs that report tunnel establishment and tear-down details are as follows:

Acme_ATTR_EVENT_TIME	Acme VSA 176
Acme_ATTR_IPSEC_EVENTS	Acme VSA 179
Acme-IPSEC_IN_SPI	Acme VSA 180
Acme_IPSEC_OUT_SPI	Acme VSA 181
Acme_ATTR_IPSEC_AUTH_METHOD	Acme VSA 186
Acme_ATTR_IPSEC_ENC_METHOD	Acme VSA 187
Acme_ATTR_IKE_AUTH_METHOD	Acme VSA 188
Acme_ATTR_IKE_ENC_METHOD	Acme VSA 189
Acme_ATTR_IKE_HASH_METHOD	Acme VSA 190
Acme_ATTR_IKE_DH_GROUP	Acme VSA 191
Acme_ATTR_IKE_COOKIE_I	Acme VSA 192
Acme_ATTR_IKE_COOKIE_R	Acme VSA 193
Acme_ATTR_IPSEC_REKEY_SPI_I	Acme VSA 194
Acme_ATTR_IPSEC_REKEY_SPI_O	Acme VSA 195
Acme_ATTR_IKE_REKEY_COOKIE_I	Acme VSA 196
Acme_ATTR_IKE_REKEY_COOKIE_R	Acme VSA 197

Accounting-Response Packets

The RADIUS accounting server sends an Accounting-Response packet to the Security Gateway to acknowledge receipt and successful processing of an Accounting-Request Early-Start, Start, Interim-Update, or Stop packet. The accounting server must not transmit any reply if it fails to process the Accounting-Request packet.

The Accounting-Response packet format is as follows.



Code

This required one octet field identifies the packet type — 5 for an Accounting Request packet.

Id

This required one octet field acts as a session identifier, and contains an 8-bit integer used to correlate RADIUS requests and responses.

The Identifier field must be changed whenever the content of the Attributes field changes, and whenever a valid reply has been received for a previous request. For retransmissions where the contents are identical, the *Id* must remain unchanged.

For Accounting-Response packets this field is a copy of the same field in the acknowledged packet.

Length

This required two octet field contains the packet length to include the entire header and any appended attributes.

Authenticator

This required four octet field contains a message-digest value used to authenticate messages between the Security Gateway and the RADIUS accounting server.

In Accounting-Request packets, the Authenticator value is a 16 octet MD5 hash, referred to as the Request Authenticator. MD5 is a hashing algorithm defined in RFC 1321, *The MD5 Message-Digest Algorithm*.

The Security Gateway and the RADIUS accounting server share a pre-configured secret that is used to calculate a one-way MD5 hash over a stream of octets consisting of the concatenation of:

the contents of the *Code* field
the contents of the *Id* field
the contents of the *Length* field
16 octets of zeros
the contents of the *Attributes* list
the shared secret

Attributes

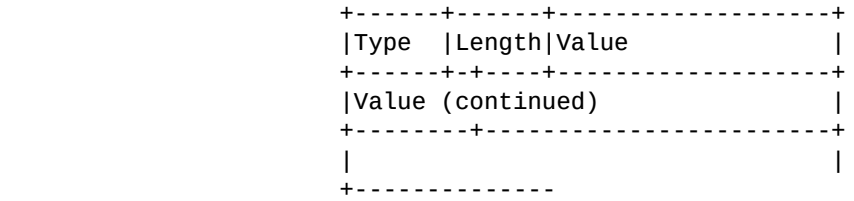
This optional field contains a sequence of attributes.

Accounting-Response packets usually do not contain attributes.

RADIUS Attribute Format

An attribute is the basic RADIUS unit of data. All data delivered by the protocol is in attribute format. Some attributes contain protocol-specific information while other contain information specific to the RADIUS accounting.

The RADIUS attribute TLV (Type/Length/Value) format is as follows.



Type

This required one octet field contains the integer that identifies a specific RADIUS attribute.

Refer to *http://www.ietf.org/assignments/radius-types/radius-types.xml* for up-to-date RADIUS attribute lists.

Length

This required one octet field contains the length of this RADIUS attribute to include the entirety of the Type, Length, and Value fields.

Value

This field contains 0 through 253 octets and contains the attribute value, which must be one of the following data types

- text — specifies 1-253 octets containing UTF-8 characters. Text of length zero (0) can not be sent.
- string — specifies 1-253 octets containing binary data (values 0 through 255decimal, inclusive). Strings of length zero (0) can not be sent.
- address — specifies a 32-bit value.
- integer — specifies an unsigned 32-bit value.
- time — specifies an unsigned 32-bit value, the number of seconds elapsed since January 1, 1970 00:00:00 UTC.

RADIUS Attributes

This section provides a description of all RADIUS attributes used in support of the IPsec accounting application.

Attributes are defined in two distinct sections, the first listing standards based RADIUS attributes, and the second listing Acme Packet VSAs. Attributes are arranged numerically within each section.

Standards-Based Attributes

User-Name

RADIUS Attribute Type 1

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.1 of 2865, *Remote Authentication Dial In User Service (RADIUS)*.

Contains a string that identifies the accounting application user, *acmepacket@wancom.com* for the Security Gateway.

NAS-IP-Address

RADIUS Attribute Type 4

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.4 of RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*.

In conjunction with *NAS-Port*, specifies the IP address/port pair used by the Security Gateway, in RADIUS operations a logical Network Access Serve (NAS), to identify itself to the accounting server.

NAS-Port

RADIUS Attribute Type 5

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.5 of RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*.

In conjunction with *NAS-IP-Address*, specifies the IP address/port pair used by the Security Gateway to identify itself to the accounting server.

Framed-IP-Address

RADIUS Attribute Type 8

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.8 of RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*.

Generated by the Security Gateway, contains the local inner IP address of the IPsec tunnel.

Class

RADIUS Attribute Type 25

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.25 of RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*.

This attribute is sent by the RADIUS accounting server to the Security Gateway during the client/server authentication process. As recommended by RFC 2865, it is included by the Security Gateway in accounting packets sent to the accounting server. The attribute is of no local significance.

Called-Station-Id

RADIUS Attribute Type 30

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.1 of RFC 2866, *RADIUS Accounting*.

Generated by the Security Gateway, contains the IPv4 address of the IKEv2 interface that supports the IPsec tunnel.

Calling-Station-Id

RADIUS Attribute Type 31

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.31 of RFC 2866, *RADIUS Accounting*.

Generated by the Security Gateway, contains either the outer IPv4 address or the FQDN of the tunnel initiator.

NAS-Identifier

RADIUS Attribute Type 32

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.32 of RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*.

Contains a string (often a FQDN) used by the Security Gateway to identify itself to the accounting server.

Acct-Status-Type

RADIUS Attribute Type 40

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.1 of RFC 2866, *RADIUS Accounting*.

Generated by the Security Gateway, contains an enumerated value that indicates how the session was terminated. Relevant values are as follows:

1 — identifies an Accounting Request Start packet

2 — identifies an Accounting Request Stop packet

3 — identifies an Accounting Request Interim-Update packet

Acct-Input-Octets

RADIUS Attribute Type 42

Found in RADIUS Accounting Interim-Update and Stop packets

Defined in Section 5.3 of RFC 2866, *RADIUS Accounting*.

Contains the number of octets received during the tunnel lifetime.

Acct-Output-Octets

RADIUS Attribute Type 43

Found in RADIUS Accounting Interim-Update and Stop packets

Defined in Section 5.3 of RFC 2866, *RADIUS Accounting*.

Contains the number of octets sent during the tunnel lifetime.

Acct-Session-Id

RADIUS Attribute Type 44

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.5 of RFC 2866, *RADIUS Accounting*.

Generated by the Security Gateway, contains a UTF-8 string that provides a unique RADIUS session identifier. The identifier is used to correlate related Start, Interim-Update, and Stop packets.

Acct-Session-Time

RADIUS Attribute Type 46

Found in RADIUS Accounting Interim-Update and Stop packets

Defined in Section 5.7 of RFC 2866, *RADIUS Accounting*.

Generated by the Security Gateway, reports the tunnel lifetime in seconds

Acct-Input-Packets

RADIUS Attribute Type 47

Found in RADIUS Accounting Interim-Update and Stop packets

Defined in Section 5.8 of RFC 2866, *RADIUS Accounting*.

Contains the number of packets received during the tunnel lifetime.

Acct-Output-Packets

RADIUS Attribute Type 48

Found in RADIUS Accounting Interim-Update and Stop packets

Defined in Section 5.9 of RFC 2866, *RADIUS Accounting*.

Contains the number of packets sent during the tunnel lifetime.

Acct-Terminate-Cause

RADIUS Attribute Type 49

Found in RADIUS Accounting Stop packets

Defined in Section 5.10 of RFC 2866, *RADIUS Accounting*.

Generated by the Security Gateway, contains an enumerated value that indicates how the session was terminated. Relevant values are as follows:

1 User Request
2 Lost Carrier
3 Lost Service
4 Idle Timeout
5 Session Timeout
6 Admin Reset
7 Admin Reboot
8 Port Error
9 NAS Error
10 NAS Request
11 NAS Reboot
12 Port Unneeded
13 Port Preempted
14 Port Suspended
15 Service Unavailable
16 Callback
17 User Error
18 Host Request

Acct-Input-Gigawords

RADIUS Attribute Type 52

Found in RADIUS Accounting Interim-Update and Stop packets

Defined in Section 5.1 of RFC 2869, *RADIUS Extensions*.

This attribute indicates how many times the *Acct-Input-Octets* counter has wrapped around 2^{32} over the course of the course of the tunnel lifetime.

Acct-Output-Gigawords

RADIUS Attribute Type 53

Found in RADIUS Accounting Interim-Update and Stop packets

Defined in Section 5.2 of RFC 2869, *RADIUS Extensions*.

This attribute indicates how many times the *Acct-Output-Octets* counter has wrapped around 2^{32} over the course of the course of the tunnel lifetime.

Event-Timestamp

RADIUS Attribute Type 55

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 5.3 of RFC 2869, *RADIUS Extensions*.

Generated by the Security Gateway, this attribute contains a time value. In an Accounting-Request Early-Start packet it contains the time at which the remote peer device initiated IKEv2 SA negotiation. In an Accounting Request Start packet, the timestamp specifies the time of tunnel creation. In a Accounting Request Stop packet, the timestamp specifies the time or tunnel tear-down. In

an Interim-Update packet, the timestamp specifies either the time of the triggering event (for example, an IPsec re-key), or the time of a synchronous update. In both cases, the time is expressed as the number of elapsed seconds since January 1, 1990 00:00 UTC.

Tunnel-Client-Endpoint

RADIUS Attribute Type 66

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 3.3 of RFC 2868, *RADIUS Attributes for Tunnel Protocol Support*.

Generated by the Security Gateway, contains either the outer IPv4 address or the FQDN of the tunnel initiator.

Tunnel-Server-Endpoint

RADIUS Attribute Type 67

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 3.4 of RFC 2868, *RADIUS Attributes for Tunnel Protocol Support*.

Generated by the Security Gateway, contains either the outer IPv4 address or the FQDN of the tunnel responder (the Security Gateway).

Acme Packet VSAs

Acme_ATTR_EVENT_TIME

Acme Packet Attribute Type 176

Found in RADIUS Accounting Interim-Update and Stop packets

This attribute contains a time value reporting the occurrence of a re-keying event. The re-keying type is identified by the *ACME_ATTR_IPSEC_EVENTS* attribute,

The time is expressed as the number of elapsed seconds since January 1, 1990 00:00 UTC.

Acme_ATTR_IPSEC_EVENTS

Acme Packet Attribute Type 179

Found in RADIUS Accounting Interim-Update, and Stop packets

Identifies a re-keying event by specific type, IPsec or IKEv2.

Acme_IPSEC_IN_SPI

Acme Packet Attribute Type 180

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Generated after initial tunnel establishment, contains the Security Parameters Index (SPI) of the inbound IPsec Security Association (SA).

Determined during IPsec SA establishment, the SPI is a value within the range 1 through 4294967295 (2^{32}-1) that, in conjunction with the local IP address, uniquely identifies the IPsec SA used to protect transmitted data. The SPI is included in the in the Encapsulating Security Protocol (ESP) and Authentication Header (AH) headers.

Acme_IPSEC_OUT_SPI

Acme Packet Attribute Type 181

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Generated after initial tunnel establishment, contains the Security Parameters Index (SPI) of the outbound IPsec Security Association (SA).

Determined during IPsec SA establishment, the SPI is a value within the range 1 through 4294967295 (2^{32}-1) that, in conjunction with the local IP address, uniquely identifies the IPsec SA used to protect transmitted data. The SPI is included in the in the Encapsulating Security Protocol (ESP) and Authentication Header (AH) headers.

Acme_ATTR_IPSEC_AUTH_METHOD

Acme Packet Attribute Type 186

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Contains a string that identifies the IPsec (tunnel) authentication method.

Acme_ATTR_IPSEC_ENC_METHOD

Acme Packet Attribute Type 187

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Contains a string that identifies the IPsec (tunnel) encryption method.

Acme_ATTR_IKE_AUTH_METHOD

Acme Packet Attribute Type 188

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Contains a string that identifies key exchange authentication method.

Acme_ATTR_IKE_ENC_METHOD

Acme Packet Attribute Type 189

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Contains a string that identifies key exchange encryption method.

Acme_ATTR_IKE_HASH_METHOD

Acme Packet Attribute Type 190

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Contains a string that identifies key exchange hash method.

Acme_ATTR_IKE_DH_GROUP

Acme Packet Attribute Type 191

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Contains a string that identifies the DIFFIE/HELLMAN Group used during key generation.

Acme_ATTR_IKE_COOKIE_I

Acme Packet Attribute Type 192

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Contains the initiator's cookie used during establishment of the initial IKEv2 connection.

Acme_ATTR_IKE_COOKIE_R

Acme Packet Attribute Type 193

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Contains the responder's cookie used during establishment of the initial IKEv2 connection.

Acme_ATTR_IPSEC_REKEY_SPI_I

Acme Packet Attribute Type 194

Found in RADIUS Accounting Interim-Update and Stop packets

Generated after an IPsec re-keying event, contains the Security Parameters Index (SPI) of the inbound IPsec Security Association (SA).

Determined during IPsec SA establishment, the SPI is a value within the range 1 through 4294967295 (2^{32}-1) that, in conjunction with the local IP address, uniquely identifies the IPsec SA used to protect the transmitted data. The SPI is included in the in the Encapsulating Security Protocol (ESP) and Authentication Header (AH) headers.

Acme_ATTR_IPSEC_REKEY_SPI_O

Acme Packet Attribute Type 195

Found in RADIUS Interim-Update and Stop packets

Generated after an IPsec re-keying event, contains the Security Parameters Index (SPI) of the outbound IPsec Security Association (SA).

Determined during IPsec SA establishment, the SPI is a value within the range 1 through 4294967295 (2^{32}-1) that, in conjunction with the local IP address, uniquely identifies the IPsec SA used to protect the transmitted data. The SPI is included in the in the Encapsulating Security Protocol (ESP) and Authentication Header (AH) headers.

Acme_ATTR_IKE_REKEY_COOKIE_I

Acme Packet Attribute Type 196

Found in RADIUS Accounting Interim-Update and Stop packets

Generated after an IKEv2 re-keying event, contains the initiator's cookie used during establishment of the new IKEv2 connection.

Acme_ATTR_IKE_REKEY_COOKIE_R

Acme Packet Attribute Type 197

Found in RADIUS Accounting Interim-Update and Stop packets

Generated after an IKEv2 re-keying event, contains the initiator's cookie used during establishment of the new IKEv2 connection.

DIAMETER Accounting

The DIAMETER protocol was derived from the RADIUS protocol, and is generally considered to be the next generation AAA protocol. DIAMETER has been widely used in the IMS (IP Multimedia Subsystem) architecture for IMS entities to exchange AAA-related information.

DIAMETER accounting is based on a client/server model running over the Transmission Control Protocol (TCP) in which the Security Gateway assumes the client role and a DIAMETER accounting server acts in the server role. In its client role, the Security Gateway is responsible for accessing one or more DIAMETER accounting servers and passing IPsec data to the server. The accounting server, in turn, is responsible for receiving the data and returning a positive or negative acknowledgement after completing data processing. All transactions between the DIAMETER client and server are conducted over TLS (transport Layer Security) or IPsec connections.

The Security Gateway initiates an accounting session in response to the successful establishment of an IPsec tunnel. At that point, the Security Gateway establishes a secure TCP connection with the DIAMETER accounting server and engages in a Capabilities Exchange, which results in the identification of the remote peer and a verification of the peer's ability to provide requested services — in this case accounting. At the successful conclusion of the Capabilities Exchange, the Security Gateway sends an Accounting Request Start message that contains the tunnel creation data available at the time of transmission. The DIAMETER accounting server returns either an acknowledgement or an error message when the message has been received and processed. When the tunnel is torn down, the Security Gateway generates an Accounting Request Stop message that contains previously reported tunnel establishment details along with traffic counts accumulated over the lifetime of the tunnel, and sends that message to the DIAMETER accounting server. As before, the accounting server returns an acknowledgement or error message when the data has been received and processed.

The DIAMETER-based IPsec accounting implementation is based upon the following RFCs.

RFC 3588, *Diameter Base Protocol*

RFC 4004, *DIAMETER Mobile IPv4 Application*

RFC 4005, *DIAMETER Network Access Server Application*

DIAMETER Messages

DIAMETER-based IPsec accounting requires two distinct message exchanges to record IPsec accounting data. Recorded data provides details of tunnel establishment and operation in addition to octet- and packet-based traffic counts covering the tunnel's lifetime.

The first message exchange, the Capabilities Exchange, consists of a Capabilities Exchange Request (CER) and a Capabilities Exchange Answer (CEA). The exchange is defined in Section 5.3 of RFC 3588, *Diameter Base Protocol*. The exchange is commenced immediately after the establishment of a TCP connection between the Security Gateway and a DIAMETER accounting server. Refer to Capabilities Exchange for messaging details.

The second message exchange, the Accounting Exchange, consists of Accounting Request Start and Stop Messages and Accounting Answer Messages. Start and Stop Messages, which contain IPsec accounting data, are transmitted by the Security Gateway. Answer Messages are transmitted by the DIAMETER accounting server in response to the receipt of a Start or Stop Message. Accounting Exchange messages are defined in Section 9.2 of RFC 3588. Refer to Accounting Exchange for messaging details.

All DIAMETER messages consist of a common header and a sequence of DIAMETER Attribute Value Pairs (AVPs).

DIAMETER Header

The common DIAMETER header format is as follows.

```
+--------+------------------------+
| ver    | message-length         |
+--------+------------------------+
| flags  | command-code           |
+--------+------------------------+
| application-id                  |
+---------------------------------+
| hop-by-hop identifier           |
+---------------------------------+
| end-to-end identifier           |
+---------------------------------+
| AVPs ...                        |
+----------------
```

ver

This one octet field specifies the DIAMETER protocol version, and must contain a value of *1*.

message-length

This three octet field specifies the length (in octets) of the DIAMETER message to include the DIAMETER header.

flags

contains command flags as follows

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|R|P|E|T|r|r|r|r|
+-+-+-+-+-+-+-+-+
```

R — Request bit, if set the message is a request, if clear the message is an answer

P — Proxy bit (not relevant for IPsec accounting operations)

E — Error bit, indicates an error (negative acknowledgement)

T — re-Transmission bit (not relevant for IPsec accounting operations)

r — reserved (not currently used)

command-code

This three octet field contains a command code that identifies the message type. Relevant command codes for DIAMETER accounting applications are:

257 — identifies Capabilities Exchange messages

271— identifies Accounting Exchange messages

application-id

This four octet field identifies the requested DIAMETER application. The value 3 identifies the DIAMETER accounting application.

hop-by-hop identifier

This four octet field, essentially a session identifier, contains an unsigned 32-bit integer used to correlate requests and answers.

end-to-end identifier

This four octet field contains an unsigned 32-bit integer used to detect duplicate messages.

DIAMETER AVP Format

An AVP is the basic DIAMETER unit of data. All data delivered by the protocol is in AVP format. Some AVPs contain protocol-specific information while other contain information specific to DIAMETER accounting.

The DIAMETER AVP format is as follows.

```
+---------------------------------+
| AVP Code                        |
+--------+------------------------+
| Flags  | AVP Length             |
+--------+------------------------+
| Vendor-Id (optional)            |
+---------------------------------+
| Data ...                        |
+---------------------------------+
```

AVP Code

This required four octet field (possibly used in conjunction with the optional *Vender-Id* field) uniquely identifies the attribute. AVP numbers 1 through 255 are reserved to provide backward RADIUS capability; numbers 256 and above are allocated by the IANA to support of DIAMETER operations. If the *Vendor-Id* field is present, the attribute is interpreted as a Vendor-specific AVP.

Flags

The required 8-bit Flags field provides handling instructions to the receiver. For the accounting application, the significant bits are the "M" (Mandatory) and "V" (Vendor-Specific) bits.

The "M" bit, when set, requires that both sender and receiver fully understand the AVP syntax. Consequently, if an AVP with its "M" bit set is received by either a DIAMETER client or server, and the recipient fails to recognize the AVP or its value, the recipient must reject the message.

The "V" bit, when set, signals the presence of the optional *Vendor-Id* field in the AVP. If the "V" bit is clear, the field is not present.

AVP Length

The required 24-bit AVP Length field specifies the AVP length (from the beginning of AVP Code through the end of the Data field) in octets.

Vendor-Id

This optional field four octet field identifies a private, non-standard AVP. If present, it contains an SMI (Structure of Management Information) Network Management Private Enterprise Code, *9148* in the case of Acme Packet.

Data

The required Data field contains the attribute value. This field is of variable length, but must end on a 32-bit boundary. If necessary, the field is padded with 0's to ensure boundary compliance.

Capabilities Exchange

RFC 3588 requires an initial Capabilities Exchange between DIAMETER devices after the establishment of a transport layer connection. The Capabilities Exchange allows the discovery of the peer's identity, and its capabilities, for example accounting.

The Capabilities Exchange Request (CER) Message is transmitted by the Security Gateway immediately after the establishment of a TCP connection with a DIAMETER accounting server. The CER identifies the Security Gateway to the accounting server and makes an explicit request for accounting services. The Capabilities Exchange Answer (CEA), transmitted by the DIAMETER accounting server to acknowledge the CER, identifies the accounting server to the Security Gateway and accepts or rejects the request for accounting services.

The CER consists of a DIAMETER header and a sequence of AVPs as follows.

```
+---------------------------------+
|                                 |
| DIAMETER Header                 |
| Command Code = 257 (REQ)        |
| Request Bit  = 1                |
|                                 |
+---------------------------------+
| Origin-Host                     |  Security Gateway hostname
+---------------------------------+
| Origin-Realm                    |  Security Gateway realm
+---------------------------------+
| Host-IP-Address                 |  Security Gateway IP address
+---------------------------------+
| Vendor-Id                       |  9148 identifies Acme Packet
+---------------------------------+
| Product-Name                    |  AcmeDiameterRf
+---------------------------------+
| Account-Application-Id          |  3 (accounting request)
+---------------------------------+
| Firmware-Revision               |  0
+---------------------------------+
```

The CEA consists of a DIAMETER header and a sequence of AVPs as follows.

```
+---------------------------------+
|                                 |
| DIAMETER Header                 |
| Command Code = 257 (REQ)        |
| Request Bit  = 0                |
|                                 |
+---------------------------------+
| Result-Code                     |  Success/Failure
+---------------------------------+
| Origin-Host                     |  Accounting Server hostname
+---------------------------------+
| Origin-Realm                    |  Accounting Server realm
+---------------------------------+
| Host-IP-Address                 |  Accounting Server IP address
+---------------------------------+
| Vendor-Id                       |  Accounting Server code
+---------------------------------+
| Product-Name                    |  Accounting Server produst
+---------------------------------+
| Account-Application-Id          |  3 (accounting request)
+---------------------------------+
| Firmware-Revision               |  Accounting Server version
+---------------------------------+
```

Refer to DIAMETER AVPs for a description of specific AVPs found in Capabilities Exchange messages.

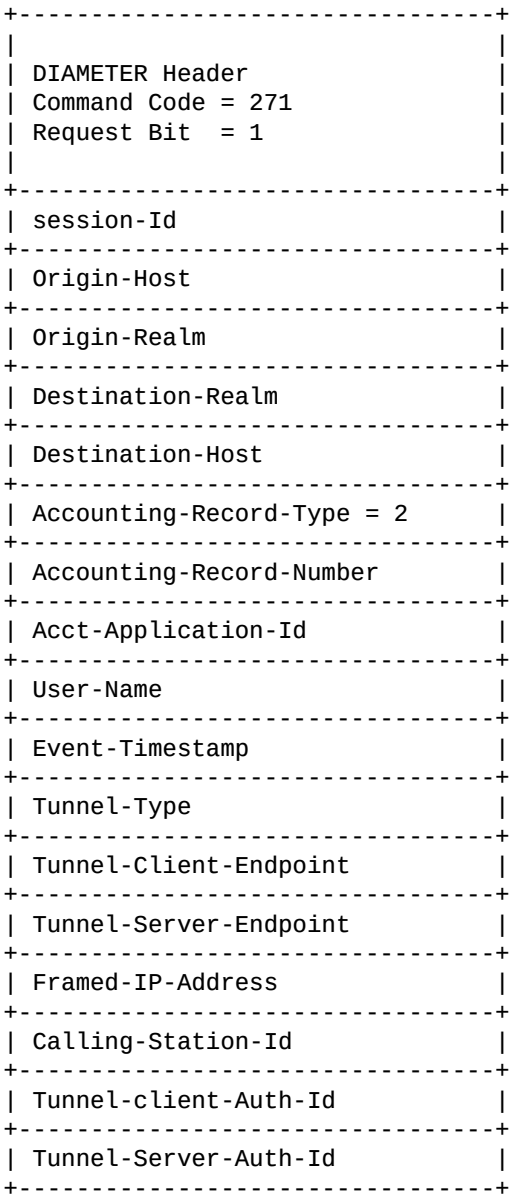
Accounting Exchange

IPsec accounting data is transmitted from the Security Gateway to a DIAMETER accounting server. Accounting data is conveyed to the server in Accounting Request Start and Stop messages. RFC 3588 requires the accounting server acknowledges receipt of accounting requests with an Accounting Answer message.

Accounting Request Start Message

Successful creation of an IPsec tunnel triggers the transmission of an Accounting Request Start Message by the Security Gateway. The Accounting Request Start Message contains AVPs that specify basic DIAMETER protocol requirements (client and server identification, requested application, session identifier, and so on), and AVPs that provide IKE/IPsec data relating to tunnel creation (tunnel type, authorization type, and so on). As the IPsec tunnel has just been created, the Accounting Request Start Message does not contains usage data.

The Accounting Request Start Message consists of a DIAMETER header and a series of AVPs as follows.

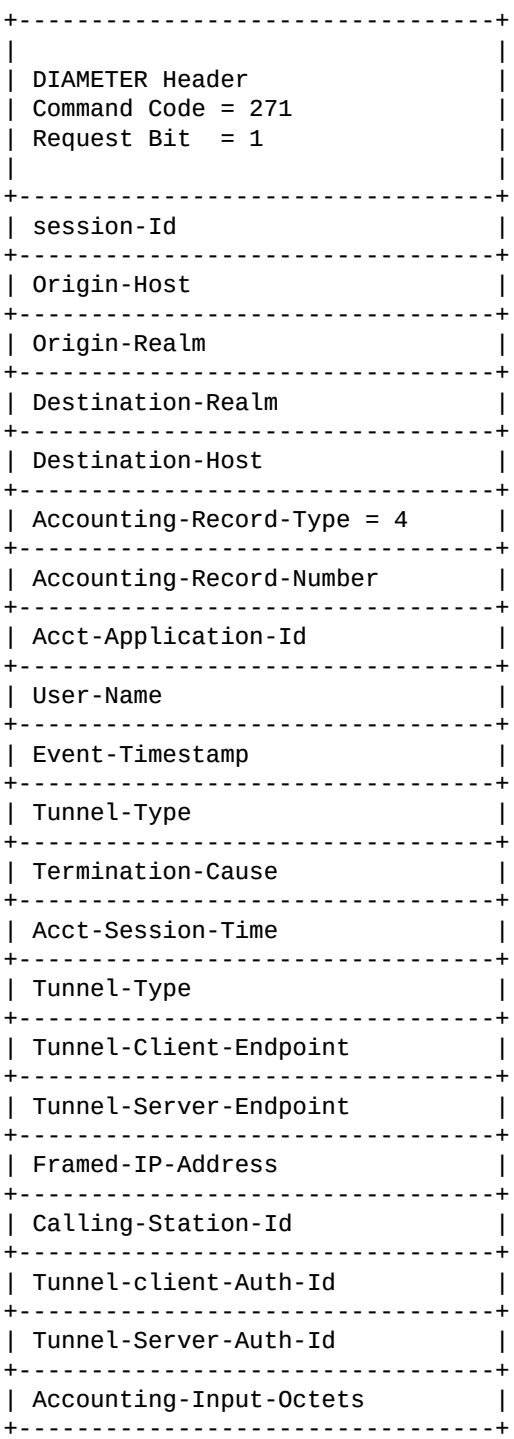


Refer to DIAMETER AVPs for descriptions of the attributes contained in the Accounting Request Start Message.

Accounting Request Stop Message

Tear-down of an IPsec tunnel triggers the transmission of an Accounting Request Stop Message by the Security Gateway. The Accounting Request Stop Message reports tunnel tear-down for whatever reason (timeout, user intervention, network failure, and so on). The message consist of the same AVPs as in the earlier Start message with the addition of new AVPs that contain usage data.

The Accounting Request Stop Message consists of a DIAMETER header and a series of AVPs as follows.



```
+--------------------------------+
| Accounting-Output-Octets       |
+--------------------------------+
| Accounting-Input-Packets       |
+--------------------------------+
| Accounting-Output-Packets      |
+--------------------------------+
```

Refer to DIAMETER AVPs for descriptions of the attributes contained in the Accounting Request Stop Message.

Accounting Request Answer Message

Receipt of an Accounting Request Start or Stop Message triggers the transmission of an Accounting Request Answer Message by the DIAMETER accounting server. The Answer message contains a sub-set of the DIAMETER-specific AVPs contained in the Start and Stop message whose receipt is being acknowledged.

The Accounting Request Answer Message consists of a DIAMETER header and a series of AVPs as follows.

```
+--------------------------------+
|                                |
| DIAMETER Header                |
| Command Code = 271             |
| Request Bit  = 0               |
|                                |
+--------------------------------+
| Session-Id                     |
+--------------------------------+
| Result-Code                    |
+--------------------------------+
| Origin-Host                    |
+--------------------------------+
| Origin-Realm                   |
+--------------------------------+
| Accounting-Record-Type = 2     |  response to Start message
| Accounting-Record-Type = 4     |  response to Stop message
+--------------------------------+
| Accounting-Record-Number       |  from Start or Stop message
+--------------------------------+
| Account-Application-Id         |
+--------------------------------+
```

Refer to DIAMETER AVPs for descriptions of the attributes contained in the Accounting Request Answer Message.

DIAMETER AVPs

This section provides a description of all DIAMETER AVPs used in support of the IPsec accounting application. AVPs are listed in alphabetical order.

Accounting-Input-Octets

DIAMETER AVP Code 363

Found in DIAMETER Accounting Request Stop Messages

Defined in Section 10.1 of RFC 4004, *DIAMETER Mobile IPv4 Application*.

Contains the number of octets received during the tunnel lifetime.

Accounting-Input-Packets

DIAMETER AVP Code 365

Found in DIAMETER Accounting Request Stop Messages

Defined in Section 10.4 of RFC 4004, *DIAMETER Mobile IPv4 Application*.

Contains the number of packets received during the tunnel lifetime.

Accounting-Output-Octets

DIAMETER AVP Code 364

Found in DIAMETER Accounting Request Stop Messages

Defined in Section 10.2 of RFC 4004, *DIAMETER Mobile IPv4 Application*.

Contains the number of octets transmitted during the tunnel lifetime.

Accounting-Output-Packets

DIAMETER AVP Code 366

Found in DIAMETER Accounting Request Stop Messages

Defined in Section 10.5 of RFC 4004, *DIAMETER Mobile IPv4 Application*.

Contains the number of packets transmitted during the tunnel lifetime.

Accounting-Record-Number

DIAMETER AVP Code 485

Found in DIAMETER Accounting Request Start, Stop, and Answer Messages

Defined in Section 9.8.3 of RFC 3588, *Diameter Base Protocol*.

Generated by the Security Gateway, contains an Unsigned32 integer that provides a sequence number for the DIAMETER session.

Accounting-Record-Type

DIAMETER AVP Code 480

Found in DIAMETER Accounting Request Start, Stop, and Answer Messages

Defined in Section 9.8.1 of RFC 3588, *Diameter Base Protocol*.

Generated by the Security Gateway, contains an enumerated value that identifies the message type. Relevant values are as follows:

2 — identifies an Accounting Request Start message

4 — identifies an Accounting Request Stop message

Acct-Application-Id

DIAMETER AVP Code 259

Found in DIAMETER Accounting Request Start, Accounting Request Stop, Accounting Request Answer, Capabilities Exchange Request, and Capabilities Exchange Answer Messages

Defined in Section 6.9 of RFC 3588, *Diameter Base Protocol*.

Generated by the Security Gateway, contains an Unsigned32 integer (3) that identifies accounting messages.

Acct-Session-Time

DIAMETER AVP Code 46

Found in DIAMETER Accounting Request Stop Messages

Defined in Section 10.1 of RFC 4004, *DIAMETER Mobile IPv4 Application*.

Contains the tunnel lifetime, from creation to tear-down, in seconds.

Calling-Station-Id

DIAMETER AVP Code 31

Found in DIAMETER Accounting Request Start and Stop Messages.

Defined in Section 4.6 of RFC 4005, *DIAMETER Network Access Server Application*.

Generated by the Security Gateway, contains the IPv4 address of the IKE interface that supports the IPsec tunnel.

Destination-Host

DIAMETER AVP Code 293

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 6.5 of RFC 3588, *Diameter Base Protocol*.

Derived by the Security Gateway from the contents of the *Origin-Host* AVP in CEA, contains a DiameterIdentity that specifies the hostname of the DIAMETER accounting server.

Destination-Realm

DIAMETER AVP Code 283

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 6.6 of RFC 3588, *Diameter Base Protocol*.

Derived by the Security Gateway from the contents of the *Origin-Realm* AVP in CEA, contains a DiameterIdentity that specifies the realm of the DIAMETER accounting server.

Event-Timestamp

DIAMETER AVP Code 55

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 8.21 of RFC 3588, *Diameter Base Protocol*.

Generated by the Security Gateway, this AVP contains a value of type *Time*. In an Accounting Request Start Message, the timestamp specifies the time of tunnel creation. In a Accounting Request Stop Message, the timestamp specifies the time or tunnel tear-down. In both cases, the time is expressed as the number of elapsed seconds since January 1, 1990 00:00 UTC.

Firmware-Revision

DIAMETER AVP Code 267

Found in DIAMETER Capabilities Exchange Request and Capabilities Exchange Answer Messages

Defined in Section 5.3.3 of RFC 3588, *Diameter Base Protocol*.

Contains a Unsigned32 that provides the firmware revision of the product; for the Security Gateway contains a value of *0*.

Framed-IP-Address

DIAMETER AVP Code 8

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 6.11.1 of RFC 4005, *DIAMETER Network Access Server Application*.

Generated by the Security Gateway, contains the inner IPv4 address of the IPsec tunnel assigned by the responder.

Host-IP-Address

DIAMETER AVP Code 257

Found in DIAMETER Capabilities Exchange Request and Capabilities Exchange Answer Messages

Defined in Section 5.3.5 of RFC 3588, *Diameter Base Protocol*.

Generated by the Security Gateway, is of type Address contains the sender's IP address.

Origin-Host

DIAMETER AVP Code 264

Found in DIAMETER Accounting Request Start, Accounting Request Stop, Accounting Request Answer, Capabilities Exchange Request, and Capabilities Exchange Answer Messages

Defined in Section 6.3 of RFC 3588, *Diameter Base Protocol.*

Derived by the Security Gateway from configuration parameters, contains a DiameterIdentity that specifies the message originator.

Origin-Realm

DIAMETER AVP Code 296

Found in DIAMETER Accounting Request Start, Accounting Request Stop, Accounting Request Answer, Capabilities Exchange Request, and Capabilities Exchange Answer Messages

Defined in Section 6.4 of RFC 3588, *Diameter Base Protocol*.

Derived by the Security Gateway from configuration parameters, contains a DiameterIdentity that specifies the realm of the message originator.

Product-Name

DIAMETER AVP Code 269

Found in DIAMETER Capabilities Exchange Request, and Capabilities Exchange Answer Messages

Defined in Section 5.3.7 of RFC 3588, *Diameter Base Protocol*.

Contains a UTF8String that provides the vendor-assigned name for the product; for the Security Gateway contains *AcmeDiameterRf*.

Result-Code

DIAMETER AVP Code 268

Found in DIAMETER Accounting Request and Capabilities Exchange Answer Messages

Defined in Section 7.1 of RFC 3588, *Diameter Base Protocol*.

Generated by the DIAMETER accounting server, contains a Unsigned32 integer that indicates whether a particular request was completed successfully, or whether an error occurred.

1xxx integers report informational errors.

1001 — DIAMETER_MULTI_ROUND_AUTH

2xxx integers report success.

2001 — DIAMETER_SUCCESS
2002 — DIAMETER_LIMITED_SUCCESS

3xxx integers report protocol errors.

3001 — DIAMETER_COMMAND_UNSUPPORTED
3002 — DIAMETER_UNABLE_TO_DELIVER
3003 — DIAMETER_REALM_NOT_SERVER
3004 — DIAMETER_TOO_BUSY
3005 — DIAMETER_LOOP_DETECTED
3006 — DIAMETER_REDIRECT_INDICATION
3007 — DIAMETER_APPLICATION_UNSUPPORTED
3008 — DIAMETER_INVALID_HEADER_BITS
3009 — DIAMETER_INVALID_AVP_BITS
3010 — DIAMETER_UNKNOWN_PEER

4xxx integers report transient failures.

4001 — DIAMETER_AUTHENTICATION_REJECTED
4002 — DIAMETER_OUT_OF_SPACE
4003 — DIAMETER_ELECTION_LOST

5xxx integers report permanent failures.

5001 — DIAMETER_AVP_UNSUPPORTED
5002 — DIAMETER_UNLNOWN_SESSION_ID
5003 — DIAMETER_AUTHORIZATION_REJECTED
5004 — DIAMETER_INVALID_AVP_VALUE
5005 — DIAMETER_MISSING_AVP
5006 — DIAMETER_RESOURCES_EXCEEDED
5007 — DIAMETER_CONTRADICTING_AVPS
5008 — DIAMETER_AVP_NOT_ALLOWED
5009 — DIAMETER_AVP_OCCURS_TOO_MANY_TIMES
5010 — DIAMETER_NO_COMMON_APPLICATION
5011 — DIAMETER_UNSUPPORTED_VERSION
5012 — DIAMETER_UNABLE_TO_COMPLY
5013 — DIAMETER_INVALID_BIT_IN_HEADER
5014 — DIAMETER_INVALID_AVP_LENGTH
5015 — DIAMETER_INVALID_MESSAGE_LENGTH
5016 — DIAMETER_INVALID_AVP_BIT_COMBO
5017 — DIAMETER_NO_COMMON_SECURITY

Refer to Sections 7.1 through 7.1.5 of RFC 3588 for additional information regarding result codes.

Session-Id

DIAMETER AVP Code 263

Found in DIAMETER Accounting Request Start, Stop, and Answer Messages

Defined in Section 8.8 of RFC 3588, *Diameter Base Protocol*.

Generated by the Security Gateway, contains a UTF8String that provides a unique DIAMETER session identifier.

Termination-Cause

DIAMETER AVP Code 295

Found in DIAMETER Accounting Request Stop Messages

Defined in Section 9.3.5 of RFC 4005, *DIAMETER Network Access Server Application*.

Values are as follows:

User Request	11
Lost Carrier	12
Lost Service	13
Idle Timeout	14
Session Timeout	15
Admin Reset	16
Admin Reboot	17
Port Error	18
NAS Error	19
NAS Request	20
NAS Reboot	21
Port Unneeded	22
Port Preempted	23
Port Suspended	24
Service Unavailable	25
Callback	26
User Error	27
Host Request	28
Supplicant Restart	29
Reauthentication Failure	30
Port Reinit	31
Port Disabled	32

Tunnel-Client-Auth-Id

DIAMETER AVP Code 90

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 7.10 of RFC 4005, *DIAMETER Network Access Server Application*.

Generated by the Security Gateway, contains the name presented by the tunnel initiator during the authentication phase of tunnel establishment.

Tunnel-Client-Endpoint

DIAMETER AVP Code 66

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 7.4 of RFC 4005, *DIAMETER Network Access Server Application*.

Generated by the Security Gateway, contains either the outer IPv4 address or the FQDN of the tunnel initiator.

Tunnel-Server-Auth-Id

DIAMETER AVP Code 91

Found in DIAMETER Accounting Request Start and Stop Messages.

Defined in Section 7.11 of RFC 4005, *DIAMETER Network Access Server Application*.

Generated by the Security Gateway, contains the name presented by the tunnel responder during the authentication phase of tunnel establishment.

Tunnel-Server-Endpoint

DIAMETER AVP Code 67

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 7.5 of RFC 4005, *DIAMETER Network Access Server Application*.

Generated by the Security Gateway, contains either the outer IPv4 address or the FQDN of the tunnel responder.

Tunnel-Type

DIAMETER AVP Code 64

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 7.2 of RFC 4005, *DIAMETER Network Access Server Application*.

Generated by the Security Gateway, contains an Unsigned32 integer value that identifies the IPsec tunnel protocol. Relevant values are as follows:

6 — IP Authentication Header (AH)

9 — IP Encapsulating Security Payload (ESP)

User-Name

DIAMETER AVP Code 1

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 8.14 of RFC 3588, *Diameter Base Protocol*.

Contains an UTF8String that identifies the accounting application user, *acmepacket@wancom.com* for the Security Gateway.

Vendor-Id

DIAMETER AVP Code 260

Found in DIAMETER Capabilities Exchange Request and Answer Messages

Defined in Section 5.3,3 of RFC 3588, *Diameter Base Protocol*.

Contains an Unsigned32 integer value that contains an SMI Network Management Private Enterprise Code, *9148* for Acme Packet.

IPsec Accounting Configuration

Configuration of IPsec accounting consists of the following steps.

1. Configure protocol-specific IPsec accounting group or groups.
2. Configure an IPsec accounting group list.
3. Configure IPsec accounting parameter list or lists.
4. Assign default IPsec account parameter lists at the global level.
5. Assign an IPsec accounting parameter list to an IKEv2 interface.

IPsec Accounting Groups

An IPsec accounting group manages transactions between the Security Gateway, acting either as a RADIUS or DIAMETER client, and one or more protocol-specific accounting servers.

An IPsec accounting group consists of a group of client parameters specific to this Security Gateway, and an associated group of server parameters specific to one or more adjacent accounting servers.

You can configure multiple IPsec accounting groups. Only two groups, however, can be active at any one time — one to manage transactions with RADIUS servers, and a second to manage transactions with DIAMETER servers.

Use the following procedure to configure one or more IPsec accounting groups.

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

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)# account-group
   ACMEPACKET(account-group)#
   ```

2. Use the required **name** parameter to assign a unique name to this IPsec accounting group.

   Later, you will use this name to assign the IPsec accounting group to an IPsec accounting group list.

   ```
   ACMEPACKET(account-group)# name IPsecRADIUSaccounting
   ACMEPACKET(account-group)#
   ```

3. Use the **hostname** parameter to specify the hostname or IP address of this Security Gateway.

   In the absence of a specific hostname or IP address, the Security Gateway provides a default value of *localhost*.

   ```
   ACMEPACKET(account-group)# hostname ragnarok
   ACMEPACKET(account-group)#
   ```

4. Use the **protocol** parameter to specify the protocol used to transfer accounting transaction to associated accounting servers.

   Select *RADIUS* (the default) or *DIAMETER*.

   ```
   ACMEPACKET(account-group)# protocol radius
   ACMEPACKET(account-group)#
   ```

5. Use the **src-port** parameter to identify the local port which supports accounting transactions.

   Provide a port number within the range *1025* through *65535*, or retain the default value, *1813*.

   ```
   ACMEPACKET(account-group)# src-port 1813
   ACMEPACKET(account-group)#
   ```

6. Use the **strategy** parameter to identify the server-selection algorithm used to choose among multiple available accounting servers.

   Retain the default value (*hunt*) if only a single server is available.

   Available algorithms are *hunt*, *failover*, *roundrobin*, *fastestrtt*, and *fewestpending*.

   hunt — by default, selects accounting servers on the basis of IP address, with the lowest address being the preferred server. If the accounting server with the lowest IP address is online and operational, all transaction are sent to it. Otherwise, the accounting server with the second lowest IP address is selected. If the first and second accounting servers are offline or non-operational, the accounting server with the next lowest address is selected, and so on through the list of accounting servers.

   failover — by default, selects accounting servers on the basis of IP address, with the lowest address being the preferred server. If the accounting server with the lowest IP address is online and operational, all transactions are sent to it until a failure occurs. Upon detection of a failure, moves to the account server with the second lowest IP address, and uses that server until a failure occurs — and so on through the list of accounting servers.

   roundrobin — by default, selects accounting servers on the basis of IP address, with the lowest address being the preferred server. Selects each accounting server in order — in theory, evenly distributing accounting transactions to each server over time.

   fastestrtt — selects the accounting server that has the fastest round trip time (RTT) observed during transactions with all servers. RTT is defined as the interim between the client transmission and receipt of the server acknowledgement.

   fewestpending — selects the accounting server that has the fewest number of unacknowledged accounting transactions.

   Select an algorithm, or retain the default value (*hunt*).

   ```
   ACMEPACKET(account-group)# strategy hunt
   ACMEPACKET(account-group)#
   ```

7. Use the **account-servers** command to access *account-servers* configuration mode.

   While in this mode you supply the information required to access one or more adjacent accounting servers. Any server identified within *account-servers* configuration mode must support the protocol (RADIUS or DIAMETER) specified by the **protocol** parameter in *account-group* configuration mode.

   ```
   ACMEPACKET(account-group)# account-servers
   ACMEPACKET(account-server)#
   ```

   The following ACLI parameters (**hostname**, **port**, **state**, **min-round-trip**, **max-inactivity**, **restart-delay**, and **priority**) are required for both RADIUS and DIAMETER accounting servers.

8. Use the **hostname** parameter to specify the hostname (in FQDN format) or IP address of an adjacent accounting server.

```
ACMEPACKET(account-server)# hostname 172.30.0.6
ACMEPACKET(account-server)#
```

9. Use the **port** parameter to identify the accounting server port which supports accounting transactions.

   Provide a port number within the range *1025* through *65535*, or retain the default value, *1813*.

```
ACMEPACKET(account-server)# port 1813
ACMEPACKET(account-server)#
```

10. Use the **state** parameter to specify the availability of this accounting server.

    Select *enabled* (the default) or *disabled*.

    Only accounting servers that are in the enabled state are considered when running the server-selection algorithm.

```
ACMEPACKET(account-server)# state enabled
ACMEPACKET(account-server)#
```

11. Use the **min-round-trip** parameter to establish eligibility for participation in the *fastestrtt* server-selection algorithm. You can safely ignore this parameter, if the strategy parameter is set to any value other than *fastest*.

    min-round-trip specifies a threshold RTT value that the accounting server must exceed to be included the server-selection algorithm. For example, assuming a configured value of 1025, the accounting server must have recorded a client/server round trip transaction in 1025 milliseconds, or less, for inclusion in the server-selection algorithm.

    Supported values are within the range 1025 through 65535 milliseconds.

```
ACMEPACKET(account-server)# min-round-trip
ACMEPACKET(account-server)#
```

12. Use the **max-inactivity** parameter to indicate the length of time that the Security Gateway waits to receive a valid response from the accounting server.

    If this timer value is exceeded, the Security Gateway marks the unresponsive accounting server as *inactive*. It then runs its server-selection algorithm, excluding any inactive or disabled servers. Upon establishing a connection with a new accounting server, the Security Gateway transmits any unacknowledged or pending accounting transactions to the newly active server.

    Supported values are within the range 1 through 300 seconds, with a default value of *60*.

```
ACMEPACKET(account-server)# max-inactivity 30
ACMEPACKET(account-server)#
```

13. Use the **restart-delay** parameter to specify a quarantine period during which an accounting server marked as inactive is ineligible to participate in the server-selection algorithm.

    Supported values are within the range 1 through 300 seconds, with a default value of *30*.

```
ACMEPACKET(account-server)# restart-delay 45
ACMEPACKET(account-server)#
```

14. Use the **priority** parameter to alter the default IP-address-based server selection algorithm. This parameter is most relevant when the **strategy** parameter is *hunt* or *failover*; it is of less relevance (serving as a tie-breaker in the event of identical round-trip times, or identical unacknowledged message counts) when the strategy is *fastestrtt* or *fewestpending*; and least relevant when the strategy is *roundrobin*, which aims to load balance accounting transactions across available servers.

    With non-default prioritization enabled, the Security Gateway substitutes absolute integer values for IP addresses in the server-selection algorithm. For example, assuming the *failover* algorithm, the Security Gateway, upon detecting a failed active accounting server, will failover, not to server with the next lowest IP address, but to the server with the next highest priority as specified by this command.

    Retain the default value (*0*) to implement the default, IP-address-based selection algorithm. Otherwise, enter an integer value to specify this accounting server's priority — the lower the integer value, the higher the priority.

    ```
    ACMEPACKET(account-server)# priority 1
    ACMEPACKET(account-server)#
    ```

    The following three ACLI parameters (**bundle-vsa**, **nas-id**, and **secret**) are required for RADIUS configurations only. They can be safely ignored by DIAMETER users.

15. Use the **bundle-vsa** parameter to enable the bundling of Acme Packet Vendor Specific Attributes (VSA), resulting in a sequence of Acme Packet VSAs conveyed within a single RADIUS Vendor-Specific attribute (Type 26).

    Retain the default value (**enable**) to enable VSA bundling; use the alternative value to disable bundling — meaning that each individual Acme Packet VSA is conveyed in an individual RADIUS Type 26 attribute.

    ```
    ACMEPACKET(account-server)# bundle-vsa enable
    ACMEPACKET(account-server)#
    ```

16. Use the **secret** parameter to specify the shared-secret known to the Security Gateway and the RADIUS accounting server.

    The shared-secret is used to calculate the contents of the Authenticator field of Accounting Request Packets generated by the Security Gateway as a RADIUS client.

    ```
    ACMEPACKET(account-server)# secret
    IamsoftsiftInanhourglassatthewallFastbutminedwithamotionadrift
    ACMEPACKET(account-server)#
    ```

17. Use the **nas-id** parameter to specify the contents of the RADIUS NAS-Identifier attribute (Type 32), which is included in Accounting Request packets generated by the Security Gateway.

    This attribute contains a unique string that identifies the Security Gateway to the associated RADIUS accounting server.

    Enter a unique Security Gateway identifier, for example its FQDN.

    ```
    ACMEPACKET(account-server)# nas-id ragnarok.acmepacket.com
    ACMEPACKET(account-server)#
    ```

18. Use **done**, **exit**, and **verify-config** to complete configuration of this IPsec accounting group.

19. Repeat Steps 1 through 18 to configure additional IPsec accounting groups, keeping in mind that only two such groups, one RADIUS-based and the other DIAMETER-based, can be simultaneously active.

IPsec Accounting Group Lists

An IPsec accounting group list enables the assignment of IPsec accounting groups to the IKEv2 protocol. As the assignment is done at a global level, the assigned IPsec accounting groups are available to individual IKEv2 interfaces.

An IPsec accounting group list contains either one or two entries, with each entry identifying an existing IPsec accounting group. An IPsec accounting group list can contain:

- one RADIUS-based accounting group
- one DIAMETER-based accounting group
- one RADIUS-based and one DIAMETER-based accounting group

Use the following procedure to configure one or more IPsec accounting group lists.

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

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-config
   ACMEPACKET(ike-config)#
   ```

2. Use the **account-group-list** parameter to designate one or two existing IPsec accounting groups as available to support IPsec accounting transactions.

   Use double quotes to bracket parameter arguments if two IPsec accounting groups are being made available; leave a space between group names.

   This command makes one presumably RADIUS-based IPsec accounting group available. DIAMETER-based service is unavailable.

   ```
   ACMEPACKET(ike-config)# account-group-list IPsecRADIUSaccounting
   ACMEPACKET(account-group)#
   ```

   This command makes one presumably DIAMETER-based IPsec accounting group available. RADIUS-based service is unavailable.

   ```
   ACMEPACKET(ike-config)# account-group-list IPsecDIAMETERaccounting
   ACMEPACKET(account-group)#
   ```

   This command makes one RADIUS-based and one DIAMETER-based IPsec accounting groups available.

   ```
   ACMEPACKET(ike-config)# account-group-list "IPsecRADIUSaccounting
   IPsecDIAMETERaccounting"
   ACMEPACKET(account-group)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete configuration and assignment of this IPsec accounting group list.

IPsec Accounting Parameter Lists

An IPsec accounting parameter list identifies specific events in the lifetime of an IPsec tunnel that trigger an accounting transaction. Additionally, for RADIUS-based accounting service only, the list identifies an interval at which the Security Gateway issues interim accounting transactions as described in Accounting-Request Packets.

Use the following procedure to configure one or more IPsec accounting parameter lists.

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

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-accounting-param
   ACMEPACKET(ike-accounting-param)#
   ```

2. Use the required **name** parameter to assign a unique name to this IPsec accounting parameter list.

   Later, you will use this name to assign the IPsec accounting parameter list to an IKEv2 interface.

   ```
   ACMEPACKET(ike-accounting-param)# name RADIUSall
   ACMEPACKET(ike-accounting-param)#
   ```

3. Use the **radius-accounting-events** parameter to specify IPsec events that trigger an IPsec accounting transaction.

   Supported values are:

none	disables RADIUS-based IPsec Accounting.
early-start	triggers an Accounting Request Start packet on initiation of IKEv2 SA negotiation.
start	triggers an Accounting Request Start packet on tunnel establishment.
stop	triggers an Accounting Request Stop packet on tunnel tear-down.
interim-ipsec-rekey	triggers an Accounting Request Interim-Update packet on IPsec tunnel re-keying.
interim-ike-rekey	triggers an Accounting Request Interim-Update packet on IKEv2 Security Association re-keying.

   The *early-start* and *start* events are mutually exclusive; you can select only one start event.

   If *early-start* is selected, the Security Gateway schedules two accounting transactions. The first transaction is an Accounting Request Start packet triggered by the start of IKEv2 SA negotiation. The second transaction depends on the success or failure of tunnel establishment. Successful tunnel establishment triggers an Interim-Update packet that provides the tunnel details usually found in the standard Accounting Request Start packet. Tunnel failure triggers an Accounting Request Stop packet.

Use double quotes to bracket parameter arguments if multiple events trigger accounting transaction; leave a space between event names.

This command triggers an accounting transaction for four reportable events.

```
ACMEPACKET(ike-accounting-param)# radius-accounting-records “start
stop interim_ipsec_rekey interim_ike_rekey”
ACMEPACKET(ike-accounting-param)#
```

4. Use the **diameter-accounting-events** parameter to specify specific IPsec events that trigger an IPsec accounting exchange.

   Supported values are:

none	disables DIAMETER-based IPsec Accounting
start	triggers an Accounting Request Start packet on tunnel establishment
stop	triggers an Accounting Request Stop packet on tunnel tear-down
interim-ipsec-rekey	not supported in this current release. Support scheduled for inclusion in a subsequent release.
interim-ike-rekey	not supported in this current release. Support scheduled for inclusion in a subsequent release.

   Use double quotes to bracket parameter arguments if multiple events trigger accounting transaction; leave a space between event names.

   This command triggers an accounting transaction for all reportable events.

   ```
   ACMEPACKET(ike-accounting-param)# diameter-accounting-records “start
   stop”
   ACMEPACKET(ike-accounting-param)#
   ```

5. For RADIUS-based IPsec accounting only, use the **intermediate-period** parameter to specify the interval at which the Security Gateway generates Accounting Request Interim-Update packets.

   Supported values are integers within the range 0 (the default) through 65535. The default value (0) disables the generation of interim packets. Any non-default value, within the allowable range, specifies the frequency, in seconds, of interim updates.

   Any value less than 60 generates a warning that such frequent transactions can impact system performance.

   ```
   ACMEPACKET(ike-accounting-param)# intermediate-period 300
   ACMEPACKET(ike-accounting-param)#
   ```

6. Use **done**, **exit**, and **verify-config** to complete configuration of this IPsec accounting parameter list.

7. Repeat Steps 1 through 6 to configure additional IPsec accounting parameter lists.

After configuring IPsec accounting parameter lists, you complete IPsec accounting configuration by assigning accounting parameter lists to IKEv2 interfaces. The locally-assigned IPsec accounting parameter list, which identifies what events are reported, works in conjunction with the globally-assigned IPsec account groups list, which identifies where events are reported.

Use the following procedure to assign an IPsec accounting parameter list to an IKEv2 interface.

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

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-interface
   ACMEPACKET(ike-interface)#
   ```

2. Use the **accounting-params-name** parameter to assign an IPsec accounting parameters list to a selected IKEv2 interface.

   Later, you will use this name to assign the IPsec accounting parameter list to an IKEv2 interface.

   ```
   ACMEPACKET(ike-interface)# accounting-params-name RADIUSall
   ACMEPACKET(ike-interface)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete assignment of the IPsec accounting parameter list to the current IKEv2 interface.

4. Repeat Steps 1 through 3 to assign an accounting parameter list to other IKEv2 interfaces.

DIAMETER Accounting on Media Interfaces

Rather than limiting accounting functions to management interfaces only, you can enhance the Security Gateway's accounting capability by creating accounting connections from the Security Gateway's media interfaces to external accounting servers. These connections are created by installing NAT flows for your accounting server(s) on the Security Gateway.

To enable accounting services on media interfaces, you set the **origin-realm** parameter in the **account-server** configuration to a valid realm and thereby enable the connection. The connection allows NAT flows (TCP for DIAMETER and UDP for RADIUS) to be installed when the Security Gateway loads its NAT table. If you do not set the applicable parameters with valid service interface names, the Security Gateway does not install NAT flows and continues to conduct accounting using just the management interface.

To facilitate accounting services on media interfaces, the Security Gateway supports two DIAMETER device watchdog messages used to detect transport failures proactively -- Device-Watchdog-Request (DWR) and Device-Watchdog-Answer (DWA). The Security Gateway processes incoming DWR messages and responds with a DWA reply. The Security Gateway does not initiate DWRs.

ACLI Instructions and Examples

This section shows you how to configure DIAMETER accounting for service interfaces using the accounting group and realm configurations.

To configure an accounting group for DIAMETER accounting service interface support:

1. From Superuser mode, use the following command sequence to access the applicable configuration.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)# account-group
   ACMEPACKET(account-group)# account-server
   ACMEPACKET(account-server)#
   ```

2. Use the **select** command to access the target the account server you want to edit.
3. **origin-realm**—Enter the name of a valid realm for which you want to enable accounting capability.

   ```
   ACMEPACKET(account-server)# origin-group diam-acct-realm
   ```

4. Type **done** to save your work, and then continue.

To configure a realm for DIAMETER accounting service interface support:

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

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

2. Use the **select** command to access the target the realm configuration you want to edit.
3. **identifier**—Enter the name of a valid service interface for which you want to enable accounting capability; this name matches the one you entered in the **account-server** configuration.

   ```
   ACMEPACKET(realm-config)# identifier diam-acct-realm
   ```

4. **network-interfaces** —Specify the service interface that the accounting connection is desired to be made from.

   ```
   ACMEPACKET(realm-config)# network-interfaces core2:0
   ```

5. Use **done**, **exit**, and **verify-config** to complete the process.

IKEv2 DDoS Protection

IKEv2-Based DDoS Attacks

Given its usual location at the network edge, and the two stage negotiation process required for the establishment of IPsec tunnels, the Security Gateway can be a target of IKEv2-based DDoS (distributed denial of service) attacks. Such attacks, which seek to overwhelm or monopolize system resources to the detriment of the gateway's functionality, can take several forms including:

- prolonging/failing to complete negotiation of the IKEv2 Security Association (SA)
- prolonging/failing to complete negotiation of the IPsec SA
- excessive renegotiation/rekeying of an established IKEv2 SA
- excessive renegotiation/rekeying of an established IPsec SA
- sabotaging the IKEv2 negotiation by failing to present a valid cookie when required to do so
- sabotaging the IKEv2 negotiation by failing to present valid credentials when required to do so

The Security Gateway provides protection against DDoS attacks by monitoring IKEv2 signaling traffic from remote endpoints (defined by an IP address:UDP port pair). All endpoints start in the *allowed* state, meaning that IKEv2 signaling received from the endpoint is accepted as valid by the Security Gateway. A group of policing parameters, and associated counters, provide protection against DDoS attacks by monitoring IKEv2 signaling from individual endpoints. The Security Gateway maintains a set of counter for each endpoint. The counters record instances of suspect traffic, which may indicate malicious intent, and periodically compare endpoint-specific traffic counts to threshold values established by the policing parameters. If endpoint counts meet or exceed threshold values, the Security Gateway places the endpoint in the *deny* state, and, if they exist, tears down the IKEv2 SA and IPsec SA associated with the endpoint. While in the deny state, the endpoint is quarantined and refused all access to the IKEv2 interface. The endpoint remains quarantined until the expiration of a pre-set timer. At timer expiration, the endpoint is transitioned to the allowed state, and granted IKEv2 interface access.

IKEv2 DDOS Protection Configuration

Configuration of IKEv2 DDOS protection consists of the following steps.

1. Configure one or more IKEv2 Access Control Templates.

   An IKEv2 Access Control Template enables protection against a DDOS attack, and provides a set of configurable timers and policing parameters used to monitor and squelch suspect IKEv2 traffic.

Two parameters set user-configurable timers; **tolerance-window** sets the interval between periodic checks of suspect traffic counts, and **deny-period** specifies the duration of the deny state.

Additional parameters (pre-ipsec-invalid-threshold, **pre-ipsec-maximum-threshold**, **invalid-cookie-threshold**, **post-ipsec-invalid-threshold**, **pre-ipsec-maximum-threshold**, and **auth-failure-threshold**) set threshold counts for suspect traffic that may be malicious in nature.

2. Assign a template to an IKEv2 interface.

   Assignment of a template to an IKEv2 interface enables protection against a DDOS attack on that specific interface.

Construct an IKEv2 Access Control Template

Use the following procedure to construct an IKEv2 Access Control Template.

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

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-access-control
   ACMEPACKET(ike-access-control)#
   ```

2. Use the required **name** parameter to assign a unique name to this IKEv2 Access Control Template instance.

   You will use this name as an identifier when assigning the template to a specific IKEv2 interface.

   ```
   ACMEPACKET(ike-access-control)# name ikev2-ddos-1
   ACMEPACKET(ike-access-control)#
   ```

3. Use the **state** parameter to enable or disable this template instance.

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

   ```
   ACMEPACKET(ike-access-control)# state enabled
   ACMEPACKET(ike-access-control)#
   ```

4. Use the **tolerance-window** parameter to specify the interval (in seconds) between checks of endpoint-specific traffic counters.

   At the specified interval, the Security Gateway checks the value of each of the counters associated with one of the policing parameters. If the counter value is less than the threshold value set by the policing parameter, the counter is cleared, and the endpoint remains is the *allowed* state. If the counter value is equal to or greater than the threshold value, the counter is cleared, and the endpoint is placed in the *deny* state.

   tolerance-window and **deny-period** must both be set to non-zero values to enable IKEv2 DDOS protection.

   Supported values are integers within the range *0*, the default, through *999999999* (packets). The default value, *0*, specifies that no IKEv2 DDOS protection is enabled.

   ```
   ACMEPACKET(ike-access-control)# tolerance-window 100
   ACMEPACKET(ike-access-control)#
   ```

5. Use the **deny-period** parameter to specify the quarantine period imposed on an endpoint that transitions to the *deny* state. During the quarantine period, the endpoint is denied all access to the IKEv2 interface.

   deny-period and **tolerance-window** must both be set to non-zero values to enable IKEv2 DDOS protection.

   Supported values are integers within the range *0* through *999999999*, with a default value of *30* (seconds).

   ```
   ACMEPACKET(ike-access-control)# deny-period 50
   ACMEPACKET(ike-access-control)#
   ```

6. Use the **pre-ipsec-invalid-threshold** parameter to enable protection against a DDOS attack that sends malformed, or otherwise invalid, packets during the IKEv2 SA negotiation process.

   pre-ipsec-invalid-threshold specifies the maximum number of malformed IKEv2 SA packets tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks can attempt to consume system resources in a futile effort to complete negotiation of IKEv2 SAs.

   If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

   Supported values are integers within the range *0*, the default, through *999999999* (packets). The default value, *0*, specifies that protection against malformed or invalid IKEv2 SA negotiation packets is not enabled.

   ```
   ACMEPACKET(ike-access-control)# pre-ipsec-invalid-threshold 10
   ACMEPACKET(ike-access-control)#
   ```

7. Use the **pre-ipsec-maximum-threshold** parameter to enable protection against an IKEv2 DDOS attack that sends excessive packets during the IKEv2 SA negotiation process.

   pre-ipsec-maximum-threshold specifies the maximum number of valid IKEv2 SA packets tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks can attempt to prolong the IKEv2 negotiation by persistently renegotiating the IKEv2 SA.

   If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

   Supported values are integers within the range *0*, the default, through *999999999* (packets). The default value, *0*, specifies that protection against valid, but excessive, IKEv2 SA negotiation packets is not enabled.

   ```
   ACMEPACKET(ike-access-control)# pre-ipsec-maximum-threshold 100
   ACMEPACKET(ike-access-control)#
   ```

8. Use the **invalid-cookie-threshold** parameter to enable protection against an IKEv2 DDOS attack that prolongs the IKEv2 SA negotiation process by having the endpoint deliberately fail to follow required protocol behavior, as defined in Section 2.6 of RFC 4306, *Internet Key Exchange (IKEv2) Protocol*.

   During the IKEv2 negotiation process, the Security Gateway can issue an *IKE_SA_INIT* response that contains a cookie notification payload. The payload mandates that the endpoint retry IKEv2 SA negotiation with the cookie value as the first payload in its response to the *IKE_SA_INIT*.

   invalid-cookie-threshold specifies the maximum number of packets containing an erroneous cookie value tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter.

   If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

   Supported values are integers within the range *0*, the default, through *999999999* (packets). The default value, *0*, specifies that protection against erroneous cookie responses is not enabled.

   ```
   ACMEPACKET(ike-access-control)# invalid-cookie-threshold 10
   ACMEPACKET(ike-access-control)#
   ```

9. Use the **post-ipsec-invalid-threshold** parameter to enable protection against an IKEv2 DDOS attack that sends malformed IKEv2 SA packets after the establishment of an IPsec tunnel.

   post-ipsec-invalid-threshold specifies the maximum number of malformed, or otherwise invalid, IKEv2 SA packets tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks can attempt to consume system resources in a futile effort to renegotiate the IKEv2 SA.

   If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

   Supported values are integers within the range *0*, the default, through *999999999* (packets). The default value, *0*, specifies that protection against malformed or invalid IKEv2 SA renegotiation packets is not enabled.

   ```
   ACMEPACKET(ike-access-control)# post-ipsec-invalid-threshold 10
   ACMEPACKET(ike-access-control)#
   ```

10. Use the **post-ipsec-maximum-threshold** parameter to enable protection against an IKEv2 DDOS attack that sends valid, but excessive, IKEv2 SA packets after the establishment of an IPsec tunnel.

    post-ipsec-maximum-threshold specifies the maximum number of valid IKEv2 SA packets tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks can attempt to consume system resources by persistently renegotiating the IKEv2 SA.

    If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

    Supported values are integers within the range *0*, the default, through *999999999* (packets). The default value, *0*, specifies that protection against valid, but excessive, IKEv2 SA negotiation packets is not enabled.

    ```
    ACMEPACKET(ike-access-control)# post-ipsec-maximum-threshold 1000
    ACMEPACKET(ike-access-control)#
    ```

11. Use the **auth-failure-threshold** parameter in conjunction with **auth-failure-threshold-report** to enable protection against an IKEv2 DDOS attack that attempts to consume system resources by overwhelming the authentication function.

    auth-failure-threshold specifies the maximum number of failed authentication attempts tolerated from a specific endpoint within the interval set by the **tolerance-window** parameter. These attacks attempt to consume system resources by persistently presenting invalid credentials during the endpoint authentication process.

    If this threshold value is reached, the endpoint is quarantined for an interval defined by the **deny-period** parameter.

    Supported values are integers within the range *0*, the default, through *999999999* (authentication attempts). The default value, *0*, specifies that protection against invalid authentications is not enabled.

    ```
    ACMEPACKET(ike-access-control)# auth-failure-threshold 10
    ACMEPACKET(ike-access-control)#
    ```

12. Use the **auth-failure-threshold-report** parameter in conjunction with the **auth-failure-threshold** to enable protection against an IKEv2 DDOS attack that attempts to consume system resources by overwhelming the authentication function.

    auth-failure-threshold-report specifies how failed authentications are reported. Supported values are:

no-reporting	(the default), authentication failures are not reported
snmp-trap-only	authentication failures are reported by generating an SNMP trap (refer to SNMP Trap for information of trap structure)
syslog-only	authentication failures are reported by sending a syslog message
snmp-trap-and-syslog	authentication failures are reported with both an SNMP trap and a syslog message

    ```
    ACMEPACKET(ike-access-control)# auth-failure-threshold-report snmp-
    trap-only
    ACMEPACKET(ike-access-control)#
    ```

13. Use **done**, **exit**, and **verify-config** to complete configuration of the IKEv2 Access Control Template.

14. Repeat Steps 1 through 13 to complete additional IKEv2 Access Control templates if required.

Assign Access Control Template to IKEv2 Interface

Use the following procedure to assign an IKEv2 Access Control Template to an IKEv2 interface. The template assignment enables IKEv2 DDOS protection on the interface.

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

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-interface
ACMEPACKET(ike-interface)#
```

2. Use the **select** command to specify the interface to which the IKEv2 Access Control Template will be assigned.

```
ACMEPACKET(ike-interface)# select
<address>:
172.30.1.150
172.30.1.151
172.30.55.127

selection: 1
ACMEPACKET(ike-interface)#
```

3. Use the **access-control-name** parameter to assign an IKEv2 Access Control Template to the interface.

```
ACMEPACKET(ike-interface)# access-control-name ikev2-ddos-1
ACMEPACKET(ike-interface)#
```

4. Use **done**, **exit**, and **verify-config** to complete IKEv2 Access Control Template assignment.

SNMP Trap

Violation of the authenticate failure threshold can generate an SNMP trap that includes the endpoint's ID or MSISDN (Mobile Station International Subscriber Directory Number), its IP address, and port number.

The SNMP trap is defined as follows:

```
apSecurityAuthFailureThresholdNotification  NOTIFICATION-TYPE
OBJECTS  {
          apSecurityUser,
          apSecurityPeerIpAddress,
          apSecurityPeerPort
         }
STATUS   current
DESCRIPTION
"This notification will be generated when IKE DDOS
auth-failure-threshold is reached and report mode includes SNMP
trap"
```

IKE counters that track suspected IKEv2 DDOS attacks are not synchronized with the standby Security Gateway. Endpoint deny status, however, is synchronized with the standby.

Pre-Populated ARP Table

In certain deployments remote IPsec endpoints may require access to core network hosts on one or more of the Security Gateways core interfaces. In these instances, the Security Gateway receives the tunneled packet, masks the received IP destination address against those of its core interfaces to determine if local delivery is possible, and, if so, dynamically ARPs for the physical destination address.

This process can be expedited by pre-populating the core-interface-specific ARP table with a list of commonly accessed hosts reachable by that interface. With the ARP table pre-populated with IP addresses, the ARP process issues ARP requests at 5 second intervals until a response is received. Once the pre-populated IP address has been resolved, periodic ARP refreshes are used to maintain the currency of the resolution.

ACLI Configuration

Use the following procedure to pre-populate a core-interface-specific ARP table.

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

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# system
ACMEPACKET(system)# network-interface
ACMEPACKET(network-interface)#
```

2. Use the **select** command to specify the network-interface to which the pre-populated ARP table will be assigned.

```
ACMEPACKET(network-interface)# select
<address>:
...
2. core1:0     99.0.0.100   gw=99.100.100.5
...
selection: 2
ACMEPACKET(network-interface)#
```

3. Use the **add-neighbor-ip** parameter to add the initial IP address to the core-interface-specific ARP table.

```
ACMEPACKET(network-interface)# add-neighbor-ip 99.0.0.101
ACMEPACKET(network-interface)#
```

4. Repeat step 3 to add an additional IP address to the core-interface-specific ARP table.

   You can add a maximum of ten IP addresses to a single network interface.

```
ACMEPACKET(network-interface)# add-neighbor-ip 99.0.0.101
ACMEPACKET(network-interface)#
```

5. Use the **show** command to examine the pre-populated ARP table, referred to as the neighbor list.

```
ACMEPACKET(network-interface)# show
network-interface
        name                            core1
        sub-port-id                     0
        description
        hostname
        ip-address                      99.0.0.100
        pri-utility-addr
        sec-utility-addr
        netmask
        gateway                         99.100.100.5
        ...
        ...
        ...
        neighbor-list                   99.0.0.101
                                        99.0.0.102
                                        99.0.0.103
                                        99.0.0.104
                                        99.0.0.105
                                        99.0.0.106
                                        99.0.0.107
                                        99.0.0.108
                                        99.0.0.109
                                        99.0.0.110
        last-modified-by                admin@console
        last-modified-date              2011-11-21 15:57:30
ACMEPACKET(network-interface)#
```

6. Use **done**, **exit**, and **verify-config** to complete configuration of the pre-populated ARP table.
7. Repeat Steps 1 through 6 to complete configure pre-populated ARP table for other core-side network interfaces as required.

After creating the pre-populated ARP table, use the **add-neighbor-ip** and **remove-neighbor-ip** parameters to change table contents as may be required.

For example:

```
ACMEPACKET(network-interface)# remove-neighbor-ip 99.0.0.101
ACMEPACKET(network-interface)# remove-neighbor-ip 99.0.0.102
ACMEPACKET(network-interface)# remove-neighbor-ip 99.0.0.109
ACMEPACKET(network-interface)# remove-neighbor-ip 99.0.0.110
ACMEPACKET(network-interface)# add-neighbor-ip 99.0.0.111
ACMEPACKET(network-interface)# add-neighbor-ip 99.0.0.113
ACMEPACKET(network-interface)# add-neighbor-ip 99.0.0.115
ACMEPACKET(network-interface)# add-neighbor-ip 99.0.0.117
```

```
ACMEPACKET(network-interface)# show
network-interface
        name                            core1
        sub-port-id                     0
        description
        hostname
        ip-address                      99.0.0.100
        pri-utility-addr
        sec-utility-addr
        netmask
        gateway                         99.100.100.5
        ...
        ...
        ...
        neighbor-list                   99.0.0.103
                                        99.0.0.104
                                        99.0.0.105
                                        99.0.0.106
                                        99.0.0.107
                                        99.0.0.108
                                        99.0.0.111
                                        99.0.0.113
                                        99.0.0.115
                                        99.0.0.117
        last-modified-by                admin@console
        last-modified-date              2011-11-21 16:05:38
ACMEPACKET(network-interface)#
```

Appendix A - MIB/SNMP Quick Reference

Overview

This chapter provides an overview of MIB/SNMP support for MSG features discussed in the Oracle Communications Security Gateway Essentials Guide Release Version M-CX2.0.0 GA. Further MIB/SNMP support for capabilities common to both SG and SBC models (for example RADIUS and DIAMETER services) is available in the MIB Reference Guide, release version 6.2.0. Please refer to that document as needed.

Enterprise Traps

The following table identifies the MSG proprietary traps that the Net-Net system supports.

apSecurityAuthFailureThresholdNotification	Generated when IKE DDos auth-failure-threshold is reached
apSecurityCrlInvalidNotification	Generated when an invalid CRL is detected
apSecurityIPsecTunCapNotification	Generated when the percentage of licensed IPsec tunnels exceeds an IPsec tunnel alarm threshold. apSecurityIPsecTunCapPct object indicates the current percentage
apSecurityIPsecTunCapClearNotification	Generated when the percentage of licensed IPsec tunnels no longer exceeds an IPsec tunnel alarm threshold. apSecurityIPsecTunCapPct object indicates the current percentage
apSecurityRadiusFailureNotification	Generated when a Radius authentication request fails
apSecurityTunnelFailureNotification	Generated when an IPSec IKEv2 tunnel cannot be established
apSecurityTunnelDPDNotification	Generated when an IPSec IKEv2 tunnel fails because of Dead Peer Detection (DPD)
apSecurityCRLRetrieveFailNotification	Generated when retrieval of CRL fails.
apSecurityCRLRetrieveClearNotification	Generated when retrieval of CRL that once failed is successful.
apSysMgmtTcaClearTrap	Generated when a Threshold Crossing Alert counter falls below the lowest configured TCA reset threshold value
apSysMgmtTcaTrap	Generated when a Threshold Crossing Alert counter crosses a configured TCA threshold

Acme Packet System Management MIB (ap-smgmt.mib)

SNMP GET Query Name	Object Identifier Name: Number	Description
Object Identifier Name: apSystemManagementModule (1.3.6.1.4.1.9148.3.2)		
Object Identifier Name: apSysMgmtMonitorObjects (1.3.6.1.4.1.9148.3.2.5)		
apSysMgmtTcaOid	apSysMgmtMonitorObjects: 1.3.6.1.4.1.9148.3.2.5.65	The Object ID for the Threshold Crossing Alert counter that is changing alert level
apSysMgmtTcaCurrent	apSysMgmtMonitorObjects: 1.3.6.1.4.1.9148.3.2.5.66	The current value of the counter associated with the TCA
apSysMgmtTcaMinorThreshold	apSysMgmtMonitorObjects: 1.3.6.1.4.1.9148.3.2.5.67	The current configured TCA minor threshold value
apSysMgmtTcaMajorThreshold	apSysMgmtMonitorObjects: 1.3.6.1.4.1.9148.3.2.5.68	The current configured TCA major threshold value
apSysMgmtTcaCriticalThreshold	apSysMgmtMonitorObjects: 1.3.6.1.4.1.9148.3.2.5.69	The current configured TCA critical threshold value

Acme Packet License MIB (ap-license.mib)

SNMP GET Query Name	Object Identifier Name: Number	Description
Object Identifier Name: apLicenseEntry (1.3.6.1.4.1.9148.3.5.1.1.1)		
apLicenseIKEFeature	apLicenseEntry: 1.3.6.1.4.1.9148.3.5.1.1.1.18	Value that indicates whether an IKE license is present. A value of True indicates that IKE licensing is enabled. A value of False indicates that IKD licensing is not enabled
apLicenseIPsecTunCapPct	apLicenseEntry: 1.3.6.1.4.1.9148.3.5.1.1.1.19	Total IPsec tunnel capacity

Acme Packet Security MIB (ap-security.mib)

SNMP GET Query Name	Object Identifier Name: Number	Description
Object Identifier Name: apSecurityMIBObjects (1.3.6.1.4.1.9148.3.9.1)		
apSecurityIPsecTunCount	apSecurityMIBObjects: 1.3.6.1.4.1.9148.3.9.1.1	Number of IPsec tunnels currently in progress
apSecurityIPsecTunCapPct	apSecurityMIBObjects: 1.3.6.1.4.1.9148.3.9.1.2	Percentage of licensed IPsec tunnels currently in progress
apSecurityCrlIssuer	apSecurityMIBObjects: 1.3.6.1.4.1.9148.3.9.1.7	CRL issuer name
apSecurityCspName	apSecurityMIBObjects: 1.3.6.1.4.1.9148.3.9.1.8	Cert-status-profile object for retrieving the CRL. If the CRL is loaded from a local file, the value is specified as File.

SNMP GET Query Name	Object Identifier Name: Number	Description
Object Identifier Name: apSecurityIkeInterfaceStatsTable (1.3.6.1.4.1.9148.3.9.1.3)		
Object Identifier Name: apSecurityIkeInterfaceStatsEntry (1.3.6.1.4.1.9148.3.9.1.3.1)		
apSecurityIkeInterfaceType	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.1	IP address version type of the IKE interface
apSecurityIkeInterfaceAddress	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.2	IP address of the IKE interface
apSecurityIkeInterfaceCpuOverloadErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.3	Number of CPU overload rejections on the IKE-interface
apSecurityIkeInterfaceInitCookieErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.4	Number of IKE COOKIE errors on the IKE interface
apSecurityIkeInterfaceAuthErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.5	Number of IKE AUTH payload errors on the IKE-interface
apSecurityIkeInterfaceEapAccessRequestErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.6	Number of IKE EAP access request errors on the IKE interface
apSecurityIkeInterfaceEapAccessChallengeErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.7	Number of IKE EAP access challenge errors on the IKE interface
apSecurityIkeInterfaceTsErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.8	Number of IKE TS errors on the IKE-interface
apSecurityIkeInterfaceCpErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.9	Number of IKE config payload errors on the IKE-interface
apSecurityIkeInterfaceKeErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.10	Number of IKE KE errors on the IKE interface
apSecurityIkeInterfaceProposalErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.11	Number of IKE proposal payload errors on the IKE interface
apSecurityIkeInterfaceSyntaxErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.12	Number of IKE syntax errors on the IKE interface
apSecurityIkeInterfaceCriticalPayloadErrors	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.13	Number of IKE critical payload errors on the IKE interface
apSecurityIkeInterfaceAuthFailureTca	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.14	Dummy object for the IKE authentication failure TCA. This object alerts the SD if IKE authentication failures cross a configured TCA threshold, and an apSysMgmtTcaTrap is generated. This object also monitors values for apSysMgmtTcaClearTrap
apSecurityIkeInterfaceTunnelRemovalsTca	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.15	Dummy object for the IPsec tunnel removals TCA. This object alerts the SD if IPsec tunnel removals cross a configured TCA threshold, and an apSysMgmtTcaTrap is generated. This object also monitors values for apSysMgmtTcaClearTrap
apSecurityIkeInterfaceDpdTca	apSecurityIkeInterfaceStatsEntry: 1.3.6.1.4.1.9148.3.9.1.3.1.16	Dummy object for the IKE dead peer detection TCA. This object alerts the SD if IKE dead peer detections cross a configured TCA threshold, and an apSysMgmtTcaTrap is generated. This object also monitors values for apSysMgmtTcaClearTrap

SNMP GET Query Name	Object Identifier Name: Number	Description
Object Identifier Name: apSecurityIkeInterfaceInfoTable (1.3.6.1.4.1.9148.3.9.1.9)		
Object Identifier Name: apSecurityIkeInterfaceInfoEntry (1.3.6.1.4.1.9148.3.9.1.9.X.1)		
apSecurityIkeInterfaceChildSaRequest	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.1	Number of Child SA Requests on the IKE-interface.
apSecurityIkeInterfaceChildSaSuccess	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.2	Number of Child SA Success on the IKE-interface.
apSecurityIkeInterfaceChildSaFail	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.3	Number of Child SA Failures on the IKE-interface.
apSecurityIkeInterfaceChildSaDelRequest	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.4	Number of Child SA Delete Requests on the IKE-interface.
apSecurityIkeInterfaceChildSaDelSuccess	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.5	Number of Child SA Delete Success on the IKE-interface.
apSecurityIkeInterfaceChildSaDelFail	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.6	Number of Child SA Delete Failures on the IKE-interface.
apSecurityIkeInterfaceChildSaRekey	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.7	Number of Child SA Rekeys on the IKE-interface."
apSecurityIkeInterfaceInitialChildSa	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.8	Number of Initial Child SA Establishments on the IKE-interface.
apSecurityIkeInterfaceDPDRecvPortChange	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.9	Number of DPD Port Change Received on the IKE-interface.
apSecurityIkeInterfaceDPDRecvIPChange	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.10	Number of DPD IP Change Received on the IKE-interface.
apSecurityIkeInterfaceDPDRespRecv	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.11	Number of DPD Responses Received on the IKE-interface.
apSecurityIkeInterfaceDPDRespNotRecv	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.12	Number of DPD Responses Not Received on the IKE-interface.
apSecurityIkeInterfaceDPDRecv	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.13	Number of DPD Packets Received on the IKE-interface.
apSecurityIkeInterfaceDPDRetran	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.14	Number of DPD Packets Retransmitted on the IKE-interface.
apSecurityIkeInterfaceDPDSent	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.15	Number of DPD Packets Sent on the IKE-interface.
apSecurityIkeInterfaceIKESAPacketSent	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.16	Number of IKE SA Packets Sent on the IKE-interface.
apSecurityIkeInterfaceIKESAPacketRcv	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.17	Number of IKE SA Packets Received on the IKE-interface.
apSecurityIkeInterfaceIKESAPacketDropped	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.18	Number of IKE SA Packets dropped on the IKE-interface.
apSecurityIkeInterfaceAuthFailure	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.19	Number of Authentication Failures on the IKE-interface.
apSecurityIkeInterfaceMsgError	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.20	Number of IKE Message Errors on the IKE-interface.
apSecurityIkeInterfaceAuthIDError	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.21	Number of Authentication ID Errors on the IKE-interface.

SNMP GET Query Name	Object Identifier Name: Number	Description
apSecurityIkeInterfaceAuthCertCheckRequest	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.22	Number of Certificate Status Requests on the IKE-interface.
apSecurityIkeInterfaceAuthCertCheckSuccess	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.23	Number of Certificate Status Success on the IKE-interface.
apSecurityIkeInterfaceAuthCertCheckFailure	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.24	Number of Certificate Status Failures on the IKE-interface.
apSecurityIkeInterfaceDDosSent	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.25	Number of DDoS Sent on the IKE-interface.
apSecurityIkeInterfaceDDosRecv	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.26	Number of DDoS Received on the IKE-interface.
apSecurityIkeInterfaceMessageRetrans	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.27	"Number of IKE Message Retransmissions on the IKE-interface.
apSecurityIkeInterfaceSAInitMsgRecv	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.28	Number of IKE_SA_INIT messages received on the IKE-interface.
apSecurityIkeInterfaceSAInitMsgSent	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.29	Number of IKE_SA_INIT messages sent on the IKE-interface.
apSecurityIkeInterfaceSAEstablishmentAttempts	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.30	"Number of IKE_SA establishment attempts on the IKE-interface.
apSecurityIkeInterfaceSAEstablishmentSuccess	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.31	Number of IKE_SA establishment success on the IKE-interface.
apSecurityIkeInterfaceTunnelRate	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.32	Number of Tunnels per second averaged over 100sec window on the IKE-interface.
apSecurityIkeInterfaceCurrentChildSaPair	apSecurityIkeInterfaceInfoEntry: 1.3.6.1.4.1.9148.3.9.1.9.X.33	Current number of Child Security Association Pairs (Tunnels) on the IKE-interface.