Oracle® Communications Mobile Security Gateway

Essentials Guide
Release M-CZ4.0.0

August 2016

Notices

Copyright© 2016, 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Oracle Legal Notices

Copyright© 2016, 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any

loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

About This Guide

Version M-CZ4.0.0 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.

Related Documentation

The following table describes the documentation set for this release.

Document Name	Document Description
Acme Packet 4500 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 4500.
Acme Packet 3820 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 3820.
Acme Packet 4600 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 4600.
Acme Packet 6100 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 6100.
Acme Packet 6300 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 6300.
Release Notes	Contains information about the current documentation set release, including new features and management changes.
ACLI Configuration Guide	Contains information about the administration and software configuration of the Service Provider Oracle Communications Mobile Security Gateway.
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 Oracle Communications Mobile Security Gateway logs, performance announcements, system management, inventory management, upgrades, working with configurations, and managing backups and archives.
MIB Reference Guide	Contains information about Management Information Base (MIBs), Oracle Communication's 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 Oracle Communications Mobile Security Gateway’s accounting support, including details about RADIUS and Diameter accounting.
HDR Resource Guide	Contains information about the Oracle Communications Mobile Security Gateway’s Historical Data Recording (HDR) feature.

Document Name	Document Description
	This guide includes HDR configuration and system-wide statistical information.
Administrative Security Essentials	Contains information about the Oracle Communications Mobile Security Gateway's support for its Administrative Security license.
Security Guide	Contains information about security considerations and best practices from a network and application security perspective for the Oracle Communications Mobile Security Gateway family of products.
Installation and Platform Preparation Guide	Contains information about upgrading system images and any pre-boot system provisioning.
Call Traffic Monitoring Guide	Contains information about traffic monitoring and packet traces as collected on the system. This guide also includes WebGUI configuration used for the SIP Monitor and Trace application.

Revision History

Date	Description
October 2015	Initial release
March 2016	M-CZ4.0.0M1
August 2016	M-CZ4.0.0M2

Known Issues

Following are the known issues for this release.

BugDB #	Description	Found in:	Resolution:	Notes
21134501	IKE-GTP data pass-through with ESP payload having inner IPv6 data fails.	M-CZ4.0.0	Fixed in: M-CZ4.0.0M1	
21493187	The time does not synchronize and there is no notification after saving and activating an IPv6 ntp-server configuration.	M-CZ4.0.0	Unresolved	Severity level downgraded from 2 to 3.
21515472	No IPv6 address allowed to be configured in push servers.	M-CZ4.0.0	Closed – Not a bug	No functional specification violation
21522805	The MSG can randomly fail when it loses connection to the primary radius server.	M-CZ4.0.0	Fixed in: M-CZ4.0.0M1	
21607928	With sg-policy policy set as DHCP, AAA radius server response missing mandatory APN or IMSI attribute; MSG fails to tear down the call.	M-CZ4.0.0	Fixed in: M-CZ4.0.0M1	
21828472	MSG does not clear local-address pool when using radius-local assignment method while testing IKE-GTP flow with radius-local IP assignment method, and Radius server failed to assign any IP addresses. In that case MSG will assign IP address to end point from its own local-address-pool. After tunnel is torn down, IP addresses assigned from local address pool are not cleared.	M-CZ4.0.0	Fixed in: M-CZ4.0.0M1	
21835122	MSG uses wrong DNS value when using local as dns-assignment method. With local as dns-assignment method and access realm as dns-realm-id, MSG assigns core side DNS server IP instead of access side.	M-CZ4.0.0	Fixed in: M-CZ4.0.0M1	
21842448	MSG does not create NAT-flow when using local as dns-assignment method.	M-CZ4.0.0	Fixed in: M-CZ4.0.0M1	
22982030	Nat entries are not installed for locally configured DNS IPs	M-CZ4.0.0M1	Unresolved	Severity level 3 - Minimal loss of service
22527871	Unable to establish IPSec connection with stock iPhone VPN client	M-CZ4.0.0M1	Unresolved	Severity level 3 - Minimal loss of service

What's New in this Release

M-CZ4.0.0

- Supports Acme Packet 4600, 6100 and 6300
- GTPv1 and PDG support
- DHCP and IKE initiated GTP tunnel set-up
- Selective IP Traffic offload
- License remediation

Baseline patch functionality: M-CX3.0.0M2

M-CZ4.0.0M1

- GTPv2 support on the s2b interface
- Software TLS license requirement eliminated

M-CZ4.0.0M2

- Offload IKE Crypto Processing to SSM3
- Authentication Fatal Error Handling for WiFi Calling
- EAP-only Authentication
- P-CSCF Discovery through IKEv2

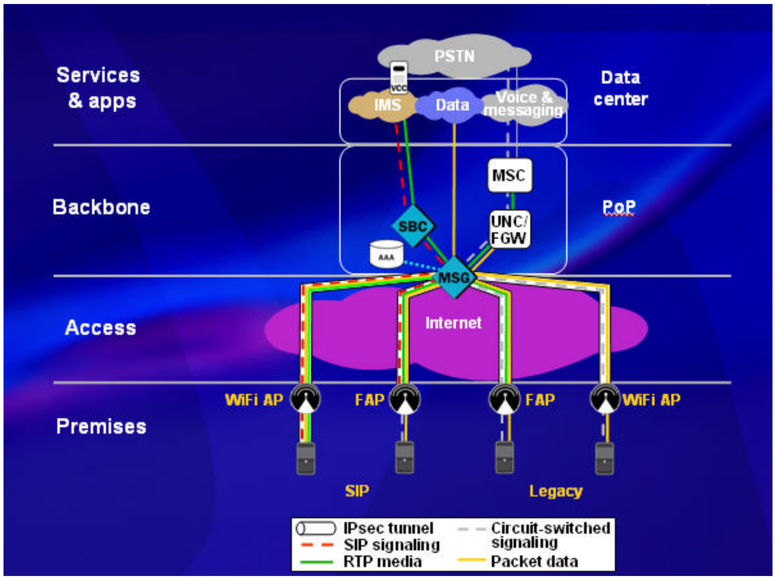
1

Oracle Communications Mobile Security Gateway

Product Overview

This chapter describes the Security Gateway services provided by this release. 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.

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



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

Oracle Communications Mobile Security Gateway Protocol Support

The following sections describe major Oracle Communications Mobile Security Gateway components

Getting Started

Before you can configure the Oracle Communications Mobile Security Gateway running release M-CZ4.0.0 you must first ensure you have the Software TLS license for all certificate related features. You do not need the Software TLS license for releases M-CZ4.0.0M1 and later. You then set the product type and provision the entitlements.

License Verification (for M-CZ4.0.0 only; does not apply for releases M-CZ4.0.0M1 and later)

For release M-CZ4.0.0 only, contact Oracle Support to obtain a Software TLS license if you do not have one.

1. Access the **license** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# system
ACMEPACKET(system)# license
ACMEPACKET(license)#
```

2. Type **show** and confirm that the software TLS license is in place:

```
ACMEPACKET(license)# show
License #1: 32000 sessions, Software TLS
no expiration
installed at 15:52:59 Aug 23 2015
ACMEPACKET(license)#
```

Product Selection

When the Oracle Communications Mobile Security Gateway software image is loaded onto the device, the device acts as a single product of type Mobile Security Gateway. Although loading the software image prevents the device from acting as anything other than a Mobile Security Gateway, you still must verify that the device is one by executing this procedure before you can provision entitlements.

1. ACMEPACKET# **show version**

```
Acme Packet Net-Net 6300 MCZ4.0.0 Beta 1 (WS Build 66)
Oracle Linux branches-6/el6-u6 {2015-06-04T07:00:00+0000}
Build Date=07/29/15
Build View=/home/acme/ccn/LENNY_WS_integration
User=WSbuilder@goose
```

2. ACMEPACKET# **setup product**

```
--------------------------------------------------------------
WARNING:
Alteration of product alone or in conjunction with entitlement
changes will not be complete until system reboot
```

```
Last Modified 2015-09-05 14:50:11
--------------------------------------------------------------
1 : Product : Security Gateway
Enter 1 to modify, d' to display, 's' to save, 'q' to exit. [s]: q
```

Entitlement Provisioning

The entitlement values will be empty or "0" (for capacities) the first time you look at them after loading the MSG software image onto the device, indicating that the features are disabled. You enter a value for each entitlement to make that feature work. To set the values, you must reboot the MSG after the first time you provision the entitlements; subsequent changes to the entitlement values do not require a reboot.

For brevity, the following example contains only one modification (to GTP Tunnel Capacity), although the final output shows the changes to all entitlements. This example reflects the M-CZ4.0.0M1 release entitlements. Earlier releases may be different.

```
ACMEPACKET# setup entitlements
 --------------------------------------------
 Entitlements for Security Gateway
 Last Modified: 2016-01-13 17:14:57
 --------------------------------------------
 1 : Security Gateway : disabled
 2 : GTP Tunnel Capacity : 0
 3 : SG Tunnel Capacity : 0

 Enter 1 - 3 to modify, d' to display, 's' to save, 'q' to exit. [s]: 2
 GTP Tunnel Capacity (0-750000) : 8888

 Enter 1 - 3 to modify, d' to display, 's' to save, 'q' to exit. [s]: s

ACMEPACKET# setup entitlements
 --------------------------------------------
 Entitlements for Security Gateway
 Last Modified: 2016-01-13 17:15:43
 --------------------------------------------
 1 : Security Gateway : enabled
 2 : GTP Tunnel Capacity : 8888
 3 : SG Tunnel Capacity : 1500000

 Enter 1 - 3 to modify, d' to display, 's' to save, 'q' to exit. [s]: q
```

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)

P-CSCF Discovery through IKEv2

The typical P-CSCF discovery as standardized VoLTE does not work for seamless hand-over between VoLTE and WiFi calling because the assigned P-CSCF for a subscriber must also be used for WiFi calling for that subscriber and it is not feasible to provide every WiFi calling end-point with a specific P-CSCF address. This feature defines two new IKEv2 attributes, per RFC7296, that allow an IPsec gateway to provide the IPv4 or IPv6 address of the P-CSCF server. These attributes can be exchanged by IKEv2 peers as part of the configuration payload exchange. The attributes follow the configuration attribute format defined in Section 3 of RFC7651. Furthermore, providing P-CSCF server addresses in IKEv2 as standard attributes enables clients to directly access IMS services behind a VPN gateway without going through the 3GPP-specific interfaces. Besides addresses received from the GPRS Tunneling Protocol (GTP Protocol Configuration Options (PCO), the MSG also includes locally configured addresses in the CP payload (CFG_REPLY) of the final IKE_AUTH response message sent to the UE.

A mobile node using WiFi calling may potentially need to access the IP Multimedia Subsystem (IMS) services in the 3GPP network, the architecture for which is described in TS23228 and TS24229. Currently, there are no attributes in IKEv2, per RFC7296, that can be used for carrying these 3GPP IMS information elements. In the absence of these attributes, the mobile node must be statically configured with this information, which is operationally unfeasible. Any other approaches for discovering these functions, such as using DNS or DHCP, obtain configuration from the access network and not from the home network. Given that the above referenced 3GPP interface is primarily for allowing the mobile node to connect to the 3GPP network through an untrusted access network, the access network may not have any relation with the home network provider and may be unable to deliver the mobile node’s home network configuration.

This feature adds new parameters **pcscf-discovery**, **pcscf-v4-addr-list**, and **pcscf-v6-addr-list** to the **sg-policy** configuration element. To enable this feature, you must configure at least one **sg-policy** configuration element that is related to the **security-interface-params** configuration element through the **sg-policy-list** parameter.

P-CSCF Discovery through IKEv2 Configuration

Use the following procedure to configure P-CSCF discovery through IKEv2:

1. Access the **sg-policy** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# sg-policy
ACMEPACKET(sg-policy)# select
```

2. **pcscf-discovery** — Set the value to **enabled** to activate P-CSCF discovery through IKEv2. The default is **disabled**.
3. **pcscf-v4-addr-list** — Enter the IPv4 addresses for the P-CSCF server. This parameter is not used when the value of **pcscf-discovery** is **disabled**.
4. **pcscf-v6-addr-list** — Enter the IPv6 addresses for the P-CSCF server. This parameter is not used when the value of **pcscf-discovery** is **disabled**.
5. Type **done** to save your configuration.

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 IPv4 or IPv6 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.

IPv6 Support

Version M-CX3.0.0 introduced support for IPv6. The following capabilities are of particular interest to SG users.

- IPv4 or IPv6 addresses on local IKEv2 interfaces
- IPv4 or IPv6 peer support
- IPv4 or IPv6 local address pools
- IPv4 or IPv6 contents in IDi, IDr, Traffic Selector, CFG_REQUEST, and CFG_REPLY payloads
- IPv4 or IPv6 tunnels — address assignment via local address pool or external RADIUS server

WiFi Offload

This feature enhances Tunnel Session Control Function (TSCF) functionality for the Oracle Communications Mobile Security Gateway by providing the capability to offload data to WiFi networks. WIFI offload offers a cost-effective mean of offloading large amounts of mobile data traffic while delivering a variety of new services. The feature adds the Gn’ interface to provide the Tunnel Terminating Gateway (TTG) functionality of a Packet Data Gateway (PDG) to interwork with existing Gateway GPRS Support Node (GGSN) deployments.

The increased use and adoption of smartphones has caused such a large increase in wireless network data usage that network operators are off-loading the data traffic from the macro Radio Access Network (RAN) onto fixed line broadband networks to access 3GPP services. To facilitate this process, this feature provides a TTG and Wireless Access Gateway (WAG) solution by adding the Gn’ interface (a subset of the Gn interface) on the core side and interworking with the existing GGSN infrastructure in 3G and 4G networks.

Secure tunnels are established with either IPsec or Tunnel Session Management (TSM) on the access side; they are connected to existing GGSNs through a GPRS Tunneling Protocol (GTP) interface using Gn' on the core side. The Oracle Communications Mobile Security Gateway currently provides IPsec and DHCP on the access side. Each secure tunnel from the access side is mapped to a single GTP tunnel on the core side. Each Security Association (SA) for the same IPsec tunnel corresponds to a single GTP tunnel. This feature creates only one Packet Data Protocol (PDP) context per GTP tunnel corresponding to each IPsec tunnel and does not create secondary PDP contexts either by using Quality of Service (QOS) or Trivial File Transfer Protocol (TFTP) parameters to differentiate the traffic type, or by any other method. GTP on the Gn interface consists of GTP-C for control signaling and GTP-U for Protocol Data Units (PDUs). The Gn' subset of the Gn interface supports only the following messages:

- Create PDP Context Request — This message is only sent by the TTG to a GGSN and is only handled by GTP-C. If it is sent by a GGSN to the TTG, it is logged and the request is silently dropped without any response.
- Create PDP Context Response — This message is only sent by a GGSN to the TTG and is only handled by GTP-C. The TTG will not send this message.
- Update PDP Context Request — This message can be sent by either the TTG or a GGSN and is only handled by GTP-C. In the current implementation the TTG will not send this message. When the TTG receives this message from a GGSN it first checks whether the optional International Mobile Subscriber Identity (IMSI) Information Element (IE) is included. If it is not included the GTP-C sends a dummy response as the current implementation does not implement any TFT or QOS. If it is included the GTP-C checks for the PDP contexts and Tunnel Endpoint Identifiers (TEIDs) associated with the IMSI IE and sends the corresponding response.
- Update PDP Context Response — This message can be sent by either the TTG or a GGSN and is only handled by GTP-C.
- Delete PDP Context Request — This message can be sent by either the TTG or a GGSN, is only handled by GTP-C, and has precedence in case of conflict with any other tunnel control message.
- Delete PDP Context Response — This message can be sent by either the TTG or a GGSN and is only handled by GTP-C.
- Error Indication — This message can be sent by either the TTG or a GGSN and is only handled by GTP-U. The TTG initiates the tunnel deletion procedure when it receives an error indication for a tunnel which is active from the TTG’s perspective. The TTG ignores the indication if the tunnel does not exist from the TTG’s perspective. The TTG sends an Error Indication with the corresponding TEID when the TTG receives a GTP-U PDU for which there is no active flow installed.
- Version Not Supported — This message can be sent by either the TTG or a GGSN and is only handled by GTP-C. When TTG receives any GTP-C messages from the GGSN with invalid version number, it sends Version Not Supported message. If TTG receives this message for any tunnel creation messages it will not create the tunnel and if it receives for any other GTP-C messages like Echo Request or Update PDP Context Request, it will initiate tunnel deletion procedures for the all the tunnels towards this GGSN.
- GTP Payload Forwarding

Configuration

Provision the following five new configuration elements to configure WiFi offload on the Oracle Communications Mobile Security Gateway:

- **gtp-config** — is used to configure system-wide global GTP parameters. Provision this element at the path **security** > **gtp** > **gtp-config**.
- **gtp-profile** — is used to configure GTP interfaces towards other GGSNs. Values for parameters in **gtp-profile** override the values for the same parameters in **gtp-config**. Provision this element at the path **security** > **gtp** > **gtp-profile**.
- **dhcp-interface** — is used to configure DHCP interfaces for obtaining client IP addresses, and connecting them to the network.
- **security-interface-params** — is used as a container for parameters affecting the elements **ike-interface** and **dhcp-interface**
- **sg-policy** — is used to determine how to route packets; that is, whether to send traffic from the UE to a GGSN server by a GTP tunnel, or to the Internet through the policy’s egress realm.

Global GTP Configuration

1. Access the **gtp-config** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# gtp
ACMEPACKET(gtp)# gtp-config
ACMEPACKET(gtp-config)# select
```

2. **log-level** — identifies the severity level at which events are logged for the GTP tasks and stack. This parameter is required; there is no default. Valid values, in decreasing order of severity, are:
   - **EMERGENCY**
   - **CRITICAL**
   - **MAJOR**
   - **MINOR**
   - **WARNING**
   - **NOTICE**
   - **INFO**
   - **TRACE**
   - **DEBUG**
   - **DETAIL**
3. **red-port** — identifies the GTP redundancy sync port. Valid values are **0** (which disables the port, and **2001**. The default is **0**.
4. **red-max-trans** — identifies the maximum number of redundancy transactions to retain on the active gateway. The range of valid values is {0 - 999999999}. The default is **10000**.
5. **red-sync-start-time** — identifies the time in milliseconds to wait before changing from active to standby or from standby to active; that is, before sending redundancy synchronization requests. The range of valid values is {0 - 999999999}. The default is **5000**.
6. **red-sync-comp-time** — identifies the time in milliseconds to wait for subsequent synchronization requests once redundancy synchronization has completed. The range of valid values is {0 - 999999999}. The default is **1000**.
7. **options** — identifies optional parameters for GTP entered in name=value format. The default is null (no default value).
8. Type **done** to save your configuration.

GTP Profile Configuration

1. Access the **gtp-profile** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# gtp
ACMEPACKET(gtp)# gtp-profile
ACMEPACKET(gtp-profile)# select
```

2. **name** — identifies the name of a valid GTP interface with a string of alpha-numeric characters. This parameter is required; there is no default.
3. **ip-address** — identifies the IPv4 or IPv6 address for the GTP interface. This parameter is required and has no default unless the parameter **hostname** is configured, in which case **ip-address** is optional with a default of null (no specified value).
4. **hostname** — identifies the name of the host for the GTP interface with a string of alpha-numeric characters. This parameter is required and has no default unless the parameter **ip-address** is configured, in which case **hostname** is optional with a default of null (no specified value). When both **ip-address** and **hostname** are configured, **ip-address** takes precedence and no DNS query will be performed. The hunt mechanism is used when a DNS query yields multiple IP addresses.
5. **realm-id** — identifies the name of the realm for the GTP interface with a string of alpha-numeric characters. This parameter is required; there is no default.
6. **remote-control-port** — identifies a valid port number for the GTP-C port on the remote GGSN. The default is **2123**.
7. **remote-user-port** — identifies a valid port number for the GTP-U port on the remote GGSN. The default is **2152**.
8. **local-control-port** — identifies a valid port number for the local GTP-C port. The default is **2123**.
9. **local-user-port** — identifies a valid port number for the local GTP-U port. The default is **2152**.

10. **version** — identifies the version of GTP to use on the interface with an integer in the range {0 - 2}. The default is **1**.
11. **gtpc-echo-request-enable** — identifies whether to enable sending echo requests to the peer from GTP-C. Possible values are **enable** and **disable**. The default value is **enable**.
12. **gtpu-echo-request-enable** — identifies whether to enable sending echo requests to the peer from GTP-U. Possible values are **enable** and **disable**. The default value is **enable**.
13. **echo-request-timeout** — identifies the time in seconds between the sending of echo requests to the peer. The range of valid values is {60 - 300}. The default is **60**.
14. **req-timeout** — identifies the time in seconds that the system waits for responses to GTP request messages. This parameter is also known as timer T3. The range of valid values is {20 - 100}. The default is **20**.
15. **max-retrans** — identifies the maximum number of times a request can be retransmitted before a response is received. This parameter is also known as counter N3. The range of valid values is {5 - 10}. The default is **5**.
16. **plmn-id** — identifies the Public Land Mobile Network for the interface as the combination of the three-digit Mobile Country Code (MCC) and the three-digit Mobile Network Code (MNC), separated by a colon (Format = MCC:MNC). The default is null (no default value).
17. **threshold-crossing-alert-group-name** — identifies the name of the threshold crossing alert group used to hold alerts for the interface with a string of alpha-numeric characters. The default is null (no default value).
18. **options** — identifies optional parameters for GTP entered in name=value format. The default is null (no default value).
19. Type **done** to save your configuration.

DHCP Interface Configuration

1. Access the **dhcp-interface** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# dhcp
ACMEPACKET(dhcp)# dhcp-interface
ACMEPACKET(dhcp-interface)# select
```

2. **state** — identifies whether the DHCP interface is enabled or disabled. The default is **enabled**.
3. **address** — identifies the IPv4 or IPv6 address for the DHCP interface. This parameter is required; there is no default.
4. **realm-id** — identifies the name of the realm for the DHCP interface with a string of alpha-numeric characters. This parameter is required; there is no default.
5. **security-interface-params-name** — identifies, with a string of alpha-numeric characters representing the previously-defined identifier **name** for the **security-interface-params** configuration element, a collection of defined parameters that can be applied to a DHCP or IKE interface. This parameter is required; there is no default.
6. **lease-time** — identifies, in seconds, the duration for which the DHCP server-assigned IP address is valid. The range of valid values is {30 - 4294967295}. The default is **3600**.
7. **options** — identifies optional parameters for DHCP entered in name=value format. The default is null (no default value).
8. Type **done** to save your configuration.

Interface Parameters Configuration

Configuring the **security-interface-param** element with non-null values for the parameters **accounting-params-name** and **account-group-list** enables accounting for IPSec on IKEv2 interfaces (by way of the **ike-accounting-param** element), or for DHCP interface transactions by way of the **dhcp-interface** element.

Use the following procedure to configure sets of common interface parameters that can affect IKEv2 and DHCP interfaces.

1. Access the **security-interface-params** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# security-interface-params
ACMEPACKET(security-interface-params)# select
```

2. **identifier** — identifies, with a string of alpha-numeric characters, the name of a container that holds parameters common to both **ike-interface** and **dhcp-interface** and acts as a single holder to define a GTP profile which can be used for IKE or DHCP interfaces. This parameter is required; there is no default.
3. **address-assignment** — identifies the method used to assign an IPv4 or IPv6 address to the user equipment (UE). The default is **local**. Possible values are:

   - **ggsn-local** — The system first checks the GGSN for the UE address. If the GGSN cannot provide the address, the system allocates one from the local address pool established with **local-address-pool-id-list**.
   - **ggsn-only** — The UE address is assigned by the GGSN for IKEv2 interfaces (this value is not supported for DHCP interfaces).
   - **local** — The system allocates an address from the local address pool established with **local-address-pool-id-list**.
   - **radius-local** — The system first checks the RADIUS server for the UE address. If the RADIUS server cannot provide the address, the system allocates one from the local address pool established with **local-address-pool-id-list**.
   - **radius-only** — The UE address is assigned by the RADIUS server.
4. **authentication-servers** — identifies a list of IPv4 and IPv6 addresses of servers to use for authentication. The default is null (no default value).
5. **authorization-servers** — identifies a list of IPv4 and IPv6 addresses of servers to use for authorization. The default is null (no default value).
6. **accounting-params-name** — identifies an individual accounting parameter name previously configured with the **ike-accounting-param** configuration element. Each individual accounting parameter name identifies specific events in the lifetime of an IPsec tunnel that trigger an accounting transaction. The default is null (no default value).
7. **account-group-list** — identifies one or two existing accounting groups that can be used for accounting transactions. Although there can be many configured accounting groups, only two groups can be active at a time — one to manage transactions with RADIUS servers, and one to manage transactions with DIAMETER servers. Use double quotes around the entry of two IPsec accounting groups and leave a space between the group names; for example, "IPsecRADIUSacctgrp IPsecDIAMETERacctgrp". When no account groups are specified, then no accounting transactions will be initiated by the Oracle Communications Mobile Security Gateway. Note that within each account-group, multiple account servers can be specified (one of which will be selected based on the account strategy).
   When either Radius accounting or Diameter accounting is required, then one account group (for Radius or Diameter) can be specified. When both Radius and Diameter accounting is required, then one account group for Radius and one Account group for Diameter can be specified. When a **security-interface-param** (with non-empty **accounting-params-name** and **account-group-list**) is associated with **IKE-interface**, it enables accounting for IPSec. Similarly a **security-interface-params** can be associated with **dhcp-interface** for enabling accounting for DHCP transactions.
8. **local-address-pool-id-list** — identifies, when the value of **address-assignment** is **local**, a list of groups of IP addresses that can be temporarily assigned to remote endpoints that request an IP address on a Security Gateway subnet, and that also specify DNS information sources made available to remote endpoints.
9. **sg-policy-list** — identifies a list of policies containing the attributes that determine the route for data from the UE.
10. **options** — identifies optional parameters entered in name=value format. The default is null (no default value).
11. Type **done** to save your configuration.

SG Policy Configuration

1. Access the **sg-policy** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# sg-policy
ACMEPACKET(sg-policy)# select
```

2. **identifier** — identifies the name of a policy containing the attributes that determine how to route data from the UE. IKE and DHCP interfaces use this element to determine how to route to a GGSN and whether to establish a GTP tunnel for that purpose. This parameter is required; there is no default.
3. **priority** — identifies the priority of the policy relative to other policies. Policies with lower values have precedence over policies with higher values. The range of valid values is {0 - 4294967295}. The default is **0**.
4. **match-field** — identifies the source of the attributes received from Authentication, Authorization and Accounting (AAA) used as criteria for policy matching. Values are not case-sensitive. The default is **none**. Possible values are:
   - **apn** — Access Point Name
   - **imsi** — International Mobile Subscriber Identity
   - **none**
5. **match-type** — identifies whether the policy match criteria identified by **match-value** should be matched exactly or with no regard to case. The default value is **exact**. Possible values are:
   - **exact** — precisely matches all characters in the string specified by **match-value**
   - **nocase** — matches all characters in the string specified by **match-value** regardless of case
   - **prefix** — matches the prefix digits (when the value of **match-field** is **imsi**) or prefix string (when the value of **match-field** is **apn**) specified in **match-value**
   - **prefix-nocase** — matches the prefix string (when the value of **match-field** is **apn**) specified in **match-value** regardless of case
   - **regex** — matches the Access Point Name or the International Mobile Subscriber Identity
6. **match-value** — identifies, with a string of alpha-numeric characters, the value of the policy match criteria received from the source identified by **match-field**. The default is null (no default value).
7. **dest-realm-id** — identifies, with a string of alpha-numeric characters, the destination realm for the target gateway. The target gateway will be a GGSN when the value of **gtp-profile-list** is non-null.The default is null (no default value).
8. **gtp-profile-list** — identifies, with a string of alpha-numeric characters, a list of names for GTP profiles previously established with the **gtp-profile** configuration element. The default is null (no default value).
9. **gtp-profile-list-selection** — identifies the method for selecting names from the list of GTP profiles. Possible values are **hunt** and **round-robin**. The default value is **hunt**.
10. **default-qos** — identifies, with a string of alpha-numeric characters, the default Quality of Service (QoS) to use if not provided by the user profile. The default is null (no default value).
11. **default-nsapi** — identifies the default Network Service Area Point Identifier (NSAPI) to use if not provided by the user profile. The range of valid values is {0 - 255}. The default is **5**.
12. **options** - identifies optional parameters for SG policy configuration entered in name=value format. The default is null (no default value).
13. Type **done** to save your configuration.

2

IKEv2 Global Configuration

The Security Gateway provides support for Version 2 of the Internet Key Exchange Protocol as defined in RFC 7296, 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.

By default, communications between the SG and the NNC management system are supported by a non-secure (unencrypted) TCP connection over the mgmt0 management interface. TCP message flow from the SG to the NNC is conveyed via port 3000, and message flow from the NNC to the SG is conveyed via port 3001. For customers who required an encrypted SG-to-NNC connection, software releases prior to version MCX3.1.0 provided the capability to establish an encrypted connection using IPsec. Because of new hardware changes, version MCX3.1.0, and later versions, no longer support IPsec as a means to encrypt the SG-to-NNC connection. In place of IPsec, MCX3.1.0 and later versions support TLS as a replacement, thus offering the ability to authenticate and encrypt the SG-to-NNC connection.

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 authentication servers (optional).
4. Configure one or more RADIUS authorization servers (optional).
5. Configure the default address pool (optional).
6. 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.

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)# ?
state                         state
ike-version                   IKE Version
log-level                     Log level for ike stack
udp-port                      port (default: 500)
negotiation-timeout           IKE Negotiation timer (secs)
event-timeout                 IKE Event timer (secs)
phase1-mode                   Ike phase1 Modes
phase1-dh-mode                Ike phase1 DH Modes
v2-ike-life-secs              (IKEv2) IKE SA lifetime (secs)
v2-ipsec-life-secs            (IKEv2) IPsec SA lifetime (secs)
v2-rekey                      whether to do IKE/IPSEC rekey as a responder
anti-replay                   enable/disable anti-replay for IKE
                              established IPSEC SA's
phase1-life-seconds           (IKEv1) Acting as IKE initiator the time
                              proposed for phase 1 SA expiration
phase1-life-secs-max          (IKEv1) Max Phase1 SA expiration time to be
                              accepted if acting as IKE responder
phase2-life-seconds           (IKEv1) Acting as IKE initiator the time
                              proposed for phase 2 SA expiration
phase2-life-secs-max          (IKEv1) Max Phase2 SA expiration time to be
                              accepted if acting as IKE responder
phase2-exchange-mode          Ike Phase2 Exchange Modes
shared-password               pre shared password
options                       optional features/parameters
eap-protocol                  Ike EAP Protocols
addr-assignment               local address assignment method
eap-bypass-identity           eap bypass identity, no-op for eap-radius-
                              passthrough
red-port                      ike redundancy sync port: 0 to disable and
                              1995 to enable
red-max-trans                 maximum redundancy transactions to keep on
                              active
red-sync-start-time           timeout for transitioning from standby to
                              active
red-sync-comp-time            sync request timeout after initial sync
                              completion
dpd-time-interval             dead peer detection time interval in secs
                              (when we initiate the DPD)
overload-threshold            Overload Threshold (Percentage) range ->
                              10-100
overload-interval             Overload Measurement Interval (Seconds)
                              range --> 1-60
```

```
overload-action              Action to be taken when IKE hits overload
                             threshold
overload-critical-threshold  Critical Overload Threshold (Percentage)
overload-critical-interval   Critical Overload Measurement Interval
                             (Seconds)
sd-authentication-method     allowed sd authentication methods
certificate-profile-id       Identity for the certificate-profile
                             container, the identity should be FQDN or
                             IP address
id-auth-type                 ID type for authentication if required
account-group-list           List of names of "account-group" elements
                             configured under session-router
select                       select ike config to edit
no                           delete ike config
show                         show ike config
done                         write ike config information
exit                         return to previous menu
ACMEPACKET(ike-config)#
```

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**.

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-related logs.

   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** or **id-auth-type** 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 verification 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 Configuring the Certificate Record 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, a remote IKEv2 peer initiates the exchange by requesting an IP address on the gateway's protected network. In response, the 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 Generation & Management

Generating certificates is a three-step process:

1. Create a certificate record configuration
2. Generate a certificate signing request (CSR)
3. Import the signed certificate.

Configuring the Certificate Record

The certificate record configuration represents either an end-entity or a CA certificate on the Security Gateway.

If it is used as an end-entity certificate, a must key should be associated with this certificate record configuration using the ACLI **generate-certificate-request** command.

No private key should be associated with the certificate record configuration if it was issued to hold a CA certificate.

A certificate can be imported to a certificate record configuration using the ACLI **import-certificate** command.

1. From Superuser mode, use this ACLI command sequence to access certificate-record configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# certificate-record
   ACMEPACKET(certificate-record)#
   ```

2. **name**—Provide the required name of the certificate record.

   ```
   ACMEPACKET(certificate-record)# name cto
   ACMEPACKET(certificate-record)#
   ```

3. **country**, **state**, **locality**, **organization**, **unit**, and **common-name**—Use these parameters to identify the certificate's subject.

   ```
   ACMEPACKET(certificate-record)# country us
   ACMEPACKET(certificate-record)# state ca
   ACMEPACKET(certificate-record)# locality San Francisco
   ACMEPACKET(certificate-record)# organization Office of the CTO
   ACMEPACKET(certificate-record)# unit cyzygy.com
   ACMEPACKET(certificate-record)# common-name www.cyzygy.org/
   emailAddress=cto@cyzygy.org
   ACMEPACKET(certificate-record)#
   ```

   Based on this ACLI sequence, the subject field of the resulting CA-issued certificate reads as follows.

   Subject: C=US, ST=California, L=San Francisco, O=Office of the CTO OU=Cyzygy,

   CN=www.cyzygy.org/emailAddress=cto@cyzygy.org

4. **key-size**—Enter the size of the key for the certificate. Allowable values are:

   - 512 | 1024 (the default) | 2048

     ```
     ACMEPACKET(certificate-record)# key-size 1024
     ACMEPACKET(certificate-record)#
     ```

5. **alternate-name**—Optionally provide one or more alternative names for the certificate holder.

   The Subject Alternative Name certificate extension is defined in section 4.2.1.6 of RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. This extension allows one or more identities to be bound to the subject of a certificate. As a result, each subject alternative name is considered as an alias for the certificate subject. These aliases may be included in addition to, or in place of the identity in the subject field of the certificate, meaning that a certificate can contain a subject name and one or more aliases, or no subject name and one or more aliases. A certificate cannot lack both a subject name and an alias. As specifically defined in RFC 5280, subject alternative names can take the form of an e-mail address, an IP address (IPv4 or IPv6), a DNS address, a Registered ID (RID), a Distinguished Name (DN), or a URI.

   RFC also 5280 allows future support for other alternative name forms, but only if such forms are specified in an IETF RFC.

   Most alternative names take one of the following three formats: IP address (which requires an IP: prefix), DNS (which requires a DNS: prefix), or email (which requires an email: prefix).

   For example:

   ```
   ACMEPACKET(certificate-record)# alternate-name IP:192.168.12.101
   ACMEPACKET(certificate-record)# alternate-name IP:13::17
   ACMEPACKET(certificate-record)# alternate-name DNS:sg1a.ba.de
   ACMEPACKET(certificate-record)# alternate-name email:my@other.address
   ACMEPACKET(certificate-record)# alternate-name copy,email:my@other.address
   ACMEPACKET(certificate-record)#
   ```

   The email form can include a special copy value that includes any email addresses contained in the certificate subject name in the extension.

   For other formats, consult RFC 5280, later RFCs that define future formats, or the OpenSSL documentation available at:

http://www.openssl.org/docs/apps/x509v3_config.html

6. **trusted**—Leave this parameters set to **enabled** to mark the certificate as trusted. Enter **disabled** to mark this certificate as untrusted.

   - enabled (default) | disabled

7. **key-usage-list**—Enter the usage extensions you want to use with this certificate record.

   Supported values are as follows:

Value	Description
digitalSignature (default with keyEncipherment)	Used when the subject public key is used with a digital signature mechanism to support security services other than non-repudiation, certificate signing, ore revocation information signing. Digital signature mechanisms are often used for entity authentication and data origin authentication with integrity.
nonRepudiation	Used with the subject public key is used to verify digital signatures that provide a non-repudiation service protecting against the signing entity falsely denying some action, excluding certificate or CRL signing.
keyEncipherment (default with digitalSignature)	Used with the subject public key is used for key transport. (For example, when an RSA key is to be used for key management.)
dataEncipherment	Used with the subject public key is used for enciphering user data other than cryptographic keys.
keyAgreement	Used with the subject public key is used key agreement. (For example, when a Diffie-Hellman key is to be used for a management key.)
encipherOnly	The keyAgreement type must also be set. Used with the subject public key is used only for enciphering data while performing key agreement.
decipherOnly	The keyAgreement type must also be set. Used with the subject public key is used only for deciphering data while performing key agreement.

8. **extended-key-usage-list**—Enter the extended key usage extensions you want to use with this certificate record.

Value	Description
serverAuth (default)	Used while the certificate is used for TLS server authentication. In SBC access-side deployments, the SBC typically acts as a TLS server accepting TLS connections. You might use this setting while generating the end-entity-cert.
clientAuth	Used while the certificate is used for TLS client authentication. In SBC core-side deployments, the SBC typically acts as a TLS client initiating TLS connections. You might use this setting while generating the end-entity-cert.

Generating a CSR

Using the ACLI **generate-certificate-request** command allows you to generate a private key and a certificate request in PKCS10 PEM format. You take this step once you have configured an ike-certificate-profile configuration object.

The Security Gateway stores the private key that is generated in the certificate record configuration in 3DES encrypted form with in internally generated password. The PKCS10 request is displayed on the screen in PEM (Base64) form.

You use this command for certificate record configurations that hold end-entity certificates. If you have configured the certificate record to hold a CA certificate, then you do not need to generate a certificate request because the CA publishes its certificate in the public domain. You import a CA certificate by using the ACLI **import-certificate** command.

To run the applicable command, you must use the value you entered in the name parameter of the certificate record configuration. You run the command from main Superuser mode command line:

```
ACMEPACKET# generate-certificate-request cto
Generating Certificate Signing Request. This can take several minutes...
-----BEGIN CERTIFICATE REQUEST-----
MIIDHzCCAoigAwIBAgIIAhMCUACEAHEwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
dXRob3JpdHkwHhcNMDUwNDEzMjEzNzQzWhcNMDgwNDEyMjEzNzQzWjBUMQswCQYD
VQQGEwJVUzELMAkGA1UECBMCTUExEzARBgNVBAcTCkJ1cmxpbmd0b24xFDASBgNV
BAoTC0VuZ2luZWVyaW5nMQ0wCwYDVQQDEwRhY21lMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQCXjIeOyFKAUB3rKkKK/+59LT+rlGuW7Lgc1V6+hfTSr0co+ZsQ
bHFUWAA15qXUUBTLJG13QN5VfG96f7gGAbWayfOS9Uymold3JPCUDoGgb2E7m8iu
vtq7gwjSeKNXAw/y7yWy/c04FmUD2U0pZX0CNIR3Mns5OAxQmq0bNYDhawIDAQAB
o4HdMIHaMBEGA1UdEQQKMAiCBnBrdW1hcjAJBgNVHRMEAjAAMB0GA1UdDgQWBBTG
tpodxa6Kmmn04L3Kg62t8BZJHTCBmgYDVR0jBIGSMIGPgBRrRhcU6pR2JYBUbhNU
2qHjVBShtqF0pHIwcDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEx
ETAPBgNVBAcTCFNhbiBKb3NlMQ4wDAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lw
aXQgVGVzdCBDZXJ0aWZpY2F0ZSBBdXRob3JpdHmCAQAwDQYJKoZIhvcNAQEFBQAD
gYEAbEs8nUCi+cA2hC/lM49Sitvh8QmpL81KONApsoC4Em24L+DZwz3uInoWjbjJ
QhefcUfteNYkbuMH7LAK0hnDPvW+St4rQGVK6LJhZj7/yeLXmYWIPUY3Ux4OGVrd
2UgV/B2SOqH9Nf+FQ+mNZOlL7EuF4IxSz9/69LuYlXqKsG4=
-----END CERTIFICATE REQUEST-----;
WARNING: Configuration changed, run save-config command.
ACMEPACKET# save-config
Save-config received, processing.
waiting 1200 for request to finish
Request to ‘SAVE-CONFIG’ has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run ‘activate-config’ or ‘reboot-activate’
ACMEPACKET# activate-config
Activate-Config received, processing.
waiting 12000 for request to finish
Add LI flows
LiSysClientMgr::handleNotifyReq
H323 Active Stack Cnt: 0
Request to ‘ACTIVATE-CONFIG’ has finished
Activate Complete
ACMEPACKET#
```

Importing a Certificate Using the ACLI

For an end-entity certificate, once a certificate is generated using the ACLI **generate-certificate-request** command, that request should be submitted to a CA for generation of a certificate in PKCS7 or X509v3 format. When the certificate has been generated, it can be imported to the Security Gateway using the **import-certificate** command.

The syntax is:

```
ACMEPACKET # import-certificate [try-all|pkcs7|x509] [certificate-record file-
name]
```

To import a certificate:

1. When you use the **import-certificate** command, you can specify whether you want to use PKCS7 or X509v3 format, or try all. In the command line, you enter the command, the format specification, and the name of the certificate record.

```
ACMEPACKET# import-certificate try-all cto
```

The following will appear:

```
Please enter the certificate in the PEM format.
Terminate the certificate with ";" to exit.......
-----BEGIN CERTIFICATE-----
MIIDHzCCAoigAwIBAgIIAhMCUACEAHEwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
dXRob3JpdHkwHhcNMDUwNDEzMjEzNzQzWhcNMDgwNDEyMjEzNzQzWjBUMQswCQYD
VQQGEwJVUzELMAkGA1UECBMCTUExEzARBgNVBAcTCkJ1cmxpbmd0b24xFDASBgNV
BAoTC0VuZ2luZWVyaW5nMQ0wCwYDVQQDEwRhY21lMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQCXjIeOyFKAUB3rKkKK/+59LT+rlGuW7Lgc1V6+hfTSr0co+ZsQ
bHFUWAA15qXUUBTLJG13QN5VfG96f7gGAbWayfOS9Uymold3JPCUDoGgb2E7m8iu
vtq7gwjSeKNXAw/y7yWy/c04FmUD2U0pZX0CNIR3Mns5OAxQmq0bNYDhawIDAQAB
o4HdMIHaMBEGA1UdEQQKMAiCBnBrdW1hcjAJBgNVHRMEAjAAMB0GA1UdDgQWBBTG
tpodxa6Kmmn04L3Kg62t8BZJHTCBmgYDVR0jBIGSMIGPgBRrRhcU6pR2JYBUbhNU
2qHjVBShtqF0pHIwcDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEx
ETAPBgNVBAcTCFNhbiBKb3NlMQ4wDAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lw
aXQgVGVzdCBDZXJ0aWZpY2F0ZSBBdXRob3JpdHmCAQAwDQYJKoZIhvcNAQEFBQAD
gYEAbEs8nUCi+cA2hC/lM49Sitvh8QmpL81KONApsoC4Em24L+DZwz3uInoWjbjJ
QhefcUfteNYkbuMH7LAK0hnDPvW+St4rQGVK6LJhZj7/yeLXmYWIPUY3Ux4OGVrd
2UgV/B2SOqH9Nf+FQ+mNZOlL7EuF4IxSz9/69LuYlXqKsG4=
-----END CERTIFICATE-----;
Certificate imported successfully....
WARNING: Configuration changed, run "save-config" command.
```

2. Save your configuration.

```
ACMEPACKET# save-config
Save-Config received, processing.
waiting 1200 for request to finish
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot activate'.
```

3. Synchronize and activate your configurations.

```
ACMEPACKET# activate-config
Activate-Config received, processing.
waiting 120000 for request to finish
Add LI Flows
LiSysClientMgr::handleNotifyReq
H323 Active Stack Cnt:  0
Request to 'ACTIVATE-CONFIG' has Finished,
Activate Complete
```

Importing a Certificate Using FTP

You can also put the certificate file in the directory /ramdrv and then execute the **import-certificate** command or by pasting the certificate in the PEM/Base64 format into the ACLI. If you paste the certificate, you might have to copy and paste it a portion at a time rather than pasting in the entire certificate at once.

To import the certificate using FTP:

1. FTP the certificate file on to the SBC (directory /ramdrv), let us say the name of the certificate file is cert.pem.
2. Once the certificate is successfully transferred to the SBC, run the **import-certificate** command.

   The syntax is:

```
ACMEPACKET# import-certificate [try-all|pkcs7|x509] [certificate-record
file-name]
```

Using the command will look like this when you have used FTP.

```
ACMEPACKET# import-certificate try-all acme cert.pem
Certificate imported successfully....
WARNING: Configuration changed, run "save-config" command.
```

3. Save your configuration.

```
ACMEPACKET# save-config
Save-Config received, processing.
waiting 1200 for request to finish
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot activate'.
```

4. Synchronize and activate your configurations.

```
ACMEPACKET# activate-config
Activate-Config received, processing.
waiting 120000 for request to finish
Add LI Flows
LiSysClientMgr::handleNotifyReq
H323 Active Stack Cnt:  0
Request to 'ACTIVATE-CONFIG' has Finished,
Activate Complete
ACMEPACKET#
```

Creating a Certificate Profile

If authentication between IKEv2 peers is certificate based, use the following procedure to create one or more certificate profiles that provide identification and verification 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)# ?
identity                 ike-certificate-profile identity, this can be IP
                         address or FQDN
end-entity-certificate   end entity certificate for the IKE connection
trusted-ca-certificates  list of trusted certificate records
verify-depth             maximum length of the certificate chain
                         [default: 3, valid range: 0-10].
batch                    enter all arguments on one line
select                   select a ike-certificate profile to edit
no                       delete selected ike-certificate profile
show                     show ike-certificate profile information
done                     write ike-certificate profile information
exit                     return to previous menu
ACMEPACKET(ike-certificate-profile)#
```

2. Use the required **identity** parameter to specify the IKEv2 entity that uses the authentication and verification 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 cto
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 semanticClass3-
   a,semanticClass3-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.

SHA-256 Signing Algorithm

With Version M-CX3.0.0 and later releases the Security Gateway supports the import of SHA-256 signed root/CA certificates using the **import-certificate** ACLI command (refer to *Importing a Certificate Using the ACLI* for procedural details). These releases also support the import of SHA-256-signed end-entity certificates, and the generation of certificate signing requests for self-generated private keys (refer to *Generating a CSR* for procedural details), and also support the import of a SHA-256 signed end-entity certificate.

A SHA-256 self-signed root/CA certificate can be created using an OpenSSL command as shown below

```
openssl req -nodes -config <openSSL-config-file-name> -days 1826
-x509 -newkey rsa:1024 -out public/LScacertsha256.pem -outform
PEM -sha256
```

An Security Gateway-generated certificate request for end-entity certificate can be SHA-256 signed by an OpenSSL root CA by configuring the default_md option in open SS L's configuration file

```
default_md = sha256
```

A sample SHA-256 signed certificate imported to the Security Gateway is shown below using the **show security certificate detail** ACLI command

```
certificate-record: SDCertSha256
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 8 (0x8)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer:
      C=us
      ST=ma
      L=burlington
      O=acme
      OU=pki
      CN=RootCASha256CN
      emailAddress=RootCASha256CN@acme.com
    Validity
      Not Before: Oct  3 18:12:17 2012 GMT
      Not After : Oct  1 18:12:17 2022 GMT
    Subject:
      C=US
      ST=MA
```

```
      O=Engineering
      CN=SDCertSha256CN
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:FALSE
      Netscape Cert Type:
        SSL Server
      Netscape Comment:
        ACME Generated Certificate
      X509v3 Subject Key Identifier:
        60:5B:03:BD:3C:66:B5:EB:92:A4:E3:FE:96:12:9A:ED:7F:FA:86:15
      X509v3 Authority Key Identifier:
        keyid:53:07:3F:1E:9B:52:89:20:D0:9A:67:E3:C9:A1:69:DA:4A:9A:A6:4A
        DirName:/C=us/ST=ma/L=burlington/O=acme/OU=pki/CN=RootCASha256CN/
emailAddress=RootCASha256CN@acme.com
        serial:A9:3B:85:9E:D2:29:0A:0F
      X509v3 Issuer Alternative Name:
        email:RootCASha256CN@acme.com
      X509v3 Subject Alternative Name:
        <EMPTY>
```

Key Usage Extension

The Key Usage extension, defined in section 4.2.1.3 of RFC 5280, Internet X.509 Public Key Infrastructure and Certificate Revocation List (CRL) Profile, defines the purpose of the key found in a certificate. In order to facilitate the construction and navigation of certificate chains, RFC 5280 conformance requires the presence of the Key Usage extension in all Certificate Authority (CA) certificates.

The Key Usage extension is implemented as a bit string of 9 bits (0 through 8) with each bit identifying a specific purpose, for example, decipher, encipher, or CRLSign.

keyCertSign Bit

RFC 5280 requirements mandate that the keyCertSign bit (bit 5) of the 9-bit KeyUsage string) must be asserted, when the public key contained in the certificate is used for signature verification of other public certificates. If the keyCertSign bit is asserted, the boolean portion of the Basic Constraints extension (refer to *Basic Constraints Extension*) must also be asserted.

With release Version M-CX3.0.0 and later releases, the Security Gateway when attempting to verify a certificate's digital signature enforces the assertion of the keyCertSign bit, meaning that every CA certificate in a received chain must have the keyCertSign bit asserted. Failure to find the bit in the asserted state results in immediate failure of the certificate verification process.

To ensure compliance with RFC 5280, testing of the keyCertSign status is enabled by default and is not user configurable.

digitalSignature Bit

Bit 0 of the KeyUsage string is referred to as the digitalSignature bit. This bit is asserted when the public key contained in the certificate is used to verify digital signatures other than signatures attached to certificates or Certificate Revocation Lists (CRL).

In certain environments, some end-entity certificates may assert the digitalSignature bit. Release Version M-CX3.0.0 and later releases provide the ability to optionally enforce the assertion of the digitalSignature bit with the **options entity-cert-keyusage** ACLI command.

In the default state, the certificate verification process pays no attention to the digitalSignature bit. However, the **options entity-cert-keyusage** command forces the certificate verification process to test the state of the digitalSignature bit. If the bit is not asserted, certificate verification fails immediately; if the bit is asserted, the process continues.

Use the following procedure to test the digitalSignature bit status.

1. From superuser mode, use the following command sequence to access ike-config configuration mode. While in this mode, you can configure enforcement of digitalSignature bit assertion using the **options** command.

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

2. Use the **options** command to enforce assertion of the digitalSignature bit.

   ```
   ACMEPACKET(ike-config)# options +entity-cert-keyusage
   ACMEPACKET(ike-config)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete configuration.

Subject Alternative Name Extension

The Subject Alternative Name extension is defined in section 4.2.1.6 of RFC 5280. This extension allows one or more identities to be bound to the subject of a certificate; that is, each subject alternative name is considered as an alias for the certificate subject. These aliases may be included in addition to, or in place of the identity in the subject field of the certificate — meaning that a certificate can contain a subject name and one or more aliases, or no subject name and one or more aliases. A certificate cannot lack both a subject name and an alias. Subject alternative names generally take the form of an e-mail address, a DNS name, an IP address, or a URI. Other options are supported, including local definitions.

As described in IKE_SA_INIT in IKEv2 peers divulge their identities through the IDi and IDr payloads. However, as noted in Section 3.5 (Identification Payloads) of RFC 7296, there is no IKEv2 protocol requirement that the IDi payload contents match the identity contained in the subject field of certificate presented for authentication.

Some environments, however, require that the IDi payload contents be verified against the certificate presented for authentication. In these environments IDi contents are verified as follows:

1. Match the contents of IDi with the contents of the certificate's Subject field.

   If the content's match, the purported identity is verified, and authentication continues.
2. If Step 1 fails, and the certificate does not include the subject alternative name extension, verification/ authentication fails.
3. If both Step 1 and Step 2 fail, the IDi contents are matched against each of the subject alternative names contained within the certificate.

   If a match is found, the purported identity is verified, and authentication continues.

   If no match is found, verification/authentication fails.

Use the following procedure to enable verification of the IDi payload contents.

By default, verification is disabled.

1. From superuser mode, use the following command sequence to access ike-config configuration mode. While in this mode, you can configure verification of the IDi payload contents using the **options** command.

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

2. Use the **options** command to enforce assertion of the digitalSignature bit.

   ```
   ACMEPACKET(ike-config)# options +idi-match-subject
   ACMEPACKET(ike-config)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete configuration.

Basic Constraints Extension

The Basic Constraints extension is defined in section 4.2.1.9 of RFC 5280. It's presence in a certificate identifies the certificate subject as a CA. Conformance to RFC 5280 requires the inclusion of the Basic Constraints extension in all CA certificates that contain public keys used for digital signature validation; conforming CAs must mark this extension as critical.

The Basic Constraints extension consists of an initial boolean value (cA) followed by an integer value referred to as the pathLenConstraint.

cA, when TRUE (asserted), indicates a CA certificate that can be used to verify certificate signatures. A FALSE value indicates a certificate that cannot be used for signature validation. Assertion of cA must be accompanied by assertion of the keySignCert bit (refer to keyCertSign Bit).

pathLenConstraint (an optional integer value, equal to or greater than 0) specifies the maximum number of non-self-issued intermediate certificates that can follow this certificate in a valid certification chain. The final certificate in the chain, which is usually, although not always, an end-entity certificate, is not included in the maximum count.

The pathLenConstraint value is certificate-specific, and changes as a certificate chain is navigated. Because pathLenConstraint is optional, it need not appear in any or all chained certificates. It's absence (or a 0 value, if present) indicates that there are no constraints on the number of certificates that follow this certificate in the chain.

When navigating a chain of intermediate CA certificates pathLenConstrain processing can be summarized as follows.

1. initialize a variable, max-path, to -1.
2. Attempt to validate the signature on the last certificate in the chain (usually as end-entity certificate), using the configured listed of trusted CAs.

   If signature validation succeeds, the certificate is verified.

   Else, proceed to Step 3.
3. Retrieve the pathLenConstraint from the first certificate in the chain.

   If not present, no path length limits are imposed by this certificate; go to the next chained certificate, and repeat this step.

   If present, proceed to Step 4.
4. If the pathLenConstraint value is less than the value of max-path, or if max-path equals -1, set max-path to the pathLenConstraint value.
5. Decrement max-path.

   If max-path equals 0, verification of the certificate chain fails.

   If max-path does not equal 0, proceed to Step 6.
6. Repeat Steps 2 through 5 for each chained certificate.

   If max-path reaches 0 when there are still unexamined certificates in the chain, verification fails.

To ensure compliance with RFC 5280, pathLenConstraint processing is enabled by default and is not user configurable.

Certificate Chain Verification

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 verification.
2. Check the maximum length of the certificate chain. If the current chain exceeds this value, authentication fails; otherwise, continue chain verification.
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 verification.

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 verification.
5. Check the X509v3 Extensions field to verify that the current certificate identifies a CA. If not so, authentication fails; otherwise, continue chain verification.
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 verification.
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 a pool of RADIUS authentication servers, with each server configuration record providing all values required for server access
- configuration of a RADIUS Authentication Servers List designating specific pool member as being available for authentication purposes
- assignment of the RADIUS Authentication Servers List to the authentication configuration object

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)# ?
address                 remote radius server address
port                    remote radius server port, default:1812
state                   administrative state (enabled/disabled),
                        default:"enabled"
secret                  radius server shared secret
nas-id                  radius server NAS ID
realm-id                radius server REALM ID
retry-limit             maximum retries, range: 1-5, default:3
retry-time              value for retry timer, range: 5-10 seconds,
                        default:5
maximum-sessions        maximum outstanding sessions, range: 1-255,
                        default:255
class                   class, "primary" or "secondary",
                        default:"primary"
dead-time               dead time, supported range: 10-10000 secs,
                        default:10 secs
```

```
authentication-methods  supported: "pap", "chap", "mschapv2", "eap"
                        and "all", default:"all"
select                  select a radius server to edit
no                      delete this radius server
show                    show radius server information
done                    write radius server configuration information
exit                    return to previous menu
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)#
name                    name of authentication-params
protocol                protocol used in the authentication
strategy                type of strategy used for authentication, "hunt"
                        or "round-robin", default:"hunt"
servers                 list of servers used for authentication requests:
                        "ip-addresses" of radius servers configured
authorization-servers   list of servers used for authorization:
                        "ip-addresses" of radius servers configured
options                 optional features/parameters
select                  select a authentication_param to edit
no                      delete this authentication_param
show                    show authentication_paramk information
done                    write authentication_param configuration information
exit                    return to previous menu
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 address 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 172.30.0.1 172.30.0.15 168.27.3.3
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 RADIUS 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.
7. From superuser mode, use the following command sequence to access authentication configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# authentication
ACMEPACKET(authentication)# ?
ACMEPACKET(authentication)#
source-port                 source port, valid values are 1645 or
                            1812(default)
type                        type of authentication, "local", "radius" or
                            "tacacs", default:"local"
protocol                    authentication protocol ( ascii (tacacs only),
                            pap, chap or mschapv2 (radius only) ),
                            default:"pap"
tacacs-authorization        Tacacs authorization state (default: "enabled")
tacacs-accounting           Tacacs accounting state (default: "enabled")
server-assigned-privilege Tacacs server assigned privilege (default:
                            "disabled")
allow-local-authorization allow the user to be authorized locally, default:
                            "disabled"
login-as-admin              login as admin user, default: "disabled"
management-strategy         type of strategy used for Management, "hunt" or
                            "round-robin", default:"hunt"
ike-radius-params-name      name of the "authentication-params" configured
                            under "security->authentication-params": string
management-servers          list of servers used for management requests:
                            "ip-addresses" of radius/tacacs+ servers
                            configured
radius-servers              radius servers for authentication
tacacs-servers              Tacacs servers for authentication
select                      select a auth profile to edit
no                          delete selected auth profile
show                        show auth profile information
done                        write auth profile information
exit                        return to previous menu
ACMEPACKET(authentication)#
```

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

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

9. 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.

Authentication Fatal Error Handling for WiFi Calling

When devices with 2G SIMs attempt to register over a WiFi network their authentication is rejected but the devices repeatedly attempt to re-register, which creates additional traffic that can lead to resource depletion in the network. This feature adds the new parameter **auth-critical-failure-threshold** to the **ike-access-control** configuration element. This parameter is used to track fatal authentication failures received from the RADIUS server. The MSG adds the peer to the DDoS (Distributed Denial of Service) deny list when the threshold is exceeded. The MSG silently drops new tunnel establishment requests from peers on the DDOS list.

Fatal authentication errors for which the MSG will add a peer to the DDoS deny list include:

- AUTHORIZATION_REJECTED
- NON_3GPP_ACCESS_TO_EPC_NOT_ALLOWED
- PLMN_NOT_ALLOWED
- RAT_TYPE_NOT_ALLOWED
- USER_UNKNOWN

The MSG uses the existing **ike-access-control** parameter **auth-failure-threshold** to track non-critical authentication failures, which include:

- NETWORK_FAILURE
- NO_APN_SUBSCRIPTION

Authentication Fatal Error Handling for WiFi Calling Configuration

1. Access the **ike-access-control** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-access-control
ACMEPACKET(ike-access-control)#
```

2. Select the **ike-access-control** object to edit.

```
ACMEPACKET(ike-access-control)# select
<name>:

ACMEPACKET(ike-access-control)#
```

3. **auth-critical-failure-threshold** — Enter the maximum number of critical authentication failures received from the RADIUS server that are allowed before the MSG adds the peer to the DDoS (Distributed Denial of Service) deny list. The range of values is **{0-999999999}**; the default is **1**. Setting the value to **0** disables the feature.
4. Type **done** to save your configuration.

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

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 Authentication 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)# ?
address                remote radius server address
port                   remote radius server port, default:1812
state                  administrative state (enabled/disabled),
                       default:"enabled"
secret                 radius server shared secret
nas-id                 radius server NAS ID
realm-id               radius server REALM ID
retry-limit            maximum retries, range: 1-5, default:3
retry-time             value for retry timer, range: 5-10 seconds,
                       default:5
maximum-sessions       maximum outstanding sessions, range: 1-255,
                       default:255
class                  class, "primary" or "secondary", default:"primary"
dead-time              dead time, supported range: 10-10000 secs,
                       default:10 secs
authentication-methods supported: "pap", "chap", "mschapv2", "eap"
                       and "all", default:"all"
select                 select a radius server to edit
no                     delete this radius server
show                   show radius server information
done                   write radius server configuration information
exit                   return to previous menu
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. Retain the default value (all) the **authentication-methods** attribute.
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)#?
name                   name of authentication-params
protocol               protocol used in the authentication
strategy               type of strategy used for authentication, "hunt"
                       or "round-robin", default:"hunt"
servers                list of servers used for authentication requests:
                       "ip-addresses" of radius servers configured
authorization-servers list of servers used for authorization: "ip-
addresses"
                       of radius servers configured
options                optional features/parameters
select                 select a authentication_param to edit
no                     delete this authentication_param
show                   show authentication_paramk information
done                   write authentication_param configuration information
exit                   return to previous menu
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 the IP address of a previously configured RADIUS authorization server to add that server to this list.

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

```
ACMEPACKET(auth-params)# servers 172.30.0.10
ACMEPACKET(auth-params)#
ACMEPACKET(auth-params)# servers 172.30.0.10 172.30.0.115 168.27.3.103
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.

Enable RADIUS Authorization

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 or IPv6 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 7296, 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 RFC 7296, 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)# ?
name                name for a data flow
realm-id            realm-id to route the upstream data flow
group-size          the number of UEs to be managed by the data flow
upstream-rate       Upstream bandwidth (KB/s) for the data flow
downstream-rate     Downstream bandwidth (KB/s) for the data flow
batch               enter all arguments on one line
select              select a data flow to edit
no                  delete selected data flow
show                show selected data flow
done                write data flow information
exit                return to previous menu
ACMEPACKET(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 or ranges of IPv4 or IPv6 addresses with an existing data-flow.

Note: An address pool can contain multiple contiguous ranges of IP addresses. However, all defined ranges must specify the same type of IP address, either IPv4 or IPv6.

You cannot include IPv4 and IPv6 addresses in the same address pool.

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)# ?
name                 name for the local address pool
address-range        address ranges for the local address pool
dns-realm-id         dns realm identifier for local-pool
data-flow            data flow name for managing data traffic with the pool
dns-assignment       DNS address assignment
batch                enter all argument on one line
select               select a local address poolto edit
no                   delete selected address pool
show                 show selected local address pool
done                 write local address pool infromation
exit                 return to previous menu
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 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 only when **radius** or **radius-local** is specified as the DNS source by the **dns-assignment** parameter. The **dns-realm-id** parameter can be safely ignored if **local** is specified as the DNS source, or if the default value, which disables responses to incoming IKE_AUTH requests for DNS information.

When a remote RADIUS server is identified as a source of DNS information provide the name of the realm that supports transit to that RADIUS server.

```
ACMEPACKET(local-address-pool)# dns-realm-id core_010
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 the address-range configuration element).

```
ACMEPACKET(local-address-pool)# address-range
ACMEPACKET(address-range)#
network-address    name for the local address pool
subnet-mask        address ranges for the local address pool
batch              enter all argument on one line
select             select a local address pool range to edit
no                 delete selected local address pool range
show               show selected local address pool range
done               write local address pool range infromation
exit               return to previous menu
ACMEPACKET(address-range)#
```

7. If constructing an address pool that consists of one or more ranges of contiguous IPv4 addresses, use **network-address** in conjunction with **subnet-mask** to define a contiguous range 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
```

   ACMEPACKET(address-range)#

   If constructing an address pool that consists of one or more ranges of contiguous IPv6 addresses, use **network-address** parameter to provide both the IPv6 address and the bit length of the network prefix (an integer within the range 1 through 128).

   With IPv6 addresses, the **subnet-mask** parameter should be left blank.

   The following ACLI sequence defines a range of IPV6 addresses from 1080:0:0:0:0:0 through 1080:0:0:0:0:0:FFFF:FFFF:FFFF.

```
ACMEPACKET(address-range)# network-address 1080::ac10:202/96
ACMEPACKET(address-range)#
```

8. Use **done** and **exit** to complete configuration of the address-range instance.
9. If required, repeat Steps 6 and 7 to add additional address ranges to this address-range instance
10. After adding all address ranges, use **done**, **exit**, and **verify-config** to complete configuration of the local-address-pool instance.
11. Repeat Steps 1 through 10 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 RFC 7296, 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.

Persistent Tunnel Addressing Configuration

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.

ike-keyid 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)#
name           name for describing the KEY-ID
keyid          KEY-ID string
presharedkey   presharedkey for the KEY-ID
batch          enter all arguments on one line
select         select a key-id to edit
no             delete selected key-id
```

```
show            show selected key-id
done            write key-id information
exit            return to previous menu
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 subelement defines a single, global Internet Key Exchange (IKE) configuration object.

Parameters

state

Enter the state (enabled or disabled) of the ike-config configuration element.

- Default: enabled
- Values: disabled | disabled

ike-version

Enter an integer value that specifies IKE version.

Select 1 for IKEV1 protocol implementation.

Select 2 for IKEV2 protocol implementation.

- Default: 2
- Values: 1 | 2

log-level

Enter the IKE log level; events of this level and other events deemed more critical are written to the system log.

- Default: info
- Values: emergency | critical | major | minor | warning | notice | info | trace | debug | detail

udp-port

Enter the UDP port used for IKEv1 protocol traffic.

- Default: 500
- Values: Min: 1025 / Max: 65535

negotiation-timeout

Enter the maximum interval between Diffie-Hellman message exchanges.

- Default: 15 (seconds)
- Values: Min: 1 / Max:4294967295 (seconds)

Note: In the event of timer expiration, the IKE initiator must restart the Diffie-Hellman exchange.

event-timeout

Enter the maximum time allowed for the duration of an IKEv1 event, defined as the successful establishment of an IKE or IPsec Security Association (SA).

- Default: 60 (seconds)
- Values: Min: 1 / Max:4294967295 (seconds)

> **Note:** In the event of timer expiration, the IKE initiator must restart the Phase 1 (IKE SA) or Phase 2 (IPsec SA) process.

phase1-mode Enter the IKE phase 1 exchange mode: aggressive or main.

- Default: main
- Values:
  - aggressive—is less verbose (requiring only three messages), but less secure in providing no identity protection, and less flexible in IKE SA negotiation
  - main—is more verbose, but provides greater security in that it does not reveal the identity of the IKE peers. Main mode requires six messages (3 requests and corresponding responses) to (1) negotiate the IKE SA, (2) perform a Diffie-Hellman exchange of cryptographic material, and (3) authenticate the remote peer

phase1-dh-mode Enter the Diffie-Hellman group used during IKE phase 1 negotiation.

- Default: first-supported
- Values:
  - dh-group1 — as initiator, propose Diffie-Hellman group 1 (768-bit primes, less secure)
  - dh-group2 — as initiator, propose Diffie-Hellman group 2 (1024-bit primes, more secure)
  - first-supported — as responder, use the first supported Diffie-Hellman group proposed by initiator

> **Note:** Diffie-Hellman groups determine the lengths of the prime numbers exchanged during the symmetric key generation process.

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

- Default: 86400 (24 hours)
- Values: Min: 1 / Max: 4294967295 (seconds)

> **Note:** This global default can be over-ridden at the IKEv2 interface level.

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

- Default: 28800 (8 hours)
- Values: Min: 1 / Max:4294967295 (seconds)

> **Note:** This global default can be over-ridden at the IKEv2 interface level.

phase1-life-seconds Set the time (in seconds) proposed for IKE SA expiration during IKE Phase 1 negotiations.

- Default: 3600 (1 hour)
- Values: Min: 1 / Max: 4294967295 (seconds)

> **Note:** Relevant only when the Oracle Communications Mobile Security Gateway is acting in the IKE initiator role.

phase2-life-seconds relevant only when the Oracle Communications Mobile Security Gateway is acting in the IKE initiator role, contains the time proposed (in seconds) for IPsec SA expiration during IKE Phase 2 negotiations.

- Default: 28800 (8 hours)

- Values: Min: 1 / Max:4294967295 (seconds)

  Note: During IKE Phase 2, the IKE initiator and responder establish the IPsec SA.

phase2-life-seconds-max Set the maximum time (in seconds) accepted for IPsec SA expiration during IKE Phase 2 negotiations.

- Default: 86400 (24 hours)
- Values: Min: 1 / Max: 4294967295 (seconds)

  Note: Relevant only when the Oracle Communications Mobile Security Gateway is acting in the IKE responder role.

phase2-exchange-mode Enter the Diffie-Hellman group used during IKE Phase 2 negotiation.

- Default: phase1-group
- Values:
  - dh-group1 — use Diffie-Hellman group 1 (768-bit primes, less secure)
  - dh-group2 — use Diffie-Hellman group 2 (1024-bit primes, more secure)
  - no-forward-secrecy — use the same key as used during Phase 1 negotiation

    Note: During IKE Phase 2, the IKE initiator and responder establish the IPsec SA.

    Diffie-Hellman groups determine the lengths of the prime numbers exchanged during the symmetric key generation process.

shared-password Enter the default PSK used during IKE SA authentication.

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

- Default: None
- Values: A string of ACSII-printable characters no longer than 255 characters (not displayed by the ACLI)

eap-protocol Enter the EAP protocol used with IKEv2.

- Default: eap-radius-passthru
- Values: eap-radius-passthru

  Note: The current software performs EAP operations by a designated RADIUS server or server group; retain the default value.

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

- 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

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

eap-bypass-identity Contains a value specifying 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 protocol exchange. 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.

- Default: disabled (requires an identity exchange)
- Values: disabled | enabled

red-port

Enter the port number monitored for IKEv2 synchronization messages; used in high-availability environments.

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: Min: 1024 / Max: 65535

red-max-trans

For HA nodes, set the maximum number of retained IKEv2 synchronization message.

- Default: 10000 (messages)
- Values: Min: 1 / Max: 4294967295 (messages)

red-sync-start-time

For HA nodes, set the timer value for transitioning from standby to active role — the amount of time (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: Min: 1 / Max:4294967295 (milliseconds)

red-sync-comp-time

For HA nodes, set the interval between synchronization attempts after the completion of an IKEv2 redundancy check.

- Default: 1000 (milliseconds)
- Values: Min: 1 / Max:4294967295 (milliseconds)

dpd-time-interval

Set the maximum period of inactivity (in seconds) 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: Min: 1 / Max:4294967295 (seconds)

overload-threshold

Set 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 of overload-critical-threshold

overload-interval

Set the interval (in seconds) between CPU load measurements while in the overload state.

- Default: 1
- Values: Min: 0 / Max: 60

overload-action

Select the action to take when the Oracle Communications Mobile Security Gateway (as a SG) CPU enters an overload state. The overload state is reached when CPU usage exceeds the percentage threshold specified by the overload-threshold

- Default: none
- Values: • drop-new-connection—use to implement call rejection
- none—use to retain default behavior (no action)

overload-critical-threshold

Set the percentage of CPU usage that triggers a critical overload state. This value must be greater than the value of overload-threshold.

- Default: 100 (disabling overload processing)
- Values: Min: 0 / Max: 100

overload-critical-interval

Set the interval (in seconds) between CPU load measurements while in the critical overload state.

- Default: shared-password
- Values: Min: 0 / Max: 60

sd-authentication-method

Select the method used to authenticate the IKEv2 SA. Two authentication methods are supported.

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

- Default: shared-password
- Values:
  - certificate—uses an X.509 certificate to digitally sign a block of data
  - shared-password—uses a PSK that is used to calculate a hash over a block of data

certificate-profile-id

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 IKEv2 interface level.
- Default: None
- Values: Name of an existing ike-certificate-profile configuration element.

Path

ike-config is a subelement under the ike element. The full path from the topmost ACLI prompt is: **configure-terminal** > **security** > **ike** > **ike-config.**

Release

First appearance: S-C6.2.0

RTC Status

Supported

Note: This is a single instance configuration element.

ike-certificate-profile ACLI Reference

ike-certificate-profile

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

Parameters

identity

Enter the local IKEv2 entity that using 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

end-entity-certificate

Enter 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.

- Default: None
- Values: Name of an existing certificate-record configuration element

trusted-ca-certificates

Enter the unique names of one or more certificate-record configuration elements referencing Certification Authority (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

Enter the maximum number of chained certificates that will be processed while authenticating the IKEv2 peer.

- Default: 10
- Values: Min: 1 | Max: 10

Path

ike-certificate-profile is a subelement under the **ike** element. The full path from the topmost ACLI prompt is: **configure-terminal** > **security** > **ike** > **ike-certificate-profile**.

Release

First appearance: S-C6.2.0

RTC Status

Supported

Note: This is a multiple instance configuration element.

radius-servers ACLI Reference

authentication > radius-servers

The radius-servers subelement defines and configures the RADIUS servers that the Oracle Communications Mobile Security Gateway communicates with.

Parameters

address

Enter the IP address for the RADIUS server. An IPv4 or IPv6 address is valid for this parameter.

port

Enter the port number on the remote IP address for the RADIUS server

- Default: 1812
- Values: 1645 | 1812

state

Enable or disable this configured RADIUS server

- Default: enabled
- Values: enabled | disabled

secret Enter the password the RADIUS server and the Oracle Communications Mobile Security Gateway share. This password is not transmitted between the two when the request for authentication is initiated.

nas-id Enter the NAS ID for the RADIUS server

realm-id Enter the RADIUS server realm ID.

retry-limit Set the number of times the Oracle Communications Mobile Security Gateway retries to authenticate with this RADIUS server

- Default: 3
- Values: Min: 1 / Max: 5

retry-time Enter the time in seconds the Oracle Communications Mobile Security Gateway waits before retrying to authenticate with this RADIUS server

- Default: 5
- Values: Min: 5 / Max: 10

maximum-sessions Enter the maximum number of sessions to maintain with this RADIUS server

- Default: 255
- Values: Min: 1 / Max: 255

class Select the class of this RADIUS server as either primary or secondary. A connection to the primary server is tried before a connection to the secondary server is tried.

- Default: primary
- Values: primary | secondary

dead-time Set the time in seconds before the Oracle Communications Mobile Security Gateway retries a RADIUS server that it has designated as dead

- Default: 10
- Values: Min: 10 / Max: 10000

authentication-methods Select the authentication method the Oracle Communications Mobile Security Gateway uses when communicating with the RADIUS server

- Default: pap
- Values: all | pap | chap | mschapv2

Path

radius-servers is a subelement under the **authentication** configuration element under the security path. The full path from the topmost prompt is: **configure terminal** > **security** > **authentication** > **radius-servers.**

Release

First appearance: 4.0 / Most recent update: S-CZ7.2.0

RTC Status

Supported

data-flow ACLI Reference

data-flow

The data-flow configuration element specifies pass-through data-traffic processing when using IKE.

Parameters

name Specify the name of this instance of the data-flow configuration element.

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

group-size Specify the maximum number of user elements grouped together by this data-flow instance. For maximum efficiency, this value should be set to a power of 2.

- Default: 128
- Values: 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256

Note:

The optional group-size parameter specifies the divisor used by this data-flow instance to segment the total address pool into smaller, individually-policed segments.

upstream-rate Specify the allocated upstream bandwidth.

- Default: 0 (allocates all available bandwidth)
- Values: Min: 0 / Max: 4294967295

downstream-rate Specify the allocated downstream (access side) bandwidth.

- Default: 0 (unlimited, no bandwidth restrictions)
- Values: Min: 0 / Max: 4294967295

Path

Data-flow is a subelement under the ike element. The full path from the topmost ACLI prompt is.**configure terminal** > **security** > **ike** > **data-flow**.

First appearance

S-C6.2.0

RTC Status

Supported

Note: This is a multiple instance configuration element.

local-address-pool ACLI Reference

local-address-pool

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.

Parameters

name Enter a unique identifier for this local-address-pool instance.

- Default None
- Values A valid configuration element name that is unique within the local-address-pool namespace.

address-range Access the address-range subelement.

dns-realm-id Enter a DNS realm that supports this local-address-pool instance.

- Default: None
- Values: Name of an existing dns-realm configuration element.

data-flow-list Enter a data-flow configuration element assaigned to this local-address-pool instance. This parameter specifies bandwidth availible to the pool of addresses specified by this local-address-pool instance.

- Default: None
- Values: Name of an existing data-flow configuration element local-address-pool is a subelement under the ike element. The full path from the topmost ACLI prompt is: **configure>terminal>security>ike>local-address-pool.**

Path

local-address-pool is a subelement under the ike element. The full path from the topmost ACLI prompt is: **configure terminal > security > ike > local-address-pool**

Release

First appearance: S-C6.2.0

RTC Status

S-C6.2.0

Note: This is a multiple instance configuration element.

local-address-pool > address-range

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

Parameters

network-address In conjunction with this parameter 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, the parameter defines a range of IPv4 addresses available for dynamic assignment.

- Default: None
- Values: A valid IPv4 subnet mask

Path

local-address-pool > **address-range** is a subelement under the ike element. The full path from the topmost ACLI prompt: **configure-terminal** > **security** > **ike** > **local-address-pool** > **address-range**.

Release

First appearance: S-C6.2.0

RTC Status

Supported

Note: This is a multiple instance configuration.

ike-keyid ACLI Reference

ike-keyid

The ike-keyid element associates a PDK with an IKEv2 asserted identity (IDi for the IKE initiator or IDr for the IKE responder) carried within the IKE Identification Payload. 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).

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.

key-id 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

presharedkey Specifies the pre-shared key for the key-id.

- Default: none
- Values: a string containing the pre-shared key

Path

ike-keyid is a subelement under the **ike** element. The full path from the top-most ACLI prompt is: **configure terminal** > **security** > **ike** > **ike-keyid**.

Release

First appearance: M-C1.0.0

RTC Status

Supported.

Note: This is a multiple instance configuration element.

3

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 access control whit/black 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 exchanges 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 cryptographic functions).

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

HDR The common IKEv2 header

SAi1 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 The initiator’s Key Exchange payload (defined in Section 3.3 of RFC 4306). It contains the initiator’s public Diffie-Hellman value.

Ni 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 cryptographic functions to increase randomness.

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

HDR The common IKEv2 header

SAr1 The Security Gateway’s Security Association payload that specifies cryptographic material used for creation of the IKEv2 SA. It selects from the proposals offered in the initiator’s corresponding payload.

KEr The Security Gateway’s Key Exchange payload. It contains the gateway’s public Diffie-Hellman value.

Nr 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 cryptographic functions to increase randomness.

[CertReq] 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 The common IKEv2 header

{ Indicates the beginning of encryption.

IDi The initiator’s Identification payload (defined in Section 3.5 of RFC 4306). It contains the identification type (for example, IPv4/IPv6 address or FQDN) and the actual address of the initiator.

[Cert] 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] 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] 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 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 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] 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 A proposed 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 A proposed 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 The common IKEv2 header

{ Indicates the beginning of encryption

IDr The responder’s Identification payload. It contains the identification type (for example, IPv4/IPv6 address or FQDN) and the actual address of the responder.

[Cert] 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 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 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] 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 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 initiator's proposed traffic selectors 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.

Note: Prior to Release M-CX3.0.0M2, the Security Gateway negotiated Traffic Selectors based upon the first TSi/TSr proposal presented by the initiator of the IKE_AUTH exchange. If either the TSi or TSr payload contained mutliple proposals, those proposals after the first, were ignored by the gateway. This processing does not comply with Section 2.9 of RFC 7296, Internet Key Exchange Protocol Version 2 (IKEv2), which requires that all proposed TSi and TSr payloads be considered in the negotiation process. Release M-CX3.0.0M2 (and later releases) comply with RFC 7296 requirements in including all proposed TSi and TSr payloads in the final Traffic Selector determination.

TSr 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 initiator's proposed traffic selectors 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 The common IKEv2 header

{ Indicates the beginning of encryption

SA 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 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 cryptographic functions to increase randomness.

[KEi] 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] 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 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 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 The common IKEv2 header

{ Indicates the beginning of encryption

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

Nr 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 cryptographic functions to increase randomness.

[KEr] 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] 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 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 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 support numerous authentication methods.

The current Security Gateway release supports for EAP-based authentication methods. Each supported method is described in the following four sections. Note that for all currently supported EAP authentication methods that the actual authentication is provided by an adjacent RADIUS server. During the EAP-based authentication exchange the Security Gateway functions as a packet relay between the authenticating client(s) 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.

Steps 11 and 12 report successful authentication.

IPv6 Inner Tunnel Address Assignment

The Security Gateway supports the assignment of IPv6 inner tunnel addresses utilizing an external RADIUS server as the IPv6 address source. During the EAP authentication of an IPsec host, neither the Security Gateway nor the RADIUS authentication server has any knowledge of the traffic type (IPv4 or IPv6) that the IPsec host intends to transmit through the tunnel. Consequently, the RADIUS authentication server may send both IPv4 and IPv6 attributes in the RADIUS
Access-Accept message, leaving it to the Security Gateway to select the appropriate attribute and ignore the other.

The Security Gateway makes its decision based on the contents of the Configuration Payload received from the IPsec host. If the payload contains an INTERNAL_IP4_ADDRESS attribute, the IPv4 address received in the Access-Accept message is forwarded to the IPsec host. In a similar fashion, if the payload contains an INTERNAL_IP6_ADDRESS attribute, the IPv6 address received in the
Access-Accept message is forwarded to the IPsec host.

Assignment of IPv6 addresses requires support for the following RADIUS attributes:

- Framed-IPv6-Prefix (Type 97) — also used in RADIUS accounting
- Framed-IPv6-Pool (Type 100)

Framed-IPv6-Pool, which can be returned by a RADIUS authentication server in an Access-Accept message, contains the name of an address pool that should be used by the Security Gateway as a source of IPv6 addresses.Use of Framed-IPv6-Pool requires the pre-configuration of the identified address pool on the Security Gateway.

EAP-only Authentication

IKEv2 specifies that Extensible Authentication Protocol (EAP) authentication must be used together with responder authentication based on public key signatures. This is necessary with old EAP methods that provide only unilateral authentication using, for example, one-time passwords or token cards. With EAP-SIM, EAP-AKA, EAP-AKA’, EAP-TTLS, and EAP-TLS, which provide mutual authentication and key agreement, extensible responder authentication for IKEv2 based on methods other than public key signatures can be used. This feature causes the MSG to default to EAP-only authentication without using public-key-based responder authentication unless the operator selects otherwise.

The Extensible Authentication Protocol, defined in RFC3748, is an authentication framework that supports multiple authentication mechanisms. One of the advantages of the EAP architecture is its flexibility. Rather than requiring the authenticator (for example, a wireless LAN access point) to be updated to support each new authentication method, EAP permits the use of a backend authentication server that may implement some or all authentication methods. The MSG uses a backend authentication server (for example, 3GPP AAA) and is in pass-through mode for EAP.

IKEv2 is a component of IPsec used for performing mutual authentication and establishing and maintaining Security Associations (SAs) for IPsec Encapsulating Security Payload (ESP) and Authentication Header (AH). In addition to supporting authentication using public key signatures and shared secrets, IKEv2 also supports EAP authentication. By using EAP, IKEv2 can leverage existing authentication infrastructure and credential databases, such as Home Subscriber Server (HSS), as EAP allows users to choose a method suitable for existing credentials, and also makes separation of the IKEv2 responder (MSG) from the EAP authentication endpoint (back-end Authentication, Authorization, and Accounting (AAA) server) easier. IKEv2 specifies that these EAP methods must also be used together with responder authentication based on public key signatures. For the public key signature authentication of the MSG to be effective, a deployment of Public Key Infrastructure (PKI) is required, which has to include management of trust anchors on all supplicants. This may not realistic in WiFi calling environments, in which the security of the MSG public key is the same as the security of a self-signed certificate. Mutually authenticating EAP methods alone can provide a sufficient level of security.

Because of these reasons, the MSG now defaults to EAP-only authentication without using public-key-based responder authentication unless the operator selects otherwise by disabling the new parameter **eap-only-support** in the **ike-interface** configuration element.

EAP-only Authentication Configuration

1. Access the **ike-interface** configuration element.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# ike
   ACMEPACKET(ike)# ike-interface
   ACMEPACKET(ike-interface)#
   ```
2. Select the **ike-interface** object to edit.

   ```
   ACMEPACKET(ike-interface)# select
   <address>:

   ACMEPACKET(ike-interface)#
   ```
3. **eapOnlyAuthSupport** — The default is **enabled**. Set the value to **disabled** to use EAP authentication together with responder authentication based on public key signatures.
4. Type **done** to save your configuration.

Debugging IKEv2 IPsec Tunnel Establishment

Prior to Release M-CX3.0.0M2, IKEv2 debug-level logging provided details of all IKE endpoints that were establishing IKEv2/IPsec tunnels. Service providers have requested finer grained controls that enable logging of target endpoints identified by IP address and
user-id . Release M-CX3.0.0M2 (and later releases) provide these requested controls.

In a typical deployment scenario, the IP address can be the public address of a NAT device that communicates with the Security Gateway; the user-id can be the user-id of a femtocell or an IKE endpoint residing behind the NAT. The user-id can be an EAP identity exchanged during EAP authentication, or the identity contained in the IDi payload of the initial IKE_AUTH message. Typically the identity in the IDi payload is an IP address, an FQDN, or an address as defined in RFC 822, Standard for the Format of ARPA Internet Text Messages.

Enabling/Disabling Targeted Debugging

Targeted debugging is enabled by the **security ike debug-logging peer-ip-userid** ACLI command which takes a single string argument in the form ipAddress:userID. For example:

```
ACMEPACKET# security ike debug-logging peer-ip-userid
172.16.20.1:12EDE12626719
ACMEPACKET#
```

With endpoint-specific logging enabled, the log.iked, log.authd, and log.secured files are populated with data pertinent to the target endpoint only and exclude date for all other endpoints. Logging is based on an exact match of the IP address and user-id provided by the argument string.

Note: This command is expensive and should be used to debug one or two endpoints at a time. The operating system imposes a hard limit of no more than 5 simultaneous targeted debugging sessions.

Use the no form of the command to stop an existing targeted debugging session

```
ACMEPACKET# security ike debug-logging peer-ip-userid
172.16.20.1:12EDE12626719 no
ACMEPACKET#
```

Use the **show security ike peer-endpoint-logging** ACLI command to display a list of configured debug-logging sessions

```
ACMEPACKET# show security ike peer-endpoint-logging
ACMEPACKET#
IPaddress : Userid
==============
172.16.20.1:12EDE12626719
ACMEPACKET#
```

High Availability Caveat

Since the security ike debug-logging peer-ip-userid command is expensive, this implementation intentionally does NOT synchronize log data on the active and standby HA devices. Consequently, in the event of a switchover from the active to the standby, no log data is available on the newly active device. To enable debug-logging on the new active device, the user should verify tunnel establishment, and then use security ike debug-logging peer-ip-userid command on the currently active member of the HA pair.

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)# ?
state                       state [enabled|disabled]
address                     IPv4/IPv6 Address
realm-id                    realm identifier
ike-mode                    allowed ike modes
local-address-pool-id-list  list of local address pools
dpd-params-name             dpd params container name
v2-ike-life-secs            (IKEv2) IKE SA lifetime (secs)
v2-ipsec-life-secs          (IKEv2) IPsec SA lifetime (secs)
v2-rekey                    whether to do IKE/IPSEC rekey as a responder
multiple-authentication     enable/disable multiple IKE Authentication
multiple-child-sa-mode      multiple-child-sa-mode
shared-password             shared password
options                     optional features/parameters
eap-protocol                Ike EAP Protocols
addr-assignment             local address assignment method
sd-authentication-method    allowed sd authentication methods
certificate-profile-id-list list of identities for the certificate-profile
                            containers, the identity should be FQDN or IP
                            address
tunnel-orig-name-list       list of names for the tunnel origination
                            parameter containers
threshold-crossing-alert-group-name   threshold-crossing-alert-group object
                                      for IKE-related counters
cert-status-check    cert-status-check   state[enabled|disabled]
cert-status-profile-list    list of cert-status-profiles for status requests
access-control-name         Name of the ike access control to be used for
                            the ike-interface
accounting-param-name       Name of the "ike-accounting-param" element
                            configured under security >> ike
traffic-selectors           config traffic-selectors (no more than 20)
add-traffic-selectors       add one or more traffic selectors, no more than
                            20 in total
```

```
remove-traffic-selectors    remove one or more traffic selectors
ip-subnets                  ip-subnets (maximum 10)
add-ip-subnet               add one or more ip-subnets, no more than 10 in
                            total
remove-ip-subnet            remove one or more ip-subnets
authorization               radius authorization [enabled|disabled]
authentication-servers      list of authentication radius server ip
                            addresses (currently only 1 server is supported)
authorization-servers       list of authorization radius server ip
                            addresses (currently only 1 server is supported)
batch                       enter all arguments on one line
select                      select a ike interface to edit
no                          delete selected ike interface
show                        show ike interface information
done                        write ike interface information
exit                        return to previous menu
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 or IPv6 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.

   sd-authentication-method can be safely ignored, if authentication utilizes any of the methods described in EAP-based Authentication.

   Two 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 Two-Factor authentication, select the method used during the initial exchange to authentication the remote IKEv2 peer (for example, a femtocell).

```
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 access control white or black list to this IKEv2 interface.

    This parameter is meaningful only when authentication uses a RADIUS server to implement the EAP-based authentication, and can otherwise be safely ignored. If authentication is performed using EAP, **access-control-name** enables the assignment of an existing white or black access control list to the current IKEv2 interface.

    White lists specify IMSI prefixes or MAC addresses that are passed through to the RADIUS authentication server. Black list also specify IMSI prefixes or MAC addresses; contents of black lists, however, are immediately denied authentication by the Security Gateway.

    Use **access-control-name**, in conjunction with the name of an existing white or black list, to assign that list to the current IKEv2 interface.

    ```
    ACMEPACKET(ike-interface)# access-control-name white_01
    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, a remote IKEv2 peer initiates the exchange by requesting an IP address on the gateway's protected network. In response, the Security Gateway returns a local address for the peer's temporary 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 containing an INTERNAL_IP4_SUBNET or INTERNAL_IP6_SUBNET attribute.

    In certain Security Gateway environments, Unlicensed Mobile Access (UMA) handsets issue a CFG_REQUEST that contains an INTERNAL_IP4_SUBNET or INTERNAL_IP6_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 IPv4 or IPv6 subnets to this IKEv2 interface. For IPv4 addresses, 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. For IPv6 addresses each list item is an IPv6 address followed by the bit-length of the network prefix. Regardless of the address family, 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.

    Note: A subnet list can contain a maximum of 12 subnet entries. However, all subnets must be of the same IP address family, either IPv4 or IPv6.

    You cannot include IPv4 and IPv6 addresses in the same subnet 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 either an INTERNAL_IP4_SUBNET or INTERNAL_IP6_SUBNET attribute in a CFG_REPLY. Use double quotes (“) to delimit the list.

    For IPv4 subnets:

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

    For IPv6 subnets:

    ```
    ACMEPACKET(ike-interface)# ip-subnets 1080::ac10:102/ 1090::ac10:202/96
    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 20 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.

While the above example processing used IPv4 addresses, the traffic selector list can contain either IPv4 or IPv6 ranges. The list supports a maximum of 20 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 (for IPv4) or prefix (for IPv6). The absence of a mask indicates that the address is to be interpreted as a host address. Use double quotes (“) to delimit the list.

For IPv4-based traffic selectors:

```
ACMEPACKET(ike-interface)# traffic-selectors 172.16.2.191 192.168.5.203/24
ACMEPACKET(ike-interface)#
```

For IPv6-based traffic selectors:

```
ACMEPACKET(ike-interface)# traffic-selectors 1080::ac10:102/
1090::ac10:202/96
ACMEPACKET(ike-interface)#
```

Version M-CX3.0.0 and subsequent releases provide for finer granularity in the the design of traffic selectors. While previous releases were limited to IP addresses and subnet masks (IPv4) or prefixes (IPv6) for traffic selector design, these later releases optionally support traffic selector design at the port and/or protocol level.

The expanded traffic selector format can be summarized as follows:

<ipAddress>/<prefix>[<startPort>~<endPort>,<protocol#>]

ipAddress	is the required IPv4 or IPv6 network address
prefix	is the required IPv4 or IPv6 prefix identifying the significant portion of the network address
[	is the start delimeter for port/protocol data, and is required when such data is present in the traffic selector

startPort	is the required floor for a contiguous range of port numbers (integers within the range 0 through 65535); defaults to 0
~	is the required seperator between start and end ports
endPort	is the required ceiling for a contiguous range of port numbers (integers within the range 0 through 65535); must be equal to or greater than the value assigned to startPort; defaults to 65535
,	is the seperator between port and protocol values, and is required
protocol#	is a required protocol number as assigned by the IANA, for example 6 for TCP, or 17 for UDP; the special value 0 (the default value) is interpreted as a wildcard, matching all protocols
]	is the end delimeter for port/protocol data, and is required when such data is present in the traffic selector

For example:

- 172.16.0.0/16, and
- 172.16.0.0/16[1~65535,0] define the same range
- 172.16.0.0/16[1~65535,17] defines the same range, restricts to UDP
- 172.16.0.0/16[1200~1200,1] defines the same range, restricts to ICMP on port 1200
- 172.16.0.0/16[1~65535,17] defines the same range, restricts to UDP
- 172.16.0.0/16[1200~1200,1] defines the same range, restricts to ICMP on port 1200

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, and only currently-supported value, eap-radius-passthru, specifies the use of a RADIUS authentication server for all EAP-based authentication. For EAP-based authentication, the Security Gateway simply relays EAP requests and EAP responses between the remote IKEv2 peer and a RADIUS authentication server. The peer’s authentication request is evaluated and then accepted or rejected by the RADIUS authentication server.

Refer to RADIUS Authentication in IKEv2 Global Configuration for information about configuring a pool of RADIUS authentication servers.

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 the optional **authentication-servers** parameter to identify a single, specific RADIUS authentication server that provides EAP-based authentication services to the current IKEv2 interface.

    By default EAP-based authentication requests are referred to a globally-defined RADIUS Authentication Servers List (refer to RADIUS Authentication Servers and RADIUS Authentication Servers List in IKEv2 Global Configuration) for evaluation and determination.

    You can alter the default behavior, by assigning an existing, previously-configured RADIUS authentication server, identified by its IP address, to the current IKEv2 interface.

    Note that if the assigned RADIUS authentication server becomes unavailable, any EAP-based authentication requests cannot be processed and will be rejected.

    In addition to the incoming authentication request, the Security Gateway also sends two RADIUS attributes to the authentication server:

    Called-Station-ID (type 30) which contains the IP address of the transmitting IKEv2 interface.

    NAS-Port (type 5) which contains the configured RADIUS server port.

    ```
    ACMEPACKET(ike-interface)# authentication-servers 172.30.0.1
    ACMEPACKET(ike-interface)#
    ```

24. Use the optional **authorization** parameter to enable RADIUS authorization on the current IKEv2 interface.

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

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

25. Use the optional **authorization-servers** parameter to identify a single, specific RADIUS authorization server that provides authorization services to the current IKEv2 interface.

    By default authorization requests are referred to a globally-defined RADIUS Authorization Servers List (refer to RADIUS Authorization Servers and RADIUS Authorization Servers List in IKEv2 Global Configuration) for evaluation and determination.

    You can alter the default behavior, by assigning an existing, previously-configured RADIUS authorization server, identified by its IP address, to the current IKEv2 interface.

    Note that if the assigned RADIUS authorization server becomes unavailable, any received authorization requests cannot be processed, and will be subsequently rejected.

    In addition to the incoming authentication request, the Security Gateway also sends two RADIUS attributes to the authentication server:

    Called-Station-ID (type 30) which contains the IP address of the transmitting IKEv2 interface.

    NAS-Port (type 5) which contains the configured RADIUS server port.

    ```
    ACMEPACKET(ike-interface)# authorization-servers 172.30.0.10
    ACMEPACKET(ike-interface)#
    ```

26. Use **done**, **exit**, and **verify-config** to complete configuration of IKEv2 interface.
27. 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.

IPsec Security Policy Configuration

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.

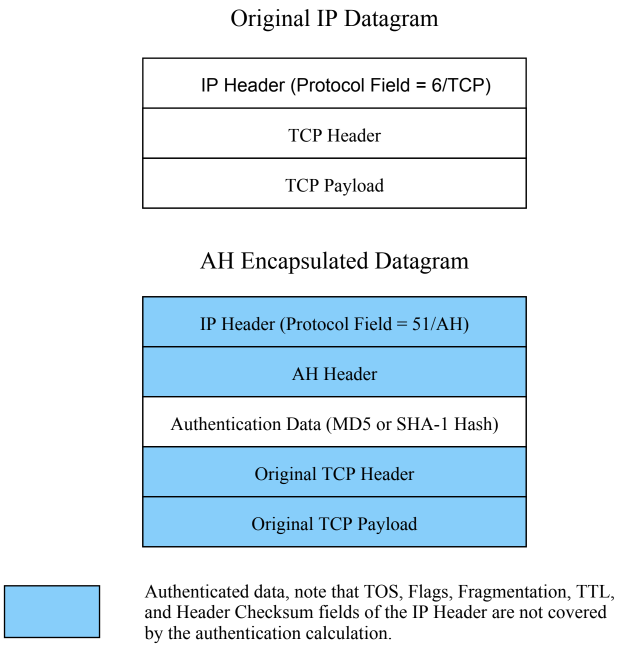
   Authentication Header (AH) — the default value — as defined by RFC 4302, IP Authentication Header, which provides authentication integrity to include the mutual identification of remote peers, non-repudiation of received traffic, detection of data that has been altered in transit, and detection of data 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

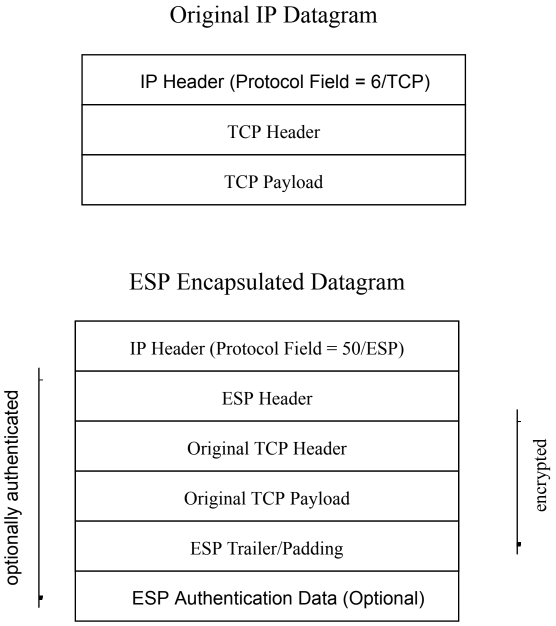
   ESP-AUTH (also RFC 4303-based), which supports ESP's optional authentication.

   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)#
```


Original IP Datagram
IP Header (Protocol Field = 6/TCP)
TCP Header
TCP Payload
AH Encapsulated Datagram
IP Header (Protocol Field = 51/AH)
AH Header
Authentication Data (MD5 or SHA-1 Hash)
Original TCP Header
Original TCP Payload
Authenticated data, note that TOS, Flags, Fragmentation, TTL, and Header Checksum fields of the IP Header are not covered by the authentication calculation.

AH Transport Mode



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

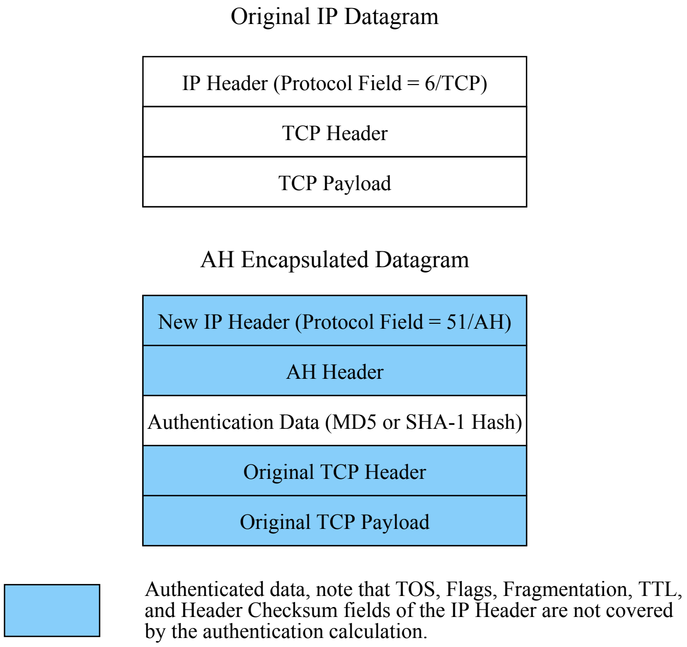
   ACMEPACKET(ike-sainfo)#

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

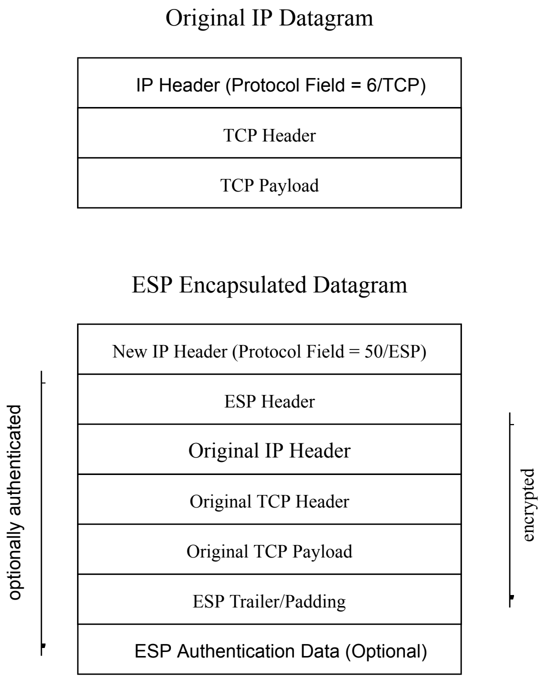
   Transport mode (the default) provides a secure end-to-end connection between two IP hosts. Transport mode encapsulates the IP payload.

   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)#
```


Original IP Datagram
IP Header (Protocol Field = 6/TCP)
TCP Header
TCP Payload
AH Encapsulated Datagram
New IP Header (Protocol Field = 51/AH)
AH Header
Authentication Data (MD5 or SHA-1 Hash)
Original TCP Header
Original TCP Payload
Authenticated data, note that TOS, Flags, Fragmentation, TTL, and Header Checksum fields of the IP Header are not covered by the authentication calculation.

AH Tunnel Mode



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. Use the value, 0.0.0.0 (matching the local IP address), for the **local-ip-mask** parameter.
13. Retain the default value, 255.255.255.255, for the **remote-ip-mask** parameter.
14. Use the **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)#
```

After a positive match between outbound traffic and the configured selectors in the security policy, the Security Gateway can perform a calculation between a set of fine-grained packet selectors and the outbound packet. The fine-grained policy masking criteria are:

- Source IP subnet mask
- Destination IP subnet mask
- VLAN mask

By default, the fine-grained security policy is set to match and pass all traffic untouched to the security association (SA) portion of IPSec processing.

Fine-grained policy selection works by performing a logical AND between outbound traffic's fine-grained selectors and the traffic's corresponding attributes. The result is then used to find the matching SA. Applying a fine-grained mask has the effect of forcing a contiguous block of IP addresses and/or ports to appear as one address and or port. During the next step of IPSec processing, when an SA is chosen, the Security Gateway in effect uses one SA lookup for a series of addresses. Without fine-grained policy selection, unique SAs need be configured for outbound packets with unique security policy selectors.

Use the following procedure to define a fine-grained security policy.

1. **outbound-sa-fine-grained-mask**—Use this command to move to outbound-sa-fine-grained-mask configuration mode.
2. Use the value, 0.0.0.0 (matching the local IP address), for the **local-ip-mask** parameter.
3. Retain the default value, 255.255.255.255 (matching all remote IP addresses), for the **remote-ip-mask** parameter.
4. **local-port-mask**—Retain the default value, 0.
5. **remote-port-mask**—Retain the default value, 0.
6. **trans-protocol-mask**—Retain the default value, 0.
7. **valid**—Retain the default value, enabled.
8. **vlan-mask**—Retain the default value, 0.

   ```
   Save your work using the ACLI done, exit, and verify-config commands.
   ```
9. Repeat Steps 1 through 21 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                0.0.0.0
remote-ip-mask               255.255.255.255
action                       allow
ike-sainfo-name
outbound-sa-fine-grained-mask
local-ip-mask                0.0.0.0
remote-ip-mask               255.255.255.255
local-port-mask              0
remote-port-mask             0
trans-protocol-mask          0
valid                        enabled
vlan-mask                    0x000
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                0.0.0.0
remote-ip-mask               255.255.255.255
action                       ipsec
ike-sainfo-name              SA-1
outbound-sa-fine-grained-mask
local-ip-mask                0.0.0.0
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
```

Tunnel Pass-Through

You can use IPsec Security Policies to enable tunnel pass-through as shown below.

Pass-through IPv4 traffic via an IPv4 tunnel	
	1. Configure IPv4 allow policy for IKE protocol traffic 2. Configure IPv4 ipsec policy for media traffic 3. Configure the IKEv2 IPv4 interface with an IPv4 local address pool, or 4. Configure the RADIUS server to return a Framed-IP-Address and/or Framed-IP-Netmask attribute
Pass-through IPv6 traffic via an IPv6 tunnel	
	1. Configure IPv6 allow policy for IKE protocol traffic 2. Configure IPv6 ipsec policy for media traffic 3. Configure the IKEv2 IPv6 interface with an IPv6local address pool, or 4. Configure the RADIUS server to return a Framed-IPv6-Prefix or Framed-IPv6-Pool attribute
Pass-through IPv4 traffic via an IPv6 tunnel	
	1. Configure IPv6 allow policy for IKE protocol traffic 2. Configure IPv4 ipsec policy for media traffic 3. Configure the IKEv2 IPv6 interface with an IPv4 local address pool, or 4. Configure the RADIUS server to return a Framed-IP Address and/or Framed-IP-Netmask attribute
Pass-through IPv6 traffic via an IPv4 tunnel	
	1. Configure IPv4 allow policy for IKE protocol traffic 2. Configure IPv6 ipsec policy for media traffic

	3. Configure the IKEv2 IPv4 interface with an IPv6local address pool, or 4. Configure the RADIUS server to return a Framed-IPv6-Prefix or Framed-IPv6-Pool attribute

Pre-Populated ARP Table

In certain topologies remote IPsec endpoints can require access to core network hosts reachable through a Security Gateway core interface. In these instances, the Security Gateway receives the tunneled packet, and masks the received IP destination address against its own local addresses to determines if direct delivery is possible. If so, the Security Gateway issues an ARP request to obtain the physical destination address.

This process can be expedited by pre-populating the interface-specific ARP table with a list of commonly accessed core network host 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 an 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. If necessary, repeat Steps 1 through 6 to configure pre-populated ARP tables for other Security Gateway IKEv2 interfaces.

   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)#
```

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 mismanagement 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. CRL updates can be deliver in either PEM (Privacy Enhanced Email) or DER (Distinguished Encoding Rules) format. PEM is base-64 encoded ASCII that provides BEGIN CERTIFICATE and END CERTIFICATE statements; DIR is a binary rendering of the PEM format. Both formats (PEM and DIR) are supported by the Security Gateway.

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.

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.

CRL-Based Certificate Verification Configuration

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 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 address. **host-name** identifies the source by its fully-qualified domain name (FQDN).

   If you provide an IP 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.

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.

Associate a CRL Source with an IKEv2 Interface

1. 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)#
    ```

2. Use **done**, **exit**, and **verify-config** to complete CRL configuration on the currentIKEv2 interface.
3. Repeat Steps 1 through 20 to configure additional IKEv2 interfaces for CRL usage.
4. 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 cert-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.

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.

OCSP-Based Certificate Verification Configuration

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

   Allowable **port** values are integers within the range 1025 through 65535. In the absence of an explicitly configured value, 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.

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.

Associate an OCSP Responder with an IKEv2 Interface

1. 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)#
   ```
2. Use **done**, **exit**, and **verify-config** to complete OCSP configuration on an IKEv2 interface.
3. Repeat Steps 1 through 5 to configure additional IKEv2 interfaces for OCSP operations.
4. 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 cert-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 Access Control

Since Release M-CX1.1.0, the MSG has supported IKEv2 access-control for EAP authenticating clients in the form of white lists that permit authentication only for a provisioned list of IMSI prefixes. Services providers have subsequently requested reverse access-control functionality in the form of MAC-address-based black lists that deny authentication to a provisioned list of MAC addresses. In response to the request for blacklisting capabilites, Release M-CX3.0.0M2 and later releases introduce support for black lists that deny access based on both IMSI prefixes and MAC addresses, while enhancing white list support to grant access based on both IMSI prefixes and MAC addresses.

Configuring 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/MAC 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.

Prior to Release MC-X3.0.0M2, these strings or regular expressions were interpreted exclusively as an IMSI or as an IMSI prefix. White lists consisted of one or more IMSI identifiers. Identifiers were constructed using the digits 0 through 9 and the ^ wild-card character, which specified any single digit. Each identifier identified 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)

With Release MC-X3.0.0M2 and later releases, these strings are interpreted as either an IMSI prefix or as a MAC address. White lists now contain either IMSI or MAC identifiers. Identifiers are constructed using the digits 0 through 9 , any hexadecimal digit, and the ^ wild-card character, which specifies any single base-10 or base-16 digit. Each identifier one or one or more subscribers eligible for authentication.

IMSI/MAC White List Configuration

Use the following procedure to create an ike-access-control configuration element. This configuration element defines a white list that filters IMSI or MAC identities presented by remote endpoints during the authentication process. Only those identities matching the literal or regular expressions contained within the 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 white_01
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. For IMSI-based whitelisting, use the **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)#
   ```

   For MAC-based whitelisting, the following double-quote delimited list identifies three specific MAC addresses.

   ```
   ACMEPACKET(ike-access-control)# identifier 0123456789AB 6789912345BF
   DA2345918290
   ACMEPACKET(ike-access-control)#
   ```

   Note: Do not configure an empty 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 4 to configure additional ike-access-control configuration elements.

Assign a White List 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 **access-control-name** parameter to identify the white list assigned to the current interface.

   ```
   ACMEPACKET(ike-interface)# access-control-name white_01
   ACMEPACKET(ike-interface)#
   ```

4. Use **done**, **exit**, and **verify-config** to complete assignment of the white list to the current IKEv2 interface.

Configuring Black Lists

Service providers have requested a black list access-control capability where an MSG can be provisioned with femtocell's EAP identities — specifically the MAC address portion. Within target environments, femtocell client addresses generally take the generic format:

<MAC ID>@cellID.serviceProvider.com

With this capability enabled, the MSG can deny authentication for such femtocells trying to establish IKE/IPsec tunnel. This feature is applicable only for femtocell clients doing EAP authentication to MSG and is not applicable for clients doing password-based or certificate-based authentication.

1. From superuser mode, use the following command sequence to access ike-access-control configuration mode. While in this mode, you configure MAC address black 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 black_01
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. For MAC-address-based black lists, use the **blacklisted-dentifiers** parameter to provide one or more MAC-based match patterns.

   The following double-quote delimited list identifies three specific MAC addresses whose authentication is summarily rejected.

```
ACMEPACKET(ike-access-control)# blacklisted-dentifiers 0123456789AB
6789912345BF DA2345918290
ACMEPACKET(ike-access-control)#
```

   This identifier, which uses the wildcard symbol (^) signifying any single hexadecimal digit, specifies two ranges of contiguous MAC addresses.

```
ACMEPACKET(ike-access-control)# blacklisted-dentifiers 0123456789A^,
^123456789AB
ACMEPACKET(ike-access-control)#
```

   For IMSI-based black lists, this example uses a double-quote delimited list of prefixes separated by spaces, to match Verizon Wireless United States networks.

```
ACMEPACKET(ike-access-control)# blacklisted-dentifiers 310004 310012
ACMEPACKET(ike-access-control)#
```

   Note: Do not configure an empty black list— that is a list that lacks any blacklisted-dentifiers parameters. Assigning an empty black list to an IKEv2 interface results in authentication eligibility 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 4 to configure additional ike-access-control configuration elements.

Assign a Black List 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 black list will be assigned.
3. Use the **access-control-name** parameter to identify the black list assigned to the current interface.

```
ACMEPACKET(ike-interface)# access-control-name black_01
ACMEPACKET(ike-interface)#
```

4. Use **done**, **exit**, and **verify-config** to complete assignment of the black list to the current IKEv2 interface.

White List/Black List Interaction

White lists and black lists may or may not be assigned to the IKEv2 interfaces. The following rules are used to support implementation of both list types.

1. If neither a white list nor a black list are assigned to an IKEv2 interface, all EAP authentication requests are forwarded to a RADIUS authentication server for final determination.
2. If only a white list is assigned to an IKEv2 interface, the incoming EAP identity is be checked against that white list. If the EAP identity is contained in the white list, the authentication request is forwarded to a RADIUS authentication server for final determination. If the EAP identity is absent, authentication is denied.
3. If only a black list is assigned to an IKEv2 interface, the incoming EAP identity is checked against that black list. If the EAP identity is contained in the black list, authentication is denied. If the EAP identity is absent, the authentication request is forwarded to a RADIUS authentication server for final determination..
4. If both a white list and a black list are assigned to an IKEv2 interface, the MSG checks both the white and the black list for incoming EAP identity.

   If the EAP identity is contained in the white list, and absent from the black list, the authentication request is forwarded to a RADIUS authentication server for final determination.

   If the EAP identity is contained in the black list and absent from the white list, authentication is rejected.

   If the EAP identity is present in both the lists, the black list takes priority. Consequently, authentication is rejected. This situation will have been previously reported by the **verify-config** ACLI command.

   If the EAP identity is absent from both the lists, the whate list takes priority. Consequently, since the EAP identity is not contained in the white list the authentication is denied.

show security ike statistics

Currently the **show security ike statistics** ACLI command displays an IKEAuthIDError counter that contains the number of authentication denials due to white list filtering. With added support for black lists, this command has been enhanced with a new counter BlacklistIKEAuthIDError that contains the number of authentication denials due to black list filtering

```
ACMEPACKET# show security ike statistics
10:12:17-59
address: 172.16.20.60
IKE Interface Statistics       ---- Lifetime ----
                               Recent      Total     PerMax
ChildSaRequest                      0          0          0
ChildSaSuccess                      0          0          0
ChildSaFail                         0          0          0
ChildSaDelRequest                   0          0          0
ChildSaDelSuccess                   0          0          0
ChildSaDelFail                      0          0          0
ChildSaRekey                        0          0          0
InitialChildSa                      0          0          0
DPDRecvPortChange                   0          0          0
DPDRecvIPChange                     0          0          0
DPDRespRecv                         0          0          0
```

```
DPDRespNotRecv                   0          0          0
DPDRecv                          0          0          0
DPDRetran                        0          0          0
DPDSent                          0          0          0
IKESAPacketSent                  0          0          0
IKESAPacketRcv                   0          0          0
IKESAPacketDropped               0          0          0
AuthFailure                      0          0          0
IKEMsgError                      0          0          0
IKEAuthIDError                   0          0          0
BlacklistIKEAuthIDError          0          0          0
IKEAuthCertCheckRequest          0          0          0
IKEAuthCertCheckSuccess          0          0          0
IKEAuthCertCheckFailure          0          0          0
IKE DDos sent                    0          0          0
IKE DDos Recv                    0          0          0
IKE Msg Retrans                  0          0          0
IKE SA Init Msgs Sent            0          0          0
IKE SA Init Msgs Rcvd            0          0          0
IKE SA Est Attempts              0          0          0
IKE SA Est Success               0          0          0
IKE Tunnel Rate                  0
IKE stats Last Updated: 2013-09-13 10:12:14
```

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.

IKEv2 Interface Management

The following two 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

The SNMP MIB is formed by appending the value in the SNMP MIB Ending column to 1.3.6.1.4.1.9148.3.9.1.9.X (apSecurityIkeInterfaceInfoTable), where X specifies the interface index. For example, the SNMP MIB for the Current Child SA Pairs is 1.3.6.1.4.1.9148.3.9.1.9.X.33, where X specifies the interface index.

Note: The range for all 32-bit counters is 0 to 4294967295.

Name	Description	Type	SNMP MIB Ending
Current Child SA Pairs	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. Note that this count is available through both an ACLI show command and an SNMP GET operation.	gauge	.33
Maximum Child SA Pairs	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.	gauge	
Last Reset Timestamp	The time that this interface was last reset -- expressed as a UNIX timestamp containing the number of seconds since January 1, 1970.	UNIX timestamp	
Child SA Request	The number of requests to add a child SA pair that were received on the interface. These requests include IPsec SA rekey requests.	counter	.1
Child SA Success	The number of requests to add a child SA pair that were successfully completed on the interface. These successes include new children created by IPsec SA rekeys.	counter	.2
Child SA Failure	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.	counter	.3
Child SA Delete Requests	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.	counter	.4
Child SA Delete Success	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.	counter	.5
Child SA Delete Failure	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.	counter	.6
Child SA Rekey	The number of child IPsec rekey exchanges transacted on the interface.	counter	.7

Name	Description	Type	SNMP MIB Ending
Initial Child SA Establishment	The number of initial child SA pair establishments, in other words, the number of successful IKE_AUTH exchanges transacted on the interface.	counter	.8
DPD Received Port Change	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.	counter	.9
DPD Received IP Change	The number of DPD messages received on the interface that contained an IP address change from the previously received message.	counter	.10
DPD Response Received	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 Security Gateway. A successful R-U-THERE/ACK exchange establishes availability on the remote IKEv2 peer.	counter	.11
DPD Response Not Received	The number of R-U-THERE messages transmitted on the interface that were not acknowledged within the DPD allowed interval.	counter	.12
DPD Received	The number of all DPD protocol messages received on the interface.	counter	.13
DPD Retransmitted	The number of R-U-THERE messages that were re-transmitted because the original R-U-THERE message was not acknowledged.	counter	.14
DPD Sent	The number of R-U-THERE messages that were sent across the interface, to include retransmitals.	counter	.15
IKE SA Packets Sent	The number of IKEv2 SA packets sent across the interface.	counter	.16
IKE SA Packets Received	The number of IKEv2 SA packets received across the interface.	counter	.17
IKE SA Packets Dropped	The number of IKEv2 SA packets dropped by the interface.	counter	.18
Authentication Failures	The number of authentication failures that occurred after the purported identity of the remote IKEv2 peer was ascertained.	counter	.19
IKE Message Errors	The number of otherwise uncharacterized IKEv2 message errors.	counter	.20
Authentication ID Errors	The number of errors that occurred during the identification stage of the authentication process.	counter	.21
Certificate Status Requests	The number of certificate status requests sent across the interface to an OCSP responder.	counter	.22

Name	Description	Type	SNMP MIB Ending
Certificate Status Success	The total number of OCSP successes, that is the number of OCSP requests that generated a good status from an OCSP responder.	counter	.23
Certificate Status Fail	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.	counter	.24
DDoS Sent	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).	counter	.25
DDoS Received	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).	counter	.26
IKE Message Retransmissions	The total number of IKEv2 message re-transmissions.	counter	.27
SA Init Messages Received	The total number of IKEv2 message re-transmissions.	counter	.28
SA Init Message Sent	The total number of IKEv2 message re-transmissions.	counter	.29
SA Establishment Attempts	The total number of IKEv2 message re-transmissions.	counter	.30
SA Establishment Success	The total number of IKEv2 SA successfully established on the IKEv2 interface.	counter	.31
Tunnel Rate	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.	gauge	.32

IKEv2 Negotiation Errors

The SNMP MIB is formed by appending the value in the SNMP MIB Ending column to 1.3.6.1.4.1.9148.3.9.1.3.X (apSecurityIkeInterfaceStatsEntry), where X specifies the interface index. For example, the SNMP MIB for the CPU Overload Errors is 1.3.6.1.4.1.9148.3.9.1.3.X.3, where X specifies the interface index.

Name	Description	SNMP MIB Ending
CPU Overload Errors	The number of IKEv2 requests that were rejected because of CPU load constraints.	.3
Init Cookie Errors	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.	.4
Auth Errors	The number of failed IKE_AUTH exchanges, regardless of the specific reason for failure.	.5

Name	Description	SNMP MIB Ending
EAP Access Request Errors	The number of authentication failures that occur ed during the EAP access phase.	.6
EAP Access Challenge Errors	The number of authentication failures that occur ed during the EAP challenge phase.	.7
TS Errors	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.	.8
CP Errors	The number of IKE_AUTH and/or CREATE_CHILD_SA exchanges that failed because of faulty, unsupported, or unknown Configuration Payload contents.	.9
IKE Errors	The number of IKE_SA_INIT and/or CREATE_CHILD_SA exchanges that failed because of faulty, unsupported, or unknown Key Exchange Payload contents.	.10
Proposal Errors	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.	.11
Syntax Errors	The number of failed negotiations, of any type, resulting from otherwise uncharacterized errors.	.12
Critical Payload Errors	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.	.13

RADIUS Protocol Operations

The SNMP MIB is formed by appending the value in the SNMP MIB Ending column to 1.3.6.1.4.1.9148.3.18.1.1.1 (aapRadiusServerStatsEntry). For example, the SNMP MIB for the Server Roundtrip Time is 1.3.6.1.4.1.9148.3.18.1.1.1.3.

Name	Description	SNMP MIB Ending
Server Roundtrip Time	Contains the average round trip time for a response from this RADIUS server.	.3
Server Malformed Access Response	Contains the number of malformed access responses received on this RADIUS server.	.4

Name	Description	SNMP MIB Ending
Server Access Requests	Contains the number of access requests received on this RADIUS server.	.5
Server Disconnect Requests	Contains the number of disconnect requests received on this RADIUS server.	.6
Server Disconnect ACKS	Contains the number of acknowledged disconnects on this RADIUS server.	.7
Server Disconnect NACKS	Contains the number of unacknowledged disconnects on this RADIUS server.	.8
Server Bad Authenticators	Contains the number of authentication rejections on this RADIUS server.	.9
Server Access Retransmissions	Contains the number of access retransmitals on this RADIUS server.	.10
Server Access Accepts	Contains the number of successful authentications on this RADIUS server.	.11
Server Timeouts	Contains the number of Response timeouts on this RADIUS server.	.12
Server Access Rejects	Contains the number of unsuccessful authentications on this RADIUS server.	.13
Server Unknown PDUTypes	Contains the number or unknown/unreadable PDUs received by this RADIUS server.	.14
Server Access Challenges	Contains the number of Access Challenges on this RADIUS server.	.15

Diameter Protocol Operations

The SNMP MIB is formed by appending the value in the SNMP MIB Ending column to 1.3.6.1.4.1.9148.3.13.1.1.2.2.X (apDiamInterfaceStatsTable), where X specifies the diameter server index. For example, the SNMP MIB for the Diameter Messages Sent is 1.3.6.1.4.1.9148.3.13.1.1.2.2.X.3, where X specifies the diameter server index.

Name	Description	SNMP MIB Ending
Diameter Messages Sent	Contains the number of messages sent by this Diameter server.	.3
Diameter Messages Sent Failed	Contains the number of unacknowledged messages sent by this Diameter server.	.4
Diameter Messages Resent	Contains the number of messages re-transmitted to this Diameter server.	.5
Diameter Messages Received	Contains the number of messages received by this Diameter server.	.6
Diameter Messages Processed	Contains the number of messages processed by this Diameter server.	.7
Diameter Connection Timeouts	Contains the number of connection timeouts on the Diameter server.	.8

Name	Description	SNMP MIB Ending
Diameter BadState Drops	Contains the number of packets dropped because of faulty state on the Diameter server.	.9
Diameter BadType Drops	Contains the number of packets dropped because of faulty type on the Diameter server.	.10
Diameter BadID Drops	Contains the number of packets dropped because of faulty ID on the Diameter server.	.11
Diameter AuthFail Drops	Contains the number of failed authentications on the Diameter server.	.12
Diameter Invalid Peer Messages	Contains the number of client messages that could not be parsed on the Diameter server.	.13

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/IPv6 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
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 4000 C-Series Historical Data Recording (HDR) Resource Guide.

IKEv2 Interface HDR

CSV header fields for IKEv2 Interface HDRs are listed below.

IKEv2 Interface HDR	Type
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

IKEv2 Interface HDR	Type
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

IKEv2 Interface HDR	Type
IKE Proposal Error	Counter
IKE Syntax Error	Counter
IKE Critica; Payload Error	Counter

RADIUS HDR

CSV header fields for RADIUS HDRs are listed below.

IKEv2 Interface HDR	Type
Time Stamp	Integer
RADIUS Sever IP Address	IP Address
RADIUS Server Port	Port Address
Round Trip Time	Counter
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.

IKEv2 Interface HDR	Type
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

IKEv2 Interface HDR	Type
Bad State Drops	Counter
Bad Type Drops	Counter
Bad ID Drops	Counter
Auth Failed Drops	Counter
Invalid Peer Messages	Counter

4

Secure Management Connection

By default, communications between the SG and the Session Delivery Manager (SDM) management system are supported by a non-secure (unencrypted) TCP connection over the mgmt0 management interface. TCP message flow from the SG to the NNC is conveyed via port 3000, while message flow from the NNC to the SG is conveyed via port 3001. For providers who required an encrypted management interface (the SG-to-NNC connection), software releases prior to version M-CX3.0.0M1 provided the capability to establish an encrypted connection using IPsec. Because of hardware changes, M-CX3.0.0M1 and subsequent versions no longer support IPsec as a means to encrypt the management interface. In place of IPsec, M-CX3.0.0M1 and later versions support Transport Layer Security (TLS), thus maintaining the ability to authenticate and encrypt the management interface.

TLS negotiation is always initiated by the TLS client, a role fulfilled by the NNC management system; during the negotiation, the SG assumes the server role. TLS connections are always encrypted, and generally authenticated, although authentication is optional. Authentication is based upon an exchange and verification of PKI certificates. Depending upon the type of authentication, TLS negotiation can assume one of three different formats which are described in following sections. Because PKI operations are CPU intensive, TLS provides the ability to cache previously negotiated cryptographic materials for re-use during subsequent TLS sessions between the SG and the NNC management system.

TLS Handshake

A TLS client and server use TLS handshake protocol to negotiate cryptographic materials (encryption/hash algorithms, random numbers, and so forth) and optionally authenticate each other. The negotiated cryptographic materials are used to generate a common symmetric key, referred to as a Master-Secret, for application data encryption and authentication after TLS handshake completes successfully.

The TLS handshake can take one of three formats:

- server authentication

  With server authentication the server presents its PKI certificate to the TLS client for verification; the client does not provide a certificate to the server.
- mutual authentication

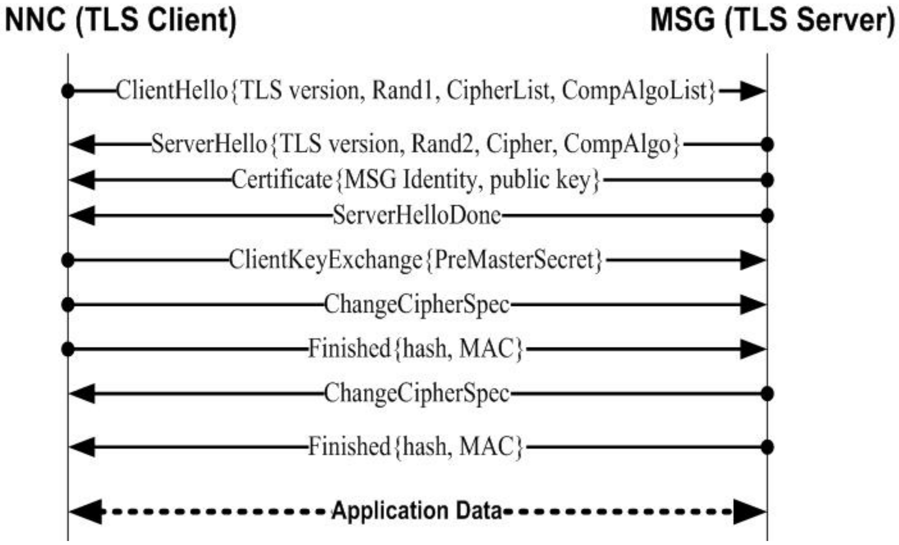
  With mutual authentication the server presents its PKI certificate to the TLS client; the client, in turn, is required to present its certificate to the server.
- TLS re-start

  With TLS re-start, the server and client use an abbreviated exchange to re-start a prior TLS connection using cached cryptographic materials.

TLS Server Authentication

In most environments, server authentication is the preferred TLS handshake format because of its simplicity and its elimination of client-side certificate requirements.

The server authentication handshake is depicted below.


NNC (TLS Client)
MSG (TLS Server)
ClientHello{TLS version, Rand1, CipherList, CompAlgoList}
ServerHello{TLS version, Rand2, Cipher, CompAlgo}
Certificate{MSG Identity, public key}
ServerHelloDone
ClientKeyExchange{PreMasterSecret}
ChangeCipherSpec
Finished{hash, MAC}
ChangeCipherSpec
Finished{hash, MAC}
Application Data

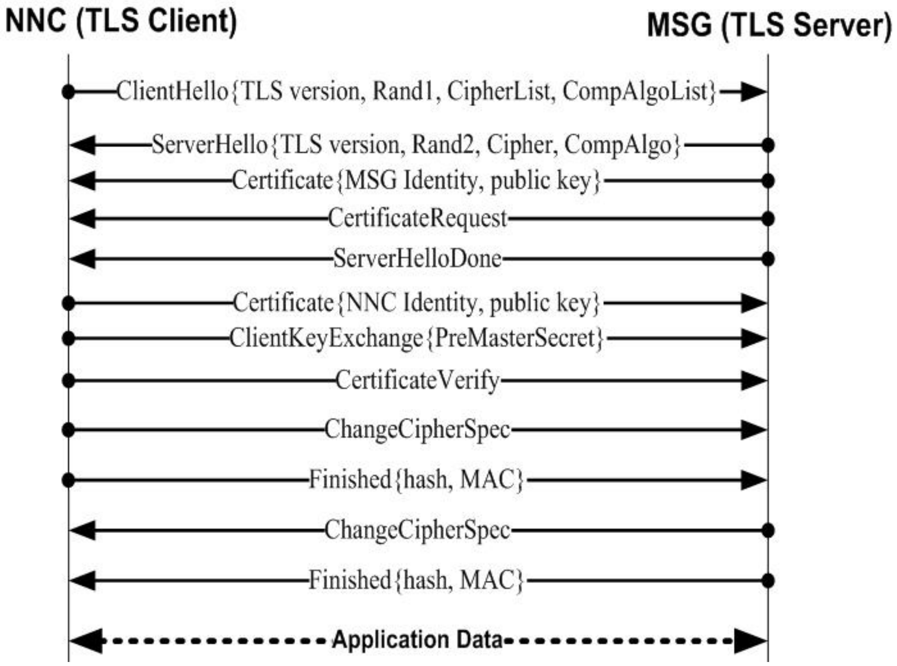
Message Type	Direction	Description
Client hello	NNC-to-SG	The client hello message initiates contact between the TLS client (the NNC management system) and the TLS server (the Security Gateway). The message contains the version of the TLS protocol that the client wants to use during the TLS session, a client-generated random value consisting of a concatenated 32-bit timestamp and 28 random bytes, a list of supported cipher suites (RSA for key exchange, AES, DES, or 3DES for encryption, and SHA for data integrity), ordered from most to least preferred, and conclude with a similarly ordered list of supported compression algorithms.
Server hello	SG-to-NNC	The server hello message responds to the client hello. Selecting from values proposed by the client, the server specifies the TLS version, the TLS cipher suites, and the compression algorithm used during the TLS session. The server hello also contains a session identifier, an arbitrary byte sequence that identifies this specific TLS session, and a server-generated random value constructed in the same fashion as the client-generated random value.
Server certificate	SG-to-NNC	The server certificate message immediately follows the server hello. It provides the client with the X.509v3 certificate used to identify the SG in its role as TLS server, and with the public key associated with that identity. The server certificate message can contain either a single certificate, or a certificate sequence which starts with the SG certificate followed by a chain of linked CA certificates — meaning that each certificate signs the one preceding it.
Server hello done	SG-to-NNC	The server hello done message indicates the completion o server hello message transmission. After transmission, the server waits for the client to provide 48-bytes of random date, referred to as the premaster secret. Both TLS client and server will use the same premaster secret together with the previously-generated client and server random values to derive cryptographic material used for encryption and authentication.
Client key exchange	NNC-to-SG	The client key exchange message contains a premaster secret encrypted with the PKI public key passed to the client in the server certificate message.

Message Type	Direction	Description
Change cipher spec	NNC-to-SG	The change cipher spec message, consisting of a single byte containing the value of 1, signals the client’s intention to utilize newly negotiated cryptographic material in its next message.
Finished	NNC-to-SG	A finished message is always sent immediately after a change cipher spec message. This finished message is the first client-initiated message that utilizes newly negotiated cryptographic material. Consequently, the TLS server (the SG) is required to verify the message’s content to ensure proper operation of encrypt/decrypt and message authentication functions. The finished message consists of a verify_data field that contains a SHA-1 Message Authentication Code (MAC) of the data included in prior handshake messages (excluding change cipher spec messages), a finished_label field that contains the string client finished, and handshake_messages field that contains the data structures from prior handshake messages.
Change cipher spec	SG-to-NNC	The change cipher spec message, consisting of a single byte containing the value of 1, signals the server’s intention to utilize newly negotiated cryptographic material in its next message.
Finished	SG-to-NNC	A finished message is always sent immediately after a change cipher spec message. This finished message is the first server-initiated message that utilizes newly negotiated cryptographic material. Consequently, the TLS client (the NNC management system) is required to verify the message’s content to ensure proper operation of encrypt/decrypt and message authentication functions. The finished message consists of a verify_data field that contains a SHA-1 Message Authentication Code (MAC) of the data included in prior handshake messages (excluding change cipher spec messages), a finished_label field that contains the string server finished, and handshake_messages field that contains the data structures from prior handshake messages.
From this point forward, all data exchanged between the TLS client (the NNC management system) and the TLS server (the Security Gateway) will be encrypted and authenticated.		

TLS Mutual Authentication

Mutual authentication is based on the exchange of certificates between both the TLS client and server. As in server authentication, the client verifies the certificate presented by the server to attest to its purported identity. In a similar manner, after requesting a certificate from the client, the server verifies the certificate presented by the client to attest to its purported identity. Because of the exchange of multiple certificates and private keys and the verification of two certificates, mutual authentication is a more expensive negotiation, when compared to server authentication, in terms of processing time and use of processing resources.

The mutual authentication handshake is depicted below.


NNC (TLS Client)
MSG (TLS Server)
ClientHello{TLS version, Rand1, CipherList, CompAlgoList}
ServerHello{TLS version, Rand2, Cipher, CompAlgo}
Certificate{MSG Identity, public key}
CertificateRequest
ServerHelloDone
Certificate{NNC Identity, public key}
ClientKeyExchange{PreMasterSecret}
CertificateVerify
ChangeCipherSpec
Finished{hash, MAC}
ChangeCipherSpec
Finished{hash, MAC}
Application Data

Message Type	Direction	Description
Client hello	NNC-to-SG	The client hello message initiates contact between the TLS client (the NNC management system) and the TLS server (the Security Gateway). The message contains the version of the TLS protocol that the client wants to use during the TLS session, a client-generated random value consisting of a concatenated 32-bit timestamp and 28 random bytes, a list of supported cipher suites (RSA for key exchange, AES, DES, or 3DES for encryption, and SHA for data integrity), ordered from most to least preferred, and conclude with a similarly ordered list of supported compression algorithms.
Server hello	SG-to-NNC	The server hello message responds to the client hello. Selecting from values proposed by the client, the server specifies the TLS version, the TLS cipher suites, and the compression algorithm used during the TLS session. The server hello also contains a session identifier, an arbitrary byte sequence that identifies this specific TLS session, and a server-generated random value constructed in the same fashion as the client-generated random value.
Server certificate	SG-to-NNC	The server certificate message immediately follows the server hello. It provides the client with the X.509v3 certificate used to identify the SG in its role as TLS server, and with the public key associated with that identity. The server certificate message can contain either a single certificate, or a certificate sequence which starts with the SG certificate followed by a chain of linked CA certificates — meaning that each certificate signs the one preceding it.
Certificate request	SG-to-NNC	The certificate request message is used only in cases of mutual authentication. If such authentication is mandated, this message contains a list of preferred certificate types ranked in order of preference, and a list of the distinguished names of preferred CAs.
Server hello done	SG-to-NNC	The server hello done message indicates the completion of server hello message transmission. After transmission, the server waits for the client to provide (1) the requested certificate, and (2) 48-bytes of random date, referred to as the premaster secret. Both TLS client and server will use the same premaster secret together with the previously-generated client and server random values to derive cryptographic material used for encryption and authentication.

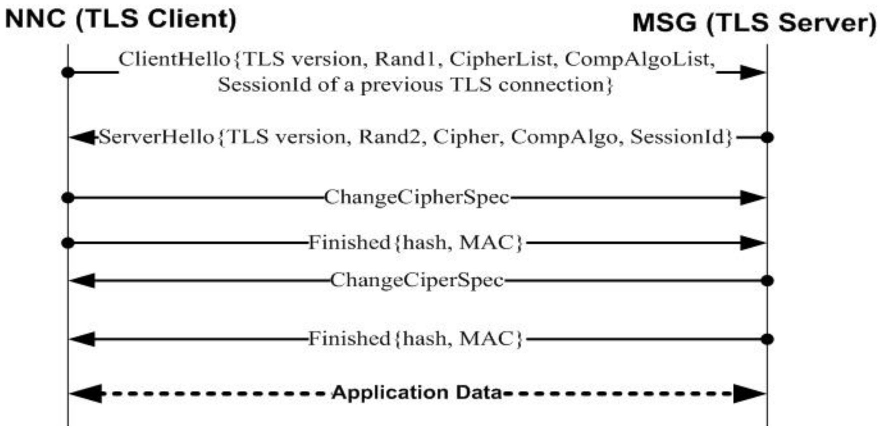
Message Type	Direction	Description
Client certificate	NNC-to-SG	The client certificate message must be the first message sent by the client after reception of a certificate request. The client certificate message provides a certificate identifying the NNC in its role as a TLS client, and with the public key associated with that identity.
Client key exchange	NNC-to-SG	The client key exchange message contains a premaster secret encrypted with the PKI public key passed to the client in the server certificate message.
Certificate verify	NNC-to-SG	The certificate verify message authenticates the TLS client. The client demonstrates possession of the private key associated with the previously provided public key by signing a value derived from the master secret and transmitted handshake messages, up to, but not including, this message. The handshake messages include the server certificate, which binds the signature to the server, and the server random value, which binds the signature to the current handshake transaction.
Change cipher spec	NNC-to-SG	The change cipher spec message, consisting of a single byte containing the value of 1, signals the client's intention to utilize newly negotiated cryptographic material in its next message.
Finished	NNC-to-SG	A finished message is always sent immediately after a change cipher spec message. This finished message is the first client-initiated message that utilizes newly negotiated cryptographic material. Consequently, the TLS server (the SG) is required to verify the message's content to ensure proper operation of encrypt/decrypt and message authentication functions. The finished message consists of a verify_data field that contains a SHA-1 Message Authentication Code (MAC) of the data included in prior handshake messages (excluding change cipher spec messages), a finished_label field that contains the string client finished, and handshake_messages field that contains the data structures from prior handshake messages.
Change cipher spec	SG-to-NNC	The change cipher spec message, consisting of a single byte containing the value of 1, signals the server's intention to utilize newly negotiated cryptographic material in its next message.
Finished	SG-to-NNC	A finished message is always sent immediately after a change cipher spec message. This finished message is the first server-initiated message that utilizes newly negotiated cryptographic material. Consequently, the TLS client (the NNC management system) is required to verify the message's content to ensure proper operation of encrypt/decrypt and message authentication functions. The finished message consists of a verify_data field that contains a SHA-1 Message Authentication Code (MAC) of the data included in prior handshake messages (excluding change cipher spec messages), a finished_label field that contains the string server finished, and handshake_messages field that contains the data structures from prior handshake messages.
From this point forward, all data exchanged between the TLS client (the NNC management system) and the TLS server (the Security Gateway) will be encrypted and authenticated.		

TLS Re-Start

A TLS re-start handshake provides an abbreviated format that allows the use of cached cryptographic material generated during previous TLS negotiations. Cached data includes, but is not limited to, all exchanged and derived cryptographic material, to include random values, the master-secret, from which working symmetric encryption, decryption, authentication keys are derived.

The re-start handshake improves latency by avoiding the most expensive (in terms of time and processing) part of the TLS handshake — namely, certificate exchange and verification.

The TLS restart handshake is depicted below.


NNC (TLS Client)
MSG (TLS Server)
ClientHello{TLS version, Rand1, CipherList, CompAlgoList, SessionId of a previous TLS connection}
ServerHello{TLS version, Rand2, Cipher, CompAlgo, SessionId}
ChangeCipherSpec
Finished{hash, MAC}
ChangeCiperSpec
Finished{hash, MAC}
Application Data

Message Type	Direction	Description
Client hello	NNC-to-SG	The client hello message initiates contact between the TLS client (the NNC management system) and the TLS server (the Security Gateway). The message contains the version of the TLS protocol that the client wants to use during the TLS session, a client-generated random value consisting of a concatenated 32-bit timestamp and 28 random bytes, a list of supported cipher suites (RSA for key exchange, AES, DES, or 3DES for encryption, and SHA for data integrity), ordered from most to least preferred, a similarly ordered list of supported compression algorithms, and concludes with a session_id that identifies a previously negotiated TLS session whose cached security parameters are offered for re-use in this current session.
Server hello	SG-to-NNC	The server checks its TLS cache for the proposed session. If a match is found, it responds with a server hello terminated by the session_id proposed by the client indicating that the cached cryptographic material will be used for the current session. If the server chooses not to re-use the cached data, it responds with a new session_id forcing the client to proceed with a complete handshake transaction.
Change cipher spec	NNC-to-SG	The change cipher spec message, consisting of a single byte containing the value of 1, signals the client's intention to utilize cached cryptographic material in its next message.
Finished	NNC-to-SG	A finished message is always sent immediately after a change cipher spec message. This finished message is the first client-initiated message that utilizes newly negotiated cryptographic material. Consequently, the TLS server (the SG) is required to verify the message's content to ensure proper operation of encrypt/decrypt and message authentication functions. The finished message consists of a verify_data field that contains a SHA-1 Message Authentication Code (MAC) of the data included in prior handshake messages (excluding change cipher spec messages), a finished_label field that contains the string client finished, and handshake_messages field that contains the data structures from prior handshake messages.
Change cipher spec	SG-to-NNC	The change cipher spec message, consisting of a single byte containing the value of 1, signals the server's intention to utilize cached cryptographic material in its next message.

Message Type	Direction	Description
Finished	SG-to-NNC	A finished message is always sent immediately after a change cipher spec message. This finished message is the first server-initiated message that utilizes newly negotiated cryptographic material. Consequently, the TLS client (the NNC management system) is required to verify the message’s content to ensure proper operation of encrypt/decrypt and message authentication functions. The finished message consists of a verify_data field that contains a SHA-1 Message Authentication Code (MAC) of the data included in prior handshake messages (excluding change cipher spec messages), a finished_label field that contains the string server finished, and handshake_messages field that contains the data structures from prior handshake messages.
From this point forward, all data exchanged between the TLS client (the NNC management system) and the TLS server (the Security Gateway) will be encrypted and authenticated.		

ACLI Configuration

Securing the management connection requires the following configuration:

- configuration of a tls-profile object (required)
- enabling TLS on the management interface (required)
- enabling the TLS cache (optional)

Configuring a TLS Profile

Use the following procedure to configure a tls-profile.

1. From Superuser mode, use the following ACLI sequence to navigate to tls-profile configuration mode.

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

2. The required **name** parameter facilitates the creation and assignment of multiple tls-profile objects. Use the parameter to assign a unique string identifier.

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

3. The required **end-entity-certificate** parameter identifies the certificate presented to the NNC management system by the SG during the TLS handshake. This certificate identifies the SG and provides its public key. TLS standards mandate that the TLS server must present a certificate to the remote client.

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

4. The optional **mutual-authenticate** parameter enables or disables mutual authentication. If the default value (**disabled**) is retained, the SG does not request a certificate from the NNC client during the TLS handshake. This approach is recommends in that it provides the simplest and fastest negotiation.

    With mutual authentication enabled, the SG requests a certificate from the NNC client during the TLS handshake. This approach increases the processing overhead of the TLS negotiation in that it adds additional steps and time to the negotiation.

    TLS standards require that the TLS client present a certificate if one is requested by the server. Failure to present a certificate requires that the connection must be torn down.

    Retain the default value (**disabled**) in most environments. If required by local conditions or policy, use **enabled** to enable mutual authentication.

5. Use the optional **trusted-ca-certificates** parameter only if mutual-authentication is enabled. If so, use this parameter to provide a list of one or more Certificate Authority (CA) certificates to use when verifying the certificate received from the NNC management system.nter the names of the trusted CA certificate records.

   This parameter can be safely ignored in mutual authentication is disabled — the default state.

   ```
   ACMEPACKET(tls-profile)# trusted-ca-certificates anyThingFromThawte
   notMuchFromGoDaddy megaCorpSelfSigned
   ACMEPACKET(tls-profile)#
   ```

6. Use the optional **tls-version** parameter to compatibility to ensure that the SG selects the most secure TLS variant supported by both the SG and the NNC management system.

   ```
   ACMEPACKET(tls-profile)# tls-version compatibility
   ACMEPACKET(tls-profile)#
   ```

7. Retain the default value (ALL) for optional **cipher-list** parameter. Retention of the default parameter provides the SG with the maximum flexibility in selected the strongest mutually supported cipher suite.
8. Other parameters can be safely ignored
9. Use **done**, **exit**, and **verify-config** to complete configuration of this tls-profile object.

Enabling TLS on the Management Interface

You enable a secure TLS-protected management interface by associating an existing TLS profile object with the management interface.

1. From Superuser mode, use the following ACLI sequence to navigate to system-config configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# system
   ACMEPACKET(system)# system-config
   ACMEPACKET(sys-config)#
   ```

2. Use the optional **acp-tls-profile** parameter to enable (require) TLS protection of the management interface. By default this parameter is empty, specifying a non-secure management interface You enable a secure management interface by assigning an existing TLS profile this parameter.

   ```
   ACMEPACKET(tls-profile)# acp-tls-profile mgmtInterface
   ACMEPACKET(tls-profile)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete tls-global enabling a secure management interface.

Configuring TLS Session Caching

Use the following procedure to enable TLS session caching.

1. From Superuser mode, use the following ACLI sequence to navigate to tls-global configuration mode.

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

2. The optional **session-caching** parameter enables or disables the caching of cryptographic and related material generated during a TLS session. By default, TLS caching is **disabled**. Set this parameter to **enabled** to enable TLS caching. With caching enabled, the TLS client and server can engage in a TLS-restart handshake which allows the NNC management system and SG to resume a TLS connection in the most expeditious manner.

   ```
   ACMEPACKET(tls-global)# session-caching enabled
   ACMEPACKET(tls-global)#
   ```

3. The optional **se ss ion-cache-timeout** parameter specifies the time (in hours) that individual TLS session (each indexed by a unique session identifiers) are retained in a TLS cache. Allowable values are integers within the range 0 through 24, with a default value of 12. The special value 0 indicates that cache entries are never cleared, and the cache can only be cleared by the **clear-cache tls** command. RFC 2246, The TLS Protocol Version 1.0, recommends that the cache-timeout parameter be set to the maximum supported value — 24 (hours).

4. Use **done**, **exit**, and **verify-config** to complete tls-global configuration.

MIB Indication of TLS Support

An entry (apTlsAcpSupportEnabled) in the apSysMgmtMIBGeneralObjects table Specifies the SG support for TLS services on the management interface. The NNC management system should read this object's value prior to initiating a request for a TLS connection.

apTlsACPSupportEnabled	**OBJECT-TYPE**
SYNTAX	TruthValue
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	A value of true indicates that TLS is supported; a value of false indicates that TLS is not supported.
::=	{apSysMGMTMIBGeneralObjects 40}
OID	1.3.6.1.4.1.9148.3.2.1.1.40
First Supported	M-CX3.0.0M1

5

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, RADIUS Extensions
- RFC 3162, RADIUS and IPv6

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 four 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    |Id      |Length         |
+--------+--------+---------------+
|Authenticator                    |
|                                 |
|                                 |
|                                 |
+---------------------------------+
|Attributes                       |
+-----------------
```

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
NAS-IPv6-Address	Attribute Type 95
Framed-IPv6-Prefix	Attribute Type 97
Framed-IPv6-Pool	Attribute Type 100

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    |Id      |Length         |
+--------+--------+---------------+
|Authenticator                    |
|                                 |
|                                 |
|                                 |
+---------------------------------+
|Attributes                       |
+-----------------
```

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  |Length|Value              |
+------+-+----+-------------------+
|Value (continued)                |
+--------+------------------------+
|                                   |
+--------------
```

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 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.

The Security Gateway assigns the value 1312 to the NAS-Port attribute in all RADIUS Accounting Start, Interim-Update, and Stop packets.

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 IP 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 IP 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 232 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 232 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 IP 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 IP address or the FQDN of the tunnel responder (the Security Gateway).

NAS-IPv6-Address

RADIUS Attribute Type 95

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 2.1 of RFC 3162, RADIUS and IPv6.

Framed-IPv6-Prefix

RADIUS Attribute Type 97

Found in RADIUS Accounting Start, Interim-Update, and Stop packets

Defined in Section 2.3 of RFC 3162, RADIUS and IPv6.

Acme Packet VSAs

Table 1: 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 (232-1) 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 (232-1) 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 (232-1) 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 (232-1) 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.

```
+----------------------------------+
|                                  |
| DIAMETER Header                  |
| Command Code = 271               |
| Request Bit  = 1                 |
|                                 |
+---------------------------------+
| session-Id                      |
+---------------------------------+
| Origin-Host                     |
+---------------------------------+
| Origin-Realm                    |
+---------------------------------+
| Destination-Realm               |
+---------------------------------+
| Destination-Host                |
+---------------------------------+
| Accounting-Record-Type = 2      |
+---------------------------------+
| Accounting-Record-Number        |
+---------------------------------+
| Acct-Application-Id             |
+---------------------------------+
| User-Name                       |
+---------------------------------+
| Event-Timestamp                 |
+---------------------------------+
| Tunnel-Type                     |
+---------------------------------+
| Tunnel-Client-Endpoint          |
+---------------------------------+
| Tunnel-Server-Endpoint          |
+---------------------------------+
| Framed-IP-Address               |
+---------------------------------+
| Calling-Station-Id              |
+---------------------------------+
| Tunnel-client-Auth-Id           |
+---------------------------------+
| Tunnel-Server-Auth-Id           |
+---------------------------------+
```

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.

```
+---------------------------------+
|                                 |
| DIAMETER Header                 |
| Command Code = 271              |
| Request Bit   = 1               |
|                                 |
+---------------------------------+
| session-Id                      |
+---------------------------------+
| Origin-Host                     |
+---------------------------------+
| Origin-Realm                    |
+---------------------------------+
| Destination-Realm               |
+---------------------------------+
| Destination-Host                |
+---------------------------------+
| Accounting-Record-Type = 4      |
+---------------------------------+
| Accounting-Record-Number        |
+---------------------------------+
| Acct-Application-Id             |
+---------------------------------+
| User-Name                       |
+---------------------------------+
| Event-Timestamp                 |
+---------------------------------+
| Tunnel-Type                     |
+---------------------------------+
| Termination-Cause               |
+---------------------------------+
| Acct-Session-Time               |
+---------------------------------+
| Tunnel-Type                     |
+---------------------------------+
| Tunnel-Client-Endpoint          |
+---------------------------------+
| Tunnel-Server-Endpoint          |
+---------------------------------+
| Framed-IP-Address               |
+---------------------------------+
| Calling-Station-Id              |
+---------------------------------+
| Tunnel-client-Auth-Id           |
+---------------------------------+
| Tunnel-Server-Auth-Id           |
+---------------------------------+
| Accounting-Input-Octets         |
+---------------------------------+
+---------------------------------+
| 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 IP 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.

Framed-IPv6-Address

DIAMETER AVP Code 97

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 6.11.6 of RFC 4005, DIAMETER Network Access Server Application.

Generated by the Security Gateway, contains the inner IPv6 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.

NAS-IPv6-Address

DIAMETER AVP Code 95

Found in DIAMETER Accounting Request Start and Stop Messages

Defined in Section 9.3.3 of RFC 4005, DIAMETER Network Access Server Application.

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 IP 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 IP 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. The **src-port** parameter can be safely ignored..

   ```
   The Security Gateway assigns a value of 1312 to the NAS-Port attribute
   contained in all Accounting Request Start, Interim, and Stop messages.
   ```

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.

Assigning an IPsec Accounting Parameter List

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.

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.

```
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.

Accounting Group Configuration

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.

Realm Configuration

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.

6

Threat 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.

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.

Constructing 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.

Assigning an Access Control Template to an 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
```

High Availability

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.

Stateless Firewall

Version M-CX3.0.0M1 and later releases provide enhanced security-policy-based filters that can be applied to data packets coming through IPSec tunnels on the protected interfaces, and to non-IPSec packets on the trusted interfaces. These filters evaluate only the IP header layer, and treat each individual packet as a discrete event. As such, the functionality they provide can be compared to that provided by a stateless firewall.

Available filters are discussed in the following sections.

ICMP Filtering

Internet Control Message Protocol (ICMP) messages can be filtered based on message Type and associated message Codes.

ICMP Policy Configuration

Use the following procedure to define security-policy-based filtering of ICMP packets. This sample policy discards ICMP message type 0, Echo Reply, code 0, Net Unreachable.

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 policy.

```
ACMEPACKET(security-policy)# name deny_icmp-type0_code0
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 M00:433
ACMEPACKET(security-policy)#
```

4. Use the optional **local-ip-addr-match** parameter 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 192.168.89.10
   ACMEPACKET(security-policy)#
   ```

5. Use the optional **local-ip-mask** parameter to specify the local address mask.

   ```
   ACMEPACKET(security-policy)# local-ip-mask 255.255.0.0
   ACMEPACKET(security-policy)#
   ```

6. Use the optional **remote-ip-addr-match** parameter to specify the local IP address of the network interface.

   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 15.0.0.0
   ACMEPACKET(security-policy)#
   ```

7. Use the optional **remote-ip-mask** parameter to specify the remote address mask.

   ```
   ACMEPACKET(security-policy)# local-ip-mask 255.0.0.0
   ACMEPACKET(security-policy)#
   ```

8. Use the optional **local-port-match** parameter in conjunction with the **local-port-match-max** parameter to specify a contiguous range of local ports to which this security policy applies.

   To specify a single local port:

   ```
   ACMEPACKET(security-policy)# local-port-match 64000
   ACMEPACKET(security-policy)# local-port-match-max 64000
   ACMEPACKET(security-policy)#
   ```

   To specify a local port range:

   ```
   ACMEPACKET(security-policy)# local-port-match 64000
   ACMEPACKET(security-policy)# local-port-match-max 64500
   ACMEPACKET(security-policy)#
   ```

   To specify all local ports:

   ```
   ACMEPACKET(security-policy)# local-port-match 0
   ACMEPACKET(security-policy)# local-port-match-max 65535
   ACMEPACKET(security-policy)#
   ```

9. Use the optional **remote-port-match** parameter in conjunction with the **remote-port-match-max** parameter to specify a contiguous range of remote ports to which this security policy applies.

   To specify a single remote port:

   ```
   ACMEPACKET(security-policy)# remote-port-match 32000
   ACMEPACKET(security-policy)# remote-port-match-max 32000
   ACMEPACKET(security-policy)#
   ```

   To specify a local port range:

   ```
   ACMEPACKET(security-policy)# remote-port-match 64000
   ACMEPACKET(security-policy)# remote-port-match-max 64500
   ACMEPACKET(security-policy)#
   ```

   To specify all local ports:

   ```
   ACMEPACKET(security-policy)# remote-port-match 0
   ACMEPACKET(security-policy)# remote-port-match-max 65535
   ACMEPACKET(security-policy)#
   ```

10. Use the optional **priority** parameter to assign a priority to this security policy.

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

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

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

11. Use the optional **trans-protocol-match** parameter to identify the filtered protocol, in this example, ICMP.

```
ACMEPACKET(security-policy)# trans-protocol-match icmp
ACMEPACKET(security-policy)#
```

12. Use the optional **trans-sub-protocol-match** parameter to identify a specific ICMP message type.

    The ICMP 8-bit Type field specifies the message type. Contents of the Type field are administered by the Internet Assigned Numbers Authority (IANA). Current Type numbers can be viewed at http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xml#icmp-parameters-types.

    The following command sequence designates the ICMP Echo Reply message.

```
ACMEPACKET(security-policy)# trans-sub-protocol-match 0
ACMEPACKET(security-policy)#
```

13. Use the optional **trans-sub-protocol-code-match** parameter to identify a specific ICMP code.

    Many ICMP message type have associated numeric codes which provide additional information pertinent to that message types. ICMP code values are administered by the Internet Assigned Numbers Authority (IANA). Up to date codes can be viewed at http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xml#icmp-parameters-codes.

    In the absence of a specifically identified code, all codes are filtered.

    The following command sequence designates the Net Unreachable code.

```
ACMEPACKET(security-policy)# trans-sub-protocol-code-match 0
ACMEPACKET(security-policy)#
```

14. Use the optional **action** parameter to specify how incoming ICMP messages that match filtering criteria are processed.

    Use **discard** to drop all ICMP messages that match filtering criteria.

    Use **allow** to pass-thru all ICMP messages that match filtering criteria.

```
ACMEPACKET(security-policy)# action discard
ACMEPACKET(security-policy)#
```

15. 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)#
```

16. Ignore the **ike-sainfo-name** parameter.
17. Use **done**, **exit**, and **verify-config** to complete configuration of the security policy.
18. Repeat Steps 1 through 17 to configure additional security policies.

SCTP Filters

Internet Control Message Protocol (ICMP) messages can be filtered based on Payload Protocol Identifiers.

SCTP Policy Configuration

Use the following procedure to define security-policy-based filtering of SCTP packets. This sample policy allows SCTP, Payload Protocol Identifier 34, Diameter in SCTP DATA chunk.

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 policy.

   ```
   ACMEPACKET(security-policy)# name allow_sctp-ppid_46
   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 M00:433
   ACMEPACKET(security-policy)#
   ```

4. Use the optional **local-ip-addr-match parameter** 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 192.168.89.10
   ACMEPACKET(security-policy)#
   ```

5. Use the optional **local-ip-mask** parameter to specify the local address mask.

   ```
   ACMEPACKET(security-policy)# local-ip-mask 255.255.0.0
   ACMEPACKET(security-policy)#
   ```

6. Use the optional **remote-ip-addr-match parameter** to specify the remote IP address of the network interface.

   Provide the remote IP address or retain the default value, 0.0.0.0, which matches all local IP addresses.

   ```
   ACMEPACKET(security-policy)# remote-ip-addr-match 192.168.89.10
   ACMEPACKET(security-policy)#
   ```

7. Use the optional remote**-ip-mask** parameter to specify the remote address mask.

   ```
   ACMEPACKET(security-policy)# remote-ip-mask 255.255.0.0
   ACMEPACKET(security-policy)#
   ```

8. Use the optional **local-port-match** parameter in conjunction with the **local-port-match-max** parameter to specify a contiguous range of local ports to which this security policy applies.

   To specify a single local port:

   ```
   ACMEPACKET(security-policy)# local-port-match 64000
   ACMEPACKET(security-policy)# local-port-match-max 64000
   ACMEPACKET(security-policy)#
   ```

   To specify a local port range:

   ```
   ACMEPACKET(security-policy)# local-port-match 64000
   ACMEPACKET(security-policy)# local-port-match-max 64500
   ACMEPACKET(security-policy)#
   ```

   To specify all local ports:

   ```
   ACMEPACKET(security-policy)# local-port-match 0
   ACMEPACKET(security-policy)# local-port-match-max 65535
   ACMEPACKET(security-policy)#
   ```

9. Use the optional **remote-port-match** parameter in conjunction with the **remote-port-match-max** parameter to specify a contiguous range of remote ports to which this security policy applies.

   To specify a single remote port:

   ```
   ACMEPACKET(security-policy)# remote-port-match 32000
   ACMEPACKET(security-policy)# remote-port-match-max 32000
   ACMEPACKET(security-policy)#
   ```

   To specify a local port range:

   ```
   ACMEPACKET(security-policy)# remote-port-match 64000
   ACMEPACKET(security-policy)# remote-port-match-max 64500
   ACMEPACKET(security-policy)#
   ```

To specify all local ports:

```
ACMEPACKET(security-policy)# remote-port-match 0
ACMEPACKET(security-policy)# remote-port-match-max 65535
ACMEPACKET(security-policy)#
```

10. Use the optional **priority** parameter to assign a priority to this security policy.

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

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

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

11. Use the optional **trans-protocol-match** parameter to identify the filtered protocol, in this example, SCTP.

    ```
    ACMEPACKET(security-policy)# trans-protocol-match sctp
    ACMEPACKET(security-policy)#
    ```

12. Use the optional **trans-sub-protocol-match** parameter to identify a specific SCTP Protocol Payload Identifier.

    SCTP DATA chunks contain a 32-bit Payload Protocol Identifier field specifying the protocol that originated the data contained in the SCTP chunk. SCTP Payload Protocol Identifiers are administered by the IANA. Current identifiers can be found at http://www.iana.org/assignments/sctp-parameters/sctp-parameters.xml#sctp-parameters-25.

    The following command sequence designates DIAMETER data.

    ```
    ACMEPACKET(security-policy)# trans-sub-protocol-match 46
    ACMEPACKET(security-policy)#
    ```

13. Ignore the optional **trans-sub-protocol-code-match** parameter, which is not currently used for SCTP filter configuration.
14. Use the optional **action** parameter to specify how incoming SCTP messages that match filtering criteria are processed.

    Use **discard** to drop all SCTP messages that match filtering criteria.

    Use **allow** to pass-thru all SCTP messages that match filtering criteria.

    ```
    ACMEPACKET(security-policy)# action allow
    ACMEPACKET(security-policy)#
    ```

15. 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)#
    ```

16. Ignore the **ike-sainfo-name** parameter.
17. Use **done**, **exit**, and **verify-config** to complete configuration of the security policy.
18. Repeat Steps 1 through 17 to configure additional security policies.

Source Routing Packets

A new **options** command is available at the global level to unconditionally discard all source routed packets. Source routed packets are identified by the presence of either a Loose Source Route/Record (LSRR) or a Strict Source Route/Record (SSRR) option within the IP header Options header.

Both options has the potential to mask malicious intent. An attacker can use the specified routes to hide the true source of a packet, or to gain access to a protected network. Consequently, such packets are often dropped upon network entry.

Use the following procedure to unconditionally discard all source routed packets.

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

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ipsec
ACMEPACKET(ipsec)# ipsec-global-config
ACMEPACKET(ipsec-global-config)#
```

2. Use the **options** command in conjunction with the **source-routing-drop** argument to unconditionally discard all source routing packets — identified by the presence of either an SSRR or LSRR option in the IP header Options header.

```
ACMEPACKET(ipsec-global-config)# options +source-routing-drop
ACMEPACKET(ipsec-global-config)#
```

3. Use the **done** and **exit** to complete configuration.

Fragmented Packets

A new **options** command is available at the global level to unconditionally discard all inbound fragmented Encapsulating Security Protocol (ESP) packets. Refer to Figure 3, "ESP Transport Mode," on page 115, and to Figure 5, ESP Tunnel Mode, on page 117 for ESP details.

Upon reception, the SG re-assembles such packets and then decrypts the re-assembled packet. After decryption, if the decrypted packet is still a fragment, the new option mandates that the packet fragment be discarded in the light of the SG's inability to do a proper policy check on an incomplete message.

Use the following procedure to unconditionally discard all fragmented ESP packets.

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

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ipsec
ACMEPACKET(ipsec)# ipsec-global-config
ACMEPACKET(ipsec-global-config)#
```

2. Use the **options** command in conjunction with the **fragmented-packet-drop** argument to unconditionally discard all inbound fragmented ESP packets.

```
ACMEPACKET(ipsec-global-config)# options +fragmented-packet-drop
ACMEPACKET(ipsec-global-config)#
```

3. Use the **done** and **exit** to complete configuration.

ACLI show Commands

Two new ACLI commands display filtering statistics.

- **show security security-policy statistics all**, which displays statistics for all filtering policies
- **show security security-policy [policyName]**, which displays statistics for a specific filtering policy

A

Appendix A: MIB SNMP Quick Reference

In addition to baseline Session Border Controller MIBs, this chapter includes Oracle Communications Mobile Security Gateway specific MIBs, organized by release where introduced.

Release M-CX 3.0.0M2

A new BlacklistIKEAuthIDError statistic is reported via SNMP.; no trap is generated.

MIB file: ap-security.mib

A new MIB object is defined for BlacklistIKEAuthIDError statistic as shown below.

```
apSecurityIkeInterfaceBlacklistIKEAuthIDError OBJECT-TYPE
        SYNTAX          Unsigned32
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
                "Number of Authentication ID Errors on the IKE-interface
    due to blacklist"
        ::= { apSecurityIkeInterfaceInfoEntry 33 }
```

This new counter is included as part of ApSecurityIkeInterfaceInfoEntry object ApSecurityIkeInterfaceInfoEntry ::=

```
SEQUENCE {
    apSecurityIkeInterfaceChildSaRequest    Unsigned32,
    apSecurityIkeInterfaceChildSaSuccess    Unsigned32,
    apSecurityIkeInterfaceChildSaFail    Unsigned32,
    apSecurityIkeInterfaceChildSaDelRequest    Unsigned32,
    apSecurityIkeInterfaceChildSaDelSuccess    Unsigned32,
    apSecurityIkeInterfaceChildSaDelFail    Unsigned32,
    apSecurityIkeInterfaceChildSaRekey    Unsigned32,
    apSecurityIkeInterfaceInitialChildSa    Unsigned32,
    apSecurityIkeInterfaceDPDRecvPortChange    Unsigned32,
    apSecurityIkeInterfaceDPDRecvIPChange    Unsigned32,
    apSecurityIkeInterfaceDPDRespRecv    Unsigned32,
    apSecurityIkeInterfaceDPDRespNotRecv    Unsigned32,
    apSecurityIkeInterfaceDPDRecv    Unsigned32,
    apSecurityIkeInterfaceDPDRetran    Unsigned32,
    apSecurityIkeInterfaceDPDSent    Unsigned32,
    apSecurityIkeInterfaceIKESAPacketSent    Unsigned32,
    apSecurityIkeInterfaceIKESAPacketRcv    Unsigned32,
    apSecurityIkeInterfaceIKESAPacketDropped    Unsigned32,
    apSecurityIkeInterfaceAuthFailure    Unsigned32,
    apSecurityIkeInterfaceMsgError    Unsigned32,
    apSecurityIkeInterfaceAuthIDError    Unsigned32,
    apSecurityIkeInterfaceBlacklistIKEAuthIDError    Unsigned32,
    apSecurityIkeInterfaceAuthCertCheckRequest    Unsigned32,
    apSecurityIkeInterfaceAuthCertCheckSuccess    Unsigned32,
    apSecurityIkeInterfaceAuthCertCheckFailure    Unsigned32,
    apSecurityIkeInterfaceDDosSent    Unsigned32,
    apSecurityIkeInterfaceDDosRecv    Unsigned32,
    apSecurityIkeInterfaceMessageRetrans    Unsigned32,
    apSecurityIkeInterfaceSAInitMsgRecv    Unsigned32,
    apSecurityIkeInterfaceSAInitMsgSent    Unsigned32,
    apSecurityIkeInterfaceSAEstablishmentAttempts    Unsigned32,
    apSecurityIkeInterfaceSAEstablishmentSuccess    Unsigned32,
    apSecurityIkeInterfaceTunnelRate    Unsigned32,
    apSecurityIkeInterfaceCurrentChildSaPair    Unsigned32
}
```

Release M-CX 3.0.0M1

M-CX 3.0.0M1 provides support for a single new MIB variable.

An entry (apTlsAcpSupportEnabled) in the apSysMgmtMIBGeneralObjects table Specifies the SG support for TLS services on the management interface. The NNC management system should read this object’s value prior to initiating a request for a TLS connection.

apTlsACPSupportEnabled	OBJECT-TYPE
SYNTAX	TruthValue
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	A value of true indicates that TLS is supported; a value of false indicates that TLS is not supported.
::=	{apSysMGMTMIBGeneralObjects 40}
OID	1.3.6.1.4.1.9148.3.2.1.1.40
First Supported	M-CX3.0.0M1

Version MCX 3.0.0GA

M-CX3.0.0GA new capabilities required no MIB/SNMP changes.

Version MCX 2.0.0GA

M-CX2.0.0GA new capabilities required enterprise traps and MIB objects as described below.

Enterprise Traps

The following table identifies the MSG proprietary traps that the 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)

To form the OID number, append the value in the OID Ending column to 1.3.6.1.4.1.9148.3.2.5. For example, the OID for apSysMgmtTcaOid is 1.3.6.1.4.1.9148.3.2.5.65.

SNMP GET Query Name	OID Ending	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	.65	The Object ID for the Threshold Crossing Alert counter that is changing alert level
apSysMgmtTcaCurrent	.66	The current value of the counter associated with the TCA
apSysMgmtTcaMinorThreshold	.67	The current configured TCA minor threshold value
apSysMgmtTcaMajorThreshold	.68	The current configured TCA major threshold value
apSysMgmtTcaCriticalThreshold	.69	The current configured TCA critical threshold value

Acme Packet License MIB (ap-license.mib)

To form the OID number, append the value in the OID Ending column to 1.3.6.1.4.1.9148.3.5.1.1.1. For example the OID for apLicenseIKEFeature is 1.3.6.1.4.1.9148.3.5.1.1.1.18.

SNMP GET Query Name	OID Ending	Description
Object Identifier Name: apLicenseEntry (1.3.6.1.4.1.9148.3.5.1.1.1)		
apLicenseIKEFeature	.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

SNMP GET Query Name	OID Ending	Description
apLicenseIPsecTunCapPct	.19	Total IPsec tunnel capacity

Acme Packet Security MIB (ap-security.mib)

To for the OID number, append the value in the OID Ending column to 1.3.6.1.4.1.9148.3.9.1 (apSecurityMIBObjects). For example, the OID for apSecurityIPsecTunCount is 1.3.6.1.4.1.9148.3.9.1.1.

SNMP GET Query Name	OID Ending	Description
Object Identifier Name: apSecurityMIBObjects (1.3.6.1.4.1.9148.3.9.1)		
apSecurityIPsecTunCount	.1	Number of IPsec tunnels currently in progress
apSecurityIPsecTunCapPct	.2	Percentage of licensed IPsec tunnels currently in progress
apSecurityCrlIssuer	.7	CRL issuer name
apSecurityCspName	.8	Cert-status-profile object for retrieving the CRL. If the CRL is loaded from a local file, the value is specified as File.
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	.3.1.1	IP address version type of the IKE interface
apSecurityIkeInterfaceAddress	.3.1.2	IP address of the IKE interface
apSecurityIkeInterfaceCpuOverloadErrors	.3.1.3	Number of CPU overload rejections on the IKE-interface
apSecurityIkeInterfaceInitCookieErrors	.3.1.4	Number of IKE COOKIE errors on the IKE interface
apSecurityIkeInterfaceAuthErrors	.3.1.5	Number of IKE AUTH payload errors on the IKE-interface
apSecurityIkeInterfaceEapAccessRequestErrors	.3.1.6	Number of IKE EAP access request errors on the IKE interface
apSecurityIkeInterfaceEapAccessChallengeErrors	.3.1.7	Number of IKE EAP access challenge errors on the IKE interface
apSecurityIkeInterfaceTsErrors	.3.1.8	Number of IKE TS errors on the IKE-interface
apSecurityIkeInterfaceCpErrors	.3.1.9	Number of IKE config payload errors on the IKE-interface
apSecurityIkeInterfaceKeErrors	.3.1.10	Number of IKE KE errors on the IKE interface
apSecurityIkeInterfaceProposalErrors	.3.1.11	Number of IKE proposal payload errors on the IKE interface
apSecurityIkeInterfaceSyntaxErrors	.3.1.12	Number of IKE syntax errors on the IKE interface

SNMP GET Query Name	OID Ending	Description
apSecurityIkeInterfaceCriticalPayloadErrors	.3.1.13	Number of IKE critical payload errors on the IKE interface
apSecurityIkeInterfaceAuthFailureTca	.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	.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	.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
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	.9.X.1	Number of Child SA Requests on the IKE-interface.
apSecurityIkeInterfaceChildSaSuccess	.9.X.2	Number of Child SA Success on the IKE-interface.
apSecurityIkeInterfaceChildSaFail	.9.X.3	Number of Child SA Failures on the IKE-interface.
apSecurityIkeInterfaceChildSaDelRequest	.9.X.4	Number of Child SA Delete Requests on the IKE-interface.
apSecurityIkeInterfaceChildSaDelSuccess	.9.X.5	Number of Child SA Delete Success on the IKE-interface.
apSecurityIkeInterfaceChildSaDelFail	.9.X.6	Number of Child SA Delete Failures on the IKE-interface.
apSecurityIkeInterfaceChildSaRekey	.9.X.7	Number of Child SA Rekeys on the IKE-interface."
apSecurityIkeInterfaceInitialChildSa	.9.X.8	Number of Initial Child SA Establishments on the IKE-interface.
apSecurityIkeInterfaceDPDRecvPortChange	.9.X.9	Number of DPD Port Change Received on the IKE-interface.

SNMP GET Query Name	OID Ending	Description
apSecurityIkeInterfaceDPDRecvIPChange	.9.X.10	Number of DPD IP Change Received on the IKE-interface.
apSecurityIkeInterfaceDPDRespRecv	.9.X.11	Number of DPD Responses Received on the IKE-interface.
apSecurityIkeInterfaceDPDRespNotRecv	.9.X.12	Number of DPD Responses Not Received on the IKE-interface.
apSecurityIkeInterfaceDPDRecv	.9.X.13	Number of DPD Packets Received on the IKE-interface.
apSecurityIkeInterfaceDPDRetran	.9.X.14	Number of DPD Packets Retransmitted on the IKE-interface.
apSecurityIkeInterfaceDPDSent	.9.X.15	Number of DPD Packets Sent on the IKE-interface.
apSecurityIkeInterfaceIKESAPacketSent	.9.X.16	Number of IKE SA Packets Sent on the IKE-interface.
apSecurityIkeInterfaceIKESAPacketRcv	.9.X.17	Number of IKE SA Packets Received on the IKE-interface.
apSecurityIkeInterfaceIKESAPacketDropped	.9.X.18	Number of IKE SA Packets dropped on the IKE-interface.
apSecurityIkeInterfaceAuthFailure	.9.X.19	Number of Authentication Failures on the IKE-interface.
apSecurityIkeInterfaceMsgError	.9.X.20	Number of IKE Message Errors on the IKE-interface.
apSecurityIkeInterfaceAuthIDError	.9.X.21	Number of Authentication ID Errors on the IKE-interface.
apSecurityIkeInterfaceAuthCertCheckRequest	.9.X.22	Number of Certificate Status Requests on the IKE-interface.
apSecurityIkeInterfaceAuthCertCheckSuccess	.9.X.23	Number of Certificate Status Success on the IKE-interface.
apSecurityIkeInterfaceAuthCertCheckFailure	.9.X.24	Number of Certificate Status Failures on the IKE-interface.
apSecurityIkeInterfaceDDosSent	.9.X.25	Number of DDoS Sent on the IKE-interface.
apSecurityIkeInterfaceDDosRecv	.9.X.26	Number of DDoS Received on the IKE-interface.
apSecurityIkeInterfaceMessageRetrans	.9.X.27	"Number of IKE Message Retransmissions on the IKE-interface.
apSecurityIkeInterfaceSAInitMsgRecv	.9.X.28	Number of IKE_SA_INIT messages received on the IKE-interface.
apSecurityIkeInterfaceSAInitMsgSent	.9.X.29	Number of IKE_SA_INIT messages sent on the IKE-interface.
apSecurityIkeInterfaceSAEstablishmentAttempts	.9.X.30	"Number of IKE_SA establishment attempts on the IKE-interface.

SNMP GET Query Name	OID Ending	Description
apSecurityIkeInterfaceSAEstablishmentSuccess	.9.X.31	Number of IKE_SA establishment success on the IKE-interface.
apSecurityIkeInterfaceTunnelRate	.9.X.32	Number of Tunnels per second averaged over 100sec window on the IKE-interface.
apSecurityIkeInterfaceCurrentChildSaPair	.9.X.33	Current number of Child Security Association Pairs (Tunnels) on the IKE-interface.

B

Appendix B: MCZ4.0.0M1

This appendix describes the features that define the MCZ4.0.0M1 maintenance release.

MSG GTPv2 Support

The MSG now supports GPRS Tunneling Protocol version 2 (GTPv2) on the s2b interface in the core side to interwork with the existing Evolved Packet Core (EPC) architecture. The MSG acts as an enhanced Packet Data Gateway (ePDG) for a client from a non-3GPP network trying to connect to a 3GPP network. On the core side, the MSG connects to a Packet Data Network (PDN) gateway through the s2b interface, which uses GTPv2.

MSG Entitlements

The MSG is self-provisionable (that is, entitlements based) and requires only the following licenses: Base (Security Gateway), IPSec Capacity, and GTP Capacity. The requirement for a Software TLS license, added for export control reasons in release MCZ4.0.0, is eliminated as the license key is redundant because the MSG runs only on the Oracle Communications Mobile Security Gateway, which is already export controlled.

C

Appendix C: MCZ4.0.0M2

This appendix describes the features that define the MCZ4.0.0M2 maintenance release.

Offload IKE Crypto Processing to SSM3

At present all encryption, decryption, HMAC and DH calculations for IKE traffic are done by software implementation of these algorithms on the host processor, which prevents the MSG from reaching the tunnel setup rate specified by the most recent performance requirements. This feature causes the MSG to offload IKE crypto processing to, when present, the optional SSM3 (security co-processor) module, which enables hardware acceleration of encryption, compression, hashing, and PK algorithms.

Authentication Fatal Error Handling for WiFi Calling

When devices with 2G SIMs attempt to register over a WiFi network their authentication is rejected but the devices repeatedly attempt to re-register, which creates additional traffic that can lead to resource depletion in the network. This feature adds the new parameter **auth-critical-failure-threshold** to the **ike-access-control** configuration element. This parameter is used to track fatal authentication failures received from the RADIUS server. The MSG adds the peer to the DDoS (Distributed Denial of Service) deny list when the threshold is exceeded. The MSG silently drops new tunnel establishment requests from peers on the DDOS list.

Fatal authentication errors for which the MSG will add a peer to the DDoS deny list include:

- AUTHORIZATION_REJECTED
- NON_3GPP_ACCESS_TO_EPC_NOT_ALLOWED
- PLMN_NOT_ALLOWED
- RAT_TYPE_NOT_ALLOWED
- USER_UNKNOWN

The MSG uses the existing **ike-access-control** parameter **auth-failure-threshold** to track non-critical authentication failures, which include:

- NETWORK_FAILURE
- NO_APN_SUBSCRIPTION

Authentication Fatal Error Handling for WiFi Calling Configuration

1. Access the **ike-access-control** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# ike
ACMEPACKET(ike)# ike-access-control
ACMEPACKET(ike-access-control)#
```

2. Select the **ike-access-control** object to edit.

```
ACMEPACKET(ike-access-control)# select
<name>:

ACMEPACKET(ike-access-control)#
```

3. **auth-critical-failure-threshold** — Enter the maximum number of critical authentication failures received from the RADIUS server that are allowed before the MSG adds the peer to the DDoS (Distributed Denial of Service) deny list. The range of values is **{0-999999999}**; the default is **1**. Setting the value to **0** disables the feature.
4. Type **done** to save your configuration.

EAP-only Authentication

IKEv2 specifies that Extensible Authentication Protocol (EAP) authentication must be used together with responder authentication based on public key signatures. This is necessary with old EAP methods that provide only unilateral authentication using, for example, one-time passwords or token cards. With EAP-SIM, EAP-AKA, EAP-AKA’, EAP-TTLS, and EAP-TLS, which provide mutual authentication and key agreement, extensible responder authentication for IKEv2 based on methods other than public key signatures can be used. This feature causes the MSG to default to EAP-only authentication without using public-key-based responder authentication unless the operator selects otherwise.

The Extensible Authentication Protocol, defined in RFC3748, is an authentication framework that supports multiple authentication mechanisms. One of the advantages of the EAP architecture is its flexibility. Rather than requiring the authenticator (for example, a wireless LAN access point) to be updated to support each new authentication method, EAP permits the use of a backend authentication server that may implement some or all authentication methods. The MSG uses a backend authentication server (for example, 3GPP AAA) and is in pass-through mode for EAP.

IKEv2 is a component of IPsec used for performing mutual authentication and establishing and maintaining Security Associations (SAs) for IPsec Encapsulating Security Payload (ESP) and Authentication Header (AH). In addition to supporting authentication using public key signatures and shared secrets, IKEv2 also supports EAP authentication. By using EAP, IKEv2 can leverage existing authentication infrastructure and credential databases, such as Home Subscriber Server (HSS), as EAP allows users to choose a method suitable for existing credentials, and also makes separation of the IKEv2 responder (MSG) from the EAP authentication endpoint (back-end Authentication, Authorization, and Accounting (AAA) server) easier. IKEv2 specifies that these EAP methods must also be used together with responder authentication based on public key signatures. For the public key signature authentication of the MSG to be effective, a deployment of Public Key Infrastructure (PKI) is required, which has to include management of trust anchors on all supplicants. This may not realistic in WiFi calling environments, in which the security of the MSG public key is the same as the security of a self-signed certificate. Mutually authenticating EAP methods alone can provide a sufficient level of security.

Because of these reasons, the MSG now defaults to EAP-only authentication without using public-key-based responder authentication unless the operator selects otherwise by disabling the new parameter **eap-only-support** in the **ike-interface** configuration element.

EAP-only Authentication Configuration

1. Access the **ike-interface** configuration element.

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

2. Select the **ike-interface** object to edit.

```
ACMEPACKET(ike-interface)# select
<address>:

ACMEPACKET(ike-interface)#
```

3. **eapOnlyAuthSupport** — The default is **enabled**. Set the value to **disabled** to use EAP authentication together with responder authentication based on public key signatures.
4. Type **done** to save your configuration.

P-CSCF Discovery through IKEv2

The typical P-CSCF discovery as standardized VoLTE does not work for seamless hand-over between VoLTE and WiFi calling because the assigned P-CSCF for a subscriber must also be used for WiFi calling for that subscriber and it is not feasible to provide every WiFi calling end-point with a specific P-CSCF address. This feature defines two new IKEv2 attributes, per RFC7296, that allow an IPsec gateway to provide the IPv4 or IPv6 address of the P-CSCF server. These attributes can be exchanged by IKEv2 peers as part of the configuration payload exchange. The attributes follow the configuration attribute format defined in Section 3 of RFC7651. Furthermore, providing P-CSCF server addresses in IKEv2 as standard attributes enables clients to directly access IMS services behind a VPN gateway without going through the 3GPP-specific interfaces. Besides addresses received from the GPRS Tunneling Protocol (GTP Protocol Configuration Options (PCO), the MSG also includes locally configured addresses in the CP payload (CFG_REPLY) of the final IKE_AUTH response message sent to the UE.

A mobile node using WiFi calling may potentially need to access the IP Multimedia Subsystem (IMS) services in the 3GPP network, the architecture for which is described in TS23228 and TS24229. Currently, there are no attributes in IKEv2, per RFC7296, that can be used for carrying these 3GPP IMS information elements. In the absence of these attributes, the mobile node must be statically configured with this information, which is operationally unfeasible. Any other approaches for discovering these functions, such as using DNS or DHCP, obtain configuration from the access network and not from the home network. Given that the above referenced 3GPP interface is primarily for allowing the mobile node to connect to the 3GPP network through an untrusted access network, the access network may not have any relation with the home network provider and may be unable to deliver the mobile node’s home network configuration.

This feature adds new parameters **pcscf-discovery**, **pcscf-v4-addr-list**, and **pcscf-v6-addr-list** to the **sg-policy** configuration element. To enable this feature, you must configure at least one **sg-policy** configuration element that is related to the **security-interface-params** configuration element through the **sg-policy-list** parameter.

P-CSCF Discovery through IKEv2 Configuration

Use the following procedure to configure P-CSCF discovery through IKEv2:

1. Access the **sg-policy** configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security
ACMEPACKET(security)# sg-policy
ACMEPACKET(sg-policy)# select
```

2. **pcscf-discovery** — Set the value to **enabled** to activate P-CSCF discovery through IKEv2. The default is **disabled**.
3. **pcscf-v4-addr-list** — Enter the IPv4 addresses for the P-CSCF server. This parameter is not used when the value of **pcscf-discovery** is **disabled**.
4. **pcscf-v6-addr-list** — Enter the IPv6 addresses for the P-CSCF server. This parameter is not used when the value of **pcscf-discovery** is **disabled**.
5. Type **done** to save your configuration.

EAP-only Authentication

IKEv2 specifies that Extensible Authentication Protocol (EAP) authentication must be used together with responder authentication based on public key signatures. This is necessary with old EAP methods that provide only unilateral authentication using, for example, one-time passwords or token cards. With EAP-SIM, EAP-AKA, EAP-AKA', EAP-TTLS, and EAP-TLS, which provide mutual authentication and key agreement, extensible responder authentication for IKEv2 based on methods other than public key signatures can be used. This feature causes the MSG to default to EAP-only authentication without using public-key-based responder authentication unless the operator selects otherwise.

The Extensible Authentication Protocol, defined in RFC3748, is an authentication framework that supports multiple authentication mechanisms. One of the advantages of the EAP architecture is its flexibility. Rather than requiring the authenticator (for example, a wireless LAN access point) to be updated to support each new authentication method, EAP permits the use of a backend authentication server that may implement some or all authentication methods. The MSG uses a backend authentication server (for example, 3GPP AAA) and is in pass-through mode for EAP.

IKEv2 is a component of IPsec used for performing mutual authentication and establishing and maintaining Security Associations (SAs) for IPsec Encapsulating Security Payload (ESP) and Authentication Header (AH). In addition to supporting authentication using public key signatures and shared secrets, IKEv2 also supports EAP authentication. By using EAP, IKEv2 can leverage existing authentication infrastructure and credential databases, such as Home Subscriber Server (HSS), as EAP allows users to choose a method suitable for existing credentials, and also makes separation of the IKEv2 responder (MSG) from the EAP authentication endpoint (back-end Authentication, Authorization, and Accounting (AAA) server) easier. IKEv2 specifies that these EAP methods must also be used together with responder authentication based on public key signatures. For the public key signature authentication of the MSG to be effective, a deployment of Public Key Infrastructure (PKI) is required, which has to include management of trust anchors on all supplicants. This may not realistic in WiFi calling environments, in which the security of the MSG public key is the same as the security of a self-signed certificate. Mutually authenticating EAP methods alone can provide a sufficient level of security.

Because of these reasons, the MSG now defaults to EAP-only authentication without using public-key-based responder authentication unless the operator selects otherwise by disabling the new parameter **eap-only-support** in the **ike-interface** configuration element.

EAP-only Authentication Configuration

1. Access the **ike-interface** configuration element.

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

2. Select the **ike-interface** object to edit.

   ```
   ACMEPACKET(ike-interface)# select
   <address>:

   ACMEPACKET(ike-interface)#
   ```

3. **eapOnlyAuthSupport** — The default is **enabled**. Set the value to **disabled** to use EAP authentication together with responder authentication based on public key signatures.
4. Type **done** to save your configuration.