

eVIP Management Guide

Evolved Virtual IP

USER GUIDE

Copyright

© Ericsson AB 2015, 2016. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



Contents

| | | |
|----------|----------------------------|-----------|
| 1 | Introduction | 1 |
| 1.1 | Prerequisites | 2 |
| 2 | Basic Concepts | 3 |
| 2.1 | eVIP Concepts | 3 |
| 2.2 | Configuration Concepts | 9 |
| 3 | Configuration | 13 |
| 3.1 | General eVIP Configuration | 13 |
| 3.2 | ALB Configuration | 17 |
| 3.3 | Traffic Configuration | 25 |





1 Introduction

Evolved Virtual IP (eVIP) is the regular method to connect a cluster to an external Data Communication Network (DCN). eVIP is a concept for collective addressing. With eVIP, a shared IP address can be used to address distributed functions in a multi-processing cluster.

eVIP can be seen as a load balancer function, as shown in Figure 1. External sources address the cluster with one address, the VIP address, which is `2011::1` in the figure. The VIP address in the figure implies IPv6, but eVIP also handles IPv4-based traffic. eVIP distributes the network traffic to the nodes in the cluster. The figure shows eVIP on a functional level. Unlike other load balancing solutions, eVIP does not usually execute as an own box-in-the-middle but is embedded in the existing cluster nodes.

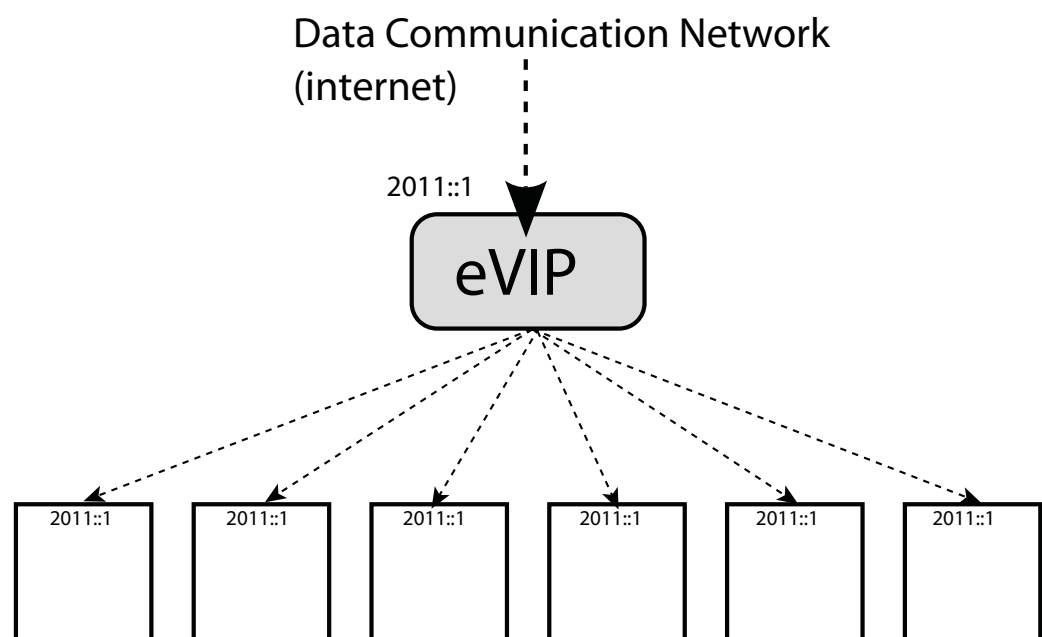


Figure 1 eVIP Load Balancer

Figure 1 also shows an important property of eVIP; the cluster nodes are configured with the VIP address, no Network Address Translation (NAT) is used. Additional eVIP properties are not mentioned in this document since they are not necessary for the configuration.

eVIP provides support for IP Security (IPsec). Some IPsec objects, like the Security Element (SE), must be configured.



1.1 Prerequisites

Ensure that the following documents have been read:

- *eVIP Internetworking*

Describes how eVIP interacts with the external DCN, for example, regarding routing configurations.

- *Managed Object Model evip_cm*

The user must be familiar with the eVIP Managed Object Model (MOM) and have it to hand when reading this document.



2 Basic Concepts

eVIP is a function in a cluster used for interfacing IP networks. A VIP address is an IP address to an abstract termination point inside the cluster, to which IP packets with a destination address corresponding to a VIP address can reach. Virtual IP simplifies network design and lets a cluster, in a network operator IP network, be addressed by a common single IP address. However, in typical deployments, a separate VIP address is often used for Operation and Maintenance (O&M) and provisioning traffic to the cluster. Thus, a cluster can have more than one VIP address if necessary; usually two or three VIP addresses are configured.

Virtual IP can also be used internally in a cluster. For example, some applications can use eVIP to communicate inside the cluster.

2.1 eVIP Concepts

Concepts pertaining to the configuration of eVIP are summarized in this section.

2.1.1 Abstract Load Balancer

The main function of eVIP is load balancing and eVIP is embedded in the existing cluster nodes. Since no physical load balancer exists, eVIP defines an Abstract Load Balancer (ALB) as shown in Figure 2.

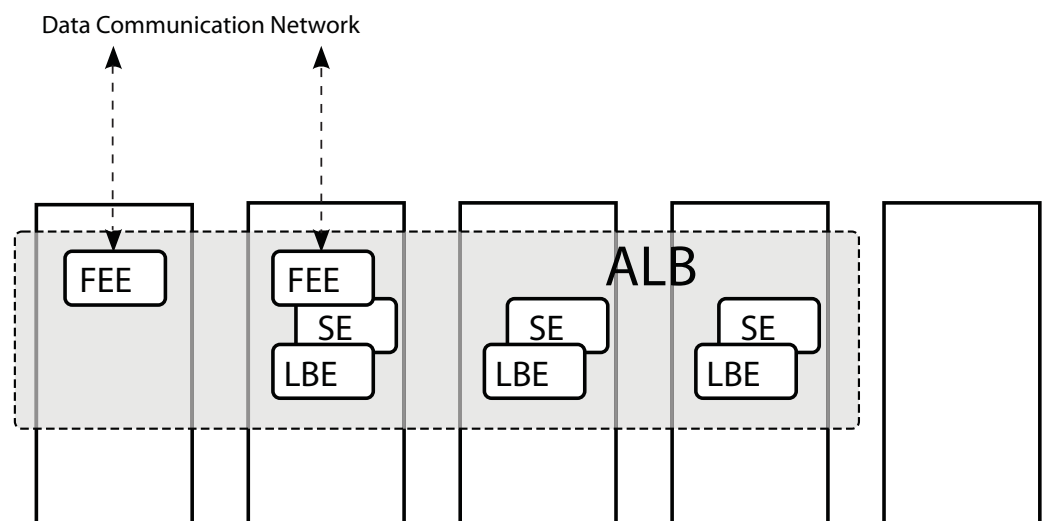


Figure 2 eVIP Abstract Load Balancer

An ALB consists of a number of Front-End Elements (FEE), a number of SEs, and a number of Load Balancer Elements (LBEs). The ALB software is executing in Linux® containers to separate the load balancing functionality

from the applications. For more information about Linux container, refer to [lxc Linux Containers](#).

There can be several ALBs configured in the cluster, each holding several VIP addresses. Network traffic belonging to different ALBs must be separated. A normal configuration is to configure one ALB for application traffic and set it as default, and a second ALB for Operations, Administration, and Maintenance (OAM) traffic that must be separated for security reasons.

ALBs have an operational state; they can be `active` or `inactive`. When a new ALB is configured, it is always `inactive`. When all necessary objects and parameters in the new ALB have been configured, the ALB can be activated. The activation can fail if the configuration is faulty. For information about configuration faults, see Section 2.2.2 Configuration Feedback on page 9.

2.1.2 Front-End Element

The FEEs handle the communication with the external DCN and must be on the nodes where the external interfaces are. The FEEs are responsible for announcing the VIP addresses to the outside world using a Routing Protocol, for example, Open Shortest Path First (OSPF). Two FEEs, configured for redundancy, are shown in Figure 2.

The FEEs utilize the Ericsson Routing Suite (ERS) for external routing; configuration is done by generic parameters in the FEE objects.

FEEs have an operational state, they can be `active` or `inactive`. When a new FEE is configured, it is always `inactive`. When a new FEE has been configured, including the external routing, it is activated. The activation can fail if the configuration is faulty.

2.1.3 Security Element

The SE is a part of the IPsec implementation. It applies security policies on the traffic flows and encrypts or decrypts traffic if necessary.

2.1.4 Load Balancer Element

A requirement on eVIP is that it has to be scalable. Therefore the network traffic is distributed among several equal LBEs, an active, or standby, solution for LBEs is not sufficient. Each LBE handles its share of the network traffic and can be configured on any node in the cluster.

When an LBE fails for some reason, its connections are redistributed among the remaining LBEs.

The LBE is implemented using the standard component IP Virtual Server (IPVS) available in the Linux kernel. The properties of the IPVS, for example, the distribution algorithms, are inherited by the LBE.



The SEs and the LBEs introduce an extra hop for network packets as shown in Figure 3. The extra hop is also taken for outgoing packets to allow the LBEs to maintain the connection state and the SEs to apply security policies.

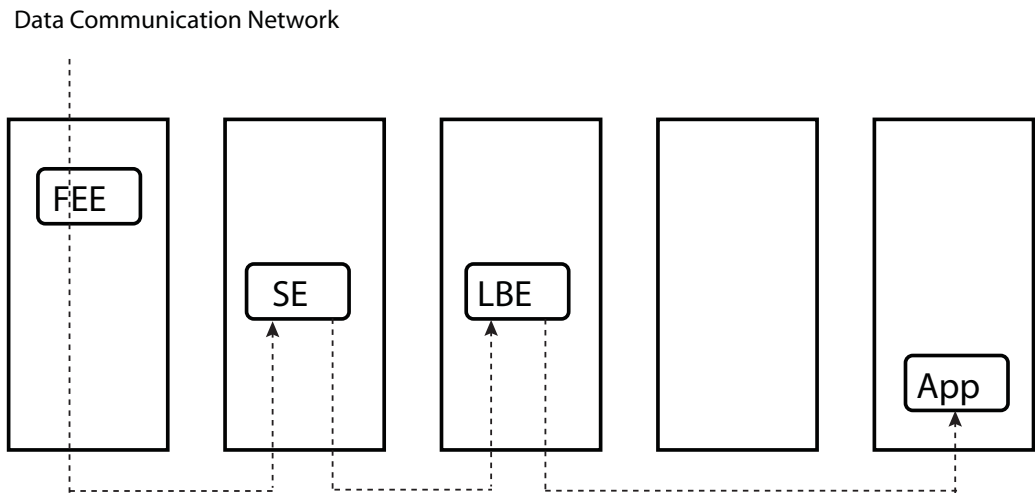


Figure 3 eVIP LBE – Extra Hop for Network Packets

The extra hop causes an increase in latency and in backplane traffic, but it is not possible to fulfill the requirement for scalability without the extra hop. There is no way to configure eVIP to avoid the extra hop.

2.1.5 Deployment of eVIP Elements

Conceptually the processing units available to eVIP can be divided into two following categories:

- Designated

A processing unit identified by a specific hostname, which is referenced in the eVIP configuration, is by convention designated and for special purpose use.

- Undesignated

A processing unit identified by a hostname, which is not referenced by the eVIP configuration, is by convention undesignated and for multi-purpose use.

The eVIP elements, through configuration, are grouped into logical eVIP nodes. The eVIP nodes are distributed across the processing units of the defined cluster.

The distribution of eVIP nodes can be done in two following distinct ways:

- Fixed

The eVIP node is fixed to a designated processing unit specified by its hostname.

- Floating

The eVIP node is dynamically assigned to one of the undesigned processing units that currently are available. This also includes that the eVIP node is automatically relocated to another active processing unit in the case of removal or failure of a currently utilized unit.

The fixed configuration is commonly used in a setup, where eVIP is deployed directly on the native physical hardware (bare metal). This provides a simple, straight forward setup which makes it possible to pin eVIP elements to processing hardware with special properties, such as physical ports and so on.

The floating configuration is for use in a virtualized environment such as cloud. In this kind of environment the support is often required for elasticity. In this context elasticity means automatically adapting the processing capacity to the current load by scaling-in or scaling-out the number of processing units available to the application. The properties of each processing unit have to be uniform across all units as the individual unit cannot be distinguished.

eVIP supports having a mixed configuration, where some eVIP nodes are fixed while others are floating. This can, for example, allow having eVIP nodes containing Front-End elements (FEEs) being fixed to non-scaling designated processing units with external connectivity, while other eVIP nodes containing SEs and LBEs are floating over other scalable undesigned units.

The eVIP FEEs can also be assigned to floating eVIP nodes and thus be relocated to another undesigned processing unit in the event of processing unit failure or removal. As the FEE provides external connectivity, special attention must be paid to network configuration to ensure a relocation of the FEE works properly. For information about network deployment, refer to *eVIP Internetworking*.

eVIP maintains a strict separation between designated and undesigned processing units; never allowing floating eVIP nodes to “invade” into designated units.

Only one eVIP node can be assigned to a processing unit. So as a consequence of scaling-in the situation can occur where the eVIP configuration contains more floating eVIP nodes than the number of undesigned processing units currently available. Each eVIP node is configured with a priority such that when this scenario arises, the eVIP control plane ensures that the highest priority nodes remain running.

Note: Priority settings together with processing unit shortage can implicate that a low priority eVIP node is swapped out with a high priority eVIP node.

Reconfiguring the cluster to have fewer Fixed Elements in favour of having more Floating Elements can be achieved by performing the following procedure.



Note: Affected processing units requires a restart for the changes to take effect.

For each Fixed Element to be removed:

1. Remove the processing unit the Fixed Element is tied to from service (for example scale-in).

In COM CLI:

- a. Remove the references to the eVIP node from the target pools.
 - b. Remove all the eVIP elements (FEE, SE, LBE) configured on the eVIP node and commit the changes.
 - c. Remove the corresponding fixed eVIP node and commit the changes.
 - d. Add an eVIP node as floating (`distribution=floating`) with a priority.
 - e. Configure eVIP elements (FEE, SE, LBE) as needed and commit the changes.
2. Return the processing unit to service or make it available again (for example scale-out).

2.1.6 Target Pool

The target pool is a group of nodes to which network traffic is distributed. The application is supposed to execute on these nodes. The traffic distribution method, for example, round robin, is defined for the target pool. The target pool contains Payload nodes each with an optional weight necessary for some distribution methods, for example, Weighted Round Robin.

In the elasticity scaling environment, when having a dynamic pool of undesignated processing units, it is not possible to preconfigure or predict on which of the processing unit the application wants to terminate traffic. For this reason the target pools can be configured to automatically adjust when scaling such that the eVIP load balancer can distribute traffic to newly added processing units. This is selected by the all-undesignated configuration option for the target pool.

Note: There is a configuration constraint not allowing an eVIP node that is configured as floating to be referenced as a payload for a target pool.

2.1.7 Flow Policy

The purpose of the flow policy is to select a part of the incoming network traffic for a particular target pool. Flow policies are network protocol-specific.



2.1.8 Selection Policy

The selection policy or ALB selection policy (as it is also known) is a configuration mechanism that is used to relate an application to an ALB. ALB selection policies should only be used in conjunction with an Adjunct Helper for the sole purpose of ALB selection when controlling a socket group. For more information, refer to *eVIP Adjunct Helper*.

Note: Selection policies have no influence upon how traffic is forwarded to, from or between ALBs. Routes or routing policies need to be added for this purpose.

2.1.9 XFRM ALB Selection Policy

When eVIP is present, some of the standard Linux commands are directed to an ALB. The XFRM ALB selection policies provide the functionality that calculates the appropriate ALB for a given command.

The XFRM ALB selection policies do not affect tunnels supervised by IKE, they affect only manual commands and any other user processes.

2.1.10 IPsec Tunnel Configuration

The IPsec tunnel configuration is actually the configuration of the IKE daemon by the ECIM-T IPsec-compliant configuration interface. The IKE daemon can be turned off if the application does not need IPsec tunnel mode with IKE or implements a key exchange protocol itself.

Setting up a tunnel requires authentication information and various tunnel parameters. A tunnel is set up between two subnetworks—local and remote—through gateways. The local and remote gateways act as tunnel endpoints. The local subnetwork on eVIP side is a VIP address, while the gateway address is also a (different) VIP address.

The authentication method can be based on preshared keys or X.509 certificates.

Note: In COM Command-Line Interface (CLI), at least one authentication method must be set for each tunnel to have a working configuration.

Other parameters provide the possibility to set encryption and integrity protection algorithms, additional restrictions of traffic selectors and various IKE parameters. If the remote side uses strongSwan version 5.0 or higher, set the `reqid` parameter on the remote side per each tunnel to avoid some known outage scenarios. For more information about setting the `reqid` parameter, refer to www.strongswan.org.

The order of steps to be followed to configure the IPsec tunnel are as follows:

1. IKE must be enabled (prerequisite).



2. Set up certificates in COM CLI if needed.
3. Create the tunnel configuration using the COM CLI.
4. Install preshared keys in COM CLI if needed.

2.2 Configuration Concepts

Configuring eVIP is done by manipulating managed objects. Generally the objects can be created, deleted, and modified. eVIP allows just a few objects to be modified, most objects are deleted or created. Deleted objects are recursive so one must double check before deleting any objects.

The allowed modification operations are as follows:

- All `EvipParam` objects can be modified.
- All `commands` attributes in any object can be modified.

2.2.1 Managed Object Model

eVIP is configured using COM. The eVIP configuration management branch is found under the `Transport` branch in the Ericsson Common Information Model (ECIM) model, see the following example:

```
ManagedElement=1,Transport=1,Evip=1.
```

For more information about eVIP configuration, refer to *Managed Object Model evip_cm*.

IPsec tunnel configuration with eVIP must be performed on another branch (IPsec fragment). The IPsec fragment is contained by the L3 infrastructure. In the L3 infrastructure, eVIP ALBs are represented as abstract hosts, for example:

```
(config-Transport=1)>show
Transport=1
Evip=1
Host=eVIP_ALB_alb_0
Host=eVIP_ALB_alb_1
```

These Hosts are instantiated automatically for each ALB. The user can select the ALB where the IPsec tunnel is instantiated by defining it under the corresponding `Host` object.

The eVIP configuration can be updated using Network Configuration (NETCONF) or the ECLI.

2.2.2 Configuration Feedback

All configuration changes are made in transaction Information Model Management (IMM) Configuration Change Bundles (CCBs). When the



transaction is committed and the data is written to the IMM, the updated configuration is pushed to eVIP. Therefore the updated configuration can contain semantic faults that are written into the IMM but are not detected until the transaction is committed.

If the configuration fails because of a semantic fault after a CCB is committed, a notification is sent by eVIP and the failure is logged in the `syslog`. For more information about the notification, refer to *eVIP, Configuration Fault*.

Note: When configuring eVIP, the operator must monitor SAF Notifications.

2.2.3 Actions

Activation and deactivation of ALBs and FEEs can be performed through “actions” on the objects. A configuration fault can be detected when the object is activated so the operator must monitor SAF Notifications during activation.

Note: Deactivating a reactivating an ALB, which is handling traffic, is not recommended. High impact on In-Service Performance (ISP) occurs, including possible node restarts.

2.2.4 Startup Commands

Startup commands can be defined in many places in the eVIP configuration. The startup commands are introduced as a way to cover unforeseen events. Normally startup commands are not necessary but in the case where some special requirement has been missed during a particular installation, then startup commands can be useful.

The actual shell commands that are executed are defined in the `EvipDeclarations` branch and are referred to by name from other places. The names of the commands to be executed are defined in a multi-value attribute. This attribute is named `commands` and is present in many eVIP managed objects. The order of execution is important so the command names must be preceded by a floating-point order number in this attribute, as follows:

```
1.0:do_init_stuff
1.5:setup
100:wrapup
```

The `commands` attribute is one of the few attributes that can be modified (using a `modify` operation). If the commands are modified, the updated commands are not executed immediately, they are executed on the next startup. The floating-point order number allows commands to be inserted anywhere in the command sequence.

For more information about the `commands` attribute, refer to *Managed Object Model evip_cm*.

The SEs do not support startup commands.



2.2.5 Parameter Objects

Parameters can be specified in various places in the eVIP configuration. The parameters are defined using the `EvipParam` class. The name of the parameter is in the Relative Distinguished Name (RDN) and the value is in the `value` attribute.

All `EvipParam` objects can be modified in runtime.





3 Configuration

This section describes the configuration of eVIP.

3.1 General eVIP Configuration

Some configurations are not directly related to load balancing or network traffic, such as the following:

- Startup command definitions
- Cluster definition
- eVIP parameters
- Port Ranges

If these `sysctls` are set to the following default values:

- `net.ipv6.neigh.default.gc_thresh1 = 128`
- `net.ipv6.neigh.default.gc_thresh2 = 512`
- `net.ipv6.neigh.default.gc_thresh3 = 1024`
- `net.ipv6.route.max_size = 4096`
- `net.ipv6.route.gc_thresh = 1024`

Then, the `sysctls` are set by eVIP as follows:

- `sysctl -w net.ipv6.neigh.default.gc_thresh1=4096`
- `sysctl -w net.ipv6.neigh.default.gc_thresh2=16384`
- `sysctl -w net.ipv6.neigh.default.gc_thresh3=32768`
- `sysctl -w net.ipv6.route.max_size=32768`
- `sysctl -w net.ipv6.route.gc_thresh=12288`

3.1.1 Specify a Startup Command

Commands are defined in the MOM under: `Evip=1,EvipDeclarations=1,EvipCommandDefinition=1`. The name (RDN) of the command is used as a reference when the command is used as a startup command at other places in the eVIP configuration. When the command is referred to as a startup

command, the name must be preceded with a floating-point order number with colon “:”, as follows:

```
1.0:init_command  
2.0:some_command
```

The `command` attribute in the object is the shell command to execute. It is executed by a shell and can contain redirections.

The following restrictions apply for the commands:

- The name (RDN) of the command is not allowed to be longer than 30 characters.
- The command string is not allowed to be longer than 100 characters.
- The number of referenced commands in one place (for example: in one node, in the LBEs or in one LBE) is not allowed to be more than 30 commands.

Delete a Command Definition

Before deleting a command definition, all referring commands must be deleted. The referring commands must be deleted in a separate `ccb`, and then the command definition can be deleted in a new `ccb`.

3.1.2 Define Cluster

The cluster is defined as `Evip=1,EvipDeclarations=1,EvipCluster=1`.

The `eVIP Cluster` object contains a mandatory `primaryInterface` attribute. The primary interface is used for all eVIP traffic between the nodes. The `primaryInterface` is redundant in some way, for instance with bonding.

Under the `eVIP Cluster`, any number of eVIP nodes can be defined. The RDN must be the `nodeID`. The nodes can be fixed or floating type, as distinguished by the distribution parameter (fixed is default).

Fixed nodes require a mandatory `hostname` attribute, whereas floating nodes do not. In the case of floating nodes, `hostname` is resolved in runtime and is not part of the configured information. The floating nodes instead require a mandatory `floatPriority` attribute.

The primary Interface on cluster level must also be replaced by a node-specific interface in this object.



Note: Regarding floating nodes:

An eVIP node swapping because of priority can cause a minor traffic disturbance, especially when few processing units are available. One way of avoiding this, is to give two or more nodes the same “full” traffic configuration and the same highest priority value. Given equal priority no node-swapping occurs and thus fewer disturbances will happen.

The `commands` attribute under the `EvipCluster` object allows the user to specify `startup` commands that they would like run against all nodes defined in the cluster. A special case exists for nodes that are undesignated, if a user wishes to run a different set of commands on these nodes, then the `commandsForAllUndesignated` attribute must be used. For more information, refer to *Managed Object Model evip_cm*.

3.1.3 Configure eVIP Parameters

The general eVIP parameters are configured under `Evip=1, EvipParams=1`.

The following parameters can be configured:

- `syslog_log_level`
- `syslog_facility`

Regarding `syslog_log_level`, the `value` attribute of this object is a number in the value range 0–7 and 10. The value range 0–7 corresponds to the standard syslog levels. Log messages with higher priority than the configured levels are suppressed.

When the `syslog_log_level` is set to 10, it begins storing of extended debug information. The extended debug information is stored in directory `/var/log/`.

Note: The collection and storage of the extended debug information leads to reduced traffic handling capability and must not be used without specific instructions from Ericsson.

Regarding `syslog_facility`, the `value` attribute of this object is a number from 0 through 7. eVIP passes this value as the `facility` parameter for logging to the Linux `syslog`.

The other eVIP parameters are intended for special conditions and must not be set or modified without specific instructions from Ericsson.

The following parameter is internal and care must be taken when modified: Maximum Transmission Unit (MTU) and Max Hot Standby.

The Maximum Transmission Unit (MTU) parameter governs the MTU of the data path in eVIP. The default value is 1452 since the eVIP internal IPv6 tunnels add an overhead of 48 bytes. This default setting works for the widest variety of available Ethernet switching infrastructure hardware and there is in general no



need to deviate from the default parameter. Changing the parameter in service leads to a cluster reboot and interruption of service.

If it is necessary to modify the MTU parameters, caution must be taken with regards to the following: To avoid unintentional modification of MTU pertaining to other backplane network than the intended eVIP backplane network in the internal infrastructure, it is highly recommended that the cluster internal network design is done in such a way that eVIP backplane network regarding MTU setting is separated from others backplane networks, for example, the LDE backplane network. This separation is achieved by configuring additional `macvlans` in the `cluster.conf` in LDE in the following way: In the `cluster.conf` file, specify a `macvlan` on top of the bonding device and use this `macvlan` for further LDE configuration.

The `Max Hot Standby` parameter governs the number of control plane that will be in hot standby state, continuously receiving system state information from the active control plane. Default value is 1 since most systems runs with only two eVIP control planes (one active and one hot standby). To keep maximum redundancy in a cloud scaling environment the value can be increased up to 7 hot standby control planes.

Note: The effect of changing the value of `Max Hot Standby` may only be seen after a cluster reboot.

3.1.4 Port Ranges

Port ranges can be defined under `Evip=1, EvipPortRanges=1`.

The port ranges are specified per protocol. The port ranges specify which ports are used for known services (well-known). Remaining ports are used for temporary connections (ephemeral). Ephemeral port ranges are not specified. Port ranges are only configured for applications with special needs, normally no port ranges are specified. The default; 32k well-known and 32k ephemeral port is OK for almost all applications.

Changing (add or delete) port ranges is allowed but requires, or causes, node reboots and thus have an impact on ISP.

The port ranges have the following constraints:

- The port range must have a size that is a multiple of 64.
- The port range must start with a port that is a multiple of 64 (64-aligned).
- Exception; the first allowed range is port 1–63 (since port 0 has special use).

For example, 100–163 is invalid since the starting port is misaligned, 64–1000 is invalid since the size is not a multiple of 64, 1–1023 is valid.

```
Evip=1, EvipPortRanges=1, EvipProtocol=tcp, EvipPortRange=1-2047
```



3.2 ALB Configuration

This section describes how to define the equivalent to a physical Load Balancer. Configuring the ALB includes the routing setup in the FEEs.

The ALBs must be created under `Evip=1, EvipAlbs=1`.

In this object, it is possible to specify startup commands that are executed on all ALB containers.

The ALB must be given a unique name (RDN), for example, `Evip=1, EvipAlbs=1, EvipAlb=<alb name>`.

The name must comply with the following rules:

- Maximum number of 13 characters.
- Allowed characters are the following:
 - a–z
 - A–Z
 - 0–9
 - _ (underscore)
 - – (dash)
- A name cannot start with a number.

The `commands` attribute can only be modified when the ALB is in state `inactive`. The attributes are described in the MOM.

The operational state can be read in the `state` attribute and be set using `actions` on the `EvipAlb` object. When an ALB is created, it is inactive. When the ALB is configured, it can be activated. The activation can fail if the configuration is faulty.

The amount of kernel memory required increases with the number of ALBs and the number of LBEs, FEEs, and SEs allocated to the ALBs.

Delete ALB Objects

To delete an ALB object:

1. Delete any IPsec tunnel in the desired ALB, under the corresponding Host object. The following example list the objects under the Host object and delete any IPsec tunnel related object:

Example:



```
>configure
(config)>ManagedElement=1,Transport=1,Host=eVIP_ALB_alb_1
(config-Host=eVIP_ALB_alb_1)>show
l3Ref="ManagedElement=1,Transport=1,Evip=1,EvipAlbs=1,EvipAlb=alb_1"
IsecProposalProfile=1
Ikev2PolicyProfile=1
IsecTunnel=1
(config-Host=eVIP_ALB_alb_1)>no IsecProposalProfile=1
(config-Host=eVIP_ALB_alb_1)>no Ikev2PolicyProfile=1
(config-Host=eVIP_ALB_alb_1)>no IsecTunnel=1
(config-Host=eVIP_ALB_alb_1)>commit
```

2. Inactivate the ALB.
3. Delete the ALB object.

3.2.1 ERSIP Parameters

Some parameters for the ALB can be defined under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipErsipParams=1`.

The following terms apply:

lo_watermark This controls when to request or release a block of resources. More precisely, controls the minimum number of unallocated ephemeral ports that the client tries to keep.

hi_watermark Controls the maximum number of unallocated ephemeral ports that the client tries to keep.

timeout_requested_resources The maximum number of seconds the client must wait after sending a request before considering the request as failed.

3.2.2 VIP Configuration

VIP addresses are configured under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipVips=1`.

The VIP address is the name (RDN) of the `EvipVip` object, for example, `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipVips=1, EvipVip=2007::1`.

Multiple VIP addresses can be defined for an ALB. The attributes are described in the MOM. `EvipVip` objects can be modified.

The `equivSrcAddr` attribute is used to cope with the problem of limited ephemeral ports and many outgoing connections. If multiple VIP addresses are specified, some of them can have the `equivSrcAddr` set to `yes`. If no port can be allocated for a certain VIP address on an outgoing connect, eVIP uses another equivalent VIP address.

The `autoActivate` attribute is used for initial auto activation taking the values `yes` or `no`. The attribute is only used when deploying the VIP-address



either at startup or when creating the VIP-address. If value of the attribute is changed the new setting will first come into use the next time the VIP-address is deployed, that is, after a cluster restart.

The activation state of VIP-address can be read in the `state` attribute and be set on-the-fly by `activate` and `deactivate` actions triggered on the `EvipVip` object. When a VIP is added, it is activated by default unless `autoActivate` is set to `no`.

Delete VIP Objects

To delete an `EvipVip` object:

1. Delete the `EvipFlowPolicy` objects that are referring to the VIP address.
2. Delete the `EvipVip` object.

3.2.3

SE Configuration

A number of SEs must be configured under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipSes=1`.

The SE must be given a unique name (RDN) and the node where the SE is located is specified in the mandatory `node` attribute. The naming restrictions for ALB names also apply for SE names. For more information, see Section 3.2 ALB Configuration on page 17.

Note: The eVIP SE objects cannot be modified once created except for the `commands` attribute.

Other attributes of the object must be specified on creation. The attributes are described in the MOM.

3.2.4

LBE Configuration

A number of LBEs must be configured under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipLbes=1`.

The LBE must be given a unique name (RDN) and the node where the LBS is located is specified in the mandatory `node` attribute. The naming restrictions for ALB names also apply for LBE names. For more information, see Section 3.2 ALB Configuration on page 17.

Note: The eVIP LBE objects cannot be modified once created except for the `commands` attribute.

Other attributes of the object must be specified on creation. The attributes are described in the MOM.



3.2.5 FEE Configuration

A number of FEEs can be configured under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipFees=1`.

The FEE must be given a unique name (RDN) and the node where the FEE is located is specified in the mandatory `node` attribute. The FEEs must be on nodes that have access to the external DCN. The naming restrictions for ALB names also apply for FEE names. For more information, see Section 3.2 ALB Configuration on page 17.

Note: The `EvipFee` objects cannot be modified once created except for the `commands` attribute.

Other attributes of the object must be specified on creation. The attributes are described in the MOM.

The operational state of the FEE can be read in the `state` attribute.

Any number of `EvipRoutingSetup` objects can be configured under the ALB. The name (RDN) of the `EvipRoutingSetup` objects defines the routing protocol. To configure routing, knowledge of the particular routing protocol is required. The individual routing protocols are not described in this document, only a brief description of the available parameters is given.

Note: The `EvipRoutingSetup` objects may not be altered (created/deleted/modified) when the FEE is active. To alter the routing configuration the ALB must be inactivated or the FEE must be deleted and recreated with updated routing configuration.

If VLAN is not used, the host interfaces that are configured to be used by the FEEs, for example, `bond0`, must be in state `up` before the FEE is instantiated.

If VLAN is used, the base interface must be in state `up` and the `vlan` interface, for example `bond0.4711`, must not be created.

Note: For the FEE to become active both IPv4 and IPv6 must be working if both are configured. This means that a faulty IPv6 configuration prevents a correct IPv4 configuration from working. If only IPv4 is used, IPv6 must not be configured in the FEE.

3.2.6 Supervised Remote Gateway

A number of `EvipSupervisedRemoteGateway` can be configured under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipFees=1, EvipFee=<FEE name>`.

The RDN must be the IP address of the remote gateway to supervise, for example: `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipFees=1, EvipFee=<FEE name>, EvipSupervisedRemoteGateway=192.168.99.7`



The sole purpose of the `EvipSupervisedRemoteGateway` object is to serve as alarming object for the `eVIP, Gateway Unavailable` alarm. To get this alarm, an `EvipSupervisedRemoteGateway` object must be created.

The description attribute in the `EvipSupervisedRemoteGateway` object is a free text field, intended for information that is useful for the alarm receiver, for example, a contact person.

3.2.7 OSPFv2 Configuration

The OSPFv2 parameters are defined under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipFees=1, EvipFee=<FEE name>, EvipRoutingSetup=ospfv2`.

The parameters are the following:

| | |
|----------------------------|--|
| local_address | The local address for the interface that is communicating with the remote router. |
| router_id | Specifies a router ID for the OSPF process, must be unique for each OSPF instance. |
| area | Specifies the area ID to use to communicate with the remote OSPF routers. OSPF routers and links are grouped logically into areas that are identified by this area ID. |
| area_type | Specifies the area type to be used; <code>stub</code> and <code>nssa</code> are supported in OSPFv2. |
| hello_interval | Used to specify the interval between <code>hello</code> packets; default interval is 10 seconds. |
| dead_interval | Used to specify the interval during which no <code>hello</code> packets are received and after which a neighbor is declared dead; default is 40 seconds. |
| retransmit_interval | Used to specify the time between Link-State Advertisement (LSA) retransmission for adjacencies belonging to the interface. The default interval is 5 seconds. |
| router_priority | Used to set the router priority to determine the designated router for the network. Default is 0, which means the FEE is <code>DRother</code> . |
| spf_delay | Specifies the SPF minimum hold time in milliseconds between two SPF calculations. Default value is 500 milliseconds. |



| | |
|-----------------------|---|
| spf_interval | Specifies the maximum delay in milliseconds between receiving a change to the SPF calculation. The value must be higher than spf_delay . Default value is 1000 milliseconds. |
| transmit_delay | Used to set the estimated time it takes to transmit a link state update packet on the interface. Default is 1 second. |

3.2.8 OSPFv2 and BFD Configuration

The OSPFv2 and Bidirectional Forwarding Detection (BFD) parameters are defined under `Evip=1,EvipAlbs=1,EvipAlb=<alb name>,EvipFees=1,EvipFee=<FEE name>,EvipRoutingSetup=bfd_ospfv2`.

The parameters are the following:

| | |
|----------------------|--|
| echo | If echo is on, then BFD echo packets are used in addition to the usual BFD control packets. |
| echo_interval | Used to set the BFD echo interval. If it is not set, but the echo parameter is on, then the default value 20 milliseconds is used. |
| slow_timer | Used to set a BFD slow timer interval. |
| bfd_interval | Value of the transmit interval, in milliseconds. |
| minrx | Value of the reception interval, in milliseconds. |
| multiplier | Integer value of the <code>hello</code> multiplier. |

3.2.9 OSPFv3 Configuration

The OSPFv3 parameters are defined under `Evip=1,EvipAlbs=1,EvipAlb=<alb name>,EvipFees=1,EvipFee=<FEE name>,EvipRoutingSetup=ospfv3`.

The parameters are the following:

| | |
|----------------------|--|
| local_address | The local address for the interface that is communicating with the remote router. This is an optional parameter, but it is recommended to set it to allow ICMPv6 to work. It has to be routable, for example, not a link-local address, but must not be a VIP address. |
| router_id | Specifies a router ID for the OSPF process; must be unique for each OSPF instance. |



| | |
|----------------------------|---|
| area | Specifies the area ID to use to communicate with the remote OSPF routers. OSPF routers and links are grouped logically into areas that are identified by this area ID. |
| area_type | Specifies the area type to be used; <code>stub</code> and <code>nssa</code> are supported in OSPFv3. |
| hello_interval | Used to specify the interval between <code>hello</code> packets. Default interval is 10 seconds. |
| dead_interval | Used to specify the interval during which no <code>hello</code> packets are received and after which a neighbor is declared dead; default is 40 seconds. |
| retransmit_interval | Used to specify the time between LSA retransmission for adjacencies belonging to the interface. The default interval is 5 seconds. |
| router_priority | Used to set the router priority to determine the designated router for the network. Default is 0, which means the FEE is <code>DRother</code> . |
| spf_delay | Specifies the SPF minimum hold time in milliseconds between two SPF calculations. Default value is 500 milliseconds. |
| spf_interval | Specifies the maximum delay in milliseconds between receiving a change to the SPF calculation. The value must be higher than <code>spf_delay</code> . Default value is 1000 milliseconds. |
| transmit_delay | Used to set the estimated time it takes to transmit a link state update packet on the interface. Default is 1 second. |

3.2.10 OSPFv3 and BFD Configuration

The OSPFv3 and BFD parameters are defined under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipFees=1, EvipFee=<FEE name>, EvipRoutingSetup=bfd_ospfv3`.

The parameters are the following:

| | |
|----------------------|--|
| echo | If echo is on, then BFD echo packets are used in addition to the usual BFD control packets. |
| echo_interval | Used to set the BFD echo interval. If it is not set, but the echo parameter is on, then the default value 20 milliseconds is used. |



| | |
|---------------------|---|
| slow_timer | Used to set a BFD slow timer interval. |
| bfd_interval | Value of the transmit interval, in milliseconds. |
| minrx | Value of the reception interval, in milliseconds. |
| multiplier | Integer value of the <code>hello</code> multiplier. |

3.2.11 Static BFD IPv4 Configuration

The static BFD IPv4 parameters are defined under `Evip=1,EvipAlbs=1,EvipAlb=<alb name>,EvipFees=1,EvipFee=<FEE name>,EvipRoutingSetup=bfd_static`.

The parameters are the following:

| | |
|----------------------|--|
| local_address | Specifies the local address, which must be in IPv4 format, for the interface that is communicating with the remote router. |
| gateway | Specifies the IP address to the remote gateway, which must be in IPv4 format. |
| echo | If echo is on, then BFD echo packets are used in addition to the usual BFD control packets. |
| echo_interval | Used to set the BFD echo interval. If it is not set, but the echo parameter is on, then the default value 20 milliseconds is used. |
| slow_timer | Used to set a BFD slow timer interval. |
| bfd_interval | Value of the transmit interval in milliseconds. |
| minrx | Value of the reception interval in milliseconds. |
| multiplier | Integer value of the <code>hello</code> multiplier. |

3.2.12 Static BFD IPv6 Configuration

The static BFD IPv6 parameters are defined under `Evip=1,EvipAlbs=1,EvipAlb=<alb name>,EvipFees=1,EvipFee=<FEE name>,EvipRoutingSetup=bfd_static6`.

The parameters are the following:

| | |
|----------------------|--|
| local_address | Specifies the local address, which must be in IPv6 format, for the interface that is communicating with the remote router. |
| gateway | Specifies the IP address to the remote gateway, which must be in IPv6 format. |



| | |
|----------------------|--|
| echo | If echo is on, then BFD echo packets are used in addition to the usual BFD control packets. |
| echo_interval | Used to set the BFD echo interval. If it is not set, but the echo parameter is on, then the default value 20 milliseconds is used. |
| slow_timer | Use d to set a BFD slow timer interval. |
| bfd_interval | Value of the transmit interval in milliseconds. |
| minrx | Value of the reception interval in milliseconds. |
| multiplier | Integer value of the hello multiplier. |

3.3 Traffic Configuration

Once the ALB is configured, the actual eVIP traffic can be configured.

3.3.1 Target Pool Configuration

Target pools are configured under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipTargetPools=1`.

The target pool must be given a unique name (RDN). This name is used when defining a flow policy to select the network traffic directed to the target pool.

Only the `allUndesignated` attribute may be modified in a `EvipTargetPool` object. To modify other attributes the target pool must be deleted and recreated with the updated parameters. The attributes are described in the MOM.

The target pool defines the nodes where the traffic handling application is executing. Both fixed and floating nodes can be part of a target pool. Fixed nodes are specified explicitly, whereas floating nodes are specified by setting the `allUndesignated` attribute to `yes`. Setting this attribute adds all floating nodes to the target pool.

An important attribute specified in the target pool is the distribution method. The distribution method defines the algorithm used for distributing network traffic among the nodes in the target pool. The LBE is implemented using the standard IPVS.

For more information about IPVS, refer to [IPVS documentation](#).

The available distribution methods in IPVS are the following:

- `round_robin`
- `weighted_round_robin`
- `least_connection`



- `weighted_least_connection`
- `locality_based_least_connection`
- `locality_based_least_connection_with_replication`
- `destination_hash`
- `source_hash`
- `shortest_expected_delay`
- `never_queue`

For more information about the distribution method in IPVS, refer to [IPVS documentation](#).

The target pool can have the `stickyGroup` attribute set. This means that all network traffic from one remote host is distributed to the same node. This can, for example, be necessary to hold together an HTTP session consisting of multiple connections. However, `stickyGroup` can cause poor distribution if there is a dominant source, and is to be avoided if possible.

Note:

The following limitations apply when using a target pool that has the `stickyGroup` attribute set:

- Connected flow policies cannot be both IPv4 and IPv6.
- Connected flow policies cannot have different destination IP addresses.
- ALB in which target pool resides cannot use 5-tuple hashing.

`stickinessTimeout` defines the time in seconds that a connection is remembered after the traffic flow stopped and the configurable IPVS timer also expired; these IPVS timers are `ipvs_tcp_timeout` for silent but unfinished TCP sessions, `ipvs_tcpfin_timeout` for finished TCP sessions and `ipvs_udp_timeout` for silent UDP pseudo sessions. During `stickinessTimeout` and the additional IPVS timeout subsequent connections from the source IP go to the same node. The maximum value for `stickinessTimeout` is 2,592,000 (30 days). `stickinessTimeout` is protocol agnostic. If `stickyGroup` is set, `stickinessTimeout` is implicit; a default of 360 (6 minutes) is used if `stickinessTimeout` is not specified.

Note: If a node belongs to an ALB, it must be included in at least one target pool for that ALB. This means that if the node is only for outgoing traffic or SCTP traffic, and does not require a target pool, the node must still be included in a target pool even if this target pool only is a “dummy” pool.



3.3.2 Flow Policy Configuration

Target pools are configured under `Evip=1, EvipAlbs=1, EvipAlb=<alb name>, EvipFlowPolicies=1`.

The purpose of a flow policy is to select what part of the incoming network traffic that is directed to a particular target pool. Flow policies are protocol-specific.

The `EvipFlowPolicy` objects cannot be modified once created. So the attributes of the object must be specified on creation. The attributes are described in the MOM.

Only 15 distinct ports or port ranges can map to one sticky target pool. This is because of a hard limit in `iptables`. Flow policies that specify ports that are in consecutive order are converted to consecutive order. For example, if there are three flow policies for `destPort` 80, 81, and 82 (otherwise identical and mapped to the same sticky target pool) these policies are converted to one range, 80–82. 15 such ranges (or individual ports) on one sticky target pool is the maximum.

3.3.3 Selection Policy Configuration

Selection policies are configured under `Evip=1, EvipSelectionPolicies=1`.

The `EvipSelectionPolicy` objects cannot be modified once created. So the attributes of the object must be specified on creation. The attributes are described in the MOM.

Selection policies, or ALB selection policies (as they are also known), can only be used in conjunction with an application supplied Adjunct Helper for ALB selection when controlling a socket group. For more information about Adjunct Helper, refer to *eVIP Adjunct Helper*.

The `SelectionPolicies` are applied in a strict order. The `SelectionPolicy` object has a mandatory `sortorder` attribute that must be a floating point number. The `SelectionPolicies` are then applied in order lowest to highest.

A good example of an ALB selection policy use case is the way in which SCTP deploys eVIP along with SS7 CAF. SS7 CAF uses an Adjunct Helper and therefore requires a selection policy to determine which ALB to use.

```
Evip, EvipSelectionPolicy=ss7_policy, , alb="ln_ss7sig_sc",
process="fe_sctp", sortorder="0.100000.
```

Example 1 SCTP with SS7 CAF

Note: The process parameter must be configured with a valid Linux process name; `fe_sctp` in this example.



3.3.4 XFRM ALB Selection Policy

This selection defines the ALB where the `ip xfrm` command is applied, as shown in Example 2 and Example 3. This configuration parameter does not affect tunnels supervised by IKE.

```
xfrm_alb_selection_policy name="control" alb="alb_0"  
order="0.100000" default="no" storage="alb" env="alb0"
```

Example 2 Selection Policy Command

The meaning of this line is that the `control` named policy has a low order. Matching commands are applied in `alb_0`, it is not the default, and the policy is transformed to the value of a Linux environment variable named `alb0`.

Possible parameters are as follows:

| | |
|---------------------|---|
| payload_node | Optional integer, denoting the payload node where this parameter is valid. If the eVIP configuration contains floating nodes, this parameter must be empty, and every XFRM ALB selection policy has an effect on every blade. |
| name | Mandatory string, name of the policy. |
| alb | Mandatory string, the target ALB |
| process | Optional string. The name of the process to which this policy maps. |
| env | Optional string. Set the <code>EVIP_XALB</code> environment variable to this value to use this policy. Note: Different policies must have a different name. |
| order | Optional real number. The order of the selection policy. The selection policy with lowest order is used if multiple policies match. |
| storage | Optional string. Describes how IPsec objects are stored. <code>local</code> = use local Linux kernel, <code>node</code> = accessible from inserter node only, <code>alb</code> = accessible from the complete ALB. On floating nodes, the node storage type is not supported. |
| default | Set to <code>yes</code> if the selection policy is default, which means that this policy matches every process name and every value. Order is still taken into account. |



```
(config-EvipXfrmSelectionPolicies=1)>show all
EvipXfrmSelectionPolicies=1
  EvipXfrmSelectionPolicy=control
    alb="alb_0"
    default="no"
    env="alb0"
    order="0.100000"
    storage="alb"
  EvipXfrmSelectionPolicy=payload
    alb="alb_1"
    default="yes"
    order="1.000000"
    storage="alb"
```

Example 3 Selection Policy Code

This is a configuration showing that, by default, security policies and security associations are expected to go to `alb_1` and must have scope `alb`.

3.3.5 IPsec Tunnel Configuration

Note: The names used in the RDN for all objects created under the `Host` object must not exceed 120 characters.

The IKE daemon must be enabled. This can be ensured by setting the `ike_enabled` parameter to `yes`. The parameter is found in `EvipParams`.

Note: In a virtualized environment (such as cloud), if both System Controllers become unavailable, IPsec Tunnel mode operation can be impacted and can result in a prolonged IPsec traffic outage.

3.3.5.1 Handling of Authentication Information

If certificates are used, the installation of certificates must be done before the tunnel configuration.

If preshared key is used, it must be installed after the corresponding `Ikev1Session` or `Ikev2Session` was committed.

3.3.5.2 Create a Tunnel

To configure IPsec tunnels in the desired ALB, the user must navigate to the corresponding `Host` object under `Transport`.

```
>configure
(config)>ManagedElement=1,Transport=1,Host=eVIP_ALB_alb_1
```

The user can define IPsec tunnels (ESP) with IKEv2 or IKEv1 key exchange.

A typical IPsec tunnel configuration with IKEv2 contains the following:

- `IpsecTunnel`



- `Ikev2Session`
- `IpsecPolicy`
- `Ikev2PolicyProfile` (referenced from `Ikev2Session`)
- `IpsecProposalProfile` (referenced from `IpsecPolicy`)

A typical IPsec tunnel configuration with IKEv1 contains the following:

- `IpsecTunnel`
- `Ikev1Session`
- `Phase2Policy`
- `Ikev1PolicyProfile` (referenced from `Ikev1Session`)
- `IpsecProposalProfile` (referenced from `Phase2Policy`)

Classes specific to IKEv1 or IKEv2 are mutually exclusive under one `IpsecTunnel`.

The following is a basic example configuration for a tunnel with IKEv2:



```

IpssecProposalProfile
-----
IpssecProposalProfile=1
ipsecProposal[@1]
diffieHellmanGroup=MODP_2048_GROUP_14
encryptionAlgorithm=ENCR_AES_CBC_256
integrityAlgorithm=AUTH_HMAC_SHA1_96

Ikev2PolicyProfile
-----
Ikev2PolicyProfile=1
ikev2Proposal[@1]
diffieHellmanGroup=MODP_2048_GROUP_14
encryptionAlgorithm=ENCR_AES_CBC_128
integrityAlgorithm=AUTH_HMAC_SHA1_96

IpssecTunnel
-----
IpssecTunnel=1
remoteAddressStr="10.128.171.101"
localAddressStr="10.0.20.101"
IpssecPolicy=1
ipsecProposalProfile="ManagedElement=1,Transport=1,
Host=eVIP_ALB_alb_1,IpssecProposalProfile=1"
localTrafficSelector[@1]
addressRange="10.0.20.1/32"
up
remoteTrafficSelector[@1]
addressRange="10.128.171.1/32"

Ikev2Session
-----
Ikev2Session=1
ikev2PolicyProfile="ManagedElement=1,Transport=1,
Host=eVIP_ALB_alb_1,Ikev2PolicyProfile=1"

```

Example 4 Tunnel with IKEv2

The user can commit all these object by object or all at once.

In case of preshared key authentication, the user must install the key under the corresponding IKE session object (Ikev1Session or Ikev2Session) with the ECIM action `installPreSharedKey`.

Note: With IKEv2, if different traffic flows are to be protected by one IPsec tunnel, the user must configure more traffic selectors under the same `IpssecPolicy` object. Using more than one `IpssecPolicy` object under the same `IpssecTunnel` object is not supported.