

eVIP Management Guide

Evolved Virtual IP

USER GUIDE

Copyright

© Ericsson AB 2015–2017. 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
2.1.1	Abstract Load Balancer	3
2.1.2	Front-End Element	4
2.1.3	Security Element	4
2.1.4	Load Balancer Element	4
2.1.5	Deployment of eVIP Elements	5
2.1.6	Resilient FEE IP Addresses	7
2.1.7	Target Pool	8
2.1.8	Flow Policy	8
2.1.9	Selection Policy	8
2.1.10	XFRM ALB Selection Policies	9
2.1.11	IPsec Tunnel Configuration	9
2.2	Configuration Concepts	9
2.2.1	Managed Object Model	10
2.2.2	Configuration Feedback	10
2.2.3	Actions	10
2.2.4	Startup Commands	11
2.2.5	Parameter Objects	11
3	Configuration	13
3.1	General eVIP Configuration	13
3.1.1	Specify a Startup Command	14
3.1.2	Delete a Command Definition	14
3.1.3	Define a Cluster	14
3.1.4	Configure an eVIP Parameter	15
3.1.5	Define a Port Range	16
3.2	ALB Configuration	17
3.2.1	Create an ALB	17
3.2.2	Delete an ALB Object	18
3.2.3	Define an ERSIP Parameter	19
3.2.4	Configure a VIP Address	19
3.2.5	Configure an SE	21
3.2.6	Configure an LBE	21
3.2.7	Configure an FEE	21
3.2.8	Configure a Supervised Remote Gateway	22
3.2.9	Configure OSPFv2 Routing	23
3.2.10	Configure OSPFv2 Routing with BFD Supervision	24
3.2.11	Configure OSPFv3 Routing	24
3.2.12	Configure OSPFv3 Routing with BFD Supervision	25
3.2.13	Configure Static Routing with BFD Supervision for IPv4	26



3.2.14	Configure Static Routing with BFD Supervision for IPv6	26
3.2.15	Configure Static Routing for IPv4	27
3.2.16	Configure Static Routing for IPv6	27
3.3	Traffic Configuration	28
3.3.1	Configure a Target Pool	28
3.3.2	Configure a Flow Policy	30
3.3.3	Configure a Selection Policy	31
3.3.4	Configure an XFRM ALB Selection Policy	31
3.3.5	Configure an IPsec Tunnel	32



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 Virtual IP (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 usually does not execute as an own box-in-the-middle but is embedded in the existing cluster nodes.

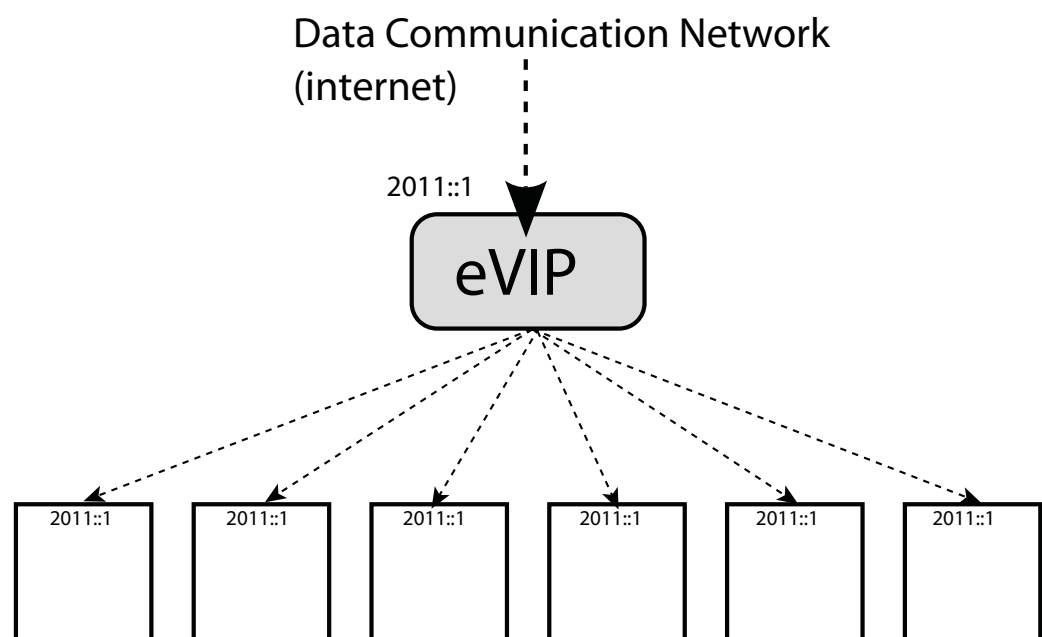


Figure 1 eVIP Load Balancer

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, as 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 document has been read:

- *eVIP Internetworking*

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

The user must be familiar with the following Managed Object Models (MOMs) and have access to them when reading this document:

- *Managed Object Model evip_cm*
- *Managed Object Model evip_ipsec*



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.

This section describes the following:

- eVIP concepts
- Configuration concepts

2.1 eVIP Concepts

2.1.1 Abstract Load Balancer

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

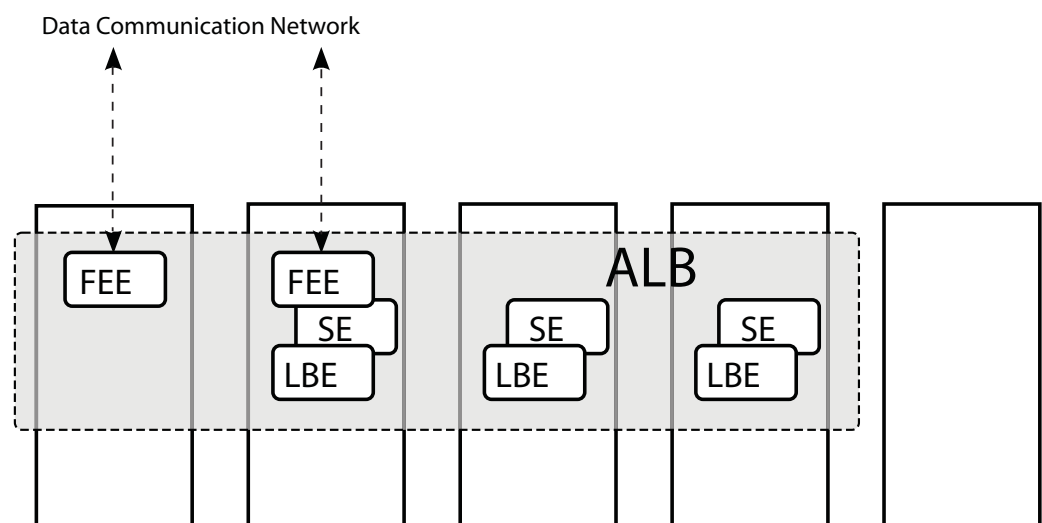


Figure 2 eVIP Abstract Load Balancer

An ALB consists of a number of Front-End Elements (FEE), Security Elements (SEs), and Load Balancer Elements (LBEs). The ALB software executes in Network Namespaces to separate the load balancing functionality from the applications. For more information about Network Namespaces, refer to [Network Namespaces](#).

Several ALBs can be 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*, which 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 10.

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. For other supported routing configuration options, see Section 3 on page 13.

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

FEEs have an operational *state*, which 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 automatically. 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 must be scalable. The network traffic is therefore 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 IPVS properties, such as 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.

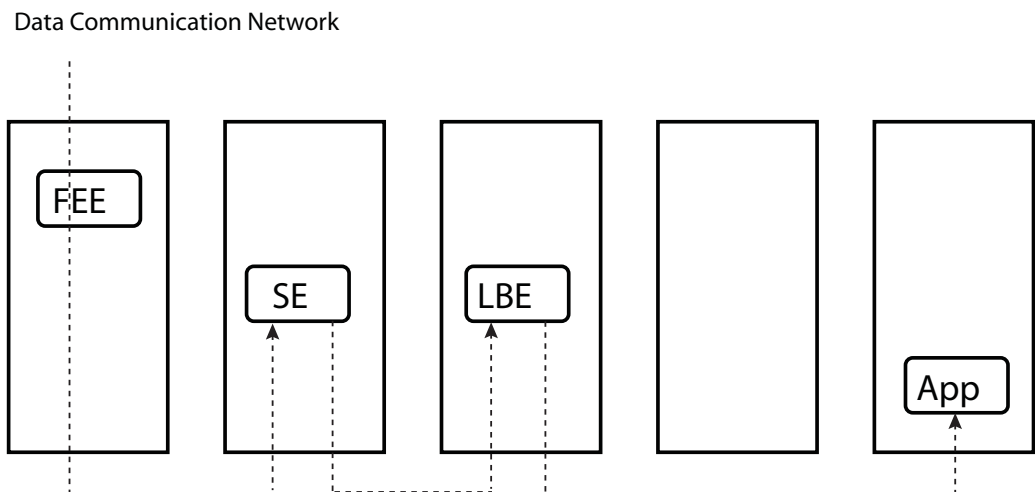


Figure 3 eVIP Load Balancer Element – Extra Hop for Network Packets

The extra hop causes an increase in latency and in backplane traffic, but the requirement for scalability cannot be fulfilled without the extra hop. eVIP cannot be configured to avoid the extra hop.

2.1.5 Deployment of eVIP Elements

Conceptually the processing units available to eVIP can be divided into the following two 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 are, through configuration, grouped into logical eVIP nodes. The eVIP nodes are distributed across the processing units of the cluster.

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

- Fixed

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

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, straightforward setup, which makes it possible to pin eVIP elements to processing hardware with special properties, such as physical ports.

- 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 event of removal or failure of a currently utilized unit.

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 must be uniform across all units, as the individual units cannot be distinguished.

eVIP supports a mixed configuration, where some eVIP nodes are fixed while others are floating. For example, some eVIP nodes with FEEs can be fixed to non-scaling designated processing units with external connectivity, while other eVIP nodes with 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 that 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 such a priority 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 mean that a low-priority eVIP node is swapped out with a high-priority eVIP node.

2.1.5.1

Remove Fixed Element

This section describes how to reconfigure the cluster to have fewer fixed elements in favor of having more floating elements.

Note: The affected processing units require 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 the Ericsson Command-Line Interface (ECLI):

- a. Remove the references to the eVIP node from the target pools.
 - b. Remove all the eVIP elements (FEE, SE, and 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 (attribute *distribution=floating*) with a priority (attribute *floatPriority*).
 - e. Configure the eVIP elements (FEE, SE, and 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

Resilient FEE IP Addresses

The end-to-end connectivity between FEEs and gateway routers must, for high availability and carrier grade performance reasons, be supervised by eVIP and gateway routers using Layer 3 next-hop supervision protocols. Bidirectional Forwarding Detection (BFD), for example, is to be used when the intermediary Layer 2 connectivity between FEEs and gateway routers is not designed for end-to-end Layer 2 carrier grade resiliency. Typically, carrier grade resiliency includes protection against both bidirectional and unidirectional faults in the intermediary connectivity infrastructure.

However, in deployments where the end-to-end Layer 2 infrastructure is carrier grade and gateway router arrangements are highly available, the need for next-hop supervision protocols between FEEs and gateway routers is not strictly required to uphold the desired high level of availability.



To provide resilience and availability for the eVIP front-end IP addresses without the use of any next-hop supervision protocol between the FEEs and gateway routers, a special mode of operation for the FEEs is provided, that is, “Resilient FEE IP Addresses”. This mode of operation can only be used in configurations where the FEEs are configured as “floating elements”, that is, the FEEs must be configured in accordance with the eVIP concept of floating elements, and the routing configuration of the FEE uses static routes.

The term “Resilient FEE IP Addresses” implies the ability of eVIP, in this mode of operation, to dynamically relocate an FEE external IP address temporarily to a “Host FEE”. Resilient FEE IP addresses are stackable on a “Host FEE”. The term IP stacking here implies that external FEE IP addresses are still available when an FEE cannot be floated to another processing unit. The stackable resilient external FEE IP addresses are reverted back to their “Home FEE” once the “Home FEE” becomes available again. For more information on IP stacking along with the system recommendations for its use, refer to *eVIP Internetworking*.

2.1.7 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 method, 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 processing unit the application wants to terminate traffic. The target pools can therefore be configured to adjust automatically when scaling so that the eVIP load balancer can distribute traffic to newly added processing units. This is selected by the *allUndesignated* attribute for the target pool (*EvipTargetPool*).

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

2.1.8 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.

2.1.9 Selection Policy

The selection policy (also called ALB selection policy) is a configuration mechanism that is used to relate an application to an ALB. ALB selection policies are to be used with an Adjunct Helper for the only purpose of ALB selection when controlling a socket group. For more information, refer to *eVIP Adjunct Helper*.



Note: Selection policies do not affect how traffic is forwarded to, from, or between ALBs. Routes or routing policies need to be added for this purpose.

2.1.10 XFRM ALB Selection Policies

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.

2.1.11 IPsec Tunnel Configuration

The IPsec tunnel configuration is the configuration of the IKE daemon using the *Managed Object Model evip_ipsec*. 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 the eVIP side is a VIP address, while the gateway address is a different VIP address.

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

Note: In the ECLI, 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 parameter `reqid` on the remote side for each tunnel to avoid some known outage scenarios. For more information about setting parameter `reqid`, refer to www.strongswan.org.

2.2 Configuration Concepts

Configuring eVIP is done by manipulating Managed Objects (MOs). Generally the objects can be created, deleted, and modified. eVIP allows just a few objects to be modified, most objects are created or deleted. As deleted objects are recursive, 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

The *Evip* configuration management branch is found under the *Transport* branch in *Managed Object Model evip_cm* as follows:

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

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

IPsec tunnel configuration with eVIP is performed on another branch (IPsec fragment), refer to *Managed Object Model evip_ipsec*. The IPsec fragment is contained by the L3 infrastructure, where 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.

eVIP is configured 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. The updated configuration can therefore 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, eVIP sends the notification *eVIP, Configuration Fault* and the failure is logged in `syslog`.

Note: When configuring eVIP, the user must monitor Service Availability Forum (SAF) notifications.

2.2.3 Actions

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



Note: Deactivating or reactivating an ALB that handles 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 if some special requirement has been missed during a particular installation, then startup commands can be useful.

The 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 the multi-value attribute `commands`, which is present in many eVIP MOs. As the order of execution is important, the command names must be preceded by a floating-point order number with colon (":") 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 parameter name 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 following:

- General eVIP configuration
- ALB configuration
- Traffic configuration

3.1 General eVIP Configuration

The following configurations are not directly related to load balancing or network traffic:

- Specify a startup command
- Delete a command definition
- Define a cluster
- Configure an eVIP parameter
- Define a port range

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 must not exceed 30 characters.
- The command string must not exceed 100 characters.
- The number of referenced commands in one place (for example in one node, in the LBEs, or in one LBE) must not exceed 30.

3.1.2 Delete a Command Definition

To delete a command definition:

1. Delete all referring commands in a separate CCB.
2. Delete the command definition in a new CCB.

3.1.3 Define a Cluster

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

The *EvipCluster* object contains a mandatory *primaryInterface* attribute. The primary interface is used for all eVIP traffic between the nodes. It is redundant in some way, for example with bonding.

Under the *EvipCluster* object, any number of *EvipNode* objects can be defined. The RDN must be the *evipNodeId*. The nodes can be fixed or floating type, as distinguished by the *distribution* attribute (*fixed* is default).

Only fixed nodes require a mandatory *hostname* attribute. For floating nodes, *hostname* is resolved in runtime and is not part of the configured information. The floating nodes instead require a mandatory *floatPriority* attribute.



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 to avoid 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 occur.

The `commands` attribute under the *EvipCluster* object allows users to specify startup commands that they would like run against all nodes defined in the cluster. A special case exists for undesigned nodes, if a user wants to run a different set of commands on these nodes, then the *commandsForAllUndesignated* attribute must be used.

3.1.4 Configure an eVIP Parameter

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

The following parameters can be configured:

- `syslog_log_level`

The *value* attribute of *EvipParam* is a number in the range 0–7 and 10. Range 0–7 corresponds to the standard `syslog` levels. Log messages with higher priority than the configured levels are suppressed.

When set to 10, storing of extended debug information begins and 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.

- `syslog_facility`

The *value* attribute of *EvipParam* is a number in the range 0–7. eVIP passes this value as parameter `facility` for logging to the Linux `syslog`.

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

Parameter `mtu`

Parameter `mtu` (Maximum Transmission Unit (MTU)) is internal and care must be taken when modified.

The parameter governs the MTU of the data path in eVIP. Defaults to 1452, as 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 value.



Note: If the parameter is changed, a cluster reboot must be initiated, which means interruption of service.

If it is necessary to modify the `mtu` parameter, caution must be taken with regards to the following. Avoid unintentional modification of `mtu` pertaining to other backplane network than the intended eVIP backplane network in the internal infrastructure. It is therefore highly recommended that the cluster internal network design is done so that the eVIP backplane network regarding MTU setting is separated from other backplane networks, for example, the Linux Distribution Extensions (LDE) backplane network. This separation is achieved by configuring additional `macvlans` in the `cluster.conf` file in LDE as follows: in the `cluster.conf` file, specify a `macvlan` on top of the bonding device and use this `macvlan` for further LDE configuration.

Parameter `max_hot_standby`

Parameter `max_hot_standby` is internal and care must be taken when modified.

The parameter governs the number of control planes that will be in hot standby state, continuously receiving system state information from the active control plane. Defaults to 1, as most systems run 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` can only be seen after a cluster reboot.

3.1.5

Define a Port Range

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). The 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 (adding or deleting) port ranges is allowed but can require, or cause, node reboots and thus has an impact on in-service performance.

The port ranges have the following constraints:

- The port range size must be 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, as port 0 has special use.

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



```
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.

This section describes the following:

- Create an ALB
- Delete an ALB object
- Define an ERSIP parameter
- Configure a VIP address
- Configure an SE
- Configure an LBE
- Configure an FEE
- Configure a supervised remote gateway
- Configure OSPFv2 routing
- Configure OSPFv2 routing with BFD supervision
- Configure OSPFv3 routing
- Configure OSPFv3 routing with BFD supervision
- Configure static routing with BFD supervision for IPv4
- Configure static routing with BFD supervision for IPv6
- Configure static routing for IPv4
- Configure static routing for IPv6

3.2.1 Create an ALB

The ALBs are created under `Evip=1,EvipAlbs=1`.

In this object, startup commands can be specified. These are executed on all ALB Network Namespaces.

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



The name must comply with the following rules:

- Maximum number of 13 characters.
- Allowed characters:
 - 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 operational state can be read in attribute `state` 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.

3.2.2 Delete an ALB Object

To delete an ALB object:

1. Delete any IPsec tunnel in the desired ALB under the corresponding `Host` object. The following example lists the objects under the `Host` object and deletes the IPsec tunnel-related objects:

```
>configure
(config)>dn 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"
  Ikev2PolicyProfile=1
  IpsecProposalProfile=1
  IpsecTunnel=1
(config-Host=eVIP_ALB_alb_1)>no Ikev2PolicyProfile=1
(config-Host=eVIP_ALB_alb_1)>no IpsecProposalProfile=1
(config-Host=eVIP_ALB_alb_1)>no IpsecTunnel=1
(config-Host=eVIP_ALB_alb_1)>commit
```

2. Inactivate the ALB:



```
(Host=eVIP_ALB_alb_1)>dn ManagedElement=1,Transport=1,⇒
Evip=1,EvipAlbs=1,EvipAlb=eVIP_alb_1
(EvipAlb=eVIP_alb_1)>inactivate
```

3. Delete the ALB object:

```
(EvipAlb=eVIP_alb_1)>up
(EvipAlbs=1)>configure
(config-EvipAlbs=1)>no EvipAlb=eVIP_alb_1
(config-EvipAlbs=1)>commit
```

3.2.3 Define an ERSIP Parameter

The following Evolved Realm Specific IP (ERSIP) parameters for the ALB can be defined under `Evip=1, EvipAlbs=1, EvipAlb=<ALB_name>, EvipErsipParams=1`:

lo_watermark	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.

3.2.4 Configure a VIP Address

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

A VIP address can be specified as an explicit VIP address or as a named VIP address, as follows:

- Explicit VIP address

In this case, the RDN name is specified explicitly with the value of an IP address, such as `10.0.0.1` or `2007::1`, that is, given in IPv4 or IPv6 textual notation. Example:

```
Evip=1,EvipAlbs=1,EvipAlb=<ALB_name>,EvipVips=1,⇒
EvipVip=2007::1
```

The VIP address is here the RDN name of the *EvipVip* object.



Note: When a VIP address is specified with an explicit IP address value, for example as shown in this example, the *address* attribute of the *EvipVip* object cannot be filled in.

- Named VIP address

In this case, the RDN name is specified as a named VIP address. That is, a given name of choice can be used as reference to the VIP address when configuring flow policies. This chosen name for the VIP address can then be used when configuring a flow policy instead of the explicit IP address value of the VIP address. When VIP addresses are specified as named VIP addresses, the chosen names must be unique within the scope of an ALB. Example:

```
Evip=1,EvipAlbs=1,EvipAlb=<ALB_name>,EvipVips=1,=>
EvipVip=vip1_alb1
```

The RDN of the *EvipVip* object specifies here the name of this VIP object as desired. The name *vip1_alb1* is referenced by the corresponding flow policy.

The IP address value of the VIP address is specified in the *address* attribute of the *EvipVip* object. Example:

```
Evip=1,EvipAlbs=1,EvipAlb=<ALB_name>,EvipVips=1,=>
EvipVip=vip1_alb1,address=10.0.0.4
```

This IP address can be modified. The value of the *address* attribute can also be provided later or removed at any time. In that case, the VIP object is kept in a *PENDING* state.

For both explicit VIP addresses and named VIP addresses, the following apply:

- Multiple VIP addresses can be defined for an ALB. *EvipVip* objects can be modified.
- Attribute *equivSrcAddr* is used to cope with the problem of limited ephemeral ports, if many outgoing connections are used. If multiple VIP addresses are specified, some of them can have this attribute set to 1 (yes). If no port can be allocated for a certain VIP address on an outgoing connection, eVIP uses another equivalent VIP address.
- Attribute *autoActivate* is used for initial auto activation and can take the values 1 (yes) or 0 (no). The attribute is only used when deploying the VIP address either at startup or when creating the VIP address. If the attribute value is changed, the new setting comes into use the next time the VIP address is deployed, that is, after a cluster restart.
- The activation state of the VIP address can be read in the *state* attribute and be set on-the-fly by the actions *activate* and *deactivate* triggered on the *EvipVip* object. When a VIP address is added, it is activated by default unless attribute *autoActivate* is set to 0 (no).



3.2.5 Configure an SE

Several 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 (see Section 3.2.1 Create an ALB on page 17) also apply for SE names.

Note: The *EvipSe* objects cannot be modified once created except for attribute *commands*.

Other attributes of the *EvipSe* object must be specified on creation.

3.2.6 Configure an LBE

Several 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 LBE is located is specified in the mandatory *node* attribute. The naming restrictions for ALB names (see Section 3.2.1 Create an ALB on page 17) also apply for LBE names.

Note: The *EvipLbe* objects cannot be modified once created except for attribute *commands*.

Other attributes of the *EvipLbe* object must be specified on creation.

3.2.7 Configure an FEE

FEEs are 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 (see Section 3.2.1 Create an ALB on page 17) also apply for FEE names.

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

Other attributes of the *EvipFee* object must be specified on creation.

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

Any number of *EvipRoutingSetup* objects can be configured under the FEE. 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 provided.

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

Only one `EvipRoutingSetup` object of each individual routing protocol can be configured in one FEE.

If Virtual LAN (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.

3.2.8 Configure a Supervised Remote Gateway

Several `EvipSupervisedRemoteGateway` objects 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 only 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.

The alarm supervised availability of the remote gateway in this context is based on that the configured address (`EvipSupervisedRemoteGateway`) is used in the FEE as a next-hop gateway address, indicating a default destination. With OSPF, this next-hop gateway address is announced by the Area Border Router (ABR) and received by the FEE.



Note: With OSPFv3, this configured next-hop gateway address is the IPv6 link-local address of the interface of the gateway router connected to the FEE.

3.2.9 Configure OSPFv2 Routing

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

Note: Only one `EvipRoutingSetup` object of OSPV2 Routing can be configured in one FEE.

The parameters are the following:

local_address	Local address for the interface that communicates with the remote router.
router_id	Router ID for the OSPF process. Must be unique for each OSPF instance.
area	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	Area type to be used; <code>stub</code> and <code>nssa</code> are supported in OSPFv2.
hello_interval	Interval between <code>hello</code> packets. Defaults to 10 seconds.
dead_interval	Interval during which no <code>hello</code> packets are received and after which a neighbor is declared dead. Defaults to 40 seconds.
retransmit_interval	Time between Link-State Advertisement (LSA) retransmission for adjacencies belonging to the interface. Defaults to 5 seconds.
router_priority	Router priority to determine the designated router for the network. Defaults to 0, which means that the FEE is <code>DRother</code> .
spf_delay	Shortest Path First (SPF) minimum hold time in milliseconds between two SPF calculations. Defaults to 500 milliseconds.
spf_interval	Maximum delay in milliseconds between receiving a change to the SPF calculation. The value must be higher than <code>spf_delay</code> . Defaults to 1000 milliseconds.



transmit_delay Estimated time it takes to transmit a link state update packet on the interface. Defaults to 1 second.

3.2.10 Configure OSPFv2 Routing with BFD Supervision

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`.

Note: Only one `EvipRoutingSetup` object of OSPVv2 Routing with BFD Supervision can be configured in one FEE.

The parameters are the following:

echo If echo is `on`, BFD echo packets are used in addition to the usual BFD control packets.

echo_interval BFD echo interval. If it is not set, but the `echo` parameter is `on`, the default value 20 milliseconds is used.

slow_timer BFD slow timer interval.

bfd_interval Transmit interval, in milliseconds.

minrx Reception interval, in milliseconds.

multiplier Integer value of the `hello` multiplier.

3.2.11 Configure OSPFv3 Routing

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

Note: Only one `EvipRoutingSetup` object of OSPVv3 Routing can be configured in one FEE.

The parameters are the following:

local_address Local address for the interface that communicates with the remote router. This is an optional parameter, but it must be set it to allow ICMPv6 to work. It must be routable, for example not a link-local address, and must not be a VIP address.

router_id Router ID for the OSPF process. Must be unique for each OSPF instance.

area 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	Area type to be used; <code>stub</code> and <code>nssa</code> are supported in OSPFv3.
hello_interval	Interval between <code>hello</code> packets. Defaults to 10 seconds.
dead_interval	Interval during which no <code>hello</code> packets are received and after which a neighbor is declared dead. Defaults to 40 seconds.
retransmit_interval	Time between LSA retransmission for adjacencies belonging to the interface. Defaults to 5 seconds.
router_priority	Router priority to determine the designated router for the network. Defaults to 0, which means that the FEE is <code>DRother</code> .
spf_delay	SPF minimum hold time in milliseconds between two SPF calculations. Defaults to 500 milliseconds.
spf_interval	Maximum delay in milliseconds between receiving a change to the SPF calculation. The value must be higher than <code>spf_delay</code> . Defaults to 1000 milliseconds.
transmit_delay	Estimated time it takes to transmit a link state update packet on the interface. Defaults to 1 second.

3.2.12

Configure OSPFv3 Routing with BFD Supervision

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

Note: Only one `EvipRoutingSetup` object of OSPFv3 Routing with BFD Supervision can be configured in one FEE.

The parameters are the following:

echo	If <code>echo</code> is <code>on</code> , BFD echo packets are used in addition to the usual BFD control packets.
echo_interval	BFD echo interval. If it is not set, but the <code>echo</code> parameter is <code>on</code> , then the default value 20 milliseconds is used.
slow_timer	BFD slow timer interval.
bfd_interval	Transmit interval, in milliseconds.
minrx	Reception interval, in milliseconds.



multiplier Integer value of the `hello` multiplier.

3.2.13 Configure Static Routing with BFD Supervision for IPv4

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

Note: Only one `EvipRoutingSetup` object of Static Routing with BFD Supervision for IPv4 can be configured in one FEE.

The parameters are the following:

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

3.2.14 Configure Static Routing with BFD Supervision for IPv6

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

Note: Only one `EvipRoutingSetup` object of Static Routing with BFD Supervision for IPv6 can be configured in one FEE.

The parameters are the following:

local_address	Local address, which must be in IPv6 format, for the interface that communicates with the remote router.
gateway	IP address, which must be in IPv6 format, to the remote gateway.



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

3.2.15 Configure Static Routing for IPv4

The static IPv4 parameters are defined under `Evip=1, EvipAlbs=1, EvipAlb=<ALB_name>, EvipFees=1, EvipFee=<FEE_name>, EvipRoutingSetup=static`.

Note: Only one `EvipRoutingSetup` object of Static Routing for IPv4 can be configured in one FEE.

The parameters are the following:

local_address	Local address, which must be in IPv4 format, for the interface that communicates with the remote router.
gateway	IP address, which must be in IPv4 format, to the remote gateway.

Note: Static routing configuration enables the feature IP stacking, refer to *eVIP Internetworking*.

3.2.16 Configure Static Routing for IPv6

The static IPv6 parameters are defined under `Evip=1, EvipAlbs=1, EvipAlb=<ALB_name>, EvipFees=1, EvipFee=<FEE_name>, EvipRoutingSetup=static6`.

Note: Only one `EvipRoutingSetup` object of Static Routing for IPv6 can be configured in one FEE.

The parameters are the following:

local_address	Local address, which must be in IPv6 format, for the interface that communicates with the remote router.
gateway	IP address, which must be in IPv6 format, to the remote gateway.



Note: Static routing configuration enables the feature IP stacking, refer to *eVIP Internetworking*.

3.3 Traffic Configuration

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

This section describes the following:

- Configure a target pool
- Configure a flow policy
- Configure a selection policy
- Configure an XFRM ALB selection policy
- Configure an IPsec tunnel

3.3.1 Configure a Target Pool

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 can be modified in a *EvipTargetPool* object. To modify the other attributes, the target pool must be deleted and recreated with the updated parameters.

The target pool defines the nodes where the traffic handling application executes. 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 *distributionMethod*. It defines the algorithm used for distributing network traffic among the nodes in the target pool. The LBE is implemented using the standard IPVS.

The available *distributionMethods* 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 IPVS and the distribution method in IPVS, refer to [IPVS documentation](#).

Attribute `stickyGroup`

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.
- The ALB in which the target pool resides cannot use 5-tuple hashing.

Attribute `stickinessTimeout`

The `stickinessTimeout` attribute defines the time, in seconds, that a connection is remembered after the traffic flow stopped and the configurable IPVS timer also has expired. The following are the IPVS timers:

- `ipvs_tcp_timeout` for silent but unfinished TCP sessions
- `ipvs_tcpfin_timeout` for finished TCP sessions
- `ipvs_udp_timeout` for silent UDP pseudo sessions

During `stickinessTimeout` and the additional IPVS time-out, subsequent connections from the source IP go to the same node. The maximum value for `stickinessTimeout` is 2,592,000 seconds (30 days). `stickinessTimeout` is protocol-agnostic. If the `stickyGroup` attribute is set, `stickinessTimeout` is implicit; a default of 360 seconds (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 Configure a Flow Policy

Flow policies 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.

Note: The *EvipFlowPolicy* objects cannot be modified once created. The attributes of the object must therefore be specified on creation

Only 15 distinct ports or port ranges can map to one sticky target pool. This is because of a hard limit in the `iptables`. Flow policies that specify ports that are in consecutive order are converted to consecutive order. For example, if three flow policies exist 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.

Attribute *dest*

The *dest* attribute in a flow policy specifies the destination (local) IP selected by this flow policy for incoming traffic.

The *dest* attribute value can be specified in either of the following textual formats:

- Explicit VIP address format, for example, `10.1.1.1` or `2007::1`
- Named VIP address format (a chosen name)

Note: The *dest* attribute must be configured in the same format as the VIP address is configured in the *EvipVip* object.

For a VIP address configured as an explicit VIP address in the *EvipVip* object, the corresponding *dest* attribute in the flow policy must also use the explicit address format.

For a VIP address configured according to the named VIP address format in the *EvipVip* object, the corresponding specified name of the VIP address must be used in the *dest* attribute of the flow policy to refer to this VIP address.

Also, if the *dest* attribute is configured with the name of a named VIP address once the IP address provided its value in attribute *address* of the *EvipVip* object for the configured named VIP address, this destination IP address is resolved internally. This provides some flexibility, because if an IP address in



the corresponding VIP object is deleted or modified, the flow policy is updated accordingly without deleting it.

Attribute `usageState`

The `usageState` attribute signifies whether this flow policy is `ACTIVE`, that is, deployed, or `IDLE`, that is, `dest` is not resolved yet, as the IP address must be provided in the corresponding VIP object.

3.3.3 Configure a Selection Policy

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

Note: The *EvipSelectionPolicy* objects cannot be modified once created. The attributes of the object must therefore be specified on creation.

Selection policies, (also called ALB selection policies) can only be used 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 selection policies are applied in a strict order. The *EvipSelectionPolicy* object has a mandatory `sortorder` attribute that must be a floating point number. The selection policies are then applied in order lowest to highest.

A good example of an ALB selection policy use case is how SCTP deploys eVIP along with Signalling System No.7 (SS7) Common Application Feature (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 Configure an XFRM ALB Selection Policy

XFRM ALB selection policies are configured under `Evip=1, EvipXfrmSelectionPolicies=1`. The XFRM selection policy defines the ALB where the `ip xfrm` command is applied, as shown in Example 2. 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`.

The attributes are as follows and are further described in *Managed Object Model evip_cm*.

payloadNode	Optional integer.
evipXfrmSelectionPolicyId	Mandatory string, name of the policy.
alb	Mandatory string.
process	Optional string.
env	Optional string.
	Note: Different policies must have a different name.
order	Optional real number.
storage	Optional string.
default	String.

```
(config-EvipXfrmSelectionPolicies=1)>show -r
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 configuration shows that, by default, security policies and security associations are expected to go to `alb_1` and must have scope `alb`.

3.3.5 Configure an IPsec Tunnel

IPsec tunnels are configured under `ManagedElement=1, Transport=1, Host`.



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

To configure an IPsec tunnel:

1. Enable the IKE daemon by setting the `EvipParams` parameter `ike_enabled` to `yes`.
2. Set up certificates in the ECLI, if needed.

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

3. Create the tunnel configuration using the ECLI, see Section 3.3.5 Configure an IPsec Tunnel on page 32.
4. Install preshared keys in the ECLI, if needed.

If preshared keys are used, they must be installed after the corresponding `Ikev1Session` or `Ikev2Session` was committed.

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

3.3.5.1

Create an IPsec Tunnel

IPsec tunnels (Encapsulating Security Payload (ESP)) can be defined 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 *IpssecTunnel* object.

To create IPsec tunnels in the desired ALB, first enter Config mode in the ECLI and navigate to the corresponding *Host* object under *Transport*, for example:

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

The following is a basic example configuration for a tunnel with IKEv2 (to simplify, the command prompts are not shown):

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

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

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=1
ikev2PolicyProfile="ManagedElement=1,Transport=1,⇒
Host=eVIP_ALB_alb_1,Ikev2PolicyProfile=1"
```

Example 4 Configuring a Tunnel with IKEv2

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

For preshared key authentication, the user must install the key under the corresponding IKE session object (*Ikev1Session* or *Ikev2Session*) with the action *installPreSharedKey* for *Ikev1Session* and *installPreSharedKey* for *Ikev2Session*.



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 *IpsecPolicy* object. Using more than one *IpsecPolicy* object under the same *IpsecTunnel* object is not supported.