

eVIP on LSB Management Guide

Evolved Virtual IP

USER GUIDE

Copyright

© Ericsson AB 2015. 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	10
3	Configuration	13
3.1	General eVIP Configuration	13
3.2	ALB Configuration	17
3.3	Traffic Configuration	25
3.4	Configuration and Validation	30
3.5	eVIP CLI Framework	30





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.

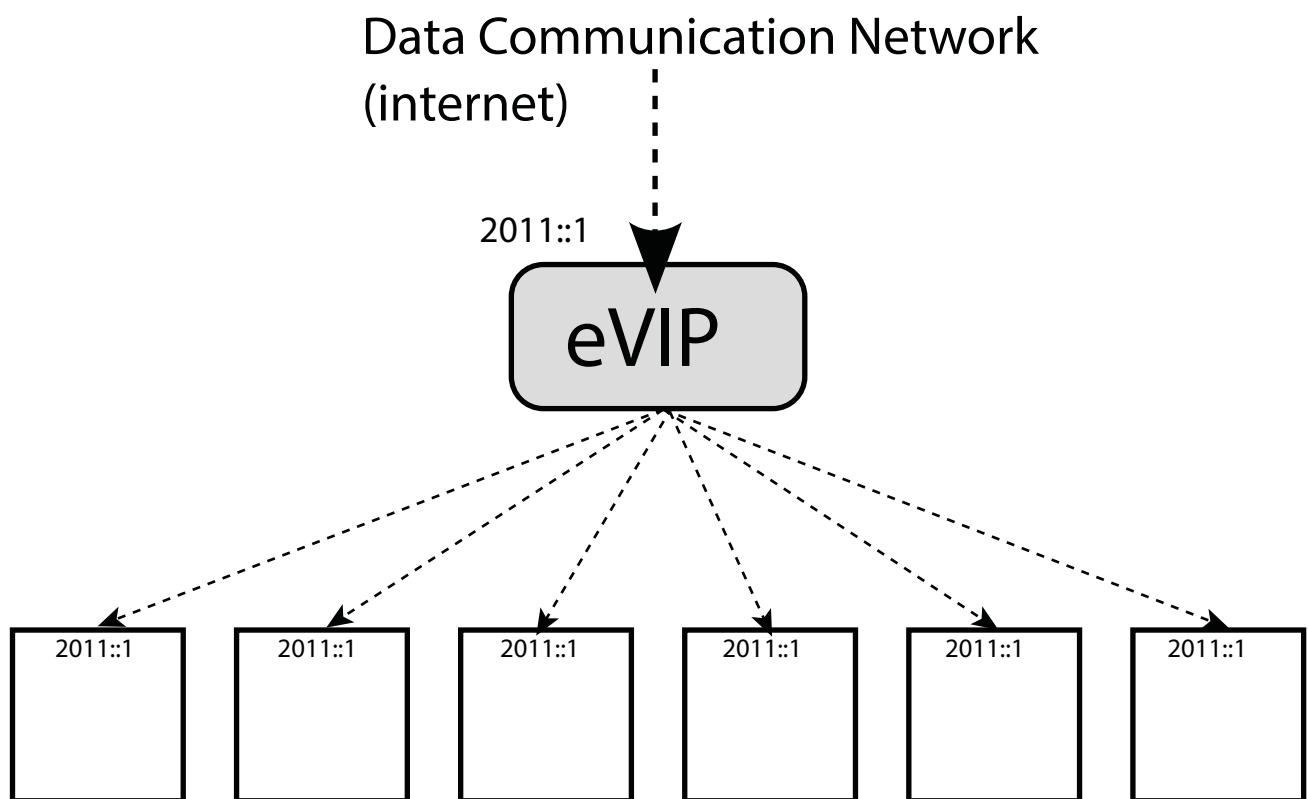


Figure 1 eVIP Load Balancer

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.



1.1 Prerequisites

Ensure that the following documents have been read:

- *eVIP on LSB Internetworking*

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

- *eVIP on LSB System Architecture Description*



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. VIP 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.

The VIP 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.

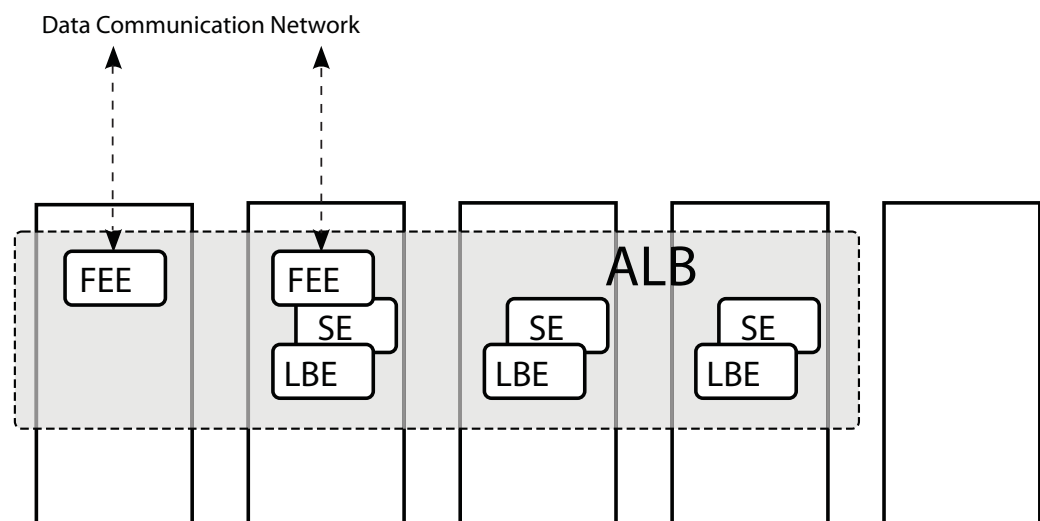


Figure 2 eVIP Abstract Load Balancer

An ALB consists of several Front-End Elements (FEEs), some Load Balancer Elements (LBEs), and some Security Elements (SEs). The ALB software is executing in Linux[®] containers to separate the load balancing functionality



from the applications. For more information about Linux containers, 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.

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. See Section 3 on page 13 for other supported routing configuration options.

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 Load Balancer Element

A requirement on eVIP is that it must 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 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.

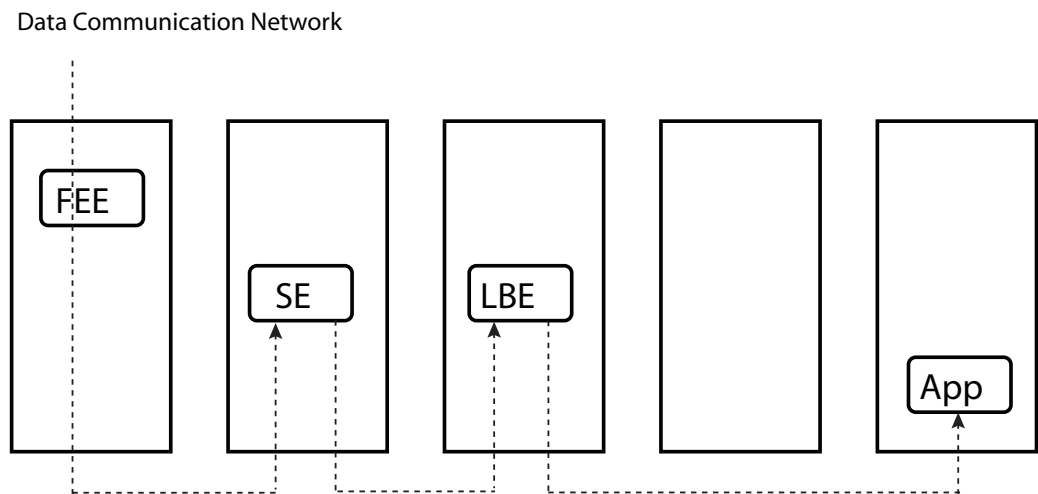


Figure 3 eVIP SE and LBE - Extra Hop for Network Packets

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.

2.1.4 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.

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

- 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 automatically is relocated to another active processing unit in the case of deletion 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 that makes it possible to pin eVIP elements to processing hardware with special properties, such as physical ports.

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 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 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 deletion. 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 on LSB Internetworking*.

eVIP maintains a strict separation between designated and undesigned processing units; never allowing floating eVIP nodes to “invade” 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 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 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 eVIP 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.
 - c. Remove the corresponding fixed eVIP node.
 - d. Add an eVIP node as floating (`distribution=floating`) with a priority.
 - e. Configure eVIP elements (FEE, SE, LBE) as needed.
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 adjust automatically adjust scaling so 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

ALB selection policy is a configuration device that is used when the application design (at design time) has not enforced the binding of sockets to VIP addresses.

For application software, which always binds sockets explicitly to a VIP address at the socket API level, there is no need to configure an ALB selection policy. The selection of ALB is automatic under this circumstance.

2.1.9 XFRM ALB Selection Policy

When eVIP is present, some of the standard Linux commands are directed to an ALB. To calculate the appropriate ALB for a given command the XFRM ALB selection policy provides a means that is transparent and backward compatible from a user application point of view.

The XFRM ALB selection policy defines the ALB where the `ip xfrm` command applied.

For a finer grained (per socket) ALB selection or eVIP bypass (local stack processing) the destination where XFRM operations take effect can be also specified by binding the netlink/XFRM socket to one of the following:

- An ALB device using the `SO_BINDTODEVICE` socket option.
- A loop-back device `lo` using the `SO_BINDTODEVICE` socket option.

For this to work, unless bound to loop back XFRM ALB, selection policy is still required.

In Example 1, policy `control` has a relatively low order and applies to `alb_0`. Value `control` is not default. The policy is transformed to the value of a Linux environment parameter named `ENV_ALB0`.

```
xfrm_alb_selection_policy name="control" alb="alb_0"
order="0.100000" default="no" storage="alb" env="ENV_ALB0"
```

Example 1 XFRM ALB Selection Policy

If no binding is done, the ALB selection works as before, so the line in Example 2 can be interpreted as the line in Example 1. However, for sockets bound to device `alb_1`, the meaning of this line is that policy `xasp_alb1` defines the storage class to be `alb`.

Matching socket binds and policies is achieved by setting `env` and `alb` parameters to the same ALB device name. Defining the storage class is the only reason why the XFRM ALB selection policy is still required if a netlink/XFRM socket is bound to an ALB.



```
xfrm_alb_selection_policy name="xasp_alb1" alb="alb_1"
order="2.000000" default="no" storage="alb" env="alb_1"
```

Example 2 XFRM ALB Selection Policy Matching Sockets Bound to alb_1

Possible parameters are as follows:

payload_node	Optional integer, denoting the payload node. 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. Note: Different policies must have different name.
alb	Mandatory string, the target ALB
process	Optional string. The name of the process to which this policy maps.
env	Mandatory string if netlink/XFRM socket is bound to an ALB device using <code>SO_BINDTODEVICE</code> . Set the name of the ALB device that the socket is bound to, which is the same as the <code>alb</code> field. The <code>EVIP_XALB</code> environment variable is ignored in this case. This is an optional string, if no binding is done. Set the <code>EVIP_XALB</code> environment variable to this value to use this policy.
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. How IPsec objects are stored. <code>local</code> = use local Linux kernel (not available if socket is bound to an ALB), <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. Note: Order is still taken into account.

The Example 3 show the selection policy command.



```
<xfrm_alb_selection_policy name="control" alb="alb_0" \
  order="0.100000" default="no" storage="alb" env="ENV_ALB0"/>
<xfrm_alb_selection_policy name="payload" alb="alb_1" \
  order="1.000000" default="yes" storage="alb"/>
```

Example 3 Selection Policy Command

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`.

2.1.10 Tunnel Configuration

IKE-supervised tunnels are not supported on LSB.

The `ike_enabled` parameter must be set to `no` in the `evip.xml`.

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 can be deleted or created. Since deleted objects are recursive, verify that they are to be deleted before actually deleting them.

Configuration of eVIP is divided into the following two parts:

- Static configuration using an XML file
- Dynamic configuration using the eVIP Command-Line Interface (CLI)

2.2.1 Configuration Feedback

Feedback on the configuration is available on both the XML file and the CLI.

XML File

For static configuration, the configuration XML file can be validated using the XML schema defined in `evipconf.xsd`. This confirms that there are no syntax errors and that the configuration is not missing any mandatory parameters.

CLI

For configuration changes through the CLI, the CLI either returns that the change was successful or that it failed.

The return values are the following:

- Configuration success: `CLI_OK`



- Configuration failed: `CLI_ERROR`

When a configuration command has failed, the CLI (in most cases) prints an error message. For example, trying to change an ALB that does not exist returns: `Alb does not exist` followed by `CLI_ERROR`.

2.2.2 Actions

Activation and deactivation of ALBs can be performed through the eVIP CLI. Activation is done by default at startup when configuring ALBs in the eVIP configuration XML file.

2.2.3 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.





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

Note: The configurations are made through the eVIP configuration XML file.

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

In the eVIP configuration XML file, commands are defined in the `command_definition` tag as shown in Example 4.

```
<command name="all_evip_nodes">echo "TestCommand"</command>
```

Example 4 Specifying a Startup Command



The name of the command is used as a reference when the command is referenced in other places in the eVIP configuration.

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.

3.1.2 Define Cluster

In the eVIP configuration XML file, the cluster is defined in the `cluster` tag.

The eVIP Cluster object contains a `primary_interface` attribute. The primary interface is used for all eVIP traffic between the nodes.

The `primary_interface` is redundant in some way, for example with bonding.

The cluster can consist of some fixed or floating nodes. For fixed nodes, there is a strict mapping between `hostname` and the eVIP internal `nodeID`. Floating nodes do not have `hostnames` defined in the eVIP configuration XML file.

Example 5 shows the defined cluster of fixed nodes.

```
<cluster primary_interface="bond0">
  <node id="1" hostname="SC-2-1"/>
  <node id="2" hostname="SC-2-2"/>
  <node id="3" hostname="PL-2-3"/>
</cluster>
```

Example 5 Defining Cluster of Fixed Nodes

Example 6 shows the defined cluster of floating nodes.

```
<cluster primary_interface="bond0">
  <node id="1" distribution="floating" float_priority="4"/>
  <node id="2" distribution="floating" float_priority="4"/>
  <node id="3" distribution="floating" float_priority="6"/>
  <node id="4" distribution="floating" float_priority="8"/>
</cluster>
```

Example 6 Defining Cluster of Floating Nodes

Example 7 shows the defined cluster of fixed and floating nodes.



```
<cluster primary_interface="bond0">
  <node id="1" hostname="SC-2-1"/>
  <node id="2" hostname="SC-2-2"/>
  <node id="3" hostname="PL-2-3"/>
  <node id="4" distribution="floating" float_priority="2"/>
  <node id="5" distribution="floating" float_priority="2"/>
  <node id="6" distribution="floating" float_priority="4"/>
</cluster>
```

Example 7 Defining Cluster of Fixed and Floating Nodes

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 happen.

The `startup` commands defined under the `cluster` tag 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 commands are listed under the `commands_for_all_undesignated` tag. For more information, refer to *eVIP on LSB - XML*.

3.1.3 Configure eVIP Parameters

In the eVIP configuration XML file, the eVIP `syslog` parameters are configured in the `syslog` tag as shown in Example 8.

```
<syslog facility="3" log_level="6"/>
```

Example 8 Configuring eVIP Parameters

Regarding `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 `log_level` is set to 10, it starts storing extended debug information. The extended debug information is stored in directory `/var/log/`.

The `syslog` level can also be changed using the CLI. For more information, see Section 3.5.1.3 General Configuration Mode on page 33.

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 `facility`, the `value` attribute of this object is a number in the value range 0–7. 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 parameters are internal and care must be taken when modifying them: `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 required to modify the MTU parameters, caution must be taken regarding 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 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 planes that is in hot standby state, continuously receiving system state information from the active control plane. Default value is 1 since 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` may only be seen after a cluster reboot.

3.1.4 Port Ranges

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 must not be specified. Port ranges are only configured for applications that have special needs. Normally no port ranges are specified. The default 32k wellknown and 32k ephemeral ports are OK for most applications.

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

The following constraints exist on the port ranges:

- 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.

Port ranges can be defined under the `port_ranges` tag, as shown in Example 9.

```
<port_ranges>
  <wellknown protocol="tcp">1-2047</wellknown>
  <wellknown protocol="udp">1-49151</wellknown>
</port_ranges>
```

Example 9 Port Ranges

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 are configured in the `albs` tag in the eVIP configuration XML file.

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, as shown in Example 10.

```
<alb name="alb1">
  ...
</alb>
```

Example 10 Naming of ALB

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.

In the eVIP CLI, command `show albs` list status of the ALBs in the system.

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.

3.2.1 ERSIP Parameters

Some parameters for the ALB can be defined under the `ersip_params` tag, as shown in Example 11.

```
<ersip_params lo_watermark="30"  
             hi_watermark="60"  
             timeout_requested_resources="30"/>
```

Example 11 ERSIP Parameters

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 the `vips` tag in the eVIP configuration XML file, as shown in Example 12.

```
<vip address="2007::1"/>  
<vip address="192.168.66.15"/>
```

Example 12 VIP Configuration

Multiple VIP addresses can be defined for an ALB. For a description of the attributes, refer to *XML Configuration Example with Comments*.

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



3.2.3 LBE Configuration

Several LBEs must be configured for the system. In the eVIP configuration XML file this is done under the `lbes` tag, as shown in Example 13.

```
<lbe node="1" name="lbe_name1"/>
```

Example 13 LBE Configuration

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

It is also possible to specify a range of LBEs together with a `name_template` attribute. eVIP then generates unique names for the LBEs in the specific range, as shown in Example 14.

```
<lbe_range node="1-9" name_template="lbe_name"/>
```

Example 14 Range of LBEs with name_template

3.2.4 SE Configuration

The SE configuration is the same as the LBE configuration, see Section 3.2.3 on page 18. A name and a node ID the identification number of the blade is required, nothing else.

The naming restrictions for ALB names also apply for SE names. For more information, see Section 3.2 ALB Configuration on page 17.

3.2.5 FEE Configuration

In the eVIP configuration XML file, several FEEs can be configured under the `fees` tag, as shown in Example 15.

```
<fee node="1" name="fee_name" external_interface="bond0.207">
  ...
</fee>
```

Example 15 FEE Configuration

The FEE must be given a unique name and the node where the FEE is located is specified in the mandatory `nodeID` 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.

Command `show fees` in the CLI gives the operational state of the FEE.

Any number of routing setups can be defined under the FEE. The name of the `routing setup` tag 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.

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 used, IPv6 must not be configured in the FEE.

3.2.6 OSPFv2 Configuration

The OSPFv2 parameters are defined as shown in Example 16.

```
<ospfv2 local_address="192.168.2.20/29"
  router_id="192.168.1.1"
  gateway="192.168.1.11"
  area="1"
  area_type="stub"
  hello_interval="10"
  dead_interval="40"
  retransmit_interval="5"
  router_priority="0"
  transmit_delay="1"/>
```

Example 16 OSPFv2 Parameters

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 used 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	Use this 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	Use this 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 DRother.
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.

3.2.7 OSPFv2 and BFD Configuration

The OSPFv2 and Bidirectional Forwarding Detection (BFD) parameters are defined as shown in Example 17.

```
<bfd_ospfv2 echo="on"
  echo_interval="100"
  slow_timer="1000"
  bfd_interval="1000"
  minrx="100"
  multiplier="5"/>
```

Example 17 OSPFv2 and BFD Parameters

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.

3.2.8

OSPFv3 Configuration

The OSPFv3 parameters are defined as shown in Example 18.

```
<ospfv3 local_address="2001:db8::1"
  router_id="225.1.1.10"
  area="2"
  area_type="stub"
  hello_interval="10"
  dead_interval="40"
  retransmit_interval="100"
  router_priority="0"
  transmit_delay="1"/>
```

Example 18 OSPFv3 Parameters

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 must set to allow ICMPv6 to work. It has to be routable, for example not a link-local address, and 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 OSPF area type to be used; <code>stub</code> area must be specified here as it is currently the only supported area type.
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 DR_{Other}.

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.

3.2.9**OSPFv3 and BFD Configuration**

The OSPFv3 and BFD parameters are defined as shown in Example 19

```
<bfd_ospfv3 echo="on"
  echo_interval="100"
  slow_timer="1000"
  bfd_interval="1000"
  minrx="100"
  multiplier="5"/>
```

Example 19 OSPFv3 and BFD Parameters

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 `hello` multiplier.



3.2.10 Static BFD IPv4 Configuration

The static BFD IPv4 parameters are defined as shown in Example 20:

```
<bfd_static local_address="192.168.71.38/30"  
  gateway="192.168.71.37"  
  echo="no"  
  bfd_interval="300"  
  minrx="300"  
  multiplier="5"/>
```

Example 20 Static BFD IPv4 Parameters

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 hello multiplier.

3.2.11 Static BFD IPv6 Configuration

The static BFD IPv6 parameters are defined as shown in Example 21.

```
<bfd_static local_address="2dec::102:8:2/112"  
  gateway="2dec::102:8:1"  
  echo="no"  
  bfd_interval="300"  
  minrx="300"  
  multiplier="5"/>
```

Example 21 Static BFD IPv6 Parameters

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	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.

3.3 Traffic Configuration

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

3.3.1 Target Pool Configuration

Target pools consist of a set of fixed nodes and a set of floating nodes. A mix of fixed and floating nodes in the same target pool is allowed.

Target pools consisting of fixed nodes are configured in the `target_pools` tag as shown in Example 22.

```
<target_pool name="sticky-target3"
  distribution_method="round_robin"
  sticky_group="yes">
  <payload node="1-8"/>
</target_pool>
```

Example 22 Target Pool Configuration

Specific floating nodes cannot explicitly be assigned to target pools. The `all_undesignated` tag represents existing set of floating nodes.

Target pool consisting of floating nodes is shown in Example 23.



```
<target_pool name="ipv6-tp"
  distribution_method="round_robin"
  all_undesignated="yes"
  sticky_group="no"
  udp_stateless="no">
</target_pool>
```

Example 23 Target Pool Configuration with Floating Nodes

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.

The target pool objects cannot be modified once created. So the attributes of the object must be specified on creation.

The target pool defines the nodes where the traffic handling application is executing. 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.

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

The target pool can have the `sticky_group` 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 a HTTP session consisting of multiple connections. However, `sticky_group` 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 `sticky_group` 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.

`stickiness_timeout` 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 `stickiness_timeout` and the additional IPVS timeout subsequent connections from the source IP go to the same node. The maximum value for `stickiness_timeout` is 2592000 (30 days). `stickiness_timeout` is protocol agnostic. If `sticky_group` is set `stickiness_timeout` is implicit; a default of 360 (6 minutes) is used if `stickiness_timeout` 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 still needs to include 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 the `flow_policies` tag as shown in Example 24.

```
<flow_policy
  name="ssh"
  protocol="tcp"
  target_pool="sticky-target3"
  dest="30.0.0.1"
  dest_port="22"/>
```

Example 24 Flow Policy Configuration

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 flow policies cannot be modified once they are created, therefore the attributes of the object must 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 `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 range 80–82. 15 such ranges (or individual ports) on one sticky target pool is the maximum.

3.3.3 ESP Traffic Configuration

SE is used as a default value for parameter `esp_termination` for determining where ESP traffic is decrypted.

Value `payload` in parameter `esp_termination` forwards the incoming ESP traffic, by the eVIP SE elements, to be decrypted on the payload node.

A requirement for having ESP termination on the payload is that a flow policy for UDP port 500 is added for each VIP that is to support ESP. This flow policy must be directed to the same, sticky, target pool as the non-ESP traffic for this VIP.

Example 25 shows a configuration where non-ESP traffic on port 5101 for both UDP and TCP to a sticky target pool is forwarded to the payload node. With the addition of the `esp-tp` flow policy, it also forwards the ESP traffic to the payload node.

```
<target_pool name="ipv4-sticky-tp" distribution_method="round_robin"
  sticky_group="yes" udp_stateless="no" stickiness_timeout="360">
  <payload node="1"/>
  <payload node="2"/>
  <payload node="3"/>
  <payload node="4"/>
</target_pool>

<flow_policy name="udp-fp" address_family="ipv4" protocol="udp"
  target_pool="ipv4-sticky-tp" dest="10.0.0.1" dest_port="5101"/>
<flow_policy name="tcp-sbg" address_family="ipv4" protocol="tcp"
  target_pool="ipv4-sticky-fp" dest="10.0.0.1" dest_port="5101"/>
<flow_policy name="esp-tp" address_family="ipv4" protocol="udp"
  target_pool="ipv4-sticky-fp" dest="10.0.0.1" dest_port="500"/>
```

Example 25 Forwarding ESP Traffic to Payload Node

Note: Parameter `esp_termination` can only be set when an ALB configuration is added, and before the ALB configuration is activated.

3.3.4 Selection Policy Configuration

Selection policies are configured under the `alb_selection_policies` tag as shown in Example 26 and Example 27.

The selection policies objects cannot be modified once they are created, therefore the attributes of the object must be specified on creation.

The purpose of a selection policy is to direct traffic to a specific ALB in cases where the application does not provide enough configuration possibilities, for example, for third party products.

The selection policies are applied in a strict order. The `SelectionPolicy` object has a mandatory `order` attribute which must be a floating point number. The selection policies are then applied in order lowest to highest.



The identified specific cases when a selection policy must be configured are the following:

- Protection of backplane networks.
- SCTP deployment with eVIP and SS7 CAF.

Example 26 shows a selection policy for isolating the backplane network 127.0.0.0/8 from any External Network.

```
<alb_selection_policy
  name="asp_exclude_local"
  alb=""
  dest="127.0.0.0/8"
  order="5.1">
</alb_selection_policy>
```

Example 26 Backplane Protection

Note: The `alb` parameter must be configured with an empty string symbol as shown in Example 26.

The `sortorder` parameter value can be assigned a floating point number.

This number governs the priority order by which the selection policy is applied.

The use of floating point numbers allows for flexible insertion of new selection policies. A “sortorder” number without decimal point is automatically converted into a floating point number.

Example 27 show the SCTP deployment with SS7 CAF.

```
<alb_selection_policy
  name="ss7_policy"
  alb="ln_ss7sig_sc"
  process="fe_sctp"
  order="0.100000">
</alb_selection_policy>
```

Example 27 SCTP Deployment with SS7 CAF

Note: The `process` parameter is configured with a Linux process name in Example 27. This process name is `fe_sctp`.

An application that by design does not enforce binding to VIP addresses must provide instructions for configuring the appropriate ALB selection policy, stating the arguments to be used for the ALB selection.



Note: Installation documents for specific applications (or application bundles) that at design time does not enforce binding to a VIP address at the API level must provide detailed instructions, which cover the needed parameters for the specific application, and instructions on how to configure the required ALB selection policies.

For examples of different matching attributes, refer to *XML Configuration Example with Comments*.

3.4 Configuration and Validation

For a complete configuration example with comments and XML schema used to validate the XML files, refer to *XML Configuration Example with Comments* and *Configuration Validation Schema*.

3.5 eVIP CLI Framework

The eVIP CLI framework is a fully featured Telnet based CLI with command-line completion, history, and hierarchical commands. The library is based on libcli by David Parrish, but was rewritten to support non-blocking, single-threaded operation.

eVIP publishes CLI on the active controller node (where `evipc` executes). The address to the CLI can be found by using the `getactivecontrol` utility. The CLI is reached with Telnet on port 25190, see Example 28.

Note: Currently, `getactivecontrol` only works on nodes that have been configured to be part of a target pool.

```
/ # /opt/vip/bin/getactivecontrol
fe80::200:ff:feff:1%evip_macvlan0
/ # telnet fe80::200:ff:feff:1%evip_macvlan0 25190
...
EVIP> show evip-status
STATUS OK
```

OK

Example 28 Checking of eVIP Status

For more information about libcli, refer to the following URLs:

- [Users Guide - libcli](#)
 - [libcli](#)
 - [GNU Lesser General Public License](#)
- This is the original libcli.



libcli is distributed under LGPL.

- [Modified libcli](#)

3.5.1 Structure and Syntax

This section describes the structure and syntax of the commands available on the eVIP CLI.

The **show** command can, on many levels be used with several arguments that are easily accessed by pressing **TAB**.

For the general commands **show**, **add**, **remove**, and **set**, a subcommand principle is used.

The **show agents** command accepts optional switches. If switches are used, they must be specified before the `<alb_name>` argument. Switches can be combined in any order.

Supported switches are the following:

- **-s, --short-states**

Show only the first letters of the agent state, for example ACTIVE UP shown as AU.

- **-c, --compact-table**

Show agent address only once in each line.

- **-w <N>, --width <N>**

Set width of the agent table to *N* characters. The default width is 100.

3.5.1.1 Top Level

Top levels are shown in Table 1.

Table 1 Top Level

Access Command	None (Top level)	
Prompt	eVIP>	
Privileged Mode	No	
Show Information	Command show shows information about the configuration	
Related Commands	Description	Required Arguments
albs	Shows ALBs	



sel-policies	Shows (ALB) selection policies	
status	Shows status	
agents	Shows agent statuses	<code><alb_name></code>
flow-policies	Shows flow policies	<code><alb_name></code>
startup-command	Shows startup commands for nodes	<code>[node <node_id> all-undesigned]</code>
target-pools	Shows target pools	<code><alb_name></code>
vips	Shows VIPs	<code><alb_name></code>
ports	Shows ports for VIP	<code>alb <alb_name></code> <code>vip <vip></code>
ports-alloc	Shows ports allocation for VIP	<code>alb <alb_name></code> <code>vip <vip></code>
evipcs	Shows eVIP-Cs	
evip-status	Shows eVIP status	
syslog-level	Shows log level	
avail-services	Shows available services	
transactions	Shows transactions	<code><alb_name></code>
f-transactions	Shows factory transactions	
keys	Shows key distribution	<code><alb_name></code>
pn-pairs	Shows peers of a repdb	<code><alb_name></code>
hosts	Shows hostname of each node	

3.5.1.2

Enabled Mode

The enable mode shown in Table 2.

Table 2 Enabled Mode

Access Level	Top Level	
Access Command	enable	
Prompt	EVIP#	
Privileged Mode	Yes	
Show Information	Command show shows information about the configuration	
Related Commands	Description	Required Arguments



albs	Shows ALBs	
sel-policies	Shows (ALB) selection policies	
status	Shows status	
agents	Shows agent statuses	<alb_name>
flow-policies	Shows flow policies	<alb_name>
startup-command	Shows startup commands for nodes	[node <node_id> all-undesigned]
target-pools	Shows target pools	<alb_name>
vips	Shows VIPs	<alb_name>
ports	Shows ports for VIP	alb <alb_name> vip <vip>
ports-alloc	Shows ports allocation for VIP	alb <alb_name> vip <vip>
evipcs	Shows eVIP-Cs	
evip-status	Shows eVIP status	
syslog-level	Shows log level	
avail-services	Shows available services	
transactions	Shows transactions	<alb_name>
f-transactions	Shows factory transactions	
keys	Shows key distribution	<alb_name>
pn-pairs	Shows peers of a repdb	<alb_name>
hosts	Show hostname of each node	
Other Command	Description	
save-config	Saves the running configuration to persistent storage	

3.5.1.3

General Configuration Mode

The general configuration mode shown in Table 3.

Table 3 General Configuration Mode

Access Level	Enabled Mode
Access Command	configure terminal
Prompt	EVIP(config)#



Privileged Mode	Yes	
Show Information	Command show shows information about the configuration	
Related Commands	Description	Required Arguments
albs	Shows ALBs	
sel-policies	Shows (ALB) selection policies	
startup-command	Shows startup commands for nodes	[node <node_id> all-undesigned]
status	Shows status	
agents	Shows agent statuses	<alb_name>
flow-policies	Shows flow policies	<alb_name>
target-pools	Shows target pools	<alb_name>
vips	Shows VIPs	<alb_name>
ports	Shows ports	alb <alb_name> vip <vip>
ports-alloc	Shows ports allocation for VIP	alb <alb_name> vip <vip>
payload-weight	Shows global payload weight	
evipcs	Shows eVIP-Cs	
evip-status	Shows eVIP status	
syslog-level	Shows log level	
avail-services	Shows available services	
transactions	Shows transactions	<alb_name>
f-transactions	Shows factory transactions	
keys	Shows key distribution	<alb_name>
pn-pairs	Shows peers of a repdb	<alb_name>
hosts	Shows hostname of each node	
Add Components	Command add adds components	
Related Commands	Description	Required Arguments
alb	Adds ALB	<alb_name>



sel-policy	Adds selection policy	alb <alb_name> name <selection_policy_name> order <order> [<addit_param_name> <value>...]
startup-command	Adds commands to be run at startup of nodes. The command must previously have been added with add command.	name <name> [node <node_id> all-undesigned]
command	Adds a startup command	name <name> <cmdstring>
node	Add a node to the cluster	<nodeid><hostname> Or floating <nodeid> > <floatPriority>
Delete Components	Command remove deletes components	
Related Commands	Description	Required Arguments
alb	Deletes ALB	<alb_name>
sel-policy	Deletes selection policy	<selection_policy_name>
startup-command	Deletes commands to be run at startup of nodes	name <name>[node <node_id> all-undesigned]
command	Deletes a startup command	name <name>
node	Deletes a node from the cluster	<nodeid>
Set Components	Command set sets components	
Related Command	Description	Required Arguments
payload-weight	Sets global payload weight	<node_id> <weight>
syslog-level	Sets log level	<log_level>
Other Command	Description	Required Argument
alb	Configures ALB	<alb_name>



3.5.1.4 ALB Configuration Mode

The ALB configuration mode shown on Table 4.

Table 4 ALB Configuration Mode

Access Level	General Configuration Mode	
Access Command	<code>alb <alb_name></code>	
Prompt	<code>EVIP(config-<alb_name>)#</code>	
Privileged Mode	Yes	
Show Information	Command <code>show</code> shows information about the ALB	
Related Commands	Description	Required Arguments
<code>esp_termination</code>	Shows where ESP traffic is decrypted	
<code>fees</code>	Shows FEEs for this ALB	
<code>lbes</code>	Shows LBEs for this ALB	
<code>flow-policies</code>	Shows flow policies for this ALB	
<code>startup-command</code>	Shows startup commands for eVIP elements	<code>group <group> fee/lbe/se <name></code>
<code>target-pools</code>	Shows target pools for this ALB	
<code>vips</code>	Shows VIPs for this ALB	
<code>deactivated-ports</code>	Shows deactivated ports	
<code>service-ip</code>	Shows service IP	
<code>pn-pairs</code>	Shows peers of a repdb	<code><alb_name></code>
<code>hosts</code>	Shows hostname of each node	
Add Components	Command <code>add</code> adds components	
Related Commands	Description	Required Arguments
<code>fee</code>	Adds FEE	<code>name <fee> node_id <node_id> interface <interface></code>



lbe	Adds LBE	name <lbe> node_id <node_id>
flow-policy	Adds flow policy	<flow_policy_name>
target-pool	Adds target pool	<target_pool_name>
vip	Adds VIP	<vip>
startup-command	Adds commands to be run at startup of eVIP elements. The command must previously have been added with add command.	name <name> [group <group> fee/lbe/se <name>]
Delete Components	Command remove deletes components	
Related Commands	Description	Required Arguments
fee	Deletes FEE	<fee>
lbe	Deletes LBE	<lbe>
flow-policy	Deletes flow policy	<flow_policy_name>
startup-command	Deletes commands to be run at startup of eVIP elements	name <name> [group <group> fee/lbe/se <name>]
target-pool	Deletes target pool	<target_pool_name>
vip	Deletes VIP	<vip>
Set Components	Command set sets components	
Related Commands	Description	Required Arguments
esp_termination	Sets where ESP traffic is decrypted	
service-ip	Sets service IP	<service_IP>
default-vip	Sets default VIP	<default_vip>
Other Commands	Description	Required Arguments
activate-alb	Activates ALB	
deactivate-alb	Deactivates ALB	
ersip	Configures eRSIP	
fee	Configures FEE	<fee_name>
target-pool	Configures target pool	<target_pool_name>
vip	Configure VIP	<vip>



3.5.1.5 Add Flow Policy Mode

The add flow policy mode shown in Table 5.

Table 5 Add Flow Policy Mode

Access Level	ALB Configuration Mode	
Access Command	add flow-policy <flow_policy_name>	
Prompt	EVIP(config-<alb_name>-<flow_policy_name>) #	
Privileged Mode	Yes	
Show Information	Command show shows committed flow-policy parameters	
Related Command	Description	
uncommitted	Shows uncommitted flow-policy parameters	
Set Components	Command set sets components	
Related Commands	Description	Required Arguments
protocol	Sets protocol parameter (mandatory)	<value>
target-pool	Sets target pool parameter (mandatory)	<value>
dest	Sets destination parameter (mandatory)	<value>
address-family	Sets address family parameter (optional)	<value>
so-grp	Sets socket group parameter (optional)	<value>
dest-port	Sets destination port parameter (optional)	<value>
src	Sets source parameter (optional)	<value>
src-port	Sets source port parameter (optional)	<value>
Other Commands	Description	
commit	Commits changes and adds flow policy	
rollback	Exits to current mode and discards uncommitted flow policy	
exit	Exits to current mode	



3.5.1.6 Add Target Pool Mode

The add target pool mode shown in Table 6.

Table 6 Add Target Pool Mode

Access Level	ALB Configuration Mode	
Access Command	add target-pool <target_pool_name>	
Prompt	EVIP(config-<alb_name>-<target_pool_name>) #	
Privileged Mode	Yes	
Show Information	Command show shows committed target-pool parameters	
Related Command	Description	
uncommitted	Shows uncommitted target-pool parameters	
Set Components	Command set sets components	
Related Commands	Description	Required Arguments
distribution-method	Sets distribution method (mandatory)	<distribution_method>
sticky-group	Set sticky group parameter (optional)	<yes/no> (default: yes)
stickiness-timeout	Sets stickiness timeout parameter (optional)	<value>
all-undesigned	Add all undesigned nodes to target pool (optional)	<yes/no> (default: no)
Other Commands	Description	
commit	Commits changes and adds target pool	
rollback	Exits to current mode and discards uncommitted target pool	
exit	Exits to current mode	

3.5.1.7 eRSIP Configuration Mode

The eRISP configuration mode shown in Table 7.



Table 7 eRSIP Configuration Mode

Access Level	ALB Configuration Mode	
Access Command	ersip	
Prompt	EVIP(config-<alb_name>-ersip) #	
Privileged Mode	Yes	
Show Information	Command show shows committed eRSIP parameters	
Related Command	Description	
uncommitted	Shows uncommitted eRSIP parameters	
Set Components	Command set sets components	
Related Commands	Description	Required Arguments
lo-watermark	Sets lo watermark	<value>
hi-watermark	Sets hi watermark	<value>
timeout-connection-session	Sets timeout connection session	<value>
timeout-requested-resources	Sets timeout requested resources	<value>
parse-ss-interval	Sets parse ss interval	<value>
lease-time	Sets lease time	<value>
lease-after-resume	Sets lease after resume	<value>
Other Command	Description	
commit	Commits changes	

3.5.1.8

FEE Configuration Mode

The FEE configuration mode shown in Table 8.

Table 8 FEE Configuration Mode

Access Level	ALB Configuration Mode	
Access Command	fee <fee_name>	
Prompt	EVIP(config-<alb_name><fee_name>) #	
Privileged Mode	Yes	
Show Information	Command show shows information about FEE	
Related Command	Description	



s-remote-gateways	Shows supervised remote gateways	
Add Components	Command add adds components	
Related Command	Description	Required Arguments
s-remote-gateway	Adds supervised remote gateway	<i><A.B.C.D/M IP address></i> (for example, 10.0.0.1/8)
Delete Components	Command remove deletes components	
Related Command	Description	Required Arguments
s-remote-gateway	Deletes supervised remote gateway	<i><A.B.C.D/M IP address></i> (for example, 10.0.0.1/8)
Other Commands	Description	
enable-fee	Enables FEE OBSOLETE	
disable-fee	Disables FEE OBSOLETE	
ospfv2	Configures routing parameters for OSPFv2	
ospfv3	Configures routing parameters for OSPFv3	
bfd-static	Configures routing parameters for an IPv4 default static route with BFD supervision	
bfd-static6	Configures routing parameters for an IPv6 default static route with BFD supervision	
bfd-ospfv2	Configures routing parameters for OSPFv2 BFD	
bfd-ospfv3	Configures routing parameters for OSPFv3 with BFD	

3.5.1.9

OSPFv2 Configuration Mode

The OSPFv2 configuration mode shown in Table 9.

Table 9 OSPFv2 Configuration Mode

Access Level	FEE Configuration Mode
Access Command	ospfv2
Prompt	EVIP(config-<alb_name>-<fee_name>-ospfv2) #
Privileged Mode	Yes



Show Information	Command <code>show</code> shows committed router parameters	
Related Commands	Description	
<code>uncommitted</code>	Shows uncommitted router parameters	
<code>routing-params</code>	Shows supported routing parameters	
Set Components	Command <code>set</code> sets components	
Related Commands	Description	Required Arguments
<code>router-id</code>	Sets router ID	<code><router_id></code>
<code>gateway</code>	Sets gateway	<code><gateway></code>
<code>local-address</code>	Sets local address	<code><local_address></code>
<code>router-priority</code>	Sets router priority	<code><router_priority></code>
<code>area</code>	Sets area	<code><area></code>
<code>area-type</code>	Sets area type	<code><area_type></code>
<code>hello-interval</code>	Sets <code>hello</code> interval	<code><hello_interval></code>
<code>dead-interval</code>	Sets dead interval	<code><dead_interval></code>
<code>transmit-delay</code>	Sets transmit delay	<code><transmit_delay></code>
<code>retransmit-interval</code>	Sets retransmit interval	<code><retransmit_interval></code>
Other Command	Description	
<code>commit</code>	Commits changes	

3.5.1.10

OSPFv3 Configuration Mode

The OSPFv3 configuration mode shown in Table 10

Table 10 OSPFv3 Configuration Mode

Access Level	FEE Configuration Mode
Access Command	ospfv3
Prompt	EVIP(config-<alb_name>-<fee_name>-ospfv3) #
Privileged Mode	Yes
Show Information	Command show shows committed router parameters
Related Commands	Description
uncommitted	Shows uncommitted router parameters



routing-params	Shows supported routing parameters	
Set Components	Command <code>set</code> sets components	
Related Commands	Description	Required Arguments
router-id	Sets router ID	<code><router_id></code>
gateway	Sets gateway	<code><gateway></code>
local-address	Sets local address	<code><local_address></code>
router-priority	Sets router priority	<code><router_priority></code>
area	Sets area	<code><area></code>
area-type	Sets area type	<code><area_type></code>
hello-interval	Sets <code>hello</code> interval	<code><hello_interval></code>
dead-interval	Sets dead interval	<code><dead_interval></code>
transmit-delay	Sets transmit delay	<code><transmit_delay></code>
retransmit-interval	Sets retransmit interval	<code><retransmit_interval></code>
Other Command	Description	
commit	Commits changes	

3.5.1.11 BFD Static Configuration Mode

The BFD Static configuration mode shown in Table 11.

Table 11 BFD Static Configuration Mode

Access Level	FEE Configuration Mode	
Access Command	bfd-static	
Prompt	EVIP(config-<alb_name>-<fee_name>-bfd-static)#	
Privileged Mode	Yes	
Show Information	Command show shows committed router parameters	
Related Commands	Description	
uncommitted	Shows uncommitted router parameters	
routing-params	Shows supported routing parameters	
Set Components	Command set sets components	
Related Commands	Description	Required Arguments



local-address	Sets local address	<local_address>
gateway	Sets gateway	<gateway>
echo	Sets echo	<echo>
echo-interval	Sets echo interval	<echo_interval>
slow-timer	Sets slow timer	<slow_timer>
bfd-interval	Sets BFD interval	<bfd_interval>
minrx	Sets minrx	<minrx>
multiplier	Sets multiplier	<multiplier>
Other Command	Description	
commit	Commits changes	

3.5.1.12

BFD OSPFv2 Configuration Mode

The BFD OSPFv2 configuration mode shown in Table 12.

Table 12 BFD OSPFv2 Configuration Mode

Access Level	FEE Configuration Mode	
Access Command	bfd-ospfv2	
Prompt	EVIP(config-<alb_name>-<fee_name>-bfd-ospfv2) #	
Privileged Mode	Yes	
Show Information	Command show shows committed router parameters	
Related Commands	Description	
uncommitted	Shows uncommitted router parameters	
routing-params	Shows supported routing parameters	
Set Components	Command set sets components	
Related Commands	Description	Required Arguments
echo	Sets echo	<echo>
echo-interval	Sets echo interval	<echo_interval>
slow-timer	Sets slow timer	<slow_timer>
bfd-interval	Sets BFD interval	<bfd_interval>
minrx	Sets minrx	<minrx>
multiplier	Sets multiplier	<multiplier>



Other Command	Description
<code>commit</code>	Commits changes

3.5.1.13 Static IPv6 Configuration Mode

The BFD Static IPv6 configuration mode shown in Table 13.

Table 13 BFD Static IPv6 Configuration Mode

Access Level	FEE Configuration Mode	
Access Command	<code>bfd-static6</code>	
Prompt	EVIP(config-<alb_name>-<fee_name>-bfd-static6)#	
Privileged Mode	Yes	
Show Information	Command <code>show</code> shows committed router parameters	
Related Commands	Description	
<code>uncommitted</code>	Shows uncommitted router parameters	
<code>routing-params</code>	Shows supported routing parameters	
Set Components	Command <code>set</code> sets components	
Related Commands	Description	Required Arguments
<code>local-address</code>	Sets local IPv6 address	<local_address>
<code>gateway</code>	Sets IPv6 gateway address	<gateway>
<code>echo</code>	Sets echo	<echo>
<code>echo-interval</code>	Sets echo interval	<echo_interval>
<code>slow-timer</code>	Sets slow timer	<slow_timer>
<code>bfd-interval</code>	Sets BFD interval	<bfd_interval>
<code>minrx</code>	Sets minrx	<minrx>
<code>multiplier</code>	Sets multiplier	<multiplier>
Other Command	Description	
<code>commit</code>	Commits changes	

3.5.1.14 BFD OSPFv3 Configuration Mode

The BFD OSPFv3 configuration mode shown in Table 14.



Table 14 BFD OSPFv3 Configuration Mode

Access Level	FEE Configuration Mode	
Access Command	bfd-ospfv3	
Prompt	EVIP(config-<alb_name>-<fee_name>-bfd-ospfv3) #	
Privileged Mode	Yes	
Show Information	Command show shows committed router parameters	
Related Commands	Description	
uncommitted	Shows uncommitted router parameters	
routing-params	Shows supported routing parameters	
Set Components	Command set sets components	
Related Commands	Description	Required Arguments
echo	Sets echo	<echo>
echo-interval	Sets echo interval	<echo_interval>
slow-timer	Sets slow timer	<slow_timer>
bfd-interval	Sets BFD interval	<bfd_interval>
minrx	Sets minrx	<minrx>
multiplier	Sets multiplier	<multiplier>
Other Command	Description	
commit	Commits changes	

3.5.1.15 Target Pool Configuration Mode

The Target Pool configuration mode shown in Table 15.

Table 15 Target Pool Configuration Mode

Access Level	ALB Configuration Mode	
Access Command	target-pool <target_pool_name>	
Prompt	EVIP(config-<alb_name>-<target_pool_name>) #	
Privileged Mode	Yes	
Show Information	Command show shows information about the target pool	
Related Commands	Description	



distribution-m	Shows distribution method	
sticky-group	Shows sticky group	
stickiness-timeout	Shows stickiness timeout	
Add Components	Command add adds component	
Related Command	Description	Required Arguments
node	Adds a node to the target pool	<node_id> [<weight>]
Delete Components	Command remove deletes component	
Related Command	Description	Required Argument
node	Deletes a node from the target pool	<node_id>

3.5.1.16 VIP Configuration Mode

The VIP configuration mode shown in Table 16.

Table 16 VIP Configuration Mode

Access Level	ALB Configuration Mode	
Access Command	vip <vip>	
Prompt	EVIP(config-<alb_name>-<vip>) #	
Privileged Mode	Yes	
Show Information	Command show shows information about the VIP	
Related Commands	Description	
ports	Shows ports for VIP	
ports-alloc	Shows ports allocation for VIP	
deactivated-ports	Shows deactivated ports for VIP	
Set Components	Command set sets component	
Related Commands	Description	Required Arguments



ephemeral-ports	Sets ephemeral ports	<code>protocol <protocol> first_port <first port> last_port <last port> block_size <block size></code>
wellknown-ports	Sets well-known ports	<code>protocol <protocol> first_port <first port> last_port <last port></code>
Other Commands	Description	
enable-vip	Enables VIP	
disable-vip	Disables VIP	