HAProxy Enterprise Documentation

Table of Contents

HAProxy Enterprise

HAProxy Enterprise is the industry's leading software load balancer. It powers modern application delivery at any scale and in any environment, providing the utmost performance, observability and security.

End-of-life dates

HAProxy Enterprise has the following product versions. Please note that dates for **End of life** in the table below refer to the last day of the month specified.

HAProxy Enterprise 3.1r1

Release date: Feb 2025

End of life: Feb 2026

Supported operating systems: AlmaLinux 8. AlmaLinux 9. Debian 11. Debian 12. RHEL 8. RHEL 9. Rocky Linux 8. Rocky Linux 9. Oracle 8. Oracle 9. SUSE 15.6. Ubuntu 22.04. Ubuntu 24.04. FreeBSD 13.4.

HAProxy Enterprise 3.0r1

Release date: Oct 2024

End of life: Feb 2029

Supported operating systems: AlmaLinux 8. AlmaLinux 9. Debian 11. Debian 12. RHEL 8. RHEL 9. Rocky Linux 8. Rocky Linux 9. Oracle 8. Oracle 9. SUSE 15.6. Ubuntu 22.04. Ubuntu 24.04. FreeBSD 13.4.

HAProxy Enterprise 2.9r1

Release date: May 2024

End of life: Feb 2025

Supported operating systems: AlmaLinux 8. AlmaLinux 9. Debian 11. Debian 12. RHEL 8. RHEL 9. Rocky Linux 8. Rocky Linux 9. Oracle 8. Oracle 9. SUSE 15.5. SUSE 15.6. Ubuntu 20.04. Ubuntu 22.04. Ubuntu 24.04. FreeBSD 13.2.

HAProxy Enterprise 2.8r1

Release date: Oct 2023

End of life: Feb 2028

Supported operating systems: AlmaLinux 8. AlmaLinux 9. Debian 11. Debian 12. Photon 3. RHEL 8. RHEL 9. Rocky Linux 8. Rocky Linux 9. Oracle 8. Oracle 9. SUSE 15.5. SUSE 15.6. Ubuntu 20.04. Ubuntu 22.04. Ubuntu 24.04. FreeBSD 13.2.

HAProxy Enterprise 2.7r1

Release date: Feb 2023

End of life: Feb 2024

Supported operating systems: AlmaLinux 8. AlmaLinux 9. Debian 11. Debian 12. Photon 3. RHEL 8. RHEL 9. Rocky Linux 8. Rocky Linux 9. Oracle 8. Oracle 9. SUSE 15.0. SUSE 15.1. SUSE 15.2. SUSE 15.3. SUSE 15.4. Ubuntu 20.04. Ubuntu 22.04. FreeBSD 12.4.

HAProxy Enterprise 2.6r1

Release date: Sep 2022

End of life: Feb 2027

Supported operating systems: AlmaLinux 8. AlmaLinux 9. Debian 10. Debian 11. Debian 12. Photon 3. RHEL 7. RHEL 8. RHEL 9. Rocky Linux 8. Rocky Linux 9. Oracle 8. Oracle 9. SUSE 15.0. SUSE 15.1. SUSE 15.2. SUSE 15.3. SUSE 15.4. Ubuntu 18.04. Ubuntu 20.04. Ubuntu 22.04. Ubuntu 24.04. FreeBSD 12.4.

HAProxy Enterprise 2.5r1

Release date: Feb 2022

End of life: Nov 2023

Supported operating systems: AlmaLinux 8. AlmaLinux 9. Debian 9. Debian 10. Debian 11. Photon 3. RHEL 7. RHEL 8. RHEL 9. Rocky Linux 8. Rocky Linux 9. Oracle 8. Oracle 9. SUSE 15.0. SUSE 15.1. SUSE 15.2. SUSE 15.3. Ubuntu 18.04. Ubuntu 20.04. Ubuntu 22.04.

HAProxy Enterprise 2.4r1

Release date: Nov 2021

End of life: Feb 2026

Supported operating systems: AlmaLinux 8. AlmaLinux 9. Debian 9. Debian 10. Debian 11. Photon 3. RHEL 7. RHEL 8. RHEL 9. Rocky Linux 8. Rocky Linux 9. Oracle 8. Oracle 9. SUSE 15.0. SUSE 15.1. SUSE 15.2. SUSE 15.3. Ubuntu 18.04. Ubuntu 20.04. Ubuntu 22.04. FreeBSD 12.4.

HAProxy Enterprise 2.3r1

Release date: Feb 2021

End of life: Nov 2022

Supported operating systems: Debian 9. Debian 10. Debian 11. Photon 3. RHEL 7. RHEL 8. Oracle 8. Oracle 9. SUSE 15.0. SUSE 15.1. SUSE 15.2. SUSE 15.3. Ubuntu 18.04. Ubuntu 20.04.

HAProxy Enterprise 2.2r1

Release date: Nov 2020

End of life: Feb 2025

Supported operating systems: AlmaLinux 8. Debian 9. Debian 10. Debian 11. Photon 3. RHEL 7. RHEL 8. Rocky Linux 8. Oracle 8. Oracle 9. SUSE 15.0. SUSE 15.1. SUSE 15.2. SUSE 15.3. Ubuntu 16.04. Ubuntu 18.04. Ubuntu 20.04. FreeBSD 12.4.

HAProxy Enterprise 2.1r1

Release date: Feb 2020

End of life: Nov 2021

Supported operating systems: Debian 8. Debian 9. Debian 10. RHEL 7. RHEL 8. Oracle 8. Oracle 9. SUSE 15.0. SUSE 15.1. Ubuntu 14.04. Ubuntu 16.04. Ubuntu 18.04. Ubuntu 20.04.

HAProxy Enterprise 2.0r1

Release date: Nov 2019

End of life: Feb 2024

Supported operating systems: AlmaLinux 8. Debian 8. Debian 9. Debian 10. RHEL 7. RHEL 8. Rocky Linux 8. Oracle 8. Oracle 9. SUSE 15.0. SUSE 15.1. SUSE 15.2. SUSE 15.3. Ubuntu 14.04. Ubuntu 16.04. Ubuntu 18.04. FreeBSD 12.4.

HAProxy Enterprise 1.9r1

Release date: Feb 2019

End of life: Nov 2020

Supported operating systems: Debian 8. Debian 9. Debian 10. RHEL 7. RHEL 8. Rocky Linux 8. Oracle 8. Oracle 9. SUSE 15.0. Ubuntu 14.04. Ubuntu 16.04. Ubuntu 18.04.

HAProxy Enterprise 1.8r2

Release date: Nov 2018

End of life: Feb 2023

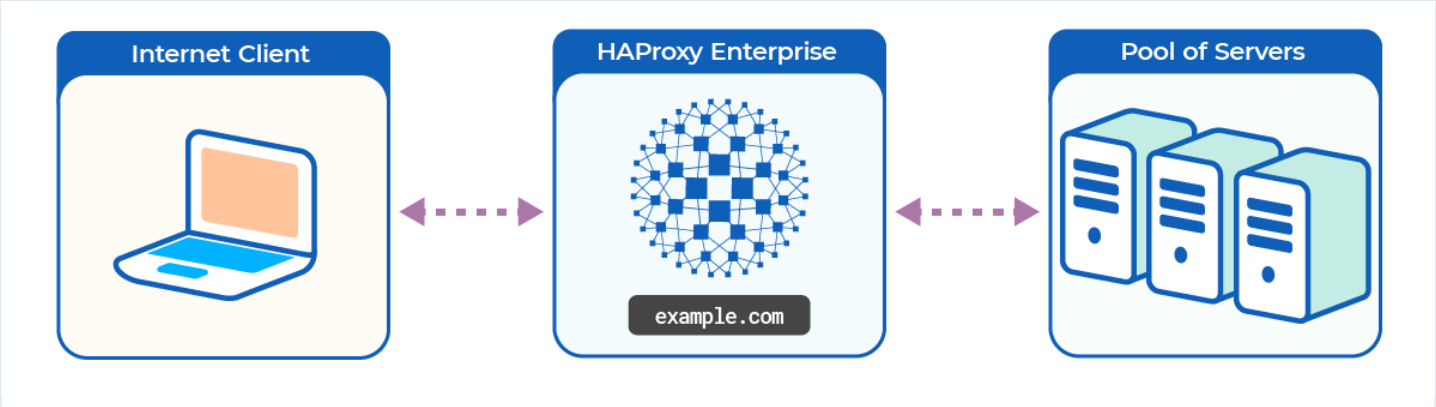
Supported operating systems: Debian 8. Debian 9. RHEL 7. Oracle 8. Oracle 9. Ubuntu 14.04. Ubuntu 16.04. Ubuntu 18.04.

Getting started

Overview of HAProxy Enterprise

Today, many web applications experience high traffic demands, and traffic spikes can lead to server overloads and a deteriorating user experience.

The HAProxy Enterprise software load balancer spreads traffic across a pool of healthy servers, allowing you to scale out your capacity for handling concurrent requests. You can then easily answer the demand, while also improving performance and availability.



You can seamlessly integrate HAProxy Enterprise with your existing infrastructure, either:

- at the edge of a network to replace traditional, hardware load balancers.
- in the cloud to replace expensive virtual load balancers.
- inside a container network as a sidecar proxy.

When you deploy HAProxy Enterprise, HAProxy Technologies, LLC. does not store or process any data of your customers related to any traffic flows through your load balancers.

The job of a load balancer

A load balancer sits in front of your web servers and receives requests directly from clients before relaying them to your servers. In this way, the load balancer can distribute requests evenly, allowing the work to be shared. This sharing of the workload prevents any backend server from becoming overworked and, as a result, your servers operate more efficiently. A load balancer differs from a web or application server in that it does not host your web application directly. Instead, its job is to spread the work across your cluster of servers.

Because all requests pass through the load balancer on their way to a server, the load balancer becomes the ideal place to redirect clients, inspect for malicious behavior, and generate traffic statistics, among other duties. HAProxy Enterprise provides security and management features in addition to load-balancing. You can use it to load balance any TCP/IP service including databases, message queues, mail servers, and IoT devices.

HAProxy Enterprise features

HAProxy Enterprise offers:

- comprehensive load balancing algorithms
- customizable routing logic
- session persistence
- device detection
- geolocation
- support for load balancer clustering and high availability
- bot management
- a Web Application Firewall
- and more

It integrates seamlessly with your existing infrastructure, able to run on virtual machines on premises or in the cloud, or as a Docker container. It routes traffic to any number of pools of servers comprising physical servers, VMs, Kubernetes pods, containers, and so on.

Download the datasheet for more details.

Installation

Install HAProxy Enterprise on AWS

This section describes how to deploy HAProxy Enterprise in Amazon Web Services.

Overview

HAProxy Enterprise is a Layer 7 load balancer that many people use to achieve high availability, security, and observability for their applications running in AWS EC2. You can use it as a replacement for other, cloud-based load balancers or in conjunction with Amazon Network Load Balancer for extra redundancy.

HAProxy Enterprise offers:

- comprehensive load balancing algorithms
- customizable routing logic
- session persistence
- device detection
- geolocation
- support for load balancer clustering and high availability
- bot management
- a Web Application Firewall
- and more

Common deployment patterns

The table below lists several common ways to deploy HAProxy Enterprise in AWS.

Deployment pattern	Description
A single HAProxy Enterprise load balancer	A single HAProxy Enterprise instance distributing traffic to web applications. This design does not include redundancy at the load balancing tier, but is useful for non-production workloads or applications that do not require extra redundancy that you would get by deploying two load balancers.
Two HAProxy Enterprise load balancers and AWS NLB	Two HAProxy Enterprise instances distributing traffic to web applications. An AWS Network Load Balancer load balances traffic to these two load balancers, giving you redundancy at the load balancing tier.

AWS supported regions

We support the following regions for deploying the HAProxy Enterprise AMI:

- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Hyderabad)
- Asia Pacific (Jakarta)
- Asia Pacific (Melbourne)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- EU (Spain)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- EU (Frankfurt)
- EU (Ireland)
- EU (London)
- EU (Milan)
- EU (Stockholm)
- EU (Zurich)
- Middle East (Bahrain)
- Middle East (UAE)
- South America (Sao Paulo)
- US East (N. Virginia)
- US East (Ohio)
- US West (N. California)
- US West (Oregon)

Get support

To get the most out of HAProxy Enterprise in AWS, activate support.

You need to activate support to get access to some parts of the documentation, such as WAF.

Contact us:

Info	Details
Email	contact@haproxy.com
Hours of operation	4am - 6pm EST/EDT
Target response time for critical issues	8 hours

If you require 24×7 support, significantly shorter SLAs, and consultative support, please activate your support account. Visit the Amazon Support Activation page to sign up for a login to the customer portal.

For support terms and related information, see HAProxy Legal Policies.

Launch the HAProxy Enterprise AMI

In this section, you will create an HAProxy Enterprise server in AWS by launching it from the AWS Marketplace.

Launch the AMI from the marketplace

A VPC with at least one public subnet is required to complete the following procedure. If you do not yet have a VPC with a public subnet, see Deploy HAProxy Enterprise in an Amazon VPC.

Create an HAProxy Enterprise server from the HAProxy Enterprise AMI:

1. Open the AWS Marketplace.

   > **Info**
   >
   > We recommend you create the server using the **AWS Marketplace** link above instead of by selecting **Launch instance** on the **EC2 Dashboard**. The **AWS Marketplace** process provides information and options not available from the **EC2 Dashboard**.

2. Click the desired HAProxy Enterprise Amazon Machine Image (AMI) product. Options exist for Ubuntu Server edition and Red Hat Enterprise Linux edition.

   The versions shown are the latest versions. If needed, you can select an earlier version in the **Configure this Software** screen described below, once you have started your subscription.

   You can estimate costs by using the pricing calculator on the marketplace product page.

3. Click **Continue to Subscribe** to start a subscription to the HAProxy Enterprise software.

4. Review the pricing and license details, then click **Continue to Configuration**.

5. On the **Configure this Software** screen, set the following fields:

Field	Description	Example value
Fulfillment option	The type of procedure used for launching the AMI in your environment.	64-bit (x86) Amazon Machine Image (AMI)
Software version	The version of HAProxy Enterprise to launch.	2.7r1-20230215 (Feb 16, 2023)
Region	The AWS region where you created your VPC.	US East (Ohio) / us-east-2

6. Click **Continue to launch**.

7. On the **Launch this software** screen, set the following fields:

Field	Description	Example value
Choose an action	How to launch the AMI.	Launch from Website
EC2 instance type	Choose an instance type with at least 4 CPUs and 4 GB RAM, but larger as needed.	c5.xlarge
VPC settings	Choose the VPC ID from the VPC you created earlier.	vpc-0146c0c368ac64143
Subnet settings	Choose one of the public subnets that was created inside the VPC.	subnet-09f29a57cffa00e48
Security group settings	Use the security group settings provided by the seller. You may want to change the Source value for port 22, which represents SSH, to be your public IPv4 address instead of Anywhere, to allow connections only from your IP address for SSH access.	Select **Create new based on seller settings**.
Key pair settings	Create an SSH key pair or use an existing key pair for connecting to EC2 instances.	-

8. Click **Launch**.

Create an elastic IP address

To associate a public, elastic IP address with your HAProxy Enterprise instance:

1. Open the Amazon EC2 console.
2. From the EC2 Dashboard, click **Elastic IPs**, then **Allocate Elastic IP address**.
3. On the **Allocate Elastic IP address** screen, click **Allocate**.
4. From the EC2 Dashboard, go to **Elastic IPs**, select the elastic IP from the list and open its settings.
5. Click **Associate Elastic IP address**.
6. Choose your HAProxy Enterprise instance from the list.
7. Click **Associate**.

Connect to the HAProxy Enterprise instance

During installation, you configured an SSH key pair that you can use to connect to the EC2 instance.

1. If necessary, change the permissions of your private key:

```
chmod 600 my-private-key.pem
```

2. To get the public IPv4 address of the instance, open the Amazon EC2 console.
3. From the EC2 Dashboard, go to **Instances** and select the HAProxy Enterprise instance from the list. Copy its public IPv4 address.
4. Connect to the HAProxy Enterprise instance through its public IP address:

Ubuntu:

```
ssh -i my-private-key.pem ubuntu@35.181.155.36
```

RedHat:

```
ssh -i my-private-key.pem ec2-user@35.181.155.36
```

Manage the HAProxy Enterprise service

The HAProxy Enterprise service runs at startup. You can manage the process with `systemctl`.

1. Connect to the HAProxy Enterprise instance through its public IP address.
2. Use `systemctl status` to check that the service is running:

```
systemctl status hapee-<VERSION>-lb
```

output

```
hapee-VERSION-lb.service - HAPEE Load Balancer
   Loaded: loaded (/usr/lib/systemd/system/hapee-VERSION-lb.service; enabled; vendor preset: disabled)
   Drop-In: /etc/systemd/system/hapee-VERSION-lb.service.d
            └─override.conf
   Active: active (running) since Wed 2020-12-09 14:07:46 UTC; 14min ago
   Main PID: 918 (hapee-lb)
   CGroup: /system.slice/hapee-VERSION-lb.service
            ├─918 /opt/hapee-VERSION/sbin/hapee-lb -Ws -f /etc/hapee-VERSION/hapee-lb.cfg -p /run/hapee-VERSION-
lb.pid -m 5212
            └─920 /opt/hapee-VERSION/sbin/hapee-lb -Ws -f /etc/hapee-VERSION/hapee-lb.cfg -p /run/hapee-VERSION-
lb.pid -m 5212
```

3. If you edit your configuration file, use `systemctl reload` to reload the load balancer configuration after making changes:

```
sudo systemctl reload hapee-<VERSION>-lb
```

Tutorial: Deploy HAProxy Enterprise in an Amazon VPC

During this procedure, you will deploy a single HAProxy Enterprise load balancer in an Amazon Virtual Private Cloud (VPC) to load balance traffic to web applications.

This design does not include redundancy at the load balancing tier, but is useful for non-production workloads or applications that do not require extra redundancy that you would get by deploying two HAProxy Enterprise load balancers.

Prerequisites

Before getting started:

- Set up an AWS account at Amazon if you have not already.
- Learn about common deployment patterns.

What you will accomplish

In this tutorial, you will:

- Create an Amazon VPC with public and private subnets, and NAT gateways.
- Create the HAProxy Enterprise instance.
- Create an EC2 instance to act as a web server.
- Optionally, add a second HAProxy Enterprise instance and a Network Load Balancer.

This tutorial should take approximately 30 minutes.

Create a VPC

The VPC will contain your load balancer on a public subnet, while your web servers will be on a private subnet accessible only through the load balancer.

To create the Amazon VPC:

1. Open the Amazon VPC console.
2. Click **Create VPC**.
3. On the **Create VPC** screen, choose the following values:

Field	Description	Example value
Resources to create	Whether to create additional resources such as subnets and availability zones with your VPC.	VPC and more
Name tag	The name to attach to resources being created.	example
IPv4 CIDR block	The IP range to assign to the VPC.	10.0.0.0/16
IPv6 CIDR block	Whether to enable IPv6 addresses.	No IPv6 CIDR block
Tenancy	Whether to use single-tenant (default) or dedicated hardware for your VPC.	Default
Number of availability zones	Choose the number of availability zones for high availability.	2
Number of public subnets	We will deploy the HAProxy Enterprise server into one public subnet so that internet traffic can access it.	2
Number of private subnets	Create a private subnet for your web servers. Traffic will go through your load balancers to reach these servers.	2
NAT gateway	Create a NAT gateway so that servers in the private subnet can reach the internet for software updates.	1 per AZ
VPC endpoints	Whether to create an S3 Gateway.	None
Enable DNS hostnames	Whether to enable DNS hostnames for your public IP addresses.	checked
Enable DNS resolution	Whether to enable DNS resolution using the Amazon DNS server.	checked

For more information about VPCs, review AWS's Virtual Private Clouds documentation.

Launch the HAProxy Enterprise AMI

Launch the HAProxy Enterprise AMI and connect to the instance.

Create a web servers security group

Create a security group that will allow the HAProxy Enterprise load balancer to communicate with the web servers over ports 22 (SSH) and 80 (HTTP):

1. Open the Amazon EC2 console.
2. From the EC2 Dashboard, click **Security Groups**, then **Create security group**.

3. On the **Create security group** screen, set the following fields:

Field	Description	Example value
Security group name	The name to assign to the security group.	webservers-security-group
Description	A description for the security group.	Security group rules for web servers
VPC	Choose the VPC ID from the VPC you created earlier.	vpc-0146c0c368ac64143

4. Add the following inbound rules:

Type	Source	Source value
HTTP	Custom	Choose the security group you assigned to the load balancer
SSH	Custom	Choose the security group you assigned to the load balancer

5. Click **Create security group**.

Launch a web server

For example purposes, create a web server that handles web requests. We will configure HAProxy Enterprise to route traffic to it.

1. From the EC2 Dashboard, click **Launch instance**.

   Choose a server AMI, such as Amazon Linux.

2. Choose the SSH key pair used to connect to the EC2 instance.

3. Under **Network settings**, click **Edit**.

4. Set the following fields:

Field	Description	Example value
VPC	Select the VPC you created.	vpc-0146c0c368ac64143
Subnet	Select one of the private subnets.	subnet-0700b54c5c1e471664
Auto-assign public IP	Whether to assign a public IP address to this instance.	Disable
Firewall	The web servers security group that you created.	Select existing security group, sg-0671c2f614fbf7d1e

5. Click **Launch instance**.

6. Connect to the web server via SSH. Because the web server is on the private subnet, you will need to connect to it via the HAProxy Enterprise server, which is on the public subnet.

   - Copy your private SSH key to the HAProxy Enterprise server.

   ```
   scp -i my-private-key.pem ./my-private-key.pem ubuntu@35.181.155.36:~/
   ```

   - Connect to the HAProxy Enterprise server through its public IP address.

   - If necessary, change the permissions of your private key that has been copied to the HAProxy Enterprise server:

   ```
   chmod 600 my-private-key.pem
   ```

   - Connect to the web server through its private IP address.

   ```
   ssh -i ~/my-private-key.pem ec2-user@10.0.148.139
   ```

   - Install the NGINX web server.

   ```
   sudo amazon-linux-extras install nginx1
   sudo systemctl enable nginx
   sudo systemctl start nginx
   ```

Add the web server to the HAProxy Enterprise configuration

To register the web server with the load balancer:

1. Connect to the HAProxy Enterprise instance through its public IP address.

2. Edit the file `/etc/hapee-<VERSION>/hapee-lb.cfg`.

3. Change the `backend be_app` section to include the private IP address of your web server.

```
backend be_app
  balance           roundrobin
  server app1       10.0.148.139:80 check
```

4. Save the file.

5. Reload the HAProxy Enterprise configuration:

```
sudo systemctl reload hapee-<VERSION>-lb
```

When browsing to the public IP address of the HAProxy Enterprise load balancer, you should see the web server's web page.

Optional: Deploy a second HAProxy Enterprise instance

You can achieve high availability for your load balancing tier by adding a second HAProxy Enterprise instance. Each subnet in a VPC resides in an availability zone. By launching HAProxy Enterprise instances in separate subnets, you gain protection from failure of a zone.

During this procedure, you will create an Amazon Network Load Balancer (NLB) to route traffic to both HAProxy Enterprise instances, doubling your load balancer capacity.

To create a second load balancer:

1. Repeat the steps in the Launch the HAProxy Enterprise AMI procedure, but assign the second instance to the other public subnet. Use the security group you already created for the first instance.

2. Copy the load balancer configuration, `/etc/hapee-<VERSION>/hapee-lb.cfg`, to the new load balancer and reload the `hapee-<VERSION>-lb` service.

3. Create a target group that the AWS NLB will use to send traffic to your HAProxy Enterprise instances:

   - Open the Amazon EC2 console.
   - From the EC2 Dashboard, click **Target groups** under **Load Balancing**, then **Create target group**.
   - On the **Specify group details** screen, set the following fields:

Field	Description	Example value
Target type	Choose how AWS NLB determines which instances to route traffic to.	Instances
Target group name	A name for the group of HAProxy Enterprise instances being targeted.	load-balancers
Protocol	The protocol by which the HAProxy Enterprise instances listen for incoming traffic.	TCP
Port	The TCP port at which the HAProxy Enterprise instances listen for incoming traffic.	80
VPC	The VPC where you created your HAProxy Enterprise instances.	vpc-0146c0c368ac64143
Health check protocol	The protocol by which the AWS NLB will send periodic health check probes.	TCP

   - Click **Next**.
   - On the **Register targets** screen, select the HAProxy Enterprise instances to include in the target group. Then click **Include as pending below**.
   - Click **Create target group**.

4. Create an AWS NLB to route traffic to both HAProxy Enterprise instances:

   - From the EC2 Dashboard, click **Load Balancers**, then **Create load balancer**.
   - Choose to create a Network Load Balancer.
   - On the **Create Network Load Balancer** screen, set the following fields:

Field	Description	Example value
Load balancer name	A name for the AWS NLB	my-nlb
Scheme	Whether the Network Load Balancer will be internet facing.	Internet-facing
IP address type	Whether your subnet uses IPv4 and IPv6 addresses, or only IPv4.	IPv4
VPC	Choose the VPC where you launched your HAProxy Enterprise instances.	vpc-0146c0c368ac64143
Mappings	Select the availability zones of your targets. Since you launched HAProxy Enterprise instances in both availability zones, select both. Then choose the public subnets.	us-east-2a, us-east-2b
Listener	Choose the protocol and port at which the AWS NLB will receive traffic. Set the **Default action** to the target group you created before.	TCP / 80

   - Click **Create load balancer**.

   Once the AWS NLB has been provisioned, you will be able to reach your web application at the new DNS name shown in the AWS NLB load balancer's details.

Install HAProxy Enterprise on Azure

In this section, you will learn how to install HAProxy Enterprise on Azure.

Get support

To get the most out of HAProxy Enterprise in Azure, activate support.

You need to activate support to get access to some parts of the documentation, such as WAF.

Contact us:

Info	Details
Email	contact@haproxy.com
Hours of operation	4am - 6pm EST/EDT
Target response time for critical issues	8 hours

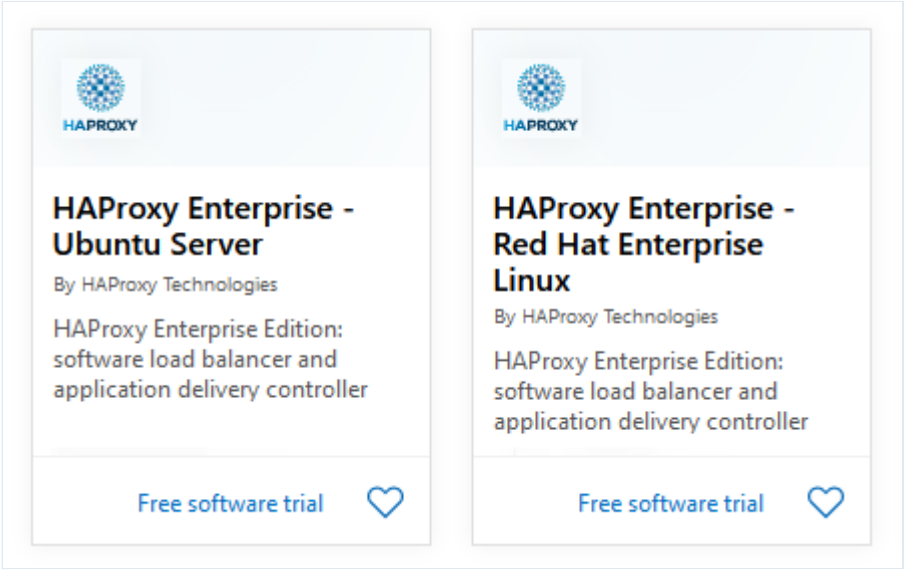
If you require 24×7 support, significantly shorter SLAs, and consultative support, please activate your support account. Visit the Azure Support Activation page to sign up for a login to the customer portal.

For support terms and related information, see HAProxy Legal Policies.

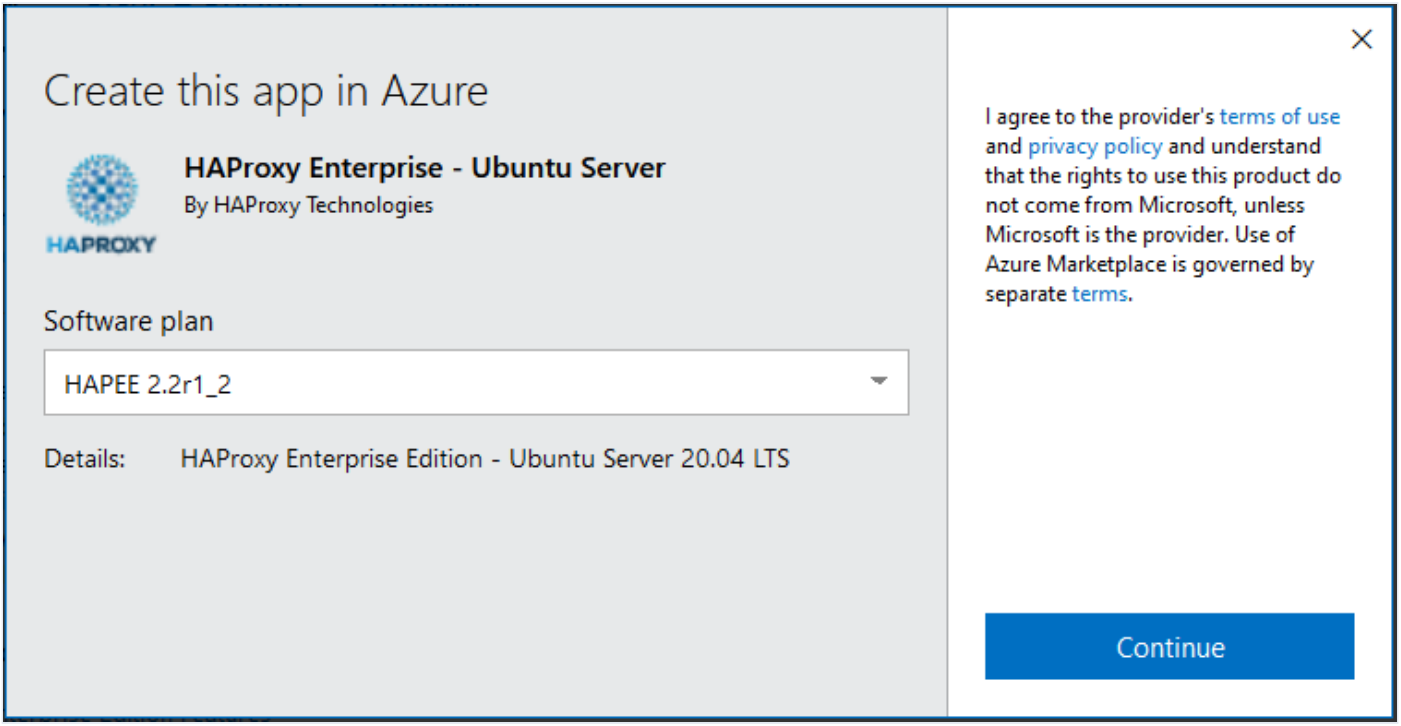
Launch HAProxy Enterprise

To launch HAProxy Enterprise directly from the Azure Marketplace:

1. From the Azure Marketplace, search for HAProxy Enterprise and choose one of the virtual machine images from the list that have the HAProxy Technologies logo. Options include an Ubuntu Server edition and a Red Hat Enterprise Linux edition.

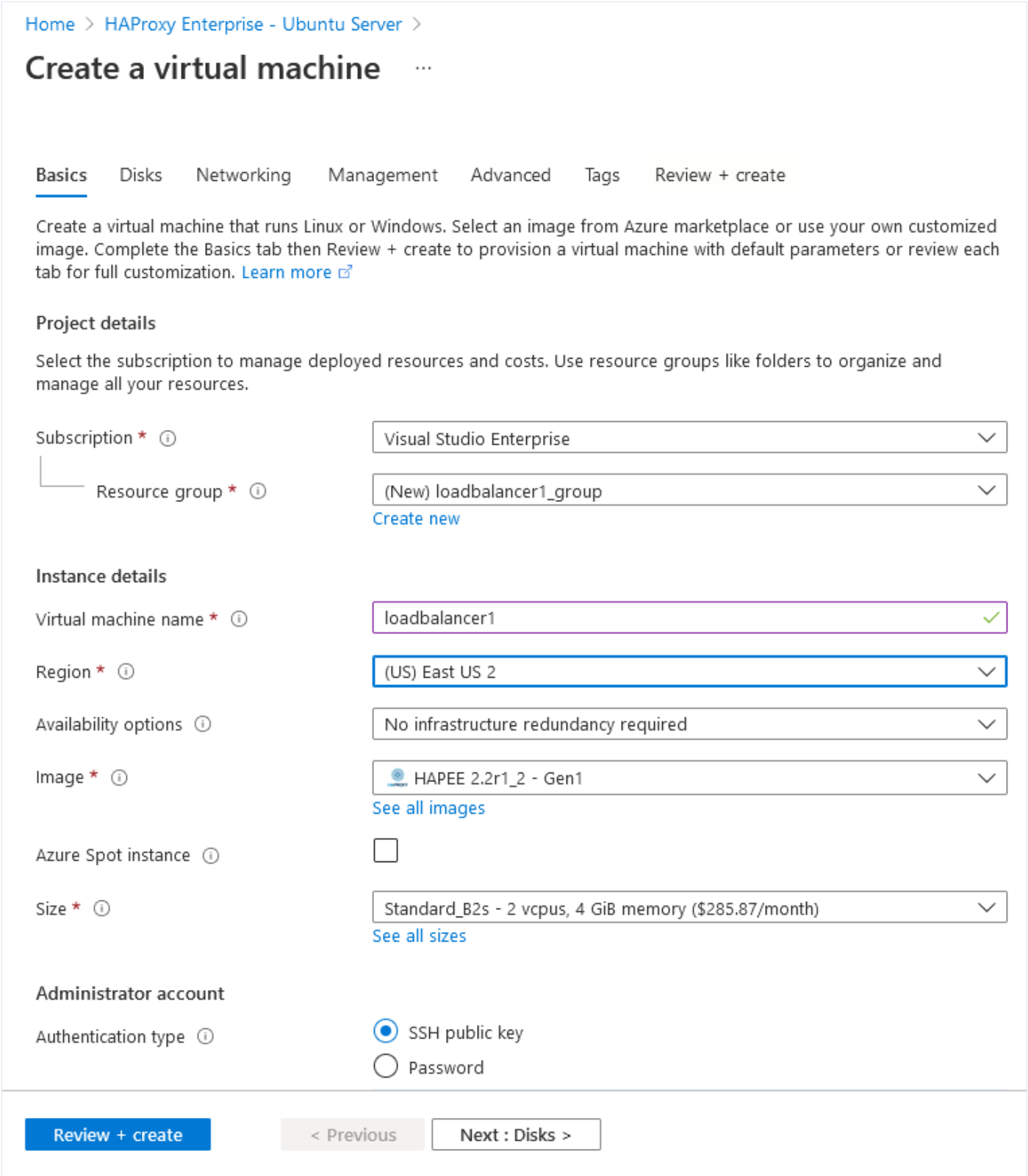


2. On the details page, click **Get It Now** and then select the software plan, which is the version of HAProxy Enterprise that you want to install.



3. On the next screen, click **Create** to launch the virtual machine. You can also click **Start with a pre-set configuration** to set up the VM with CPU and memory-optimized for a common workload.

4. On the **Create a virtual machine** screen, fill in the fields for the resource group, virtual machine name, region, availability zone, image size, and Administrator account. We recommend setting up an SSH keypair here so that you can access your machine using SSH.



5. Continue through to the **Networking** screen. Choose whether you want to assign a public IP address to the load balancer.

   - You can give the virtual machine a public IP, which makes it accessible from the Internet.
   - Or, you can set the **Public IP** field to `None` and select under **Load balancing**, `Place this virtual machine behind an existing load-balancing solution`, which allows you to place two or more HAProxy Enterprise servers behind an Azure Load Balancer. This allows you to run your HAProxy Enterprise load balancers in an active-active setup for high availability. You must create the Azure Load Balancer before you can complete this step.

6. Continue through the other tabs until you reach the **Review + Create** screen. Click **Create** to create the virtual machine.

Connect to the HAProxy Enterprise instance

During installation, you had the option to configure an SSH keypair that you can use to connect to the virtual machine.

1. If necessary, change the permissions on the private key file that you downloaded:

   ```
   chmod 600 my-haproxy-enterprise_key.pem
   ```

2. Copy the public IPv4 address for the instance from the Azure portal.

3. Connect to the HAProxy Enterprise instance through its public IP using SSH. For example:

   ```
   ssh -i my-haproxy-enterprise_key.pem azureuser@20.80.234.150
   ```

   If you set up an Azure Load Balancer in front of your instance, then you will need to go to the Load balancers screen and create an inbound NAT rule that maps a port for SSH (e.g. 222) on the Azure Load Balancer to port 22 on the HAProxy Enterprise instance.

Manage the HAProxy Enterprise service

The HAProxy Enterprise service runs at startup. You can manage the process with `systemctl`.

For example, use `systemctl status` to check that the service is running:

```
sudo systemctl status hapee-<VERSION>-lb
```

output

```
hapee-VERSION-lb.service - HAPEE Load Balancer
   Loaded: loaded (/lib/systemd/system/hapee-VERSION-lb.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/hapee-VERSION-lb.service.d
           └─override.conf
   Active: active (running) since Tue 2021-03-30 15:47:08 UTC; 47min ago
 Main PID: 983 (hapee-lb)
    Tasks: 3 (limit: 4615)
   Memory: 12.2M
   CGroup: /system.slice/hapee-VERSION-lb.service
           ├─983 /opt/hapee-VERSION/sbin/hapee-lb -Ws -f /etc/hapee-VERSION/hapee-lb.cfg -p /run/hapee-VERSION-lb.pid -
m 2623
           └─993 /opt/hapee-VERSION/sbin/hapee-lb -Ws -f /etc/hapee-VERSION/hapee-lb.cfg -p /run/hapee-VERSION-lb.pid -
m 2623
```

Install HAProxy Enterprise on BSD

Using HAProxy Fusion?

If you're using HAProxy Fusion, then see the HAProxy Fusion - Nodes topic instead.

This section describes how to install HAProxy Enterprise on BSD.

Supported operating systems

HAProxy Enterprise is distributed through the Operating System package manager for the following Linux distributions:

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
3.1r1	Feb 2025	Feb 2026	• FreeBSD 13.4 (amd64)

- Other HAProxy Enterprise versions

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
3.1r1	Feb 2025	Feb 2026	• FreeBSD 13.4 (amd64)
3.0r1	Oct 2024	Feb 2029	• FreeBSD 13.4 (amd64)
2.9r1	May 2024	Feb 2025	• FreeBSD 13.2 (amd64)
2.8r1	Oct 2023	Feb 2028	• FreeBSD 13.2 (amd64)
2.7r1	Feb 2023	Feb 2024	• FreeBSD 12.4 (amd64)
2.6r1	Sep 2022	Feb 2027	• FreeBSD 12.4 (amd64)
2.4r1	Nov 2021	Feb 2026	• FreeBSD 12.4 (amd64)
2.2r1	Nov 2020	Feb 2025	• FreeBSD 12.4 (amd64)
2.0r1	Nov 2019	Feb 2024	• FreeBSD 12.4 (amd64)

Install HAProxy Enterprise

The following procedure adds package repositories and installs HAProxy Enterprise 3.1r1.

1. Optional: To verify the integrity of the script before installing, first download the script and its SHA hash to a local directory:

   ```
   wget https://www.haproxy.com/static/install_haproxy_enterprise.sh
   wget https://www.haproxy.com/static/install_haproxy_enterprise.sh.sha512.asc
   ```

   Then run these commands to verify the checksum of the script against the SHA hash:

```
gpg --keyserver hkp://keyserver.ubuntu.com --recv-keys 0xCA2DF14657C5A207
gpg --verify ./install_haproxy_enterprise.sh.sha512.asc
```

Check for the output Good signature.

2. To install HAProxy Enterprise, run the following command, replacing `<HAProxy Enterprise Key>` with your HAProxy Enterprise license key.

```
wget https://www.haproxy.com/static/install_haproxy_enterprise.sh
sudo bash ./install_haproxy_enterprise.sh \
  --version 3.1r1 \
  --key <HAProxy Enterprise key>
```

3. Enable and start the HAProxy Enterprise service:

```
sudo service hapee_31_lb onestart
```

Install additional modules

HAProxy Enterprise comes with additional native and third-party modules.

Search for additional modules

```
sudo pkg search -g -r 'HAPEE31R1' '*'
sudo pkg search -g -r 'HAPEEEXTRAS' '*'
```

Install a module

For example, install the `update` module:

```
sudo pkg install hapee-3.1r1-lb-update
```

See other parts of this documentation for instructions on how to enable and configure each package.

Locate installed directories

HAProxy Enterprise files are installed following these rules:

- Binaries and documentation are in `/usr/local/opt/hapee-3.1/`.

```
/usr/local/opt/hapee-3.1/
  |-- bin
  |-- dev
  |-- doc
  |-- modules
  |-- sbin
  |-- version
```

- Configuration is installed in `/usr/local/etc/hapee-3.1/`.

```
/usr/local/etc/hapee-3.1/
  |-- hapee-lb.cfg
```

- `init` scripts are installed in `/usr/local/etc/rc.d/`.

```
/usr/local/etc/rc.d/
  |-- hapee_31_lb
```

Install HAProxy Enterprise on Docker

Using HAProxy Fusion?

If you're using HAProxy Fusion, then see the HAProxy Fusion - Nodes topic instead.

The HAProxy Enterprise Docker image lets you run the load balancer as a container.

Install the HAProxy Enterprise Docker container

The haproxy-enterprise Docker image hosts HAProxy Enterprise and the HAProxy Data Plane API. Follow these steps to install it:

1. If you do not have a HAProxy Enterprise license key, get one by registering and requesting a free trial of HAProxy Enterprise at https://www.haproxy.com/downloads/hapee-trial/.

2. Log into the HAProxy Enterprise Docker registry using your HAProxy Enterprise license key as both the username and password.

   ```
   sudo docker login https://hapee-registry.haproxy.com
   ```

3. Pull the HAProxy Enterprise image.

   ```
   sudo docker pull hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
   ```

4. Create an HAProxy Enterprise configuration file (i.e. `hapee-lb.cfg`) that defines your load balancer settings.

   You can run a container temporarily, just to get the HAProxy Enterprise configuration file from it to use as a starting point.

   ```
   sudo docker run --rm -d --name hapee hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
   sudo docker cp hapee:/etc/hapee-3.1 ./
   sudo docker stop hapee
   cd hapee-3.1
   ```

5. Start the Docker container, referencing the directory containing your configuration file as a volume by using the `-v` flag.

```
sudo docker run \
  --name hapee-3.1 \
  -d \
  -p 80:80 \
  -p 443:443 \
  -p 5555:5555 \
  -v $(pwd):/etc/hapee-3.1  \
  --restart=unless-stopped \
  hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
```

Access the Data Plane API

The Data Plane API listens at port 5555. However, you must retrieve the Basic authentication credentials to access it.

1. Use the following command to read the username and password from the Data Plane API configuration file:

```
sudo docker exec -it hapee-3.1 grep user: -A3 /etc/hapee-extras/dataplaneapi.yml
```

output

```
user:
- insecure: true
  password: v7xkLr4Z4Qlc
  name: admin
```

Then you can verify that the API is working by calling the `info` function. In the example below, we use the username and password that we retrieved during the last step:

```
curl -X GET --user admin:v7xkLr4Z4Qlc http://localhost:5555/v2/info
```

output

```
{"api":{"build_date":"2021-06-01T21:18:15.000Z","version":"v2.3.2-ee2 8fdcfba"},"system":{}}
```

Reload or stop the services

When you change your `hapee-lb.cfg` configuration file, you will need to reload the service for the changes to take effect. We use s6-overlay to run HAProxy Enterprise and the HAProxy Data Plane API as services.

Reload the services

To reload the services:

1. Make sure that the following line is included in the `global` section of your HAProxy Enterprise configuration file. It ensures that you can perform a hitless reload, which means no traffic is dropped.

   ```
   global
     stats socket /var/run/hapee-3.1/hapee-lb.sock user hapee-lb group hapee mode 660 level admin expose-fd listeners
   ```

2. Reload the HAProxy Enterprise process with this command:

   ```
   sudo docker exec hapee-3.1 s6-svc -2 /var/run/s6/services/haproxy
   ```

   The `-2` argument sends a `SIGUSR2` signal to the HAProxy Enterprise container, and when the load balancer receives the signal it will reload.

3. (Optional): To restart the Data Plane API, use this command:

   ```
   sudo docker exec hapee-3.1 s6-svc -t /var/run/s6/services/dataplaneapi
   ```

Stop the services

To perform a graceful shutdown of the load balancer you can issue this command:

```
sudo docker exec hapee-3.1 s6-svc -1 /var/run/s6/services/haproxy
```

The `-1` argument sends a `SIGUSR1` signal to the HAProxy Enterprise container. Upon receiving the signal, the load balancer unbinds its listening ports so that it will not receive any additional connections, but will continue processing existing connections until all connections close. After all connections have closed, the load balancer process terminates gracefully.

Tip

You can configure the maximum time the load balancer will wait for connections to close after issuing the command for a graceful shutdown. Use the global directive `hard-stop-after`, specifying the desired maximum time (default: milliseconds). For example, to set the maximum time to 30 seconds, add the following line to the `global` section in your load balancer configuration.

```
global
  hard-stop-after 30s
```

If needed, you can perform a hard restart by issuing the following command:

```
sudo docker exec hapee-3.1 s6-svc -t /var/run/s6/services/haproxy
```

The `-t` argument sends a `SIGTERM` signal to the HAProxy Enterprise container. When the load balancer receives the `SIGTERM` signal, it does not shut down gracefully, but rather, it immediately terminates and closes all established connections. Note that performing a hard restart in this way may result in dropped traffic.

Installing with a custom Docker file

To install a modified HAProxy Enterprise environment, create a custom Docker file. Specify the desired HAProxy Enterprise base image as in the following example, which specifies version 3.1r1:

```
FROM hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
```

Then add any further Dockerfile commands as needed.

The following example Dockerfile builds a HAProxy Enterprise version 3.1r1 image that includes a custom set of tools. The custom tool installation script, `install_mytools-4.3.sh`, must reside in the current directory.

```
FROM hapee-registry.haproxy.com/haproxy-enterprise:3.1r1

# install internal tools
WORKDIR /tmp
ARG mytools_version="4.3"
COPY install_mytools-"$mytools_version".sh .
RUN  /tmp/install_mytools-"$mytools_version".sh
```

See also

- To set the maximum amount of time allowed for a clean soft-stop, see hard-stop-after reference.
- For complete information on stopping and restarting HAProxy, see the Management Guide.

Install HAProxy Enterprise as a virtual machine image

Install HAProxy Enterprise as a virtual machine in OpenStack

In this guide, you'll learn how to install HAProxy Enterprise in OpenStack. This tutorial demonstrates the procedure using DevStack; DevStack is the official way to install OpenStack onto a single server.

Set up your OpenStack environment

Create an OpenStack environment for installing HAProxy Enterprise:

1. Determine the required size for your Linux server. The Linux server onto which you'll install OpenStack should provide enough CPU, memory, and hard drive resources to support running virtual machines. For example, the HAProxy Enterprise virtual machine we'll create in OpenStack will have 2 CPUs, 4 GB RAM, and a 40 GB hard drive. So, the server hosting OpenStack should have at least twice that much.

2. Create a Linux virtual server having the required configuration. The DevStack Quick Start recommends using Ubuntu 22.04 (Jammy).

3. Follow the DevStack Quick Start to install the hypervisor software onto your Linux server.

   - Part of the installation is to define the file `local.conf`:

   local.conf

   ```
   [[local|localrc]]
   ADMIN_PASSWORD=secret
   DATABASE_PASSWORD=$ADMIN_PASSWORD
   RABBIT_PASSWORD=$ADMIN_PASSWORD
   SERVICE_PASSWORD=$ADMIN_PASSWORD
   PUBLIC_INTERFACE=lo
   HOST_IP=0.0.0.0
   ```

   Be sure to:

   - Set the network interface to use for assigning floating IP addresses, which you'll use to access virtual machines. We do that by setting the `PUBLIC_INTERFACE` field to the loopback interface, `lo`, to access VMs from the host. As described in the DevStack Networking guide, you can set a different interface to make guest virtual machines accessible on your network, such as `eth0`.
   - Set the IP address on which to serve the OpenStack user interface. We accept traffic on all bound IP addresses by setting `HOST_IP` to `0.0.0.0`.

   The final step of installing DevStack is to call `./stack.sh`, which will start the OpenStack services and make the CLI command `openstack` available.

Deploy an HAProxy Enterprise VM

In this tutorial, we'll deploy HAProxy Enterprise by invoking `openstack` commands from the Linux console. But you could instead perform similar steps through the OpenStack user interface, which listens at port 80.

1. Connect to your OpenStack server.

2. To download the HAProxy Enterprise virtual machine image (`.qcow2`) that you want to install, go the URL below. Replace `<HAPROXY ENTERPRISE KEY>` with your HAProxy Enterprise license key to access the repository.

   ```
   https://www.haproxy.com/download/hapee/key/<HAPROXY ENTERPRISE KEY>-openstack
   ```

   You can use the command `wget <URL of file to download>` to download a file.

3. To upload the `.qcow2` virtual machine image to OpenStack, call `openstack image create`. Replace `<image name>` with the name of the file you downloaded:

   ```
   openstack image create \
     --os-auth-url "http://localhost/identity" \
     --os-project-name demo \
     --os-project-domain-name Default \
     --os-auth-type=v3password \
     --os-username admin \
     --os-password secret \
     --os-user-domain-name Default \
     --disk-format qcow2 \
     --container-format bare \
     --public \
     --file ./<image name>.qcow2 \
     haproxy-enterprise
   ```

4. To create a security group, which acts as a firewall for the virtual machine you'll create, call `openstack security group create`.

```
openstack security group create \
  --os-auth-url "http://localhost/identity" \
  --os-project-name demo \
  --os-project-domain-name Default \
  --os-auth-type=v3password \
  --os-username admin \
  --os-password secret \
  --os-user-domain-name Default \
  --description "Load balancer security group" \
  load-balancer-sg
```

5. Call `openstack security group rule create` to add rules to the security group to allow traffic to reach the virtual machine. For example, here we add rules that allow SSH (port 22), HTTP (port 80), and ICMP traffic:

```
openstack security group rule create \
  --os-auth-url "http://localhost/identity" \
  --os-project-name admin \
  --os-project-domain-name default \
  --os-user-domain-name default \
  --os-auth-type=v3password \
  --os-username admin \
  --os-password secret \
  --protocol tcp \
  --dst-port 22 \
  load-balancer-sg

openstack security group rule create \
  --os-auth-url "http://localhost/identity" \
  --os-project-name admin \
  --os-project-domain-name default \
  --os-user-domain-name default \
  --os-auth-type=v3password \
  --os-username admin \
  --os-password secret \
  --protocol tcp \
  --dst-port 80 \
  load-balancer-sg

openstack security group rule create \
  --os-auth-url "http://localhost/identity" \
  --os-project-name admin \
  --os-project-domain-name default \
  --os-user-domain-name default \
  --os-auth-type=v3password \
  --os-username admin \
  --os-password secret \
  --protocol icmp \
  --dst-port 0 \
  load-balancer-sg
```

6. Create an SSH key pair that will allow you to connect to the HAProxy Enterprise virtual machine. For example, here we create one for a user named `fusion` by calling `ssh-keygen`. The files are saved to your home directory's `.ssh` directory.

```
ssh-keygen -t ed25519 -m PEM -C "fusion@example.com"
chmod 600 ~/.ssh/id_ed25519
```

7. Create a file named `cloud-init.yaml` that defines users to add to the virtual machine. For example, here we create a user named `fusion` and set `ssh_authorized_keys` to the public key we just created, which is saved to `~/.ssh/id_ed25519.pub`:

cloud-init.yaml

```
#cloud-config
users:
- name: fusion
  gecos: Fusion
  primary_group: fusion
  groups: wheel,users,sudo
  sudo: ALL=(ALL) NOPASSWD:ALL
  shell: /bin/bash
  lock_passwd: false
  plain_text_passwd: mypassword
  ssh_authorized_keys:
  - ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGbr3mjpystKU/4GQrh6AKggqM20vg9JZaxFJQQYU3m4 fusion@example.com
```

8. You can list available virtual machine flavors, which define an instance's compute, memory, and storage capacity, by calling `openstack flavor list`.

```
openstack flavor list \
  --os-auth-url "http://localhost/identity" \
  --os-project-name demo \
  --os-project-domain-name default \
  --os-auth-type=v3password \
  --os-username admin \
  --os-password secret \
  --os-user-domain-name default
```

output

```
+----+-----------+-------+------+-----------+-------+-----------+
| ID | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+-------+------+-----------+-------+-----------+
| 1  | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2  | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3  | m1.medium |  4096 |   40 |         0 |     2 | True      |
| 4  | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 42 | m1.nano   |   192 |    1 |         0 |     1 | True      |
| 5  | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
| 84 | m1.micro  |   256 |    1 |         0 |     1 | True      |
| c1 | cirros256 |   256 |    1 |         0 |     1 | True      |
| d1 | ds512M    |   512 |    5 |         0 |     1 | True      |
| d2 | ds1G      |  1024 |   10 |         0 |     1 | True      |
| d3 | ds2G      |  2048 |   10 |         0 |     2 | True      |
| d4 | ds4G      |  4096 |   20 |         0 |     4 | True      |
+----+-----------+-------+------+-----------+-------+-----------+
```

9. Create the HAProxy Enterprise virtual machine by calling `openstack server create`. Here, we create an `m1.medium` machine using the security group, VM image, and `cloud-init.yaml` file we created previously:

```
openstack server create \
  --os-auth-url "http://localhost/identity" \
  --os-project-name demo \
  --os-project-domain-name Default \
  --os-auth-type=v3password \
  --os-username admin \
  --os-password secret \
  --os-user-domain-name Default \
  --flavor m1.medium \
  --network private \
  --image haproxy-enterprise \
  --security-group load-balancer-sg \
  --user-data cloud-init.yaml \
  loadbalancer1
```

   This could take several minutes to complete.

10. To create a floating IP address that will allow you to access the virtual machine from the host server, call `openstack floating ip create`.

```
openstack floating ip create \
  --os-auth-url "http://localhost/identity" \
  --os-project-name demo \
  --os-project-domain-name Default \
  --os-auth-type=v3password \
  --os-username admin \
  --os-password secret \
  --os-user-domain-name Default \
  public
```

    The floating IP address `172.24.4.248` was created:

output

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2025-03-25T21:17:20Z                 |
| description         |                                      |
| dns_domain          |                                      |
| dns_name            |                                      |
| fixed_ip_address    | None                                 |
| floating_ip_address | 172.24.4.248                         |
| floating_network_id | 5bb7d124-0227-4ea6-bede-2d276911b53b |
| id                  | bc0557c7-3123-44d2-bcdb-e6b351171fe8 |
| name                | 172.24.4.248                         |
| port_details        | None                                 |
| port_id             | None                                 |
| project_id          | 4a36ab79893548a6a424e8daf92f5d8b     |
| qos_policy_id       | None                                 |
| revision_number     | 0                                    |
| router_id           | None                                 |
| status              | DOWN                                 |
| subnet_id           | None                                 |
| tags                | []                                   |
| updated_at          | 2025-03-25T21:17:20Z                 |
```

11. Assign the floating IP address to the virtual machine:

```
openstack server add floating ip \
  --os-auth-url "http://localhost/identity" \
  --os-project-name demo \
  --os-project-domain-name Default \
  --os-auth-type=v3password \
  --os-username admin \
  --os-password secret \
  --os-user-domain-name Default \
  loadbalancer1 \
  172.24.4.248
```

12. At this point, your HAProxy Enterprise virtual machine should be running. To check, run `openstack server list`.

```
openstack server list \
  --os-auth-url "http://localhost/identity" \
  --os-project-name demo \
  --os-project-domain-name Default \
  --os-auth-type=v3password \
  --os-username admin \
  --os-password secret \
  --os-user-domain-name Default
```

output

```
+-------------+--------------+--------+--------------------------------+-------------------+-----------+
| ID          | Name         | Status | Networks                       | Image             | Flavor    |
+-------------+--------------+--------+--------------------------------+-------------------+-----------+
| 2eff8628... | loadbalancer1 | ACTIVE | private=10.0.0.19, 172.24.4.248 | haproxy-enterprise | m1.medium |
+-------------+--------------+--------+--------------------------------+-------------------+-----------+
```

From the OpenStack server, make an SSH connection to it:

```
ssh -i ~/.ssh/id_ed25519 fusion@172.24.4.248
```

output

```
The authenticity of host '172.24.4.248 (172.24.4.248)' can't be established.
ED25519 key fingerprint is SHA256:zLCERG3GGz6iXS3ngrRmHZhYxuza1yz09GeM+Kg9vnA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

Warning: Permanently added '172.24.4.248' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Wed Mar  5 07:49:59 UTC 2025

  System load:  0.400390625       Processes:             104
  Usage of /:   7.0% of 19.20GB   Users logged in:       0
  Memory usage: 2%                IPv4 address for ens3: 192.168.222.194
  Swap usage:   0%
```

13. If you're using HAProxy Fusion, next add the HAProxy Enterprise instance as a Fusion node.

Install HAProxy Enterprise as a virtual machine in VMware vSphere

In this guide, you'll learn how to install HAProxy Enterprise in VMware vSphere.

Download and verify the OVA image

Download the `.ova` image and verify it using the checksum.

1. Download the `.ova` file and the `SHA256SUMS` file from the repository:

```
https://www.haproxy.com/download/hapee/key/<HAPROXY ENTERPRISE KEY>-vmware
```

2. The `SHA256SUMS` file contains checksums for the `.ova` files:

SHA256SUMS

```
ae43142589b3ee...  hapee-rhel-8-amd64-2.6r1-20250121.ova
06ca84c7da61bf...  hapee-rhel-9-amd64-2.8r1-20250121.ova
f01a2a6263ebfe...  hapee-rhel-9-amd64-2.9r1-20250121.ova
...
```

3. Get the checksum for the `.ova` file you downloaded. For example:

Linux:

Use the command `shasum` to get the checksum of the `.ova` file:

```
shasum -a 256 hapee-rhel-9-amd64-3.0r1-20250121.ova
```

output

```
eac5f1f35a386dec73c0bef8769848014f76d0cf7688a651538c1d89cc3e4e4b  hapee-rhel-9-amd64-3.0r1-20250121.ova
```

Windows:

Use the PowerShell command `Get-FileHash` to get the checksum of the `.ova` file:

```
Get-FileHash hapee-rhel-9-amd64-3.0r1-20250121.ova
```

output

```
Algorithm       Hash
---------       ----
SHA256          EAC5F1F35A386DEC73C0BEF8769848014F76D0CF7688A651538C1D89CC3E4E4B
```

4. Verify that the checksum matches the line for that `.ova` in the `SHA256SUMS` file.

Install the OVA image

To install the OVA image:

1. Import one of the `.ova` files from the VM Templates repository into your VMware hypervisor. Each file installs a different version of HAProxy Enterprise on Ubuntu or Red Hat Enterprise Linux. Replace `<HAPROXY ENTERPRISE KEY>` in the following URL with your HAProxy Enterprise license key to access the repository.

   ```
   https://www.haproxy.com/download/hapee/key/<HAPROXY ENTERPRISE KEY>-vmware
   ```

2. When you import the OVA file into your VMware hypervisor to create a new virtual machine, you will be asked to provide:

   - a name for the virtual machine (example: `hapee-vm`)
   - a storage path for the virtual machine (prefilled automatically, but you can change it)
   - a unique instance ID for the VM instance (example: `hapee1`)
   - a hostname for the VM instance (example: `hapeevm`)
   - an SSH public key for connecting to the instance (example: `ssh-ed25519 ABCD... My PC`)
   - the default user's password (example: `mypassword`)

   If you don't provide a password, a random one will be generated. It will display above the sign-in prompt in the terminal.

3. Start the VM.

4. Log in with the username `ubuntu` or `cloud-user` and the password you set during the installation.

5. Verify the network interface driver is the `VMXNET3` driver. We recommend you use this driver because it provides superior performance.

> **Info**
>
> The OVA image is already configured to use the `VMXNET3` network interface driver, but if you add more interfaces, the hypervisor may provide a different driver as the default. Make sure you specify the `VMXNET3` interface driver instead. In particular, avoid the `E1000` driver, which is not optimized for use in a hypervisor.

Install HAProxy Enterprise on Linux

Using HAProxy Fusion?

If you're using HAProxy Fusion, then see the HAProxy Fusion - Nodes topic instead.

This section describes how to install HAProxy Enterprise on Linux.

Hardware recommendations

The hardware requirements for HAProxy Enterprise depend on the workload it needs to manage:

- Only CPU and memory are taken into consideration.
- Disk size depends on your operating system and the volume of logs you want to keep.
- The indications below are for information only. Please contact us for assistance in sizing your servers.

Low-level workload

- TCP or HTTP traffic
- Up to 1000 conn/s
- Very low SSL traffic or gzip compression

This type of workload can be achieved either by a Virtual Machine or a bare metal server. You need at least:

- 1 CPU core
- 2 GB of RAM

Mid-level workload

- TCP or HTTP traffic (including HTTP manipulation)
- Up to 4000 conn/s
- Low SSL traffic or gzip compression

This type of workload can be achieved either by a Virtual Machine or a bare metal server. You need at least:

- 2 CPU cores
- 2 GB of RAM

High-level workload

- TCP or HTTP traffic (including HTTP manipulation)
- Up to 20000 conn/s
- 10% of traffic ciphered (SSL) or compressed

This type of workload can be achieved by a bare metal server only. You need at least:

- 2 CPU cores, as fast as possible
- 4G of RAM
- powerful network card

Supported operating systems

HAProxy Enterprise is distributed through the Operating System package manager for the following Linux distributions:

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
3.1r1	Feb 2025	Feb 2026	• AlmaLinux 8 (x86_64) AlmaLinux 9 (x86_64) • Debian 11 (amd64) Debian 12 (amd64) • RHEL 8 (x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.6 (x86_64) • Ubuntu 22.04 (amd64, arm64) Ubuntu 24.04 (amd64, arm64)

- Other HAProxy Enterprise versions

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
3.1r1	Feb 2025	Feb 2026	• AlmaLinux 8 (x86_64) AlmaLinux 9 (x86_64) • Debian 11 (amd64) Debian 12 (amd64) • RHEL 8 (x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.6 (x86_64) • Ubuntu 22.04 (amd64, arm64) Ubuntu 24.04 (amd64, arm64)
3.0r1	Oct 2024	Feb 2029	• AlmaLinux 8 (x86_64) AlmaLinux 9 (x86_64) • Debian 11 (amd64) Debian 12 (amd64) • RHEL 8 (x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.6 (x86_64) • Ubuntu 22.04 (amd64, arm64) Ubuntu 24.04 (amd64, arm64)
2.9r1	May 2024	Feb 2025	

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
			• AlmaLinux 8 (x86_64) AlmaLinux 9 (x86_64) • Debian 11 (amd64) Debian 12 (amd64) • RHEL 8 (x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.5 (x86_64) SUSE 15.6 (x86_64) • Ubuntu 20.04 (amd64, arm64) Ubuntu 22.04 (amd64, arm64) Ubuntu 24.04 (amd64, arm64)
2.8r1	Oct 2023	Feb 2028	• AlmaLinux 8 (aarch64, x86_64) AlmaLinux 9 (x86_64) • Debian 11 (amd64) Debian 12 (amd64) • Photon 3 (x86_64) • RHEL 8 (aarch64, x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (aarch64, x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.5 (x86_64) SUSE 15.6 (x86_64) • Ubuntu 20.04 (amd64, arm64) Ubuntu 22.04 (amd64, arm64) Ubuntu 24.04 (amd64, arm64)
2.7r1	Feb 2023	Feb 2024	

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
			• AlmaLinux 8 (x86_64) AlmaLinux 9 (x86_64) • Debian 11 (amd64) Debian 12 (amd64) • Photon 3 (x86_64) • RHEL 8 (x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.0 (x86_64) SUSE 15.1 (x86_64) SUSE 15.2 (x86_64) SUSE 15.3 (x86_64) SUSE 15.4 (x86_64) • Ubuntu 20.04 (amd64, arm64) Ubuntu 22.04 (amd64, arm64)

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
2.6r1	Sep 2022	Feb 2027	• AlmaLinux 8 (aarch64, x86_64) AlmaLinux 9 (x86_64) • Debian 10 (amd64) Debian 11 (amd64) Debian 12 (amd64) • Photon 3 (x86_64) • RHEL 7 (aarch64, x86_64) RHEL 8 (aarch64, x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (aarch64, x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.0 (x86_64) SUSE 15.1 (x86_64) SUSE 15.2 (x86_64) SUSE 15.3 (x86_64) SUSE 15.4 (x86_64) • Ubuntu 18.04 (amd64) Ubuntu 20.04 (amd64, arm64) Ubuntu 22.04 (amd64, arm64) Ubuntu 24.04 (amd64, arm64)
2.5r1	Feb 2022	Nov 2023	

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
			• AlmaLinux 8 (x86_64) AlmaLinux 9 (x86_64) • Debian 9 (amd64) Debian 10 (amd64) Debian 11 (amd64) • Photon 3 (x86_64) • RHEL 7 (aarch64, x86_64) RHEL 8 (x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.0 (x86_64) SUSE 15.1 (x86_64) SUSE 15.2 (x86_64) SUSE 15.3 (x86_64) • Ubuntu 18.04 (amd64) Ubuntu 20.04 (amd64, arm64) Ubuntu 22.04 (amd64, arm64)

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
2.4r1	Nov 2021	Feb 2026	• AlmaLinux 8 (aarch64, x86_64) AlmaLinux 9 (x86_64) • Debian 9 (amd64) Debian 10 (amd64) Debian 11 (amd64) • Photon 3 (x86_64) • RHEL 7 (aarch64, x86_64) RHEL 8 (aarch64, x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (aarch64, x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.0 (x86_64) SUSE 15.1 (x86_64) SUSE 15.2 (x86_64) SUSE 15.3 (x86_64) • Ubuntu 18.04 (amd64) Ubuntu 20.04 (amd64, arm64) Ubuntu 22.04 (amd64, arm64)
2.3r1	Feb 2021	Nov 2022	• Debian 9 (amd64) Debian 10 (amd64) Debian 11 (amd64) • Photon 3 (x86_64) • RHEL 7 (x86_64) RHEL 8 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.0 (x86_64) SUSE 15.1 (x86_64) SUSE 15.2 (x86_64) SUSE 15.3 (x86_64) • Ubuntu 18.04 (amd64) Ubuntu 20.04 (amd64)
2.2r1	Nov 2020	Feb 2025	

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
			• AlmaLinux 8 (x86_64) • Debian 9 (amd64) Debian 10 (amd64) Debian 11 (amd64) • Photon 3 (x86_64) • RHEL 7 (x86_64) RHEL 8 (x86_64) • Rocky Linux 8 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.0 (x86_64) SUSE 15.1 (x86_64) SUSE 15.2 (x86_64) SUSE 15.3 (x86_64) • Ubuntu 16.04 (amd64) Ubuntu 18.04 (amd64) Ubuntu 20.04 (amd64)
2.1r1	Feb 2020	Nov 2021	• Debian 8 (amd64) Debian 9 (amd64) Debian 10 (amd64) • RHEL 7 (x86_64) RHEL 8 (ppc64le, x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.0 (x86_64) SUSE 15.1 (x86_64) • Ubuntu 14.04 (amd64) Ubuntu 16.04 (amd64) Ubuntu 18.04 (amd64) Ubuntu 20.04 (amd64)
2.0r1	Nov 2019	Feb 2024	

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
			• AlmaLinux 8 (x86_64) • Debian 8 (amd64) Debian 9 (amd64) Debian 10 (amd64) • RHEL 7 (x86_64) RHEL 8 (x86_64) • Rocky Linux 8 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.0 (x86_64) SUSE 15.1 (x86_64) SUSE 15.2 (x86_64) SUSE 15.3 (x86_64) • Ubuntu 14.04 (amd64) Ubuntu 16.04 (amd64) Ubuntu 18.04 (amd64)
1.9r1	Feb 2019	Nov 2020	• Debian 8 (amd64) Debian 9 (amd64) Debian 10 (amd64) • RHEL 7 (x86_64) RHEL 8 (x86_64) • Rocky Linux 8 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.0 (x86_64) • Ubuntu 14.04 (amd64) Ubuntu 16.04 (amd64) Ubuntu 18.04 (amd64)
1.8r2	Nov 2018	Feb 2023	• Debian 8 (amd64) Debian 9 (amd64) • RHEL 7 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • Ubuntu 14.04 (amd64) Ubuntu 16.04 (amd64) Ubuntu 18.04 (amd64)

Installation

The following procedure adds package repositories and installs HAProxy Enterprise 3.1r1.

1. Download the installer:

```
wget https://www.haproxy.com/static/install_haproxy_enterprise.sh
```

2. Optional: To verify the integrity of the install script before installing, download the SHA hash to a local directory and use it to verify the install script's checksum:

```
wget https://www.haproxy.com/static/install_haproxy_enterprise.sh.sha512.asc
gpg --keyserver hkp://keyserver.ubuntu.com --recv-keys 0xCA2DF14657C5A207
gpg --verify ./install_haproxy_enterprise.sh.sha512.asc
```

   Check for the output `Good signature`.

3. To install HAProxy Enterprise, run the following command, replacing `<HAProxy Enterprise Key>` with the key you were given when you registered (see HAProxy Enterprise license key). Register for a trial.

```
sudo bash ./install_haproxy_enterprise.sh --version "3.1r1" --key "<HAProxy Enterprise key>"
```

   To see other arguments, run `./install_haproxy_enterprise.sh --help`.

4. Enable and start the HAProxy Enterprise service:

```
sudo systemctl enable hapee-3.1-lb
sudo systemctl start hapee-3.1-lb
```

   Messages may appear, stating that backend servers are not available. This condition is expected and occurs because the default configuration file contains stubs for backend servers. Later you will modify the configuration and replace the stubs with valid server addresses.

About package repositories

HAProxy Enterprise adds package repositories via the file `haproxy-tech.list` or `haproxy-tech.repo`. The table below describes these repositories.

Package repository	Description
Common	Contains the primary components for HAProxy Enterprise.
Plus	Contains add-on modules that extend HAProxy Enterprise.
Extras	Contains supporting software: SNMP, RHI, VRRP, etc.

Search for additional modules

The list of additional modules is also available by running the following commands:

Apt:

```
apt-cache search hapee-<VERSION>
apt-cache search hapee-extras
```

Example for HAProxy Enterprise 3.1r1:

```
apt-cache search hapee-3.1r1
apt-cache search hapee-extras
```

Yum:

```
yum search hapee-<VERSION>
yum search hapee-extras
```

Example for HAProxy Enterprise 3.1r1:

```
yum search hapee-3.1r1
yum search hapee-extras
```

Zypper:

```
zypper search hapee-<VERSION>
zypper search hapee-extras
```

Example for HAProxy Enterprise 3.1r1:

```
zypper search hapee-3.1r1
zypper search hapee-extras
```

See other parts of this documentation for instructions on how to enable and configure each package.

Locate installed directories

Binaries and documentation

```
/opt/hapee-3.1/
  |-- bin
  |-- doc
  |-- modules
  |-- sbin
  |-- version
```

Configuration files

```
/etc/hapee-3.1/
  |-- hapee-lb.cfg
```

Init scripts

```
/etc/init.d/
        |-- hapee-3.1-lb
```

View module dependencies

This section applies to:

- HAProxy Enterprise 2.4r1 and newer

To view the module dependencies for your installed version of HAProxy Enterprise, you can use the `hapee-lb-rdepends` tool that is installed with HAProxy Enterprise. This tool is located at `/opt/hapee-3.1/bin/hapee-lb-rdepends`.

To use the tool:

1. Run the `hapee-lb` executable with the `-v` option to identify the version and build of your HAProxy Enterprise installation.

   ```
   /opt/hapee-2.8/sbin/hapee-lb -v
   ```

   output

   ```
   HAProxy version 2.8.0-1.0.0-310.418 2023/12/14 - https://haproxy.org/
   [...]
   ```

   The version is the first part of the output after "HAProxy version". In this example it is `2.8`. The build is the numbers following the first dash (`-`). In this example it is `1.0.0-310.418`.

2. Run the `hapee-lb-rdepends` tool, providing the values for the parameters `--version`, `--build`, and `--key` as follows:

   - `--version` is the version you retrieved in the previous step. For this example, version `2.8` we will specify `2.8r1` as the version.
   - `--build` is the build you retrieved in the previous step
   - `--key` is your HAProxy Enterprise license key

   ```
   /opt/hapee-3.1/bin/hapee-lb-rdepends -v "2.8r1" --build "1.0.0-310.418" --key "<HAProxy Enterprise key>"
   ```

output

```
hapee-2.8r1-lb-fingerprint=1.0.0-342.6
hapee-2.8r1-lb-wafadvanced=1.0.0-358.1
hapee-2.8r1-lb-update=1.0.0-596.3
hapee-2.8r1-lb-da-update=1.0.0-347.1
hapee-2.8r1-lb-wafoffloader=1.0.0-279.0
hapee-2.8r1-lb-send-metrics=1.0.0-438.1
hapee-2.8r1-lb-wurfl=1.0.0-277.418
hapee-2.8r1-lb-maxmind=1.0.0-443.2
hapee-2.8r1-lb-wurfl-update=1.0.0-340.1
hapee-2.8r1-lb-51d-update=1.0.0-469.1
hapee-2.8r1-lb-antibot=1.0.0-343.11
hapee-2.8r1-lb-da=1.0.0-280.418
hapee-2.8r1-lb-modsecurity=1.0.0-312.0
hapee-2.8r1-lb-htmldom=1.0.0-235.0
hapee-2.8r1-lb-fingerprint-ssl=1.0.0-141.0
hapee-2.8r1-lb-extensions=1.0.0-13.1
hapee-2.8r1-lb-51d=1.0.0-283.418
hapee-2.8r1-lb-netacuity=1.0.0-448.1
```

The dependencies for the specific version and build are listed.

There are some additional parameters you can provide for the `hapee-lb-rdepends` tool:

Option	Description
`--version VERSION`	HAProxy Enterprise major version (for example: `2.7r1`)
`--build BUILD`	HAProxy Enterprise build version (for example: `1.0.0-293.382`)
`--key KEY`	HAProxy Enterprise subscription key
`--arch ARCH`	HAProxy Enterprise target architecture (default: `amd64`)
`--distro DISTRO`	HAProxy Enterprise target OS distribution (default: try all supported)
`--rpm`	When this option is provided, the output will be in RPM format, for example: `hapee-2.8r1-lb-extensions-1.0.0-13.1` instead of in DEB format, for example: `hapee-2.8r1-lb-extensions=1.0.0-13.1`

Install HAProxy Enterprise manually

The following section gives detailed information on how to install HAProxy Enterprise 3.1r1 and its associated components manually on all supported Operating Systems.

Use this procedure if our installation script is not suited for your infrastructure or if you want to customize your installation.

Debian:

1. Update the repository cache and install required dependencies:

```
sudo apt-get update
sudo apt-get install --yes apt-transport-https dirmngr gnupg-agent curl
```

2. Create a new file `/etc/apt/sources.list.d/haproxy-tech.list` if it does not exist and add the contents below. Replace `<HAProxy Enterprise Key>` with the key you were given when you registered (see HAProxy Enterprise license key). Replace `<CODENAME>` with your operating system's codename (for example, `bullseye`).

haproxy-tech.list

```
deb [arch=amd64 signed-by=/etc/apt/keyrings/HAPEE-key-3.1r1.asc] https://www.haproxy.com/download/hapee/key/
<HAProxy Enterprise Key>-common/3.1r1/debian-<CODENAME>/amd64/ <CODENAME> main
deb [arch=amd64 signed-by=/etc/apt/keyrings/HAPEE-key-3.1r1.asc] https://www.haproxy.com/download/hapee/key/
<HAProxy Enterprise Key>-plus/3.1r1/debian-<CODENAME>/amd64/ <CODENAME> main
deb [arch=amd64 signed-by=/etc/apt/keyrings/HAPEE-key-extras.asc] https://www.haproxy.com/download/hapee/key/
<HAProxy Enterprise Key>-plus/extras/debian-<CODENAME>/amd64/ <CODENAME> main
```

3. The packages that HAProxy Technologies provides are signed. We encourage you to validate the fingerprints first before installing them onto your system.

```
wget https://pks.haproxy.com/linux/enterprise/HAPEE-key-3.1r1.asc
wget https://pks.haproxy.com/linux/enterprise/HAPEE-key-extras.asc
```

Then, compare the output of the following commands with the list of expected fingerprints below:

With gpg versions < 2.1.16:

```
gpg --with-fingerprint HAPEE-key-3.1r1.asc
gpg --with-fingerprint HAPEE-key-extras.asc
```

With gpg versions > 2.1.16:

```
gpg --import --import-options show-only HAPEE-key-3.1r1.asc
gpg --import --import-options show-only HAPEE-key-extras.asc
```

output

```
# HAProxy Enterprise 3.1r1
FC381713A1C783AC76EB2005CD6DD5ABF28C0C38

# Extras
77A66FDC5D4D779E9CB9D5809ABA76BB03A731D6
```

For the PGP fingerprints of older versions, see the chart here.

4. Import the public keys:

```
sudo mkdir -p /etc/apt/keyrings
sudo curl -s -L "https://pks.haproxy.com/linux/enterprise/HAPEE-key-3.1r1.asc" -o /etc/apt/keyrings/HAPEE-key-3.1r1.asc
sudo curl -s -L "https://pks.haproxy.com/linux/enterprise/HAPEE-key-extras.asc" -o /etc/apt/keyrings/HAPEE-key-extras.asc
```

5. Update the repository cache:

```
sudo apt-get update
```

6. To install the load balancer, run:

```
sudo apt-get install hapee-3.1r1-lb
```

output

```
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
hapee-3.1r1-base hapee-3.1r1-libotc hapee-3.1r1-libotcpp libpcre2-posix2
The following NEW packages will be installed:
hapee-3.1r1-base hapee-3.1r1-lb hapee-3.1r1-libotc hapee-3.1r1-libotcpp
libpcre2-posix2
0 upgraded, 5 newly installed, 0 to remove and 25 not upgraded.
Need to get 2907 kB of archives.
After this operation, 18.7 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

7. To start HAProxy Enterprise, run:

```
sudo systemctl enable hapee-3.1-lb
sudo systemctl start hapee-3.1-lb
```

8. If you have installed Rsyslog, restart it now to begin collecting HAProxy Enterprise logs:

```
sudo systemctl restart rsyslog
```

Ubuntu:

1. Update the repository cache and install required dependencies:

```
sudo apt-get update
sudo apt-get install --yes apt-transport-https dirmngr gnupg-agent curl
```

2. Create a new file `/etc/apt/sources.list.d/haproxy-tech.list` if it does not exist and add the contents below. Replace `<HAProxy Enterprise Key>` with the key you were given when you registered (see HAProxy Enterprise license key). Replace `<VERSION>` with your operating system version number (for example, `22.04`). Replace `<CODENAME>` with your operating system's codename (for example, `jammy`).

haproxy-tech.list

```
deb [arch=amd64 signed-by=/etc/apt/keyrings/HAPEE-key-3.1r1.asc] https://www.haproxy.com/download/hapee/key/
<HAProxy Enterprise Key>-common/3.1r1/ubuntu-<VERSION>/amd64/ <CODENAME> main
deb [arch=amd64 signed-by=/etc/apt/keyrings/HAPEE-key-3.1r1.asc] https://www.haproxy.com/download/hapee/key/
<HAProxy Enterprise Key>-plus/3.1r1/ubuntu-<VERSION>/amd64/ <CODENAME> main
deb [arch=amd64 signed-by=/etc/apt/keyrings/HAPEE-key-extras.asc] https://www.haproxy.com/download/hapee/key/
<HAProxy Enterprise Key>-plus/extras/ubuntu-<VERSION>/amd64/ <CODENAME> main
```

3. The packages that HAProxy Technologies provides are signed. We encourage you to validate the fingerprints first before installing them onto your system.

```
wget https://pks.haproxy.com/linux/enterprise/HAPEE-key-3.1r1.asc
wget https://pks.haproxy.com/linux/enterprise/HAPEE-key-extras.asc
```

Then, compare the output of the following commands with the list of expected fingerprints below:

With gpg versions < 2.1.16:

```
gpg --with-fingerprint HAPEE-key-3.1r1.asc
gpg --with-fingerprint HAPEE-key-extras.asc
```

With gpg versions > 2.1.16:

```
gpg --import --import-options show-only HAPEE-key-3.1r1.asc
gpg --import --import-options show-only HAPEE-key-extras.asc
```

output

```
# HAProxy Enterprise 3.1r1
FC381713A1C783AC76EB2005CD6DD5ABF28C0C38

# Extras
77A66FDC5D4D779E9CB9D5809ABA76BB03A731D6
```

For the PGP fingerprints of older versions, see the chart here.

4. Import the public keys:

```
sudo mkdir -p /etc/apt/keyrings
sudo curl -s -L "https://pks.haproxy.com/linux/enterprise/HAPEE-key-3.1r1.asc" -o /etc/apt/keyrings/HAPEE-key-3.1r1.asc
sudo curl -s -L "https://pks.haproxy.com/linux/enterprise/HAPEE-key-extras.asc" -o /etc/apt/keyrings/HAPEE-key-extras.asc
```

5. Update the repository cache:

```
sudo apt-get update
```

6. To install the load balancer, run:

```
sudo apt-get install hapee-3.1r1-lb
```

output

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
        hapee-3.1r1-base openssl
Suggested packages:
        ca-certificates
The following NEW packages will be installed:
        hapee-3.1r1-base hapee-3.1r1-lb openssl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
[...]
```

7. To start HAProxy Enterprise, run:

```
sudo systemctl enable hapee-3.1-lb
sudo systemctl start hapee-3.1-lb
```

8. If you have installed Rsyslog, restart it now to begin collecting HAProxy Enterprise logs:

```
sudo systemctl restart rsyslog
```

Redhat:

1. Create a new file `/etc/yum.repos.d/haproxy-tech.repo` if it does not exist and add the contents below. Replace `<HAProxy Enterprise Key>` with the key you were given when you registered (see HAProxy Enterprise license key). Replace `<VERSION>` with your operating system's version number (for example, `8`).

   haproxy-tech.repo

   ```
   [hapee-base]
   name=hapee-base
   enabled=1
   baseurl=https://www.haproxy.com/download/hapee/key/<HAProxy Enterprise Key>-common/3.1r1/rhel-<VERSION>/
   $basearch/bin/
   gpgcheck=1

   [hapee-plus]
   name=hapee-plus
   enabled=1
   baseurl=https://www.haproxy.com/download/hapee/key/<HAProxy Enterprise Key>-plus/3.1r1/rhel-<VERSION>/$basearch/
   bin/
   gpgcheck=1

   [hapee-plus-extras]
   name=hapee-plus-extras
   enabled=1
   baseurl=https://www.haproxy.com/download/hapee/key/<HAProxy Enterprise Key>-plus/extras/rhel-<VERSION>/
   $basearch/bin/
   gpgcheck=1
   ```

2. The packages that HAProxy Technologies provides are signed. To install them, you first must import the public key.

   Run the following commands:

   ```
   rpm --import https://pks.haproxy.com/linux/enterprise/HAPEE-key-3.1r1.asc
   rpm --import https://pks.haproxy.com/linux/enterprise/HAPEE-key-extras.asc
   ```

   We encourage you to validate the fingerprints first before installing them onto your system.

   ```
   wget https://pks.haproxy.com/linux/enterprise/HAPEE-key-3.1r1.asc
   wget https://pks.haproxy.com/linux/enterprise/HAPEE-key-extras.asc
   ```

   Then, compare the output of the following commands with the list of expected fingerprints below:

   With gpg versions < 2.1.16:

```
gpg --keyid-format long --with-fingerprint HAPEE-key-3.1r1.asc
gpg --keyid-format long --with-fingerprint HAPEE-key-extras.asc
```

With gpg versions > 2.1.16:

```
gpg --import --import-options show-only HAPEE-key-3.1r1.asc
gpg --keyid-format long --with-fingerprint HAPEE-key-extras.asc
```

output

```
# HAProxy Enterprise 3.1r1
FC381713A1C783AC76EB2005CD6DD5ABF28C0C38

# Extras
77A66FDC5D4D779E9CB9D5809ABA76BB03A731D6
```

For the PGP fingerprints of older versions, see the chart here.

3. Update the repository cache:

```
yum makecache
```

4. To install the load balancer, run:

```
yum install -y hapee-3.1r1-lb
```

Output of a successful installation:

output

```
[...]
Running Transaction
Installing : hapee-3.1r1-base-3.1r1.0-16.0.noarch      1/2
Note: you should edit /etc/sysctl.conf for system tuning.
Installing : hapee-3.1r1-lb-3.1r1.0-67.20.x86_64       2/2
Verifying  : hapee-3.1r1-base-3.1r1.0-16.0.noarch      1/2
Verifying  : hapee-3.1r1-lb-3.1r1.0-67.20.x86_64       2/2
Installed:
hapee-3.1r1-lb.x86_64 0:3.1r1.0-67.20
Dependency Installed:
hapee-3.1r1-base.noarch 0:3.1r1.0-16.0
Complete!
```

5. To start HAProxy Enterprise, run:

```
sudo systemctl enable hapee-3.1-lb
sudo systemctl start hapee-3.1-lb
```

6. If you have installed Rsyslog, restart it now to begin collecting HAProxy Enterprise logs:

```
sudo systemctl restart rsyslog
```

Suse:

1. Create a new file `/etc/zypp/repos.d/haproxy-tech.repo` if it does not exist and add the contents below. Replace `<HAProxy Enterprise Key>` with the key you were given when you registered (see HAProxy Enterprise license key). Replace `<VERSION>` with your operating system's version number (for example, `15.5`).

```
[hapee-base]
name=HAProxy Enterprise Base
enabled=1
autorefresh=1
baseurl=https://www.haproxy.com/download/hapee/key/<HAProxy Enterprise Key>-common/3.1r1/suse-<VERSION>/x86_64/
bin/
path=/
type=rpm-md
keeppackages=0

[hapee-plus]
name=HAProxy Enterprise Base
enabled=1
autorefresh=1
baseurl=https://www.haproxy.com/download/hapee/key/<HAProxy Enterprise Key>-plus/3.1r1/suse-<VERSION>/x86_64/
bin/
path=/
type=rpm-md
keeppackages=0

[hapee-plus-extras]
name=HAProxy Enterprise Base
enabled=1
autorefresh=1
baseurl=https://www.haproxy.com/download/hapee/key/<HAProxy Enterprise Key>-plus/extras/suse-<VERSION>/x86_64/
bin/
path=/
type=rpm-md
keeppackages=0
```

2. Import the keys for the HAProxy Enterprise repositories.

```
sudo rpm --import https://pks.haproxy.com/linux/enterprise/HAPEE-key-3.1r1.asc
```

3. Update the repository cache:

```
sudo zypper refresh
```

The operation may report that repositories are signed with unknown keys. When prompted whether to continue, enter `yes`.

4. Install the load balancer:

```
sudo zypper install -y hapee-<VERSION>-lb
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install -y hapee-3.1r1-lb
```

Example of installation output for openSUSE 15.5:

output

```
Refreshing service 'Basesystem_Module_x86_64'.
Refreshing service 'Containers_Module_x86_64'.
Refreshing service 'Desktop_Applications_Module_x86_64'.
Refreshing service 'Development_Tools_Module_x86_64'.
Refreshing service 'Public_Cloud_Module_x86_64'.
Refreshing service 'Python_3_Module_x86_64'.
Refreshing service 'SUSE_Linux_Enterprise_Server_x86_64'.
Refreshing service 'Server_Applications_Module_x86_64'.
Refreshing service 'Web_and_Scripting_Module_x86_64'.
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following 7 NEW packages are going to be installed:
  hapee-3.1r1-base hapee-3.1r1-lb hapee-3.1r1-libotc hapee-3.1r1-libotcpp insserv-compat libpcre2-posix3
sysvinit-tools

The following 4 packages have no support information from their vendor:
  hapee-3.1r1-base hapee-3.1r1-lb hapee-3.1r1-libotc hapee-3.1r1-libotcpp

7 new packages to install.
Overall download size: 7.4 MiB. Already cached: 0 B. After the operation, additional 33.3 MiB will be used.

Backend:  classic_rpmtrans
Continue? [y/n/v/...? shows all options] (y): y
Retrieving: libpcre2-posix3-10.42-150600.1.26.x86_64 (SLE-Module-Basesystem15-SP6-
Pool)                                                                    (1/7),  30.1 KiB
Retrieving: libpcre2-
posix3-10.42-150600.1.26.x86_64.rpm ..................................................................
.....................................[done]
Retrieving: sysvinit-tools-2.99-1.1.x86_64 (SLE-Module-Basesystem15-SP6-
Pool)                                                                   (2/7), 132.3 KiB
Retrieving: sysvinit-
tools-2.99-1.1.x86_64.rpm ............................................................................
.....................................[done]
Retrieving: insserv-compat-0.1-4.6.1.noarch (SLE-Module-Basesystem15-SP6-
Pool)                                                                   (3/7),  15.0 KiB
Retrieving: insserv-
compat-0.1-4.6.1.noarch.rpm ..........................................................................
.....................................[done]
Retrieving: hapee-3.1r1-base-1.0.0-110.0.noarch (HAProxy Enterprise
Base)                                                                       (4/7),  43.5 KiB
Retrieving: hapee-3.1r1-
base-1.0.0-110.0.suse-15.6.noarch.rpm ................................................................
.....................[done (37.6 KiB/s)]
Retrieving: hapee-3.1r1-libotcpp-1.0.0-15.2.x86_64 (HAProxy Enterprise
Base)                                                                   (5/7), 296.1 KiB
```

```
Retrieving: hapee-3.1r1-
libotcpp-1.0.0-15.2.suse-15.6.x86_64.rpm ..........................................................
.....................[done (33.4 KiB/s)]
Retrieving: hapee-3.1r1-libotc-1.0.0-21.15.x86_64 (HAProxy Enterprise
Base)                                                                          (6/7), 973.9 KiB
Retrieving: hapee-3.1r1-
libotc-1.0.0-21.15.suse-15.6.x86_64.rpm ..........................................................
.....................[done (70.2 KiB/s)]
Retrieving: hapee-3.1r1-lb-1.0.0-329.537.x86_64 (HAProxy Enterprise
Base)                                                                           (7/7),   6.0 MiB
Retrieving: hapee-3.1r1-
lb-1.0.0-329.537.suse-15.6.x86_64.rpm ............................................................
.......................[done (1.7 MiB/s)]

Checking for file
conflicts: .......................................................................................
.......................................[done]
(1/7) Installing: libpcre2-
posix3-10.42-150600.1.26.x86_64 ..................................................................
................................[done]
(2/7) Installing: sysvinit-
tools-2.99-1.1.x86_64 ............................................................................
................................[done]
(3/7) Installing: insserv-
compat-0.1-4.6.1.noarch ..........................................................................
................................[done]
    Note: you should edit /etc/sysctl.d/hapee-3.1.conf for system tuning.
(4/7) Installing: hapee-3.1r1-
base-1.0.0-110.0.noarch ..........................................................................
.............................[done]
(5/7) Installing: hapee-3.1r1-
libotcpp-1.0.0-15.2.x86_64 .......................................................................
.............................[done]
(6/7) Installing: hapee-3.1r1-
libotc-1.0.0-21.15.x86_64 ........................................................................
.............................[done]
(7/7) Installing: hapee-3.1r1-
lb-1.0.0-329.537.x86_64 ..........................................................................
.............................[done]
```

5. To start HAProxy Enterprise, run:

```
sudo systemctl enable hapee-3.1-lb
sudo systemctl start hapee-3.1-lb
```

6. If you have installed Rsyslog, restart it now to begin collecting HAProxy Enterprise logs:

```
sudo systemctl restart rsyslog
```

System tuning

To get the best performance for your particular environment, consider the following recommendations for tuning your system.

It is advisable to disable swap for performance reasons.

Enable SYSCTL features

In Linux, you can use the program `sysctl` to read and/or modify the attributes of the system kernel, including its maximum limits and security settings.

When you install HAProxy Enterprise, some recommended `sysctl` settings are written to its configuration file. These `sysctl` settings are disabled by default.

1. Open the configuration file `/etc/sysctl.d/30-hapee-3.1.conf`
2. Enable the settings by un-commenting them (remove the prefixing hash sign).
3. Reload the file using `systemctl restart systemd-sysctl`.

```
# Limit the per-socket default receive/send buffers to limit memory usage
# when running with a lot of concurrent connections. Values are in bytes
# and represent minimum, default and maximum. Defaults: 4096 87380 4194304
#
# net.ipv4.tcp_rmem            = 4096 16060 262144
# net.ipv4.tcp_wmem            = 4096 16384 262144

# Allow early reuse of a same source port for outgoing connections. It is
# required above a few hundred connections per second. Defaults: 0
#
# net.ipv4.tcp_tw_reuse        = 1

# Extend the source port range for outgoing TCP connections. This limits early
# port reuse and makes use of 64000 source ports. Defaults: 32768 61000
#
# net.ipv4.ip_local_port_range = 1024 65023

# Increase the TCP SYN backlog size. This is generally required to support very
# high connection rates as well as to resist SYN flood attacks. Setting it too
# high will delay SYN cookie usage though. Defaults: 1024
#
# net.ipv4.tcp_max_syn_backlog = 60000

# Timeout in seconds for the TCP FIN_WAIT state. Lowering it speeds up release
# of dead connections, though it will cause issues below 25-30 seconds. It is
# preferable not to change it if possible. Default: 60
#
# net.ipv4.tcp_fin_timeout     = 30

# Limit the number of outgoing SYN-ACK retries. This value is a direct
# amplification factor of SYN floods, so it is important to keep it reasonably
# low. However, too low will prevent clients on lossy networks from connecting.
# Using 3 as a default value gives good results (4 SYN-ACK total) and lowering
# it to 1 under SYN flood attack can save a lot of bandwidth. Default: 5
#
# net.ipv4.tcp_synack_retries  = 3

# Set this to one to allow local processes to bind to an IP which is not yet
# present on the system. This is typically what happens with a shared VRRP
# address, where you want both primary and backup to be started even though the
# IP is not yet present. Always leave it to 1. Default: 0
#
# net.ipv4.ip_nonlocal_bind    = 1

# Serves as a higher bound for all of the system's SYN backlogs. Put it at
# least as high as tcp_max_syn_backlog, otherwise clients may experience
# difficulties to connect at high rates or under SYN attacks. Default: 128
#
# net.core.somaxconn           = 60000
```

Migration

- Migrate from HAProxy

Migrate from HAProxy to HAProxy Enterprise

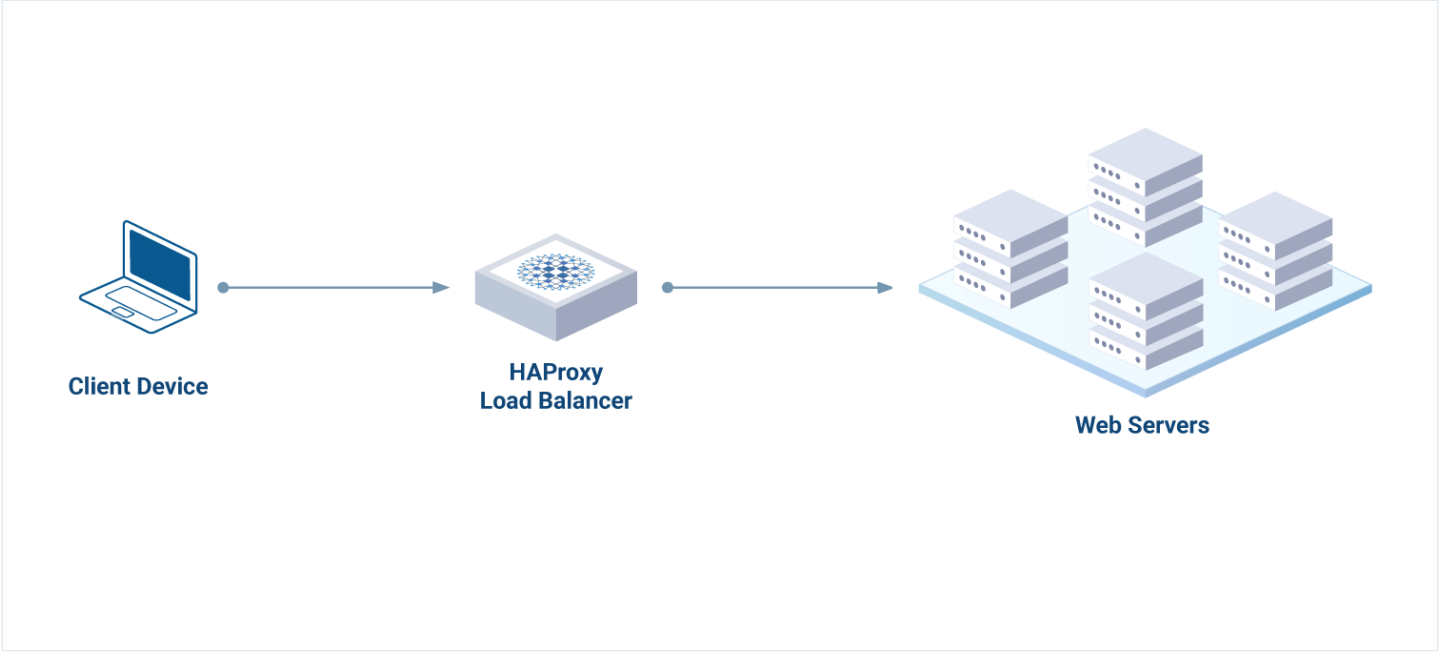
Migrate an existing HAProxy (Community edition) load balancer to use HAProxy Enterprise instead with a blue-green deployment strategy.

The following migration strategy is a high-level overview and doesn't cover every use case or application. You will need to adapt your migration plan to your specific use case. It's recommended to test a migration plan in a pre-production environment before deploying changes to production.

Notable changes migrating from HAProxy to HAProxy Enterprise:

- The HAProxy Enterprise configuration is located at `/etc/hapee-3.1/hapee-lb.cfg` instead of HAProxy's `/etc/haproxy/haproxy.cfg`
- The Runtime API socket is located at `/var/run/hapee-3.1/hapee-lb.sock` instead of HAProxy's `/var/run/haproxy.sock`
- The main service file is called `hapee-3.1-lb.service` instead of `haproxy.service`
- The logs are located at `/var/log/hapee-3.1/lb-access-{YEAR,MONTH,DAY}.log` and `lb-admin-{YEAR,MONTH,DAY}.log` instead of HAProxy's `/var/log/haproxy.log`

In this overview, you are migrating an HAProxy load balancer to a new HAProxy Enterprise load balancer on two different machines; here's what the example setup looks like:



Validate your HAProxy configuration file

Issues with the HAProxy configuration can cause the load balancer to fail after migration. For example, directives or keywords that are permitted in HAProxy now may not be permitted in the target release of HAProxy Enterprise.

1. Validate the HAProxy configuration file before proceeding with migration.

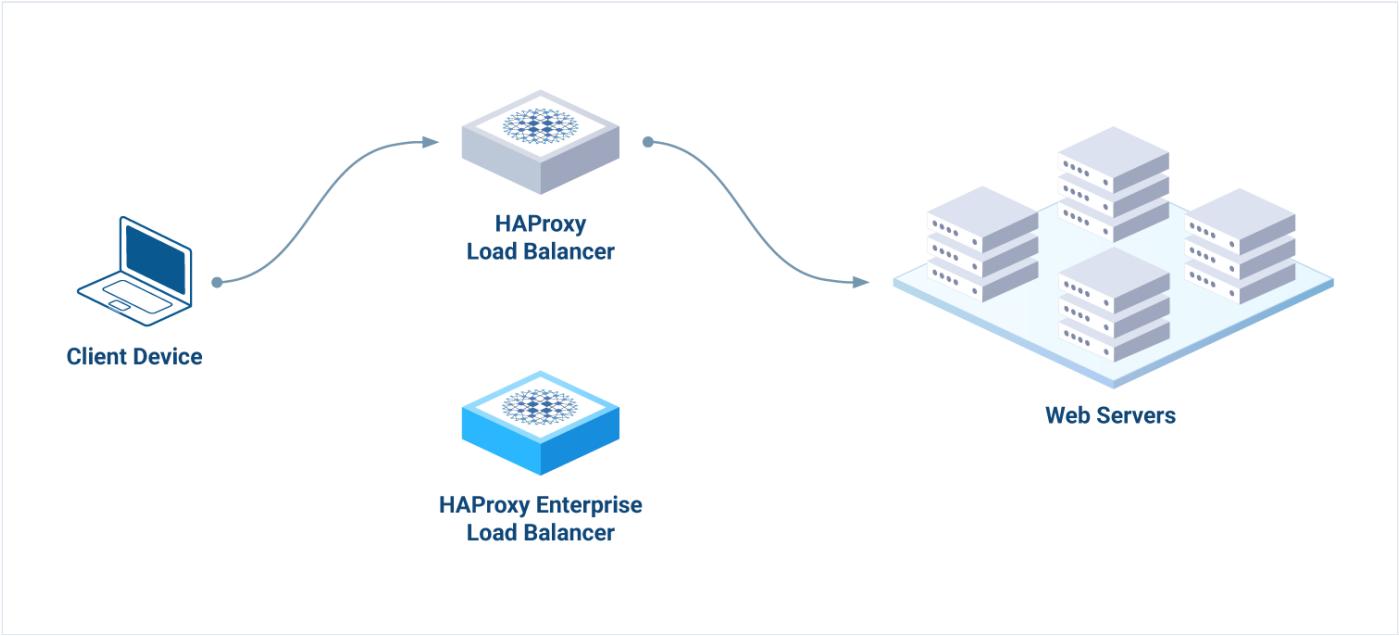
```
/usr/local/sbin/haproxy -c -V -f /etc/haproxy/haproxy.cfg
```

Alerts and warnings that appear in the console output explain the errors encountered. An alert is a fatal condition and prohibits the service from starting. A warning indicates a condition that is not fatal now but may become fatal in the target release. Fix both types of conditions before proceeding.

Migrate with a blue-green deployment strategy

1. Obtain an HAProxy Enterprise license key. Don't have one? Request a free HAProxy Enterprise trial to obtain your trial license key.

2. Install HAProxy Enterprise on a new machine following the Linux Installation procedure.



3. Transfer the necessary contents of the HAProxy configuration file to the new, default HAProxy Enterprise configuration file line-by-line. Here's a list of recommendations:

   - Don't overwrite the default HAProxy Enterprise configuration file with HAProxy's configuration file.
   - Don't transfer the `global` section from HAProxy; instead, use the default `global` section that comes with HAProxy Enterprise.
   - Move necessary HAProxy directories and files to the new machine, and rename them accordingly. These files may include TLS certificates, Map files, and ACL files.
   - Update any path names in the HAProxy Enterprise configuration file accordingly.
   - Confirm hard-coded IP addresses in the `frontend` section.
   - Configure the HAProxy Enterprise load balancer to run on a different IP address than the HAProxy load balancer. This allows you to run both products simultaneously and have the ability to rollback if an issue were to arise.

4. Validate the HAProxy Enterprise configuration file.

   ```
   /opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg -V
   ```

   Alerts and warnings that appear in the console output explain the errors encountered. An alert is a fatal condition and prohibits the service from starting. A warning indicates a condition that is not fatal now but may become fatal in a future release. Address all the errors until you receive the successful output: `Configuration file is valid`.

5. Restart the HAProxy Enterprise service.

   ```
   sudo systemctl restart hapee-3.1-lb
   ```

6. Confirm that the HAProxy Enterprise process is running.

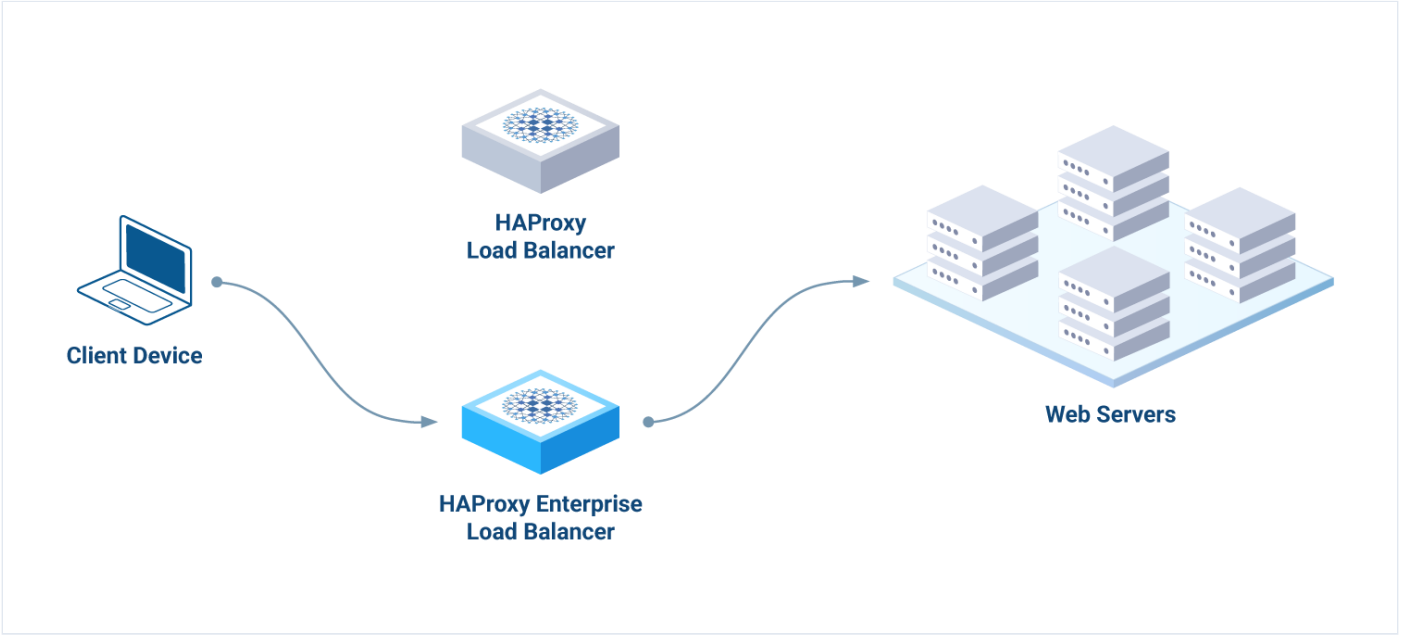
   ```
   systemctl status hapee-3.1-lb
   ```

7. Confirm that HAProxy Enterprise is listening on the right ports.

   ```
   ss -tlpn
   ```

8. Check the HAProxy Enterprise admin log files at `/var/log/hapee-3.1/lb-admin-{YEAR,MONTH,DAY}.log` for messages about startup and the backend server health checks.

9. Perform the migration by diverting traffic from the existing HAProxy load balancer to only the HAProxy Enterprise load balancer using its IP address.

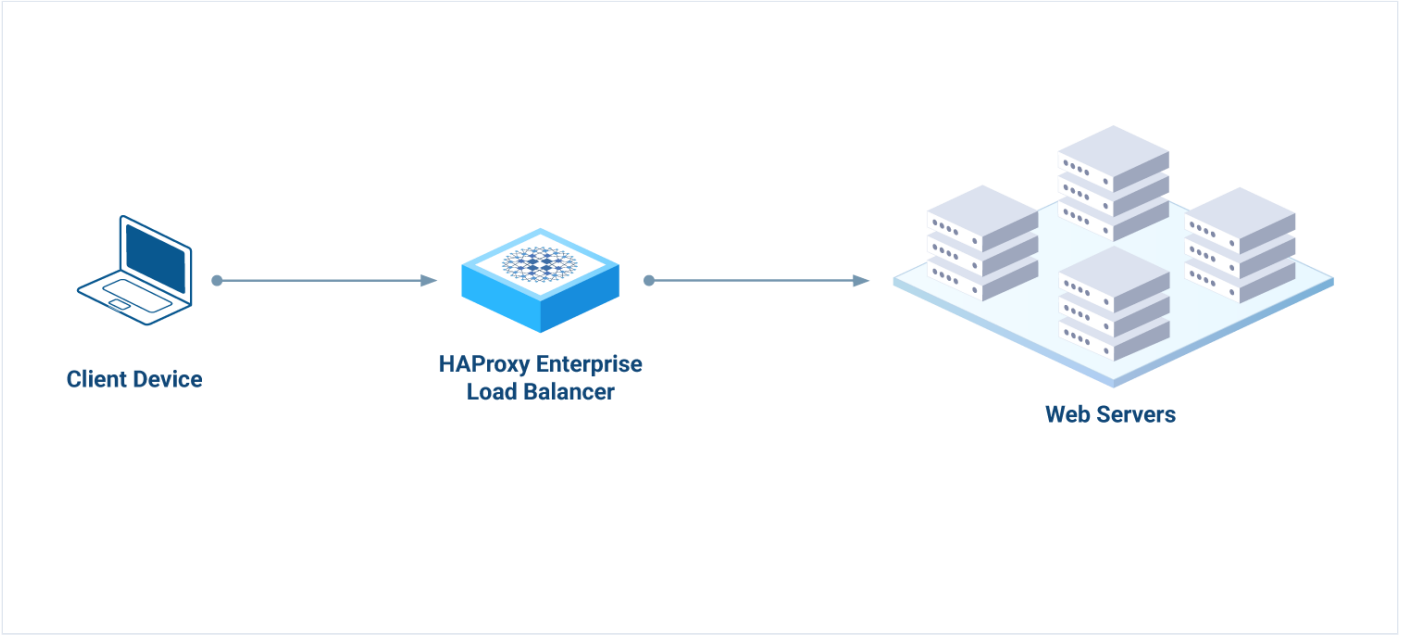


10. Verify no warnings or errors by tailing the HAProxy Enterprise access logs:

```
sudo tail /var/log/hapee-3.1/lb-access-{YEAR,MONTH,DAY}.log
```

11. Perform tests to confirm that your application is working as expected. Divert traffic back to the HAProxy load balancer if issues cannot be addressed.

12. If your application is working as expected and a rollback is no longer needed, then you can tear down HAProxy and the old machine.



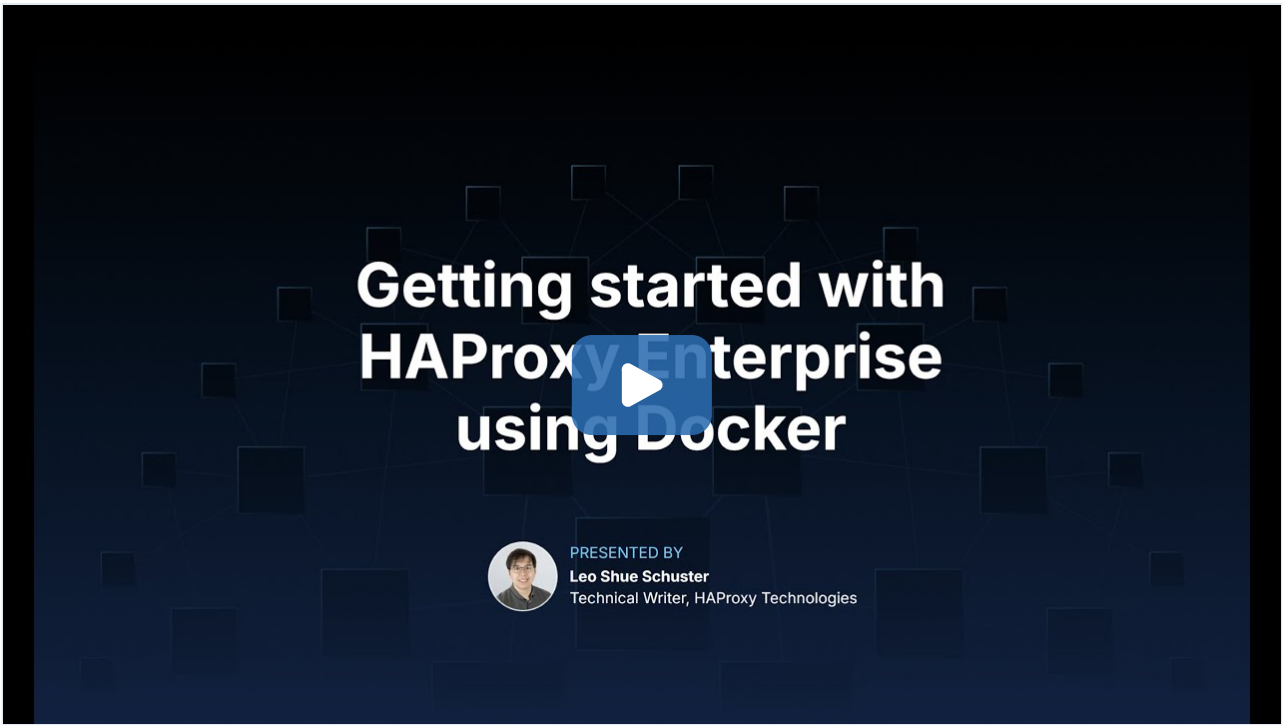
If this blue-green deployment strategy is not possible for your use case, contact our Support team to consider alternatives based on your specific needs.

Tutorials

- Docker tutorial

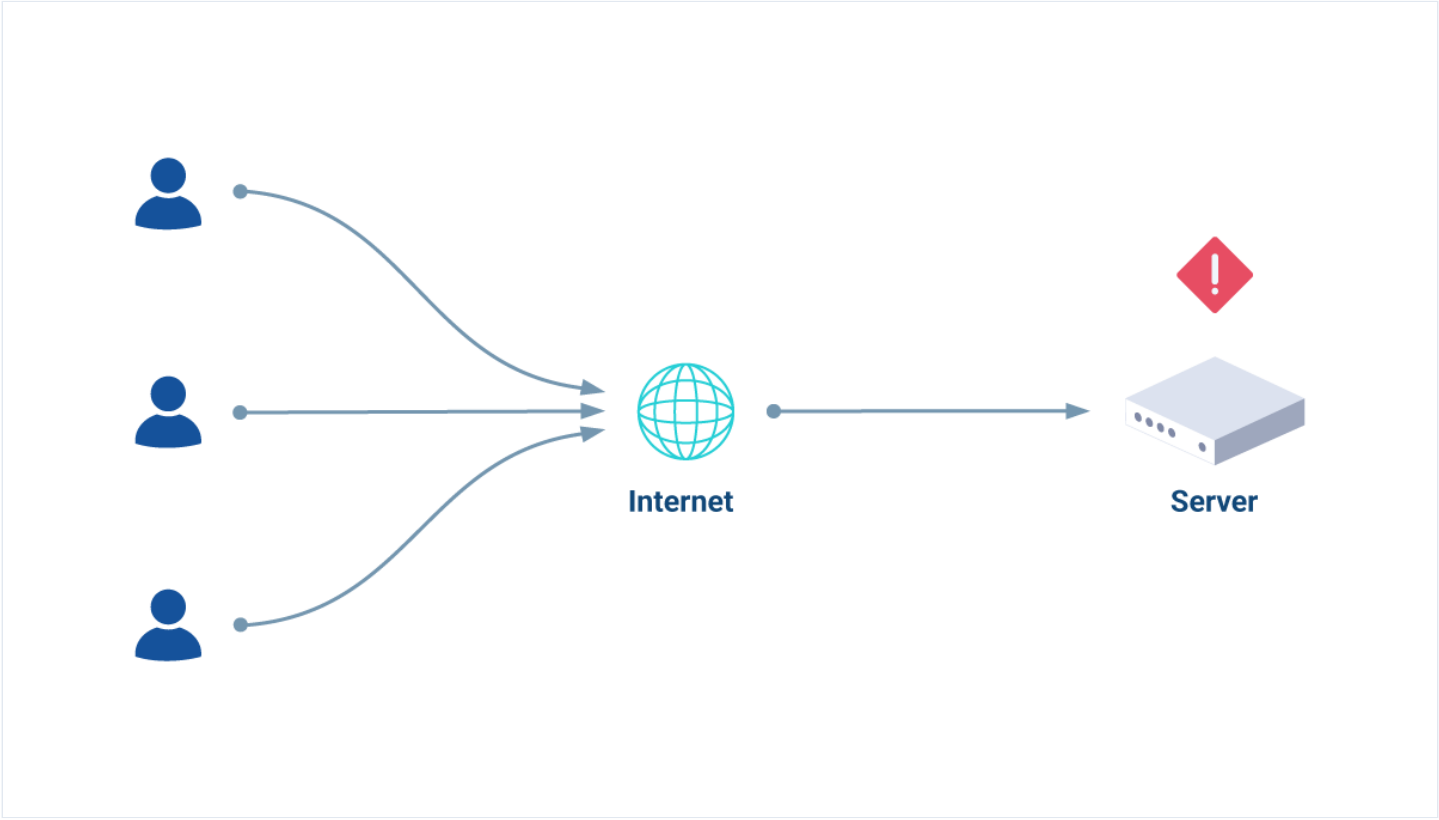
A step-by-step tutorial to set up HAProxy Enterprise using Docker for the first time

Welcome to the tutorial on getting started with HAProxy Enterprise using Docker.

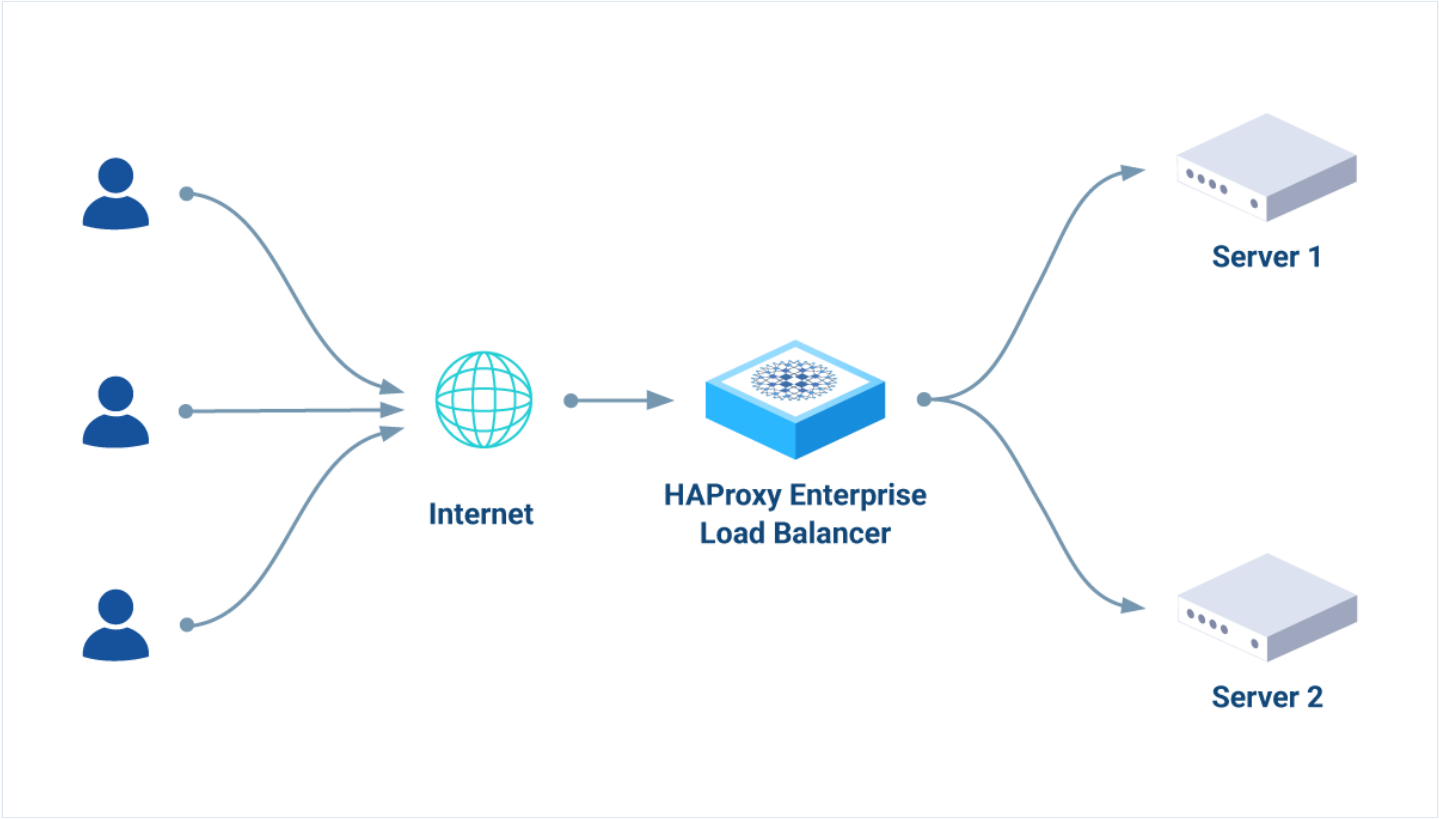


(https://www.youtube.com/watch?v=I7uGVA9414c)

In this tutorial, you will walk through an example use case and set up HAProxy Enterprise in a development environment. Your use case involves end-users wanting to access your website. There's just one problem: only one web server is handling all the web traffic right now, and there are signs of it being overwhelmed.



Your website is gaining popularity, and it's time to scale. You have been tasked with implementing a load balancer between the end-users and two identical web servers. A load balancer will distribute requests evenly so that the two web servers share the work. Your goal is to have an HAProxy Enterprise load balancer handle all the web traffic being sent from end-users and forward that traffic to your web servers.



You will see some code examples throughout this tutorial. These code examples are designed to offer real-world, working code as a place to start implementing HAProxy Enterprise.

The following steps will walk you through setting up HAProxy Enterprise with Docker.

Step 0 - Check your prerequisites

You must have access to a stable internet connection and a computer with Docker Desktop installed (installation instructions here).

Docker is an open platform for developing, shipping, and running applications using containers. Containers are lightweight, standalone, executable packages of software that include everything you need to run an application (which can include code, runtime, system tools, system libraries, and settings). You will use Docker to get an HAProxy Enterprise Docker container up and running.

Open a Terminal session. Create a directory called `hapee-tutorial` in your preferred location, and change directory into it:

```
mkdir hapee-tutorial
cd hapee-tutorial
```

> **“hapee” stands for “HAProxy Enterprise Edition”**

You are ready to continue to Step 1 if you have a trial license key.

Don’t have one? Request a free HAProxy Enterprise trial to obtain your trial license key. Once you have one, return here and continue to Step 1.

Step 1 - Obtain an HAProxy Enterprise Docker image

A Docker image is a standardized package that includes all of the files, binaries, libraries, and configurations to run a container. The haproxy-enterprise image hosts HAProxy Enterprise, and you can obtain it by using Terminal.

Log into the hapee-registry.haproxy.com Docker registry using your HAProxy Enterprise license key as both the username and password:

```
docker login https://hapee-registry.haproxy.com
```

If login is successful, you will see the following message: `Login Succeeded`.

Pull the HAProxy Enterprise image:

```
docker pull hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
```

output

```
3.1r1: Pulling from haproxy-enterprise
00d67a470c5: Pull complete
6bb6daa6e42b: Pull complete
49543f4059: Pull complete
51b3f827b3e5: Pull complete
Digest: sha256:bd32fa7e4b0a2e8da4a3c1ecf66c125868f8f86bc65fe44a2f860a3d2331g
Status: Downloaded newer image for hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
```

You have obtained an HAProxy Enterprise Docker image.

Step 2 - Create an HAProxy Enterprise configuration file

An HAProxy Enterprise configuration file stores settings for an HAProxy Enterprise load balancer. This is where you will make changes to the load balancer so that it can fit your needs.

Create a directory for the HAProxy Enterprise load balancer and change directory into it:

```
mkdir hapee-3.1
cd hapee-3.1
```

In the `hapee-3.1` directory, create an HAProxy Enterprise configuration file called: `hapee-lb.cfg`.

Open the configuration file with your preferred text editor and insert the following code:

```
#---------------------------------------------------------------------
# Process global settings
#---------------------------------------------------------------------
global
  stats socket /var/run/hapee-3.1/hapee-lb.sock user hapee-lb group hapee mode 660 level admin
  log stdout format raw local0 info

#---------------------------------------------------------------------
# Common defaults that the 'backend' section will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
  mode                http
  log                 global
  timeout connect     10s
  timeout client      300s
  timeout server      300s

#---------------------------------------------------------------------
# main frontend which forwards to the backend
#---------------------------------------------------------------------
frontend  fe_main
  bind :80            # direct HTTP access
  default_backend     web_servers

#---------------------------------------------------------------------
# default round-robin balancing in the backend
#---------------------------------------------------------------------
backend web_servers
  balance           roundrobin
  server s1         172.16.0.11:80 check
  server s2         172.16.0.12:80 check
```

Save and close this file. You configured your first HAProxy Enterprise configuration file!

- Detailed explanation of this HAProxy Enterprise configuration file

What does each line of your configuration file do?

```
global
```

The `global` section defines parameters for process-wide security and performance tunings. See Global.

```
stats socket /var/run/hapee-3.1/hapee-lb.sock user hapee-lb group hapee mode 660 level admin
```

The `stats socket` parameter enables the HAProxy Runtime API.

```
log stdout format raw local0 info
```

The `log` parameter enables logging. To understand how logging works, see Manage HAProxy Enterprise logs.

```
defaults
```

The `defaults` section helps reduce code duplication by applying its settings to all of the `frontend` and `backend` sections that come after it. See Defaults.

```
mode                http
```

Sets HTTP as the running mode for the load balancer, as opposed to TCP or UDP. See HTTP, TCP, and Load balance UDP with HAProxy Enterprise.

```
log                 global
```

This setting tells each subsequent frontend to use the log setting defined in the global section.

```
timeout connect     10s
timeout client      300s
timeout server      300s
```

`timeout connect` sets the amount of time that HAProxy will wait for a connection to a backend server to be established. `timeout client` sets how long to wait during client inactivity. The `timeout server` sets how long to wait for backend server inactivity. See Timeouts.

```
frontend  fe_main
```

You are defining a `frontend` with the name `fe_main`. This section defines the IP addresses and ports that clients can connect to. See Frontends.

```
bind :80       # direct HTTP access
```

A `bind` setting assigns a listener to `localhost:80`. See Bind options.

```
default_backend    web_servers
```

The `default_backend` setting will send traffic to the specified `backend` called `web_servers`. See Backends.

```
backend web_servers
  balance          roundrobin
  server s1        172.16.0.11:80 check
  server s2        172.16.0.12:80 check
```

The `backend` section you named `web_servers` defines two web servers to handle requests with a round-robin algorithm (see other available algorithms in Change the load balancing algorithm). You have specified an IP address for each web server. The `check` argument monitors a server to check if it's healthy; see HTTP health checks.

Step 3 - Create an `index.html` file for each web server

To demonstrate request traffic being sent to web servers in your development environment, you will create two Apache HTTP servers with Docker. Apache is an open-source HTTP web server application. By creating an `index.html` file for each web server, you will visualize the request traffic that end-users will send to your web servers and how HAProxy Enterprise handles those requests.

Back in the `hapee-tutorial` directory, create a directory for the first web server and change directory into it:

```
cd ..
mkdir public-html-web1
cd public-html-web1
```

Create an HTML file called `index.html`, and insert the following HTML code into that file:

```
<html><body><h1>It works! Web server 1 received your request this time.</h1></body></html>
```

Save and close the file. Move back to the `hapee-tutorial` directory and repeat the steps for the second web server:

```
cd ..
mkdir public-html-web2
cd public-html-web2
```

Create an HTML file called `index.html`, and insert the following HTML code into that file:

```
<html><body><h1>It works! Web server 2 received your request this time.</h1></body></html>
```

Note how this text says “Web server 2” instead of “Web server 1”. Save and close this file. You’ve created an `index.html` file for each web server.

Step 4 - Docker Compose

Docker Compose is a tool for configuring and running many Docker containers at once. Compose makes development easier with the use of a single YAML configuration file. You’ll use it to start, stop, and rebuild services with Docker.

Move back to the `hapee-tutorial` directory:

```
cd ..
```

Create a YAML file called `docker-compose.yml`, and insert the following code:

```
---
networks:
  my_custom_network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.16.0.0/16
          gateway: 172.16.0.1


services:
  hapee-3.1:
    image: hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
    ports:
      - "80:80"
    volumes:
      - "./hapee-3.1:/etc/hapee-3.1"
    networks:
      my_custom_network:
        ipv4_address: 172.16.0.10
    container_name:
        hapee-3.1
  web1:
    image: httpd
    volumes:
      - "./public-html-web1:/usr/local/apache2/htdocs/"
    networks:
      my_custom_network:
        ipv4_address: 172.16.0.11
    container_name:
        web1
  web2:
    image: httpd
    volumes:
      - "./public-html-web2:/usr/local/apache2/htdocs/"
    networks:
      my_custom_network:
        ipv4_address: 172.16.0.12
    container_name:
        web2
```

Save and close this file. You have now created a Docker Compose YAML file.

- Detailed explanation of this Docker Compose YAML file

  What does each line of your YAML file do?

```
networks:
```

Docker Compose will set up a single network for your services to communicate with each other.

```
my_custom_network:
  driver: bridge
```

Compose will create a bridge network named `my_custom_network` that will connect the load balancer and two web servers all on one network. A bridge network is a software bridge that allows containers connected to the same bridge network to communicate. In turn, that means it provides isolation from other containers that are not connected to the bridge network.

```
ipam:
  config:
    - subnet: 172.16.0.0/16
      gateway: 172.16.0.1
```

`ipam` will specify a custom IPAM configuration in `my_custom_network`. `config` will contain a configuration element with a `subnet` in CIDR format to represent a network segment and a `gateway` of IPv4 for the subnet.

```
services:
  hapee-3.1:
    image: hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
    ports:
      - "80:80"
    volumes:
      - "./hapee-3.1:/etc/hapee-3.1"
    networks:
      my_custom_network:
        ipv4_address: 172.16.0.10
    container_name:
      hapee-3.1
  web1:
    image: httpd
    volumes:
      - "./public-html-web1:/usr/local/apache2/htdocs/"
    networks:
      my_custom_network:
        ipv4_address: 172.16.0.11
    container_name:
        web1
  web2:
    image: httpd
    volumes:
      - "./public-html-web2:/usr/local/apache2/htdocs/"
    networks:
      my_custom_network:
        ipv4_address: 172.16.0.12
    container_name:
        web2
```

The `services` section defines the three different containers Docker will create: a `hapee-3.1` HAProxy Enterprise load balancer and two Apache web servers, `web1` and `web2`.

The `image` lines will run services using a pre-built image, and you are specifying their image locations.

`ports` is used to map a container's port to the host machine.

`volumes` is used to mount disks in Docker. In this development environment, the `hapee-tutorial` directory will contain the mounted disks.

`networks` connect the `my_custom_network` bridge network with each service on their own IP addresses as specified.

`container_name` is where you specify the name of the container for Docker.

Run Docker Compose from the `hapee-tutorial` directory:

```
docker compose -f "docker-compose.yml" up -d --build
```

output

```
[+] Running 8/8
web1 Pulled                                      4.9s
web2 Pulled                                      4.9s
 efc2b5ad9eed Pull complete                      2.7s
 fc31785eb818 Pull complete                      2.7s
 4f4fb700ef54 Pull complete                      2.8s
 f214daa0692g Pull complete                      2.9s
 05383fd8b2b4 Pull complete                      3.3s
 88ad12232aa2 Pull complete                      3.3s
[+] Running 4/4
Network hapee-tutorial_my_custom_network  Created  0.0s
Container hapee-3.1                       Started  0.9s
Container web1                            Started  0.9s
Container web2                            Started  0.7s
```

Verify that Docker Compose has created and started your containers:

```
docker ps
```

output

```
CONTAINER ID  IMAGE  COMMAND  CREATED  STATUS  PORTS  NAMES
e12f4825e548  httpd  "httpd-foreground"  4 seconds ago  Up 3 seconds  80/tcp  web2
e83f7965c044  hapee-registry.haproxy.com/haproxy-enterprise:3.1r1  "/init"  4 seconds ago  Up 3 seconds  443/tcp,
0.0.0.0:80->80/tcp, 5555/tcp  hapee-3.1
d9e82d201a71  httpd  "httpd-foreground"  4 seconds ago  Up 3 seconds  80/tcp  web1
```

Docker Compose has created a network and three containers as specified, and Docker is actively running the services.

Step 5 - Send request traffic to web servers through an HAProxy Enterprise load balancer

Launch your preferred web browser, and navigate to the following web address: `http://localhost:80`.

The web result is a request being sent to the HAProxy Enterprise load balancer, which in turn forwards the request to Web Server 1.

If you refresh the webpage on `http://localhost:80`, it will make a new request to the HAProxy Enterprise load balancer. Notice how it returns Web Server 2:

This development environment demonstrates an end-user making a request to port 80 and the HAProxy Enterprise load balancer relaying the traffic to the Apache web servers in the backend using the default round-robin algorithm. That means each subsequent request will be relayed to a different web server, effectively distributing the load across your two web servers evenly.

Round-robin algorithm in action

Refresh the webpage again to see the request go to a different web server:

Each subsequent refresh will relay the request traffic to the web servers in a round-robin fashion, thanks to the configuration you specified in the HAProxy Enterprise load balancer.

Congratulations! You have HAProxy Enterprise running after following these steps. HAProxy Enterprise is serving and load balancing end-user traffic with Docker.

That concludes your walkthrough in this development environment. When you're done experimenting, run the following command from the `hapee-tutorial` directory to stop Docker services and remove the containers:

```
docker compose -f "docker-compose.yml" down
```

output

```
[+] Running 4/4
Container hapee-tutorial-web1             Removed  1.6s
Container hapee-tutorial-web2             Removed  1.5s
Container hapee-tutorial-hapee-3.1        Removed  3.9s
Network hapee-tutorial_my_custom_network  Removed  0.2s
```

Logging

Need help troubleshooting? Logs give you insight into issues and errors.

HAProxy Enterprise logs

HAProxy Enterprise generates two types of logs: access logs and administrative logs. See Manage HAProxy Enterprise logs.

Docker Compose logs

The following Docker Compose CLI command will show you the log output of Docker services and containers, helpful for troubleshooting if you run into any issues while building:

```
docker compose logs
```

example output

```
web1       | [Mon Jul 15 23:27:09.251161 2024] [mpm_event:notice] [pid 1:tid 1] AH00489: Apache/2.4.61 (Unix)
configured -- resuming normal operations
hapee-3.1  | [cont-init.d] executing container initialization scripts...
hapee-3.1  | [cont-init.d] 01-hapee: executing...
hapee-3.1  | [cont-init.d] 01-hapee: exited 0.
hapee-3.1  | [cont-init.d] done.
hapee-3.1  | [services.d] starting services
hapee-3.1  | [services.d] done.
hapee-3.1  | [WARNING]  (227) : Failed to connect to the old process socket '/var/run/hapee-3.1/hapee-lb.sock'
hapee-3.1  | [ALERT]    (227) : Failed to get the sockets from the old process!
hapee-3.1  | [NOTICE]   (227) : New worker (258) forked
hapee-3.1  | [NOTICE]   (227) : Loading success.
hapee-3.1  | time="2024-07-15T23:27:09Z" level=info msg="Build date: 2024-07-03T11:54:03Z"
web2       | [Mon Jul 15 23:27:09.250890 2024] [mpm_event:notice] [pid 1:tid 1] AH00489: Apache/2.4.61 (Unix)
configured -- resuming normal operations
```

Conclusion

With this development environment setup, the end-users' requests are sent to one HAProxy Enterprise load balancer. The requests are then forwarded to two web servers in a round-robin fashion. If one of the web servers were to go down, HAProxy Enterprise will keep your website available by automatically detecting the loss and routing request traffic to only the available web server.

Where to go from here? You can scale this use case by adding more web servers in the backend for higher availability, redundancy, and performance improvements. In addition, you can add another HAProxy Enterprise load balancer so that the load balancing layer also has higher availability and redundancy; having two load balancers is our recommended set up for production environments.

Uninstallation

Uninstall HAProxy Enterprise on BSD

Follow these steps to uninstall HAProxy Enterprise from a BSD system.

1. Stop HAProxy Enterprise.

   ```
   sudo systemctl stop hapee-<VERSION>-lb
   ```

   For example, to stop HAProxy Enterprise 3.1r1:

   ```
   sudo service hapee_31_lb onestop
   ```

2. Back up your configuration files.

   The HAProxy Enterprise configuration directory may contain files that you wish to keep: SSL/TLS certificates, map files, and other files that you may wish to preserve after uninstallation. Ensure that any files you wish to keep are copied to a safe location.

   The following command will copy the contents of `/etc/hapee-3.1` to a time-stamped directory in `/tmp`:

   ```
   cp -r /etc/hapee-3.1 /tmp/hapee-3.1-lb.backup.$(date +%F_%R)
   ```

3. Query to see your installed packages.

   ```
   sudo pkg info hapee*
   ```

   output

   ```
   hapee-3.1r1-lb-1.0.0.237.0
   ```

4. Remove all HAProxy Enterprise packages.

   ```
   sudo pkg remove hapee-*
   ```

output

```
Checking integrity... done (0 conflicting)
Deinstallation has been requested for the following 1 packages (of 0 packages in the universe):

Installed packages to be REMOVED:
        hapee-3.1r1-lb: 1.0.0.237.0

Number of packages to be removed: 1

The operation will free 5 MiB.

Proceed with deinstalling packages? [y/N]: y
```

Uninstall HAProxy Enterprise on Docker

Uninstall and remove all HAProxy Enterprise Docker containers.

1. List all of your containers:

```
sudo docker ps -a --no-trunc
```

2. List all of your containers that match the string `hapee`:

```
sudo docker ps -a --no-trunc | grep hapee
```

   Verify that the string `hapee` exactly matches the list of containers that you wish to remove, as matching containers will be permanently removed.

3. Stop the HAProxy Enterprise containers that match the string `hapee`:

```
sudo docker ps -a --no-trunc | \
  grep hapee | \
  awk '{print $1}' | \
  xargs -r --no-run-if-empty sudo docker stop
```

4. Remove the HAProxy Enterprise containers that match the string `hapee`:

```
sudo docker ps -a --no-trunc | \
  grep hapee | \
  awk '{print $1}' | \
  xargs -r --no-run-if-empty sudo docker rm
```

5. Optional: Remove the corresponding container images:

```
sudo docker images --no-trunc | \
  grep hapee | \
  awk '{print $3}' | \
  xargs -r --no-run-if-empty sudo docker rmi
```

Uninstall HAProxy Enterprise on Linux

Follow these steps to uninstall HAProxy Enterprise from a Linux system.

1. Stop HAProxy Enterprise.

   ```
   sudo systemctl stop hapee-<VERSION>-lb
   ```

   For example, to stop HAProxy Enterprise 3.1r1:

   ```
   sudo systemctl stop hapee-3.1-lb
   ```

2. Back up your configuration files.

   The HAProxy Enterprise configuration directory may contain files that you wish to keep: SSL/TLS certificates, map files, and other files that you may wish to preserve after uninstallation. Ensure that any files you wish to keep are copied to a safe location.

   The following command will copy the contents of `/etc/hapee-3.1` to a time stamped directory in `/tmp`:

   ```
   cp -r /etc/hapee-3.1 /tmp/hapee-3.1-lb.backup.$(date +%F_%R)
   ```

3. Query to see your installed packages.

   Debian:

   ```
   sudo apt list --installed | grep hapee
   ```

   output

   ```
   WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

   hapee-3.1r1-base/unknown,now 1.0.0.93.0 all [installed,automatic]
   hapee-3.1r1-lb/unknown,now 1.0.0-239.210 amd64 [installed]
   ```

Ubuntu:

```
sudo apt list --installed | grep hapee
```

output

```
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

hapee-3.1r1-base/unknown,now 1.0.0.93.0 all [installed,automatic]
hapee-3.1r1-lb/unknown,now 1.0.0-239.210 amd64 [installed]
```

Redhat:

```
sudo yum list --installed hapee-*
```

output

```
[...]
Installed Packages
hapee-3.1r1-base.noarch   1.0.0-93.0       @hapee-base
hapee-3.1r1-lb.x86_64     1.0.0-239.210    @hapee-base
```

Suse:

```
sudo zypper search --installed-only hapee
```

output

```
Loading repository data...
Reading installed packages...

S  | Name             | Summary                                       | Type
---+------------------+-----------------------------------------------+--------
i  | hapee-3.1r1-base | HAPEE BASE : Common dependencies              | package
i+ | hapee-3.1r1-lb   | HAPEE LB : Layer 7 load-balancing (HAProxy) | package
```

4. Remove all HAProxy Enterprise packages.

Debian:

```
sudo apt-get purge hapee-*
```

output

```
The following packages will be REMOVED:
  hapee-3.1r1-base* hapee-3.1r1-lb*
0 upgraded, 0 newly installed, 2 to remove and 95 not upgraded.
After this operation, 4,556 kB disk space will be freed.
Do you want to continue? [Y/n] y
```

Ubuntu:

```
sudo apt-get purge hapee-*
```

output

```
The following packages will be REMOVED:
  hapee-3.1r1-base* hapee-3.1r1-lb*
0 upgraded, 0 newly installed, 2 to remove and 95 not upgraded.
After this operation, 4,556 kB disk space will be freed.
Do you want to continue? [Y/n] y
```

Redhat:

```
sudo yum remove hapee-*
```

output

```
Loaded plugins: fastestmirror
Resolving Dependencies
--> Running transaction check
---> Package hapee-3.1r1-base.noarch 0:1.0.0-93.0 will be erased
---> Package hapee-3.1r1-lb.x86_64 0:1.0.0-239.210 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
Package               Arch        Version               Repository        Size
================================================================================
Removing:
hapee-3.1r1-base      noarch      1.0.0-93.0            @hapee-base       19 k
hapee-3.1r1-lb        x86_64      1.0.0-239.210         @hapee-base      3.8 M

Transaction Summary
================================================================================
Remove  2 Packages

Installed size: 3.8 M
Is this ok [y/N]: y
```

Suse:

```
sudo zypper remove --clean-deps hapee-*
```

output

```
Reading installed packages...
Resolving package dependencies...

The following 3 packages are going to be REMOVED:
  hapee-3.1r1-base hapee-3.1r1-lb libpcreposix0

3 packages to remove.
After the operation, 15.3 MiB will be freed.
Continue? [y/n/v/...? shows all options] (y): y
```

5. Confirm the operation by repeating the query to list installed packages. If any `hapee` packages remain, change to your home directory and repeat the command to remove packages.

6. If you're using HAProxy Fusion, remove any other supporting files. Doing so avoids errors that may occur should you decide to re-add the node in the future. Use this command:

```
sudo rm -rf /etc/hapee-extras/* /var/run/hapee-* /var/lib/dataplaneapi/
```

Upgrade

Upgrade HAProxy Enterprise on AWS

The following procedures outline upgrading an instance of HAProxy Enterprise deployed in AWS via HAProxy Enterprise AMI.

Validate the HAProxy Enterprise configuration file

1. Issues with the HAProxy Enterprise configuration can cause the load balancer to fail after an upgrade. Even if the service is performing as desired before the upgrade, the upgrade could introduce changes that cause a failure afterward.

   Validate the HAProxy Enterprise configuration file before proceeding with the upgrade.

   Depending on your version, the command's output indicates that the file is valid in these ways:

   - In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
   - In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

   > **Multiple configuration files**
   >
   > If you have multiple configuration files in your application, be sure to check them all in the correct order.

   Example: validate the configuration for version 2.9:

   AWS/BSD/Linux:

   ```
   sudo /opt/hapee-2.9/sbin/hapee-lb -c -f /etc/hapee-2.9/hapee-lb.cfg -V
   ```

   Docker:

   ```
   sudo docker exec hapee /opt/hapee-2.9/sbin/hapee-lb -c -f /etc/hapee-2.9/hapee-lb.cfg -V
   ```

   Alerts and warnings that appear in the console output explain the errors encountered. An alert is a fatal condition and prohibits the service from starting. A warning indicates a condition that is not fatal now but may become fatal in the target release. Address all the errors until you receive the successful output: `Configuration file is valid`.

Create a new instance with the latest AMI

The following procedure will produce a new instance of HAProxy Enterprise using the latest AMI. The new instance running the latest version of HAProxy Enterprise will replace your existing instance.

Launch a new instance with the latest AMI

1. Follow the steps at Launch the HAProxy Enterprise AMI to launch a new instance of HAProxy Enterprise via AMI. You can select the existing VPC, subnets, and security groups associated with your existing instance of HAProxy Enterprise.
2. Connect to your existing HAProxy Enterprise instance through its public IP address.
3. Copy the configuration file `/etc/hapee-3.1/hapee-lb.cfg` from your existing instance to your new instance.
4. Validate the HAProxy Enterprise configuration file.

   ```
   sudo /opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg -V
   ```

   Alerts and warnings that appear in the console output explain the errors encountered. Address all the errors until you receive the successful output: `Configuration file is valid`.
5. Use `systemctl reload` to reload the load balancer configuration with the copied configuration file:

   ```
   sudo systemctl reload hapee-3.1-lb
   ```

Enable the new instance

There are two options for changing your configuration to point to your new instance.

Option 1: Your instance uses an elastic IP address

If, when you created your instance running the previous version of HAProxy Enterprise, you assigned it an elastic IP, follow these steps to reassign that elastic IP to the new instance.

If for some reason your existing instance did not have an elastic IP, or you have no elastic IPs available, create an elastic IP address.

1. Open the Amazon EC2 console.
2. From the EC2 Dashboard, go to **Network & Security > Elastic IPs**, and in the list, find the elastic IP associated with your existing HAProxy Enterprise instance.
3. Click on the elastic IP address to open its settings.

4. Click **Associate Elastic IP address**.
5. Choose your new instance from the list.
6. Check the box to **Allow this Elastic IP to be reassociated**. This will remove the association with your previous instance and associate the elastic IP with your new instance.

Option 2: Update your DNS server

If you are not using elastic IP addresses, update your DNS server to use the public IPv4 address of the new instance running the latest version of HAProxy Enterprise.

1. To get the public IPv4 address of the instance, open the Amazon EC2 console.
2. From the EC2 Dashboard, go to **Instances**, and select the HAProxy Enterprise instance from the list. Copy its public IPv4 address.

Your new instance running the latest version of HAProxy Enterprise is now active. You can now terminate your other instance running the previous version of HAProxy Enterprise.

Terminate the previous instance

Terminate an instance that is no longer needed.

1. Open the Amazon EC2 console.
2. From the EC2 Dashboard, go to **Instances**, and in the list, find the HAProxy Enterprise instance you no longer need.
3. Right-click on the instance and select **Terminate instance**.
4. Click **Terminate**. AWS will terminate the instance and remove it from the list.

Optional: Re-configure your Amazon Network Load Balancer (NLB)

If you have multiple HAProxy Enterprise instances configured for use with AWS NLB, you need to update your NLB targets to add your new HAProxy Enterprise instance and to remove the instance you no longer need.

Add an instance to a target group

To add a target to a target group:

1. Open the Amazon EC2 console.
2. From the EC2 Dashboard, go to **Load Balancing > Target Groups**, and select your target group.

3. Select the **Targets** tab, and then **Register targets**.
4. On the **Register targets** screen, select the new HAProxy Enterprise instance to include in the target group. Then click **Include as pending below**.
5. Click **Register pending targets**.

The instance will appear in the list for **Registered targets**.

Remove an instance from a target group

To remove a target from a load balancing target group:

1. Open the Amazon EC2 console.
2. From the EC2 Dashboard, go to **Load Balancing > Target Groups**, and select your target group.
3. Select the **Targets** tab.
4. Check the box for the instance you no longer need to select it, and then click **Deregister**.

The instance will no longer be associated with the target group for the AWS NLB.

Upgrade HAProxy Enterprise on BSD

You can upgrade HAProxy Enterprise to version 3.1r1 on the following operating systems:

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
3.1r1	Feb 2025	Feb 2026	• FreeBSD 13.4 (amd64)

Upgrade to HAProxy Enterprise 3.1r1

The following **upgrade** procedure installs a new major version of HAProxy Enterprise.

1. Issues with the HAProxy Enterprise configuration can cause the load balancer to fail after an upgrade. Even if the service is performing as desired before the upgrade, the upgrade could introduce changes that cause a failure afterward.

   Validate the HAProxy Enterprise configuration file before proceeding with the upgrade.

   Depending on your version, the command's output indicates that the file is valid in these ways:

   - In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
   - In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

   > **Multiple configuration files**
   > If you have multiple configuration files in your application, be sure to check them all in the correct order.

   Example: validate the configuration for version 2.9:

   AWS/BSD/Linux:

   ```
   sudo /opt/hapee-2.9/sbin/hapee-lb -c -f /etc/hapee-2.9/hapee-lb.cfg -V
   ```

Docker:

```
sudo docker exec hapee /opt/hapee-2.9/sbin/hapee-lb -c -f /etc/hapee-2.9/hapee-lb.cfg -V
```

Alerts and warnings that appear in the console output explain the errors encountered. An alert is a fatal condition and prohibits the service from starting. A warning indicates a condition that is not fatal now but may become fatal in the target release. Address all the errors until you receive the successful output: `Configuration file is valid`.

2. If the HAProxy Enterprise server is part of a high-availability cluster, take it out of the cluster:

   - If using Active/Active clustering, update your network to stop sending traffic to the load balancer. For example, if using DNS roundrobin, remove the load balancer from DNS.
   - If using Active/Standby clustering, then on the active load balancer edit `/etc/hapee-extras/hapee-vrrp.cfg`. Lower the `priority` value to less than the standby load balancer's `priority`. This will put the active load balancer into the standby state and simultaneously activate the standby load balancer. Then restart the `hapee-extras-vrrp` service.

3. Install HAProxy Enterprise 3.1r1 by running the following command and replacing `<HAProxy Enterprise Key>` with your HAProxy Enterprise license key:

```
wget https://www.haproxy.com/static/install_haproxy_enterprise.sh
sudo bash ./install_haproxy_enterprise.sh \
  --version 3.1r1 \
  --key <HAProxy Enterprise key>
```

4. Update additional modules. You may be running additional modules with HAProxy Enterprise, which you must also reinstall when you upgrade to a new release. Install a package using its name, for example, upgrade the `lb-update` package:

```
sudo pkg install hapee-3.1r1-lb-update
```

5. Copy `/etc/hapee-<PREVIOUS VERSION>/hapee-lb.cfg` over to `/etc/hapee-3.1/hapee-lb.cfg` and any associated files (maps, certificates, etc).

6. Optionally, you might want to validate your configuration against the new load balancer before disabling the old load balancer. Use the `-c` option to check the configuration.

   Depending on your version, the command's output indicates that the file is valid in these ways:

   - In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
   - In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

> **Multiple configuration files**
>
> If you have multiple configuration files in your application, be sure to check them all in the correct order.

Example:

```
sudo /opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg -V
```

Alerts and warnings that appear in the console output explain the errors encountered. Address all the errors until you receive the successful output: `Configuration file is valid`.

7. Stop the old HAProxy Enterprise daemon and start the new daemon:

```
sudo service hapee_<PREVIOUS VERSION>_lb onestop
sudo service hapee_31_lb onestart
```

8. Run `tail -f /var/log/messages` (or distribution equivalent) to check for warnings or errors.
9. Run `curl localhost` to ensure that HAProxy Enterprise is responding. Adjust address/port as needed, and use a local address that is currently active on the box and not a VRRP IP.
10. Put the upgraded server back into the cluster.
11. **Repeat this procedure on secondary servers**.

Post-upgrade

After you upgrade, see the section on enterprise modules on how to enable and configure each one.

After the new HAProxy Enterprise version has been running for a while, you can uninstall the old HAProxy Enterprise version. To uninstall a package:

```
sudo pkg delete hapee-<PREVIOUS VERSION>-lb
```

Update HAProxy Enterprise

The **update** procedure installs the latest build for your current HAProxy Enterprise version in order to benefit from the latest bug fixes.

```
sudo pkg update
sudo pkg upgrade
```

Upgrade HAProxy Enterprise on Docker

You can upgrade HAProxy Enterprise to version 3.1r1 as a Docker container.

Prepare for the upgrade

1. Issues with the HAProxy Enterprise configuration can cause the load balancer to fail after an upgrade. Even if the service is performing as desired before the upgrade, the upgrade could introduce changes that cause a failure afterward.

   Validate the HAProxy Enterprise configuration file before proceeding with the upgrade.

   Depending on your version, the command's output indicates that the file is valid in these ways:

   - In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
   - In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

   > **Multiple configuration files**
   >
   > If you have multiple configuration files in your application, be sure to check them all in the correct order.

   Example: validate the configuration for version 2.9:

   AWS/BSD/Linux:

   ```
   sudo /opt/hapee-2.9/sbin/hapee-lb -c -f /etc/hapee-2.9/hapee-lb.cfg -V
   ```

   Docker:

   ```
   sudo docker exec hapee /opt/hapee-2.9/sbin/hapee-lb -c -f /etc/hapee-2.9/hapee-lb.cfg -V
   ```

   Alerts and warnings that appear in the console output explain the errors encountered. An alert is a fatal condition and prohibits the service from starting. A warning indicates a condition that is not fatal now but may become fatal in the target release. Address all the errors until you receive the successful output: `Configuration file is valid`.

2. Preserve application files in the old container by copying them to the host computer. For example, copy the configuration directory for version 2.9:

```
sudo docker cp hapee:/etc/hapee-2.9 ./
```

Copy map files, ACL files, certificates, WAF rule files, and so on.

Upgrade to HAProxy Enterprise 3.1r1

The following procedure installs a new major version of HAProxy Enterprise.

1. Log into the `hapee-registry.haproxy.com` Docker registry using your HAProxy Enterprise license key as the username and password.

```
sudo docker login https://hapee-registry.haproxy.com
```

2. Pull the HAProxy Enterprise image.

```
sudo docker pull hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
```

3. Stop and remove any previously running HAProxy Enterprise containers:

```
sudo docker stop <OLD CONTAINER NAME>
sudo docker rm <OLD CONTAINER NAME>
```

4. Start the new Docker container, referencing the directory containing your applications files as a volume by using the `-v` flag.

```
sudo docker run \
  --name hapee-3.1 \
  -d \
  -p 80:80 \
  -p 443:443 \
  -p 5555:5555 \
  -v $(pwd):/etc/hapee-3.1r1  \
  --restart=unless-stopped \
  hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
```

Review the configuration to ensure HAProxy Enterprise successfully locates application files such as the configuration file, map files, ACL files, TLS certificates, and so on.

5. Validate the HAProxy Enterprise configuration file against the new version:

```
sudo docker exec hapee /opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg -V
```

Alerts and warnings that appear in the console output explain the errors encountered. Address all the errors until you receive the successful output: `Configuration file is valid`.

Upgrade HAProxy Enterprise on Linux

Using HAProxy Fusion?

If you're using HAProxy Fusion, then see the HAProxy Fusion - Nodes topic instead.

To upgrade to a new, major version of HAProxy Enterprise (e.g. 3.1r1) from an older version, please use the **Upgrade** procedure. However, if you want to install the latest patches for your current version without upgrading to a new, major version, please use the **Update** procedure.

You can upgrade HAProxy Enterprise to version 3.1r1 on the following operating systems:

HAProxy Enterprise version	Release date	End of life	Supported OS (architecture)
3.1r1	Feb 2025	Feb 2026	• AlmaLinux 8 (x86_64) AlmaLinux 9 (x86_64) • Debian 11 (amd64) Debian 12 (amd64) • RHEL 8 (x86_64) RHEL 9 (x86_64) • Rocky Linux 8 (x86_64) Rocky Linux 9 (x86_64) • Oracle 8 (x86_64) Oracle 9 (x86_64) • SUSE 15.6 (x86_64) • Ubuntu 22.04 (amd64, arm64) Ubuntu 24.04 (amd64, arm64)

Upgrade to a new version

The following **upgrade** procedure installs a new major version of HAProxy Enterprise.

1. Issues with the HAProxy Enterprise configuration can cause the load balancer to fail after an upgrade. Even if the service is performing as desired before the upgrade, the upgrade could introduce changes that cause a failure afterward.

   Validate the HAProxy Enterprise configuration file before proceeding with the upgrade.

Depending on your version, the command's output indicates that the file is valid in these ways:

- In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
- In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

> **Multiple configuration files**
>
> If you have multiple configuration files in your application, be sure to check them all in the correct order.

Example: validate the configuration for version 2.9:

AWS/BSD/Linux:

```
sudo /opt/hapee-2.9/sbin/hapee-lb -c -f /etc/hapee-2.9/hapee-lb.cfg -V
```

Docker:

```
sudo docker exec hapee /opt/hapee-2.9/sbin/hapee-lb -c -f /etc/hapee-2.9/hapee-lb.cfg -V
```

Alerts and warnings that appear in the console output explain the errors encountered. An alert is a fatal condition and prohibits the service from starting. A warning indicates a condition that is not fatal now but may become fatal in the target release. Address all the errors until you receive the successful output: `Configuration file is valid`.

2. If the HAProxy Enterprise server is part of a high-availability cluster, take it out of the cluster:
   - If using Active/Active clustering, update your network to stop sending traffic to the load balancer. For example, if using DNS roundrobin, remove the load balancer from DNS.
   - If using Active/Standby clustering, then on the active load balancer edit `/etc/hapee-extras/hapee-vrrp.cfg`. Lower the `priority` value to less than the standby load balancer's `priority`. This will put the active load balancer into the standby state and simultaneously activate the standby load balancer. Then restart the `hapee-extras-vrrp` service.
3. Install HAProxy Enterprise by running the following command, specifying the desired version with `--version`, and replacing `<HAProxy Enterprise Key>` with your HAProxy Enterprise license key. Below, we install HAProxy Enterprise 3.1r1:

```
wget https://www.haproxy.com/static/install_haproxy_enterprise.sh
sudo bash ./install_haproxy_enterprise.sh \
  --version 3.1r1 \
  --key <HAProxy Enterprise key>
```

4. Update additional modules: you may be running additional modules with HAProxy Enterprise, which you must also reinstall when you upgrade to a new release. Install a package using its name, for example:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-update
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-update
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-update
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-update
```

5. Copy `/etc/hapee-<PREVIOUS VERSION>/hapee-lb.cfg` over to `/etc/hapee-3.1/hapee-lb.cfg` and any associated files (maps, certificates, etc).

6. Optionally, you might want to validate your configuration against the new load balancer before disabling the old load balancer. Use the `-c` option to check the configuration.

   ```
   sudo /opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg -V
   ```

   Alerts and warnings that appear in the console output explain the errors encountered. Address all the errors until you receive the successful output: `Configuration file is valid`.

7. Disable the old HAProxy Enterprise version from starting with the system, as follows:

   ```
   sudo systemctl disable hapee-<PREVIOUS VERSION>-lb
   ```

8. Ensure that the new HAProxy Enterprise version starts with the system, as follows:

   ```
   sudo systemctl enable hapee-3.1-lb
   ```

9. Stop the old HAProxy Enterprise daemon and start the new daemon:

   ```
   sudo systemctl stop hapee-<PREVIOUS VERSION>-lb
   sudo systemctl start hapee-3.1-lb
   ```

10. Run `tail -f /var/log/syslog` (or distribution equivalent) to check for warnings or errors.

11. Run `curl localhost` to ensure that HAProxy Enterprise is responding. Adjust address/port as needed, and use a local address that is currently active on the box and not a VRRP IP.

12. Put the upgraded server back into the cluster.

13. **Repeat this procedure on secondary load balancers after restoring the current load balancer to active service.**

Post-upgrade

After you upgrade, see the section on enterprise modules on how to enable and configure each one.

After the new HAProxy Enterprise version has been running for a while, you can uninstall the old HAProxy Enterprise version. To uninstall a package:

Debian:

```
sudo apt purge hapee-<PREVIOUS VERSION>-lb
sudo rm /etc/rsyslog.d/<PREVIOUS VERSION>.conf
sudo systemctl restart rsyslog
```

Ubuntu:

```
sudo apt purge hapee-<PREVIOUS VERSION>-lb
sudo rm /etc/rsyslog.d/<PREVIOUS VERSION>.conf
sudo systemctl restart rsyslog
```

Redhat:

```
sudo yum remove hapee-<PREVIOUS VERSION>-lb
sudo rm /etc/rsyslog.d/<PREVIOUS VERSION>.conf
sudo systemctl restart rsyslog
```

Suse:

```
sudo zypper remove hapee-<PREVIOUS VERSION>-lb
```

Update HAProxy Enterprise

The **update** procedure installs the latest build for your current HAProxy Enterprise version in order to benefit from the latest bug fixes.

1. Optional: Save the current load balancer configuration as a backup. For example, on HAProxy Enterprise 3.1r1:

```
sudo cp /etc/hapee-3.1/hapee-lb.cfg /etc/hapee-3.1/hapee-lb.cfg.current
```

2. Optional: Simulate the package upgrade to check for potential errors:

Debian:

```
sudo apt update
sudo apt upgrade --simulate
```

Ubuntu:

```
sudo apt update
sudo apt upgrade --simulate
```

Redhat:

```
sudo yum check-update
```

Suse:

```
sudo zypper update --dry-run
```

Search the output for `hapee`, which will show the new package version. If there are no errors, perform the upgrade.

3. Upgrade the packages on your system. Note that a prompt may display asking whether the system can restart service automatically during the upgrade. To avoid any disruption to service, you can uncheck HAProxy Enterprise from this list and then manually call `systemctl reload hapee-3.1-lb`, which will not drop traffic.

Debian:

```
sudo apt update
sudo apt upgrade
```

Ubuntu:

```
sudo apt update
sudo apt upgrade
```

Redhat:

```
sudo yum update
```

Suse:

```
sudo zypper update
```

4. Reload the HAProxy Enterprise service to start the new version. For example, on HAProxy Enterprise 3.1r1:

```
sudo systemctl reload hapee-3.1-lb
```

Administration

Manage HAProxy Enterprise configuration files

This page applies to:

- HAProxy Enterprise - all versions

By default, the HAProxy Enterprise service loads only the configuration file, `/etc/hapee-<VERSION>/hapee-lb.cfg`.

Validate your configuration

You can validate a configuration file using the `hapee-lb` program with the `-c` flag.

Depending on your version, the command's output indicates that the file is valid in these ways:

- In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
- In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

Example:

```
sudo /opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg -V
```

Alerts and warnings that appear in the console output explain the errors encountered. An alert is a fatal condition and prohibits the service from starting. A warning indicates a condition that is not fatal now but may become fatal in the target release. Fix both types of conditions before proceeding.

Optional: Split the load balancer configuration file

By default, the HAProxy Enterprise service loads only one configuration file, `/etc/hapee-<VERSION>/hapee-lb.cfg`. If you have a complex configuration and edit it yourself, you may want to consider splitting the configuration into multiple configuration files.

Info

You cannot split a configuration into multiple files if you use the Data Plane API to make configuration changes. The Data Plane API operates only on single-file configurations.

A multi-file configuration is typically used for load balancers that support multiple applications. In such a configuration, the default configuration file would contain the `global` directives shared by all applications, and the `defaults`, `frontend`, and `backend` sections for each application would be contained in their own configuration files. Thus, for example, if there were two applications, there might be one file for `global` directives, a second file containing the `defaults`, `frontend`, and `backend` sections for application A, and a third file containing the `defaults`, `frontend`, and `backend` sections for application B.

To split the load balancer configuration:

1. Create the directory `/etc/hapee-<VERSION>/lb-conf.d`. For example:

   ```
   sudo mkdir /etc/hapee-3.1/lb-conf.d/
   ```

2. In the `lb-conf.d` directory, create a configuration file for one application, for example, `app-A.cfg`.

3. Move the application A sections from the main configuration file, `/etc/hapee-<VERSION>/hapee-lb.cfg`, to the application A configuration file `/etc/hapee-<VERSION>/lb-conf.d/app-A.cfg`.

4. Repeat this process for the remaining applications.

   > **Info**
   >
   > - Do not split configuration sections. A configuration section (such as `global`, `defaults`, `frontend`, or `backend`) must be completely contained in a single file and cannot be split across multiple files.
   > - Configuration files must have the suffix `.cfg`.
   > - The lexical sort order of the names determines the order in which they are loaded at start time.

5. Edit the service environment file:

   - On Debian/Ubuntu: `/etc/default/hapee-<VERSION>-lb`
   - On other operating systems: `/etc/sysconfig/hapee-<VERSION>-lb`

   Change the `OPTIONS` line so that it contains the following `-f` argument, which loads the files in the `lb-conf.d` directory when the HAProxy Enterprise service starts:

   hapee-3.1-lb

   ```
   OPTIONS="-f /etc/hapee-3.1/lb-conf.d/"
   ```

6. Restart the service:

```
sudo systemctl restart hapee-3.1-lb
```

With this configuration, when the load balancer starts, it first loads the default file containing the `global` section. Then it loads the files in the `lb-conf.d` directory in lexical sort order.

Validating the configuration

1. Use the `hapee-lb` program's `-c` flag to validate the configuration files in configurations split as described above.

   Depending on your version, the command's output indicates that the file is valid in these ways:

   - In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
   - In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

   ```
   sudo /opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg -f /etc/hapee-3.1/lb-conf.d/ -V
   ```

   Alerts and warnings that appear in the console output explain the errors encountered. An alert is a fatal condition and prohibits the service from starting. A warning indicates a condition that is not fatal now but may become fatal in the target release. Address all the errors until you receive the successful output: `Configuration file is valid`.

High availability

Active/Active clustering

This page applies to:

- HAProxy Enterprise - all versions

In an active-active cluster, two or more HAProxy Enterprise nodes receive traffic in a load-balanced rotation. This allows you to scale out your load-balancing capacity. There are several ways to achieve this setup, including:

- Using DNS round-robin
- Deploy two tiers of load balancers
- Using Anycast with the HAProxy Enterprise Route Health Injection module (see Route Health Injection)

Option 1: DNS round-robin

GSLB for DNS round-robin

You can use an HAProxy Enterprise server for DNS round-robin. See GSLB.

To create an active-active cluster of load balancers, you can use DNS round-robin to send traffic to each load balancer in a rotation.

To remove unhealthy HAProxy Enterprise nodes from round-robin load-balancing rotation, you need a way to detect failure. If your DNS server supports health checking upstream servers, you can use that feature to remove unhealthy nodes. However, in this section, we assume that your DNS server does not support health checking. Instead, you need to configure the HAProxy Enterprise VRRP module to assign a virtual IP address to each load balancer. In the event that one of your load balancers fails, its virtual IP address will transfer to a healthy load balancer node.

A side effect of this failover is that your healthy node, which had been receiving its own share of traffic already, will temporarily take on more traffic until the other node recovers.

In this guide, we set up two load balancers: both actively receiving traffic.

Configure VRRP

1. Install the VRRP module using your system's package manager on both HAProxy Enterprise nodes that will participate in your load balancer cluster:

Apt:

```
sudo apt-get install hapee-extras-vrrp
```

Yum:

```
sudo yum install hapee-extras-vrrp
```

Zypper:

```
sudo zypper install hapee-extras-vrrp
```

Pkg:

```
sudo pkg install hapee-extras-vrrp
```

2. Decide on a unique Virtual Router Identifier (VRID) for each of the two clusters we will create. A VRID can be any number between 1 and 255. It allows the instances in a cluster to share a virtual router and virtual IP address. The VRID must be the same on all instances in a cluster. When there are multiple VRRP clusters in the environment, as there are in the active-active configuration, the VRID must be unique for each cluster.

   When defining a new cluster, do not use a VRID already in use for another cluster. If the `vrrp` service has been started and enabled, you can list VRIDs already in use by executing the following command:

```
sudo tcpdump -vvvenns0 -c 5 -i eth0 vrrp | grep -o "vrid [0-9]*"
```

output

```
[...]
5 packets captured
6 packets received by filter
0 packets dropped by kernel
vrid 161
vrid 155
```

   In our example configuration, we will use VRID `51` for one VRRP cluster and VRID `52` for the other cluster.

3. Decide on a unique password for each cluster. A VRRP password is a clear-text string for use by all instances in a VRRP cluster. It does not provide security, but it ensures that VRRP instances from other clusters cannot join this cluster due to errors in configuration, such as a duplicate VRID. In our example configuration, we'll use `1234` as the password for the cluster with VRID `51` and `5678` as the password for the cluster with VRID `52`.

4. On the first load balancer, edit the file `/etc/hapee-extras/hapee-vrrp.cfg`. The file contains one `vrrp_instance` block. Delete the block and insert the following code in its place:

hapee-vrrp.cfg

```
vrrp_instance vrrp_1 {
  interface enp0s8                    # Change network interface name
  state MASTER
  virtual_router_id 51           # VRID unique to this cluster
  authentication {
    auth_type PASS
    auth_pass 1234               # Password unique to this cluster
  }
  priority 101
  virtual_ipaddress_excluded {
    192.168.50.10                # NEW IP address
  }
  track_interface {
    enp0s8 weight -2             # Change network interface name
  }
  track_script {
    chk_sshd
    chk_lb
  }
}

vrrp_instance vrrp_2 {
  interface enp0s8               # Change network interface name
  state BACKUP
  virtual_router_id 52           # VRID unique to this cluster
  authentication {
    auth_type PASS
    auth_pass 5678               # Password unique to this cluster
  }
  priority 100
  virtual_ipaddress_excluded {
    192.168.50.11                # NEW IP address
  }
  track_interface {
    enp0s8 weight -2             # Change network interface name
  }
  track_script {
    chk_sshd
    chk_lb
  }
}
```

5. Make the following modifications to these two `vrrp_instance` blocks:

   - Replace the `interface` values with the name of the network interface on which this server receives traffic. The interface is the same for the two instances.
   - Replace the `virtual_router_id` fields with the VRIDs determined previously. Each instance should have a different VRID.
   - Replace the `auth_pass` fields with the passwords determined previously. Each instance should have a different password.
   - Replace the IP addresses listed in the `virtual_ipaddress_excluded` blocks with addresses you'd like to use to receive traffic. These new addresses should fall within the interface's IP subnet, but should not be assigned to any server already. Each instance should have a different IP address.
   - In the `track_interface` blocks, specify the same interface name that is specified in the `interface` fields.

   With these modifications, the configuration creates two virtual IP addresses: `192.168.50.10` and `192.168.50.11`. The first has its `state` set to `MASTER`, while the second is set to `BACKUP`. This means that this load balancer owns the first IP address but will only bind to the second IP address as a backup if its primary owner fails.

6. Copy the file `/etc/hapee-extras/hapee-vrrp.cfg` to the second load balancer.

7. On the second load balancer, edit the file and make the following changes:

   - Swap the `state` values so that the first `vrrp_instance` block is set to `BACKUP`, while the second is set to `MASTER`.
   - Swap the `priority` values.

hapee-vrrp.cfg

```
vrrp_instance vrrp_1 {
  interface enp0s8                  # Change network interface name
  state BACKUP
  virtual_router_id 51          # VRID unique to this cluster
  authentication {
    auth_type PASS
    auth_pass 1234              # Password unique to this cluster
  }
  priority 100
  virtual_ipaddress_excluded {
    192.168.50.10               # NEW IP address
  }
  track_interface {
    enp0s8 weight -2            # Change network interface name
  }
  track_script {
    chk_sshd
    chk_lb
  }
}

vrrp_instance vrrp_2 {
  interface enp0s8              # Change network interface name
  state MASTER
  virtual_router_id 52          # VRID unique to this cluster
  authentication {
    auth_type PASS
    auth_pass 5678              # Password unique to this cluster
  }
  priority 101
  virtual_ipaddress_excluded {
    192.168.50.11               # NEW IP address
  }
  track_interface {
    enp0s8 weight -2            # Change network interface name
  }
  track_script {
    chk_sshd
    chk_lb
  }
}
```

With these configurations, each load balancer serves as a backup for the other. If one fails, its virtual IP will transfer to the other node, which will at that point answer requests from both addresses until the failed node recovers.

8. Start the `hapee-extras-vrrp` service on both servers:

```
sudo systemctl start hapee-extras-vrrp
sudo systemctl enable hapee-extras-vrrp
```

9. Edit your HAProxy Enterprise configurations to listen on the virtual IP addresses. Note that you must append the field `transparent`, indicating that the address will be bound even if it does not belong to the local machine, which is necessary since the virtual IP will float to the active load balancer.

```
frontend myfrontend
  mode http
  bind 192.168.50.10:80 transparent
  bind 192.168.50.11:80 transparent
  default_backend web_servers

backend web_servers
  server s1 192.168.50.20:80
```

Alternatively, if you prefer to not add `transparent` to every `bind` line, you can set the kernel parameter `ip_nonlocal_bind`. Edit the file `/etc/sysctl.conf` and add the line `net.ipv4.ip_nonlocal_bind=1`:

sysctl.conf

```
net.ipv4.ip_nonlocal_bind=1
```

Then run `sudo sysctl -p` to reload the configuration.

As a third option, you can listen on all IP addresses assigned to the server by omitting the IP address:

```
frontend myfrontend
  mode http
  bind :80
  default_backend web_servers
```

In essence, we have created an active-active cluster by creating two active-standby clusters, one in each direction. Each load balancer has an active IP address, but also serves as a standby server for the other IP address.

Configure DNS round-robin

DNS allows you to rotate which IP address is returned in response to a DNS query. When clients request your service by its domain name, they will receive the IP address of the next load balancer in the list. Clients tend to cache DNS results, so once a client receives a DNS answer, it will likely continue making requests to the same load balancer until the DNS answer expires.

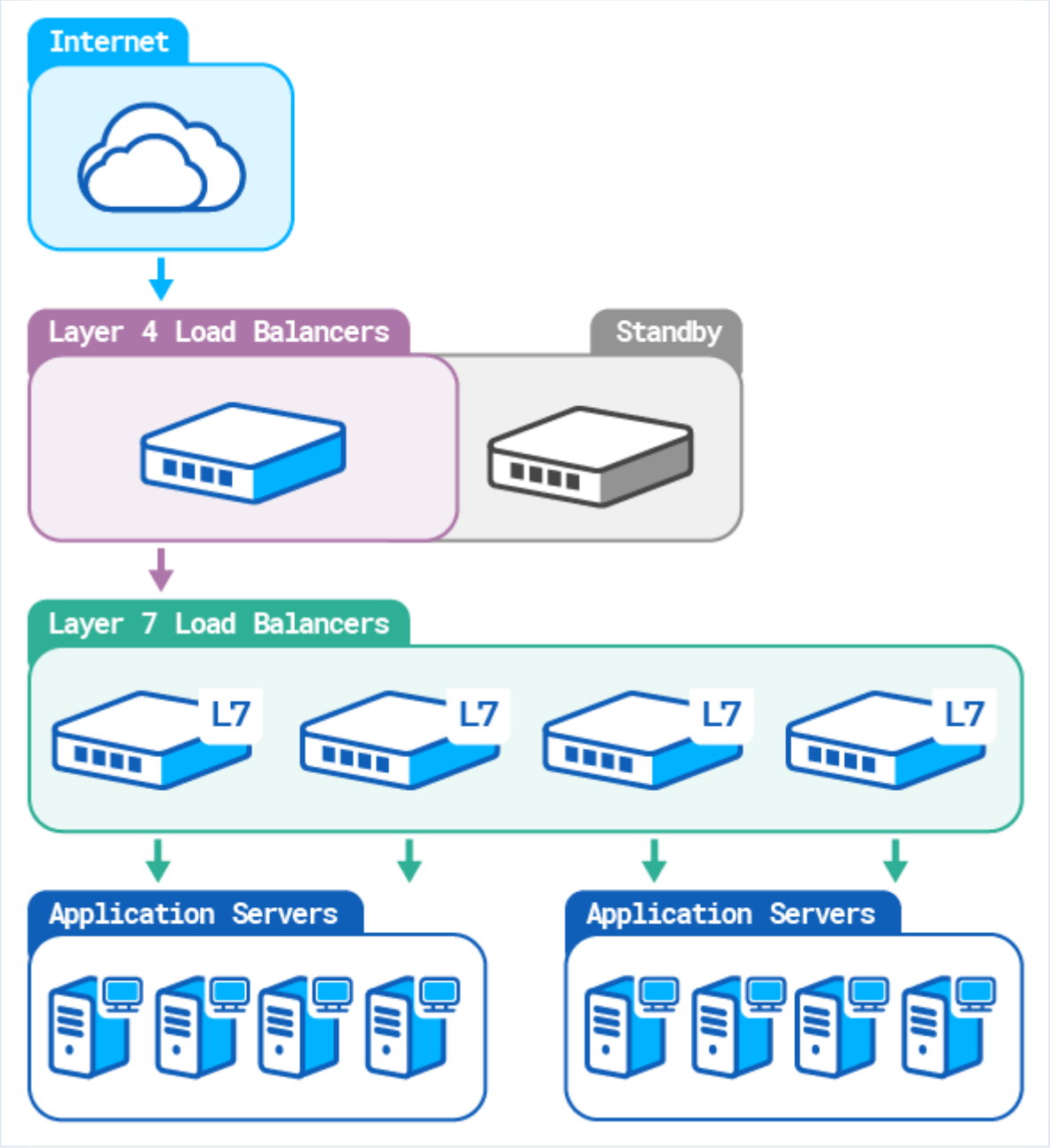
Create an `A` record for each load balancer IP address. For example, consider the following DNS zone file:

```
$ORIGIN example.local.

@       3600 IN SOA dns1.example.local. admin.example.local. (
            2017042745 ; serial
            1800       ; refresh (30 minutes)
            900        ; retry (15 minutes)
            1209600    ; expire (2 weeks)
            60       ; minimum (1 minute)
         )

        3600 IN NS dns1.example.local.

@     60 IN A     192.168.50.10
@     60 IN A     192.168.50.11
www   60 IN CNAME @
```

Depending on your DNS server, you may need to enable load-balancing of the `A` records. It is also a good idea to set a shorter TTL for these records to avoid staying cached in intermediate nameservers for long.

Option 2: Two tiers of load balancers

To create an active-active cluster of load balancers, you can place another tier of HAProxy Enterprise load balancers in front. These load balancers operate at Layer 4, the Network layer, and load balance the load balancers. Because HAProxy Enterprise has the ability to check the health of the lower tier of load balancers, it can remove unhealthy nodes as needed.



Deploy Layer 7 load balancers

Create a tier of load balancers that will, themselves, be load-balanced:

1. Deploy at least two servers to host HAProxy Enterprise. Add one or more static IP addresses to each server.
2. Install HAProxy Enterprise on each server.
3. Configure them to receive traffic on the assigned IP addresses and relay it to backend application servers. Other than the addresses, their configurations should match.

Deploy Layer 4 load balancers

Create a tier of load balancers that will load balance the layer 7 HAProxy Enterprise servers:

1. Deploy one or more servers to host HAProxy Enterprise. These will load balance the other load balancers.
2. If you deploy multiple load balancers in this tier, configure them in active-standby mode.
3. Configure them to relay traffic to the layer 7 load balancers. In the following example, a `listen` section defines the pool of upstream load balancers:

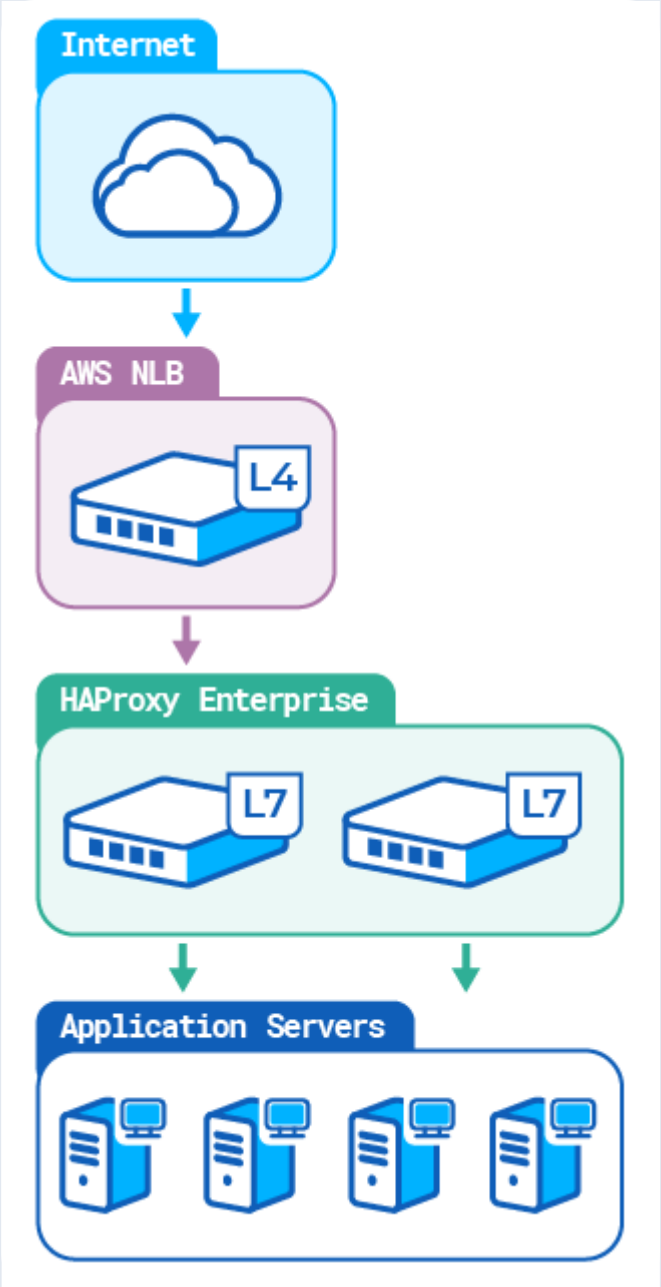
```
listen load_balancer_cluster
  mode tcp
  bind :80
  option tcplog
  balance roundrobin
  server lb1 192.168.1.25:80 check
  server lb2 192.168.1.26:80 check
```

Option 3: ECMP with the Route Health Injection module

Use Equal-cost multi-path routing (ECMP) to load balance traffic to two load balancers. Combine this with the Route Health Injection module to ensure that load balancers are removed from load balancing if they lose connectivity to backend servers. For details, see Route Health Injection module.

Active/active setup on AWS

To create an active-active cluster of HAProxy Enterprise load balancers in AWS, you can place a Network Load Balancer in front. This load balancer operates at Layer 4, the Network layer, and load balances the HAProxy Enterprise load balancers. Because AWS Network Load Balancer can check the health of the lower-tier of load balancers, it can remove unhealthy load balancers as needed.



We will:

- Create a virtual network dedicated to your AWS account (VPC).
- Deploy several HAProxy Enterprise AMIs.
- Create an AWS target group.
- Create an AWS Network Load Balancer in a single Availability Zone.
- Test the setup.

Create a VPC

You can launch HAProxy Enterprise nodes in a Virtual Private Cloud (VPC), which is a virtual network similar to a traditional network.

1. Open the Amazon VPC console, then click **Launch VPC Wizard**.
2. Create a **VPC with a Single Public Subnet** with **Public subnet's Availability Zone** set to the availability zone of your choice, for example, `eu-west-3a`. In other fields, keep the default values or select other values that better suit your needs.
3. In the **Virtual Private Cloud** section on the left, select **Your VPCs**, then write down the ID of the new VPC. For example, `vpc-0deecc96935b9ef73`.

Deploy HAProxy Enterprise AMIs

You can launch HAProxy Enterprise nodes directly from the AWS Marketplace. Create two or more HAProxy Enterprise nodes on AWS, with the following characteristics:

Field	Value
Software version	The version of HAProxy Enterprise.
Region	Region to which your Availability Zone belongs.
VPC Settings	ID of the VPC you previously created. For example, `vpc-0deecc96935b9ef73`.
Subnet Settings	Public subnet that belongs to the VPC.
Security Group Settings	Create a new security group based on seller settings for the first HAProxy Enterprise node. Then select the same security group for other nodes.
Key Pair Settings	Create a new EC2 key pair, or select an existing one. Select the same key pair for all HAProxy Enterprise nodes.

In other fields, keep the default values or select other values that better suit your needs.

Create an AWS target group

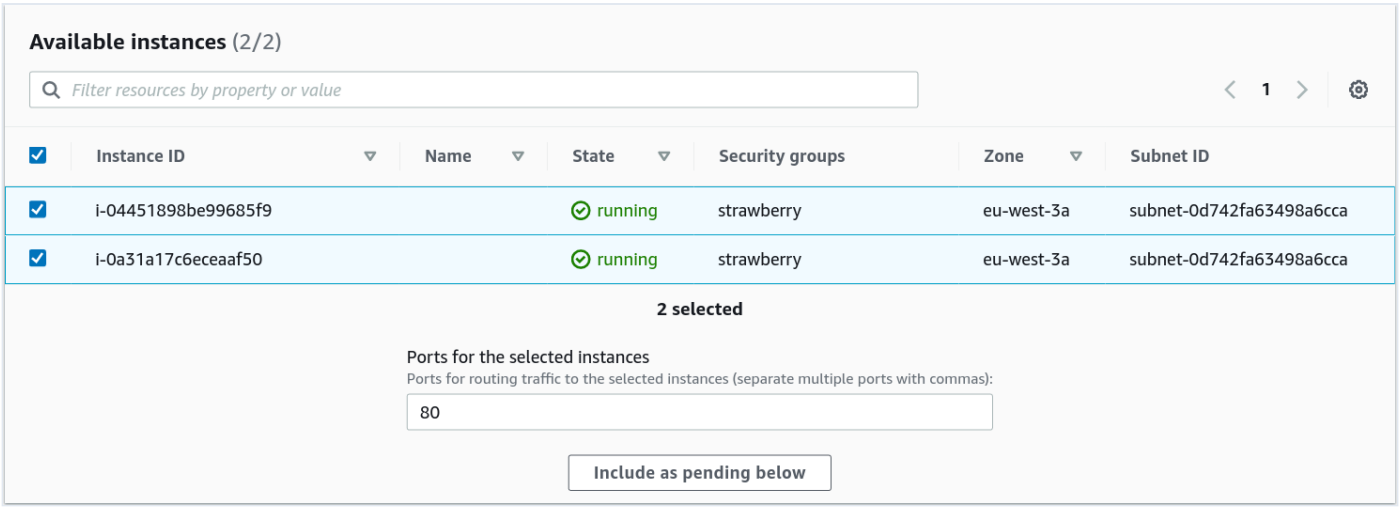
A target group routes requests to one or more registered targets, such as HAProxy Enterprise nodes, using the TCP protocol and the port number that you specify.

1. Open the Amazon EC2 console.
2. In the **Load Balancing** section on the left, select **Target Groups**.
3. Create a target group with the following characteristics, then click **Next**.

Field	Value
Target type	Instances
Protocol	TCP
VPC	The virtual private cloud (VPC) you created previously. For example, `vpc-0deecc96935b9ef73`.

In the other fields, keep the default values or select other values that better suit your needs.

4. On the **Register targets** page, select the HAProxy Enterprise nodes you created previously. Then click **Include as pending below**.



5. Click **Create target group**.

Create an AWS Network Load Balancer

The AWS Network Load Balancer selects a target HAProxy Enterprise node using a flow hash algorithm based on the source and destination IP addresses and ports, the protocol, and the TCP sequence number.

1. In the **Load Balancing** section on the left, select **Load Balancers**.

2. Create a Network Load Balancer with the following characteristics:

Field	Value
Scheme	Internet-facing.
VPC	The virtual private cloud (VPC) you created previously. For example, `vpc-0deecc96935b9ef73`.
Mappings	Select your Availability Zone and the corresponding subnet.
Listeners	Default (a listener that accepts TCP traffic on port 80).
Default action	Select the target group you created previously.

In the other fields, keep the default values or select other values that better suit your needs.

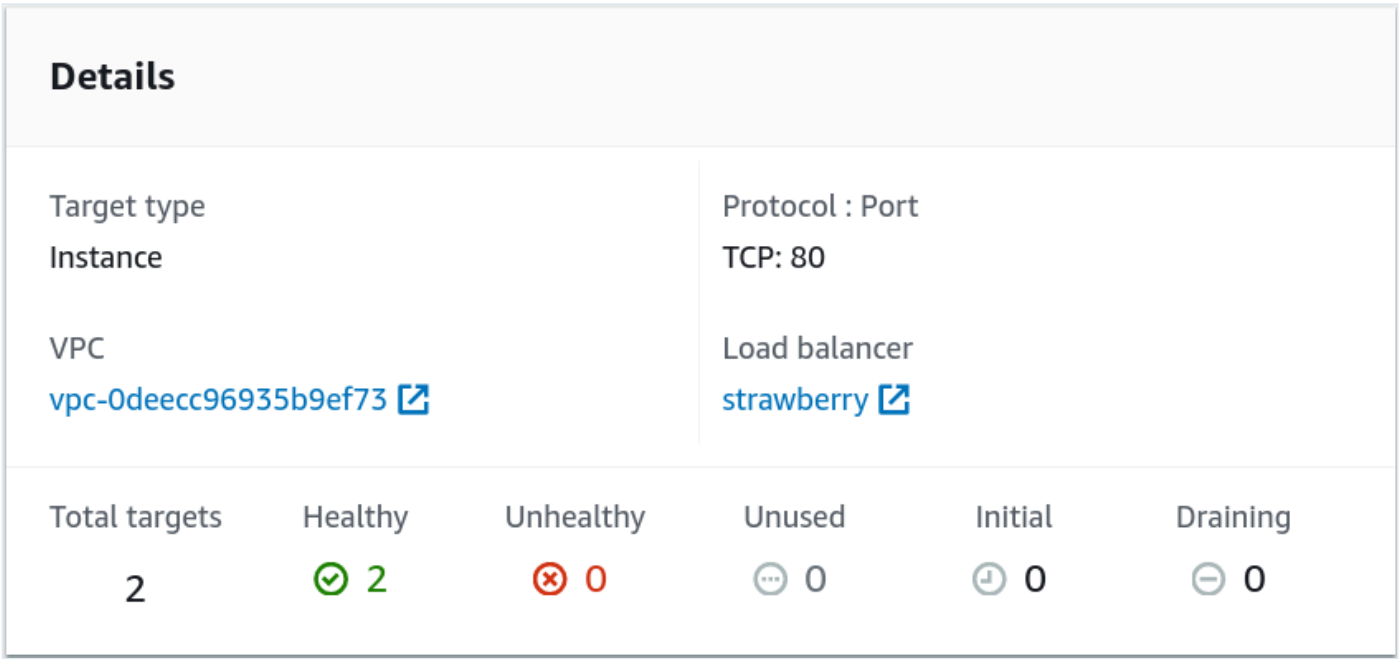
3. Click **Create load balancer**.

Test your setup

Once you have created a Network Load Balancer, wait a few minutes and check that the HAProxy Enterprise nodes in your target group have passed the initial health check. You can then test that the layer 4 AWS Network Load Balancer sends traffic to your layer 7 HAProxy Enterprise load balancers.

1. In the **Load Balancing** section on the left, select **Target Groups**.

2. Select the newly created target group, and check that your HAProxy Enterprise nodes are ready.



3. In the **Load Balancing** section on the left, select **Load Balancers**.

4. Select the newly created load balancer.

5. Copy the DNS name of the load balancer and paste it into the address field of a web browser.

   For example, `strawberry-f9f565c7eb5b3cd3.elb.eu-west-3.amazonaws.com`.

   The browser displays the statistics page of your HAProxy Enterprise node.

   If you launched nodes with different versions of HAProxy Enterprise, paste the DNS name of the Network Load Balancer into the address field of another web browser or in a private browser window. The browser should display the statistics page of one of your other HAProxy Enterprise nodes, which runs another version of HAProxy Enterprise. Since each TCP connection is routed to a single target as long as the connection is active, you may have to try connecting several times.

You can now:

- configure your backend server pool.
- edit the `frontend` section of your HAProxy configuration file (each AWS Network Load Balancer in the Availability Zone has a static IP).
- edit the `backend` section of your HAProxy configuration file.
- copy the configuration file to all HAProxy Enterprise nodes in the AWS target group.

See also

- For complete information about the Global Server Load Balancing module, see GSLB.
- For complete information about NS1 integration with HAProxy, see NS1 integration.
- For complete information about the Route Health Injection module, see Route Health Injection.
- To understand Amazon's regions and availability zones, see the AWSEC2 User Guide.
- To understand and work with the Amazon Network Load Balancer, see the Amazon Elastic Load Balancing User Guide.
- For an introduction to the Amazon VPC, see the Amazon VPC User Guide.

Configure two HAProxy Enterprise servers for active/standby clustering

This page applies to:

- HAProxy Enterprise - all versions

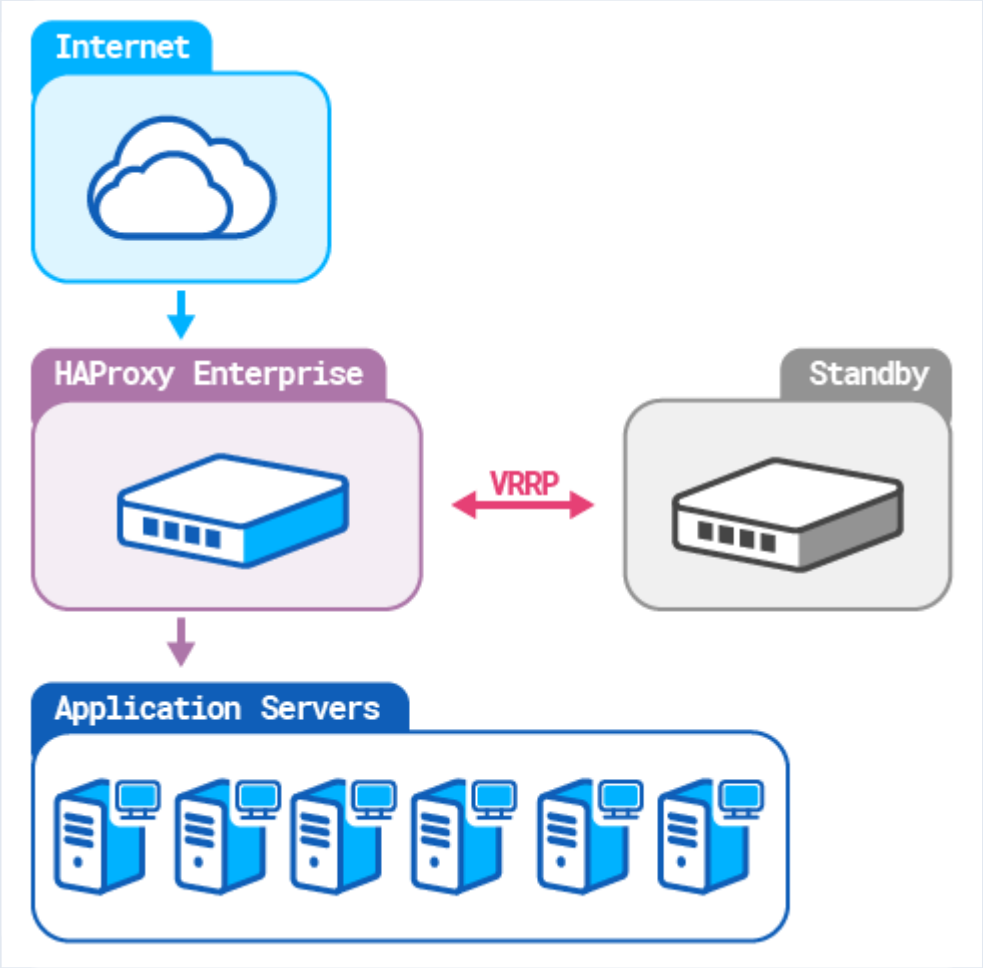
Using HAProxy Fusion?

If you're using HAProxy Fusion, then see the HAProxy Fusion - High availability topic instead.

In an active-standby cluster, one HAProxy Enterprise node receives traffic while a second node is put on standby. In case the active node fails, the standby takes over. With HAProxy Enterprise, you configure an active-standby cluster by installing the Virtual Router Redundancy Protocol (VRRP) module.

The HAProxy Enterprise VRRP module assigns a virtual, static IP address to your load balancer node. If the node fails, the IP address will instantly transfer to a backup node, avoiding any break in service. Your router can continue to send traffic to the same IP address for the load balancer, never knowing that there was a failover. The module utilizes a stable version of Keepalived, which implements VRRP.

In this guide, we will set up two load balancers: one active and the other on standby.



Prerequisites

If you have a firewall running on the server, such as `firewalld` (the default on Red Hat Enterprise Linux) or `iptables`, then be sure to add an exception for VRRP traffic.

- Add an exception to firewalld

  To add an exception for VRRP to `firewalld`:

  1. Check if `firewalld` is running:

     ```
     sudo firewall-cmd --state
     ```

output

```
running
```

2. Check which `firewalld` zones are active for your network interfaces. Below, our active zone is named `public`:

```
sudo firewall-cmd --get-active-zones
```

output

```
public
  interfaces: eth0 eth1
```

3. To ensure that existing runtime rules (those that are in effect) are persisted, save them as permanent before going further:

```
sudo firewall-cmd --runtime-to-permanent
```

output

```
success
```

4. Add the VRRP protocol to the list of rules. Below, we use the zone named `public`:

```
sudo firewall-cmd --zone=public --add-protocol=vrrp --permanent
sudo firewall-cmd --reload
```

output

```
success
```

5. Verify that the protocol has been added. Below, we use the zone named `public`:

```
sudo firewall-cmd --zone=public --list-all | grep vrrp
```

```
protocols: vrrp
```

6. Make these changes persist:

```
sudo firewall-cmd --runtime-to-permanent
```

output

```
success
```

7. If you need to revert this change, run these commands:

```
sudo firewall-cmd --zone=public --remove-protocol=vrrp --permanent
sudo firewall-cmd --reload
```

- Add an exception to iptables

To add an exception for VRRP to `iptables`:

1. Save a backup of your current `iptables` configuration:

```
sudo apt install iptables-persistent
sudo su -c 'iptables-save > /etc/iptables.backup.rules'
```

2. Add the VRRP protocol to the list of rules. Below, we specify the interface `eth0`:

```
sudo iptables -A INPUT -p vrrp -i eth0 -j ACCEPT && sudo iptables -A OUTPUT -p vrrp -j ACCEPT
```

3. Verify that the protocol number 112 (VRRP) has been added:

```
sudo iptables -nvL | grep 112
```

output

```
0     0 ACCEPT     112  --  eth0   *       0.0.0.0/0            0.0.0.0/0
0     0 ACCEPT     112  --  *      *       0.0.0.0/0            0.0.0.0/0
```

4. Save the configuration:

```
sudo su -c 'iptables-save > /etc/iptables.vrrp.rules'
```

5. If you need to revert this change, run this command:

```
sudo iptables-restore < /etc/iptables.backup.rules
```

Installation

Follow the steps on your active and backup load balancers.

Active load balancer

On the active load balancer, follow these steps:

1. Install the VRRP module using your system's package manager:

Apt:

```
sudo apt-get install hapee-extras-vrrp
```

Yum:

```
sudo yum install hapee-extras-vrrp
```

Zypper:

```
sudo zypper install hapee-extras-vrrp
```

Pkg:

```
sudo pkg install hapee-extras-vrrp
```

2. Decide on a unique Virtual Router Identifier (VRID) for the cluster. A VRID can be any number between 1 and 255. It allows the instances in a cluster to share a virtual router and virtual IP address. The VRID must be the same on all instances in a cluster. If there are multiple VRRP clusters in the environment, the VRID must be unique for each cluster.

   Do not use a VRID already in use for another cluster. If the `vrrp` service has been started and enabled, you can list VRIDs already in use by executing the following command:

   ```
   sudo tcpdump -vvvenns0 -c 5 -i eth0 vrrp | grep -o "vrid [0-9]*"
   ```

   output

   ```
   [...]
   5 packets captured
   6 packets received by filter
   0 packets dropped by kernel
   vrid 161
   vrid 155
   ```

   In our example configuration, we will use VRID `51`.

3. Decide on a password for the cluster. The password is a clear-text string for use by all instances in this VRRP cluster. It does not provide security, but it ensures that VRRP instances from other clusters cannot join this cluster due to errors in configuration, such as a duplicate VRID.

4. Edit the file `/etc/hapee-extras/hapee-vrrp.cfg`.

   - In the `vrrp_instance` section, change the `interface` value from `eth0` to the name of a network interface that receives traffic on the server. This is the same interface that you the bind HAProxy Enterprise's `frontend` to. For example, if traffic reaches the load balancer using interface `enp0s8`. Set this line to `interface enp0s8`.
   - For the `virtual_router_id` value, specify the VRID determined previously.
   - Change the authentication password in `auth_pass` to the password determined previously.
   - Change the IP address listed in the `virtual_ipaddress_excluded` block to the virtual IP address you'd like to assign to this interface. HAProxy Enterprise will bind to this address to receive traffic. If needed, add more IP addresses here, each on its own line. The new address(es) should fall within the interface's IP subnet but shouldn't already be assigned to any server.
   - In the `track_interface` block, specify the same interface name that is specified in the `interface` field.

hapee-vrrp.cfg

```
vrrp_instance vrrp_1 {
  interface enp0s8       # Change network interface name
  state MASTER
  virtual_router_id 51   # VRID unique to this cluster
  authentication {
    auth_type PASS
    auth_pass 1234       # Text string unique to this cluster
  }
  priority 101
  virtual_ipaddress_excluded {
    192.168.50.10        # New IP address
  }
  track_interface {
    enp0s8 weight -2     # Change network interface name
  }
  track_script {
    chk_sshd
    chk_lb
  }
}
```

5. Enable and start the `hapee-extras-vrrp` service:

```
sudo systemctl enable hapee-extras-vrrp
sudo systemctl unmask hapee-extras-vrrp
sudo systemctl start hapee-extras-vrrp
```

6. Allow the server to bind to the virtual IP address.

   Edit the file `/etc/sysctl.conf` and add the `net.ipv4.ip_nonlocal_bind=1` directive. This directive allows the server to accept connections for IP addresses that aren't bound to any of its interfaces, enabling the use of a floating, virtual IP.

sysctl.conf

```
net.ipv4.ip_nonlocal_bind=1
```

7. Configure your `bind` line in the HAProxy Enterprise configuration to use the virtual IP address.

```
frontend myfrontend
  mode http
  bind 192.168.50.10:80
  default_backend web_servers
```

8. Reboot the server.

Standby load balancer

On the standby load balancer, follow these steps:

1. Install the VRRP module using your system's package manager:

Apt:

```
sudo apt-get install hapee-extras-vrrp
```

Yum:

```
sudo yum install hapee-extras-vrrp
```

Zypper:

```
sudo zypper install hapee-extras-vrrp
```

Pkg:

```
sudo pkg install hapee-extras-vrrp
```

2. Edit the VRRP configuration `/etc/hapee-extras/hapee-vrrp.cfg`. It must be exactly the same as the file on the active load balancer, except for these differences:

   - In the `vrrp_instance` block, change the `interface` value to the name of the network interface that receives traffic on the server.
   - Change the `state` value to `BACKUP`.
   - Change the `priority` field to have a lower number than the active node. The load balancer with the highest priority is promoted to be the active node. For example, if the priority on the active node is 101, then set the backup node's priority to 100.
   - In the `track_interface` block, specify the same interface name that is specified in the `interface` field.

hapee-vrrp.cfg

```
vrrp_instance vrrp_1 {
  interface enp0s8          # Change network interface name
  state BACKUP              # Change to BACKUP
  virtual_router_id 51
  authentication {
    auth_type PASS
    auth_pass 1234
  }
  priority 100              # A lower priority value than the primary
  virtual_ipaddress_excluded {
    192.168.50.10           # Same IP address as on primary
  }
  track_interface {
    enp0s8 weight -2        # Change network interface name
  }
  track_script {
    chk_sshd
    chk_lb
  }
}
```

3. Enable and start the `hapee-extras-vrrp` service:

```
sudo systemctl enable hapee-extras-vrrp
sudo systemctl unmask hapee-extras-vrrp
sudo systemctl start hapee-extras-vrrp
```

4. Allow the server to bind to the virtual IP address.

   Edit the file `/etc/sysctl.conf` and add the `net.ipv4.ip_nonlocal_bind=1` directive. This directive allows the server to accept connections for IP addresses that aren't bound to any of its interfaces, enabling the use of a floating, virtual IP.

sysctl.conf

```
net.ipv4.ip_nonlocal_bind=1
```

5. Configure your `bind` line in the HAProxy Enterprise configuration to use the virtual IP address.

```
frontend myfrontend
  mode http
  bind 192.168.50.10:80
  default_backend web_servers
```

6. Reboot the server.

Verify the setup

To check that the VRRP service is running, use `systemctl status`:

```
sudo systemctl status hapee-extras-vrrp --no-pager
```

output

```
● hapee-extras-vrrp.service - HAPEE VRRP : VRRP daemon (Keepalived)
     Loaded: loaded (/lib/systemd/system/hapee-extras-vrrp.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2023-10-18 18:01:59 UTC; 10min ago
   Main PID: 1191 (hapee-vrrp)
      Tasks: 2 (limit: 4555)
     Memory: 1.5M
        CPU: 4.632s
     CGroup: /system.slice/hapee-extras-vrrp.service
             ├─1191 /opt/hapee-extras/sbin/hapee-vrrp -D -f /etc/hapee-extras/hapee-vrrp.cfg -p /var/run/hapee-extras/
hapee-vrrp.pid
             └─1192 /opt/hapee-extras/sbin/hapee-vrrp -D -f /etc/hapee-extras/hapee-vrrp.cfg -p /var/run/hapee-extras/
hapee-vrrp.pid

Oct 18 18:02:01 hapee01 Keepalived_vrrp[1192]: VRRP_Instance(vrrp_1) forcing a new MASTER election
Oct 18 18:02:02 hapee01 Keepalived_vrrp[1192]: VRRP_Instance(vrrp_1) Transition to MASTER STATE
Oct 18 18:02:03 hapee01 Keepalived_vrrp[1192]: VRRP_Instance(vrrp_1) Entering MASTER STATE
Oct 18 18:02:03 hapee01 Keepalived_vrrp[1192]: VRRP_Instance(vrrp_1) setting protocol E-VIPs.
Oct 18 18:02:03 hapee01 Keepalived_vrrp[1192]: VRRP_Instance(vrrp_1) Sending gratuitous ARPs on eth0 for 172.1.1.4
```

Manual failover

To activate the standby load balancer while simultaneously putting the active load balancer into standby mode:

1. Edit the VRRP configuration `/etc/hapee-extras/hapee-vrrp.cfg` on the active load balancer.
2. Lower the `priority` value to less than the standby load balancer's `priority`. This will put the active load balancer into the standby state and simultaneously activate the standby load balancer. For example, change it from 101 to 90.
3. Restart the `hapee-extras-vrrp` service.

   Reverse these actions to restore the original state.

Failover triggers

The following events can trigger a failover:

- The active node lowers its weight below one of the backup nodes due to a failed health check.
- A backup node is reconfigured with a weight larger than the current active node.
- The active node stops emitting its heartbeat packet to the cluster.

VRRP health check scripts

In the VRRP configuration, the `vrrp_script` blocks define health checks that can trigger a failover. By default, our configuration checks whether the SSH daemon is running and whether HAProxy Enterprise is running. You may want to add more checks.

Consider this block, which checks the status of the SSH daemon every five seconds. If the service is not available, the VRRP instance's weight (its priority) is reduced by 4 points:

hapee-vrrp.cfg

```
vrrp_script chk_sshd {
  script "pkill -0 sshd"  # pkill -0 is cheaper than pidof
  interval 5              # check every 5 seconds
  weight -4               # remove 4 points of prio if missing
  fall 2                  # check twice before setting down
  rise 1                  # check once before setting up
}
```

The HAProxy Enterprise health check is similar, except if the `hapee-lb` process is running, it adds 6 points. The reason is, in case one load balancer has SSH working but not `hapee-lb`, while the other has `hapee-lb` but not SSH, the one with `hapee-lb` will become the active node, even though it is not manageable remotely.

hapee-vrrp.cfg

```
vrrp_script chk_lb {
  script "pkill -0 hapee-lb"  # pkill -0 is cheaper than pidof
  interval 1                  # check every second
  weight 6                    # add 6 points of prio if present
  fall 2                      # check twice before setting down
  rise 1                      # check once before setting up
}
```

The following fields define a `vrrp_script` block:

Name	Argument type	Description
`script`	string	Command to run. Running `pkill -0` checks if any process with the given name is running. If it returns 0, the check passes; if it returns 1, the check fails.
`interval`	integer	Interval (in seconds) between two checks.
`weight`	integer	Points to add or remove to instance weight.
`fall`	integer	Number of consecutive negative checks before considered as down.
`rise`	integer	Number of consecutive positive checks before considered as up.

VRRP instance reference

The following fields define a `vrrp_instance` block:

Name	Argument type	Description
`vrrp_instance`	string	Describes a new VRRP instance.
`interface`	string	Interface managed by the VRRP module.
`state`	string	State at startup until VRRP priority negotiation completes. Can be either `MASTER` or `BACKUP`.
`virtual_router_id`	integer	VRRP instance identifier, which must be common to all nodes of the same cluster. Each cluster must have a unique ID. The value can range from 1 to 255.
`priority`	integer	Weight of local node in this VRRP instance. **NOTE** This won't work on cloud provider networks that disable IP multicast.
`virtual_ipaddress`	string	List of virtual IP addresses to add when the state changes to `MASTER` or remove when it changes to `BACKUP`. All VRRP nodes in a cluster must own the same IP. **NOTE** This won't work on cloud provider networks that disable IP multicast.
`virtual_ipaddress_excluded`	string	The same as `virtual_ipaddress`, but IPs aren't announced in the VRRP heartbeat packet, which reduces bandwidth usage when sending packets.
`track_interface`	string	Tracks interface status and updates priority accordingly.
`track_script`	string	Runs the health check scripts and updates priority accordingly.

Troubleshooting

Does a packet capture reveal that packets are reaching the server, but the VRRP service doesn't receive them?

- Check if there is a firewall running that may be blocking the packets. See Prerequisites.

See also

- The VRRP module is a wrapper around the Keepalived software. The syntax we use is different, but you may still find the Keepalived documentation useful.

HAProxy Enterprise license keys

This page applies to:

- HAProxy Enterprise - all versions

This section offers tips for managing your HAProxy Enterprise license key.

Keep your key secret

Your license key is part of the contract defining your organization's rights and responsibilities regarding the use of HAProxy Technologies software and services. Do not share this key outside of your organization.

Retrieve your license key

Some operations, such as package installation and updates, may require your HAProxy Enterprise license key. To retrieve your HAProxy Enterprise license key, execute these commands on a system where you have installed HAProxy Enterprise. Use the commands for your platform.

Apt:

```
sed 's/.*]//' /etc/apt/sources.list.d/haproxy-tech.list | awk -F'/' '{print $7}' | sed 's/\(.*\)-[^-]*/\1/' | uniq
```

Yum:

```
grep '^baseurl=https://www.haproxy.com/download/hapee/key/' /etc/yum.repos.d/haproxy-tech.repo | sed
's~baseurl=https://www.haproxy.com/download/hapee/key/~~' | sed 's~/.*~~' | sed 's/\(.*\)-[^-]*/\1/' | uniq
```

Zypper:

```
grep '^baseurl=https://www.haproxy.com/download/hapee/key/' /etc/zypp/repos.d/haproxy-tech.repo | sed
's~baseurl=https://www.haproxy.com/download/hapee/key/~~' | sed 's~/.*~~' | sed 's/\(.*\)-[^-]*/\1/' | uniq
```

Manage HAProxy Enterprise logs

This page applies to:

- HAProxy Enterprise - all versions

Using HAProxy Fusion?

If you're using HAProxy Fusion, then see the HAProxy Fusion - Logs topic instead.

HAProxy Enterprise generates two types of logs: access logs and administrative logs.

Enable access logs

Access logs, also called traffic logs, record information about client connections and requests. The logs are stored in the `/var/log/hapee-<VERSION>` directory and have file names prefixed with `lb-access-` plus a timestamp.

To enable access logs:

1. Add the following `log` directive in the `global` section of the HAProxy Enterprise configuration file, if one does not already exist:

   ```
   global
     log 127.0.0.1 local0
   ```

   This definition, which does not set a severity level for events, captures all events and tags them with the `local0` syslog facility code. An rsyslog configuration file at `/etc/rsyslog.d/hapee-<VERSION>-lb.conf` routes these events to the access log file.

   You can change the IP address to send logs to a remote rsyslog server over UDP.

2. In your `frontend` or `defaults` section, add a `log global` directive, which applies the `log` directive from the `global` section to the frontend.

   ```
   frontend fe_main
     log global
   ```

Enable administrative logs

Administrative logs record information about load balancer process events such as when the load balancer starts or stops. The logs are stored in the `/var/log/hapee-<VERSION>` directory and have file names prefixed with `lb-admin-` plus a timestamp.

To enable administrative logs:

1. Add the following `log` directive in the `global` section of the HAProxy Enterprise configuration file, if one does not already exist:

   ```
   global
     log 127.0.0.1 local1 notice
   ```

   This definition, which sets its severity level to `notice`, captures most log events but omits the more verbose traffic events captured by the access logs. It tags those events with the syslog facility code `local1`. An rsyslog configuration file at `/etc/rsyslog.d/hapee-<VERSION>-lb.conf` routes these events to the admin log file. Basically, they have the same contents as access logs but without the `info` level details about connection and request events.

   You can change the IP address to send logs to a remote rsyslog server over UDP.

2. In your `frontend` or `defaults` section, add a `log global` directive, which applies the `log` directive from the `global` section to the frontend.

   ```
   frontend fe_main
     log global
   ```

Enable log compression and rotation

> **This section applies to:**
>
> - HAProxy Enterprise 2.0r1 and newer

The `hasavelog` utility, disabled by default, can automatically compress and delete old HAProxy Enterprise logs. The enabled, out-of-the-box defaults for the `hasavelog` utility will:

- compress log files using the `gzip` compression method
- operate in HAProxy Enterprise mode (enforces nocycle, nodate, notouch, nocreate, and noclean)
- create and write a new compressed log file daily into the directory `/var/log/hapee-<VERSION>`
- operate in quiet mode
- keep all log files indefinitely unless specified

To enable the `hasavelog` utility:

1. Open the `hasavelog` utility's configuration file. Located at:

Debian/Ubuntu:

```
/etc/default/hapee-<VERSION>-hasavelog
```

RHEL:

```
/etc/sysconfig/hapee-<VERSION>-hasavelog
```

2. The `HASAVELOG_ENABLE` field accepts either 1 (enabled) or 0 (disabled). Set `HASAVELOG_ENABLE="1"` and save.

3. Start the `hasavelog` service with:

```
sudo systemctl start hapee-<VERSION>-hasavelog
```

Tune the `hasavelog` utility with additional arguments

The `HASAVELOG_ARGS` field tunes the `hasavelog` utility. `HASAVELOG_ARGS` has a default configuration of `-e -r /var/log/hapee-<VERSION> -q`, where logs are stored in `/var/log/hapee-<VERSION>`.

`HASAVELOG_ARGS` can set the following arguments:

Argument	Description
`-e`	Operate in HAProxy Enterprise mode (enforces nocycle, nodate, notouch, nocreate, and noclean). Default: Operates in HAProxy Enterprise mode
`-r <roll-directory-path>`	Sets the roll directory path to `<roll-directory-path>` instead of the current directory. The roll directory path is where compressed files get stored. Default: Rolls log files to `/var/log/hapee-<VERSION>`
`-q`	Operate in quiet mode. Default: Operates in quiet mode
`-l`	Do not compress log files. Default: Compress log files
`-x <path-to-shell-script>`	Invoke a shell script at `<path-to-shell-script>` after a successful log file compression. For example, create `/hook/script.sh` with a script to send notifications to IRC/Slack or send HUP to a syslogd. Default: null
`-s <integer>`	If a log file size is under MB, then that log file will not be compressed. Default: All logs will be compressed no matter the size
`-Z`	Compress with `pigz` using threads to make use of multiple processors and cores. Default: Compress with `gzip`

To tune `HASAVELOG_ARGS`:

1. Open the `hasavelog` utility's configuration file. Located at:

   Debian/Ubuntu:

   ```
   /etc/default/hapee-<VERSION>-hasavelog
   ```

   RHEL:

   ```
   /etc/sysconfig/hapee-<VERSION>-hasavelog
   ```

2. Change the `HASAVELOG_ARGS` field to include the arguments you want to set and save. For example:

   ```
   HASAVELOG_ARGS="-e -r /var/log/hapee-@@HAPEE_MAJORVERSION@@ -q -Z -s 1"
   ```

   In addition to the out-of-the-box defaults, compresses log files over 1 MB in size with the `pigz` compression method.

3. Restart the `hasavelog` service after any saved changes to the `hasavelog` utility configuration file:

   ```
   sudo systemctl restart hapee-<VERSION>-hasavelog
   ```

Set the retention period for logs

The `HASAVELOG_RETAIN` field sets the retention period. It keeps all log files indefinitely with the default value of an empty string (`""`). Using an integer, set `HASAVELOG_RETAIN` to delete logs older than that number of days.

1. Open the `hasavelog` utility's configuration file. Located at:

Debian/Ubuntu:

```
/etc/default/hapee-<VERSION>-hasavelog
```

RHEL:

```
/etc/sysconfig/hapee-<VERSION>-hasavelog
```

2. Change the value of `HASAVELOG_RETAIN` to the number of days you would like to keep log files for. For example, setting `HASAVELOG_RETAIN="30"` will automatically continue to delete any logs older than 30 days.

3. Restart the `hasavelog` service after any saved changes to the `hasavelog` utility configuration file:

```
sudo systemctl restart hapee-<VERSION>-hasavelog
```

Enable audit logging

The `HASAVELOG_AUDIT` field, disabled by default, allows the `hasavelog` utility to log file deletions for compliance. It's printed to stdout. The `HASAVELOG_AUDIT` field accepts either `1` (enabled) or `0` (disabled).

When the field is enabled, the `hasavelog` utility offers more verbose logging:

1. Open the `hasavelog` utility's configuration file. Located at:

Debian/Ubuntu:

```
/etc/default/hapee-<VERSION>-hasavelog
```

RHEL:

```
/etc/sysconfig/hapee-<VERSION>-hasavelog
```

2. The `HASAVELOG_AUDIT` field accepts either 1 (enabled) or 0 (disabled). Set `HASAVELOG_AUDIT="1"` and save.

3. Restart the `hasavelog` service after any changes to the `hasavelog` utility configuration file:

```
sudo systemctl restart hapee-<VERSION>-hasavelog
```

An example output with `HASAVELOG_AUDIT="1"` and `HASAVELOG_RETAIN="2"`:

```
systemctl status hapee-<VERSION>-hasavelog.service
```

output

```
[...]
Jul 22 14:34:27 hostname1 hasavelog[1569]: hasavelog: `find -delete -print' in /var/log/hapee-<VERSION>
Jul 22 14:34:27 hostname1 hasavelog[1575]: hasavelog: /var/log/hapee-<VERSION>/lb-access-20240719.log
Jul 22 14:34:27 hostname1 hasavelog[1575]: hasavelog: /var/log/hapee-<VERSION>/lb-admin-20240719.log
Jul 22 14:34:27 hostname1 systemd[1]: hapee-<VERSION>-hasavelog.service: Succeeded.
Jul 22 14:34:27 hostname1 systemd[1]: Finished Compress HAProxy Enterprise log files.
```

The highlighted lines represent what you will see with audit loggging enabled. On July 22nd, the `hasavelog` utility deleted the logs from July 19th because it is over 2 days old. It then printed the deletion events to stdout.

Set logging levels

Log levels are the same as the severity levels used by the `syslog` service. Below, we set the level to `notice`, which captures all events at that level and above (up to `emerg`):

```
global
  log 127.0.0.1 local1 notice
```

HAProxy Enterprise tags each loggable event with a severity level. For example, it categorizes log messages related to connections and HTTP requests with the `info` severity level. Other events are categorized using one of the other, less verbose levels. From most to least verbose, the severity levels are:

Level	Events at this level
emerg	Errors such as running out of operating system file descriptors.
alert	Some rare cases where something unexpected has happened, such as being unable to cache a response.
crit	Not used.
err	Errors such as being unable to parse a map file, being unable to parse the HAProxy configuration file, and when an operation on a stick table fails.
warning	Certain important, but non-critical, errors such as failing to set a request header or failing to connect to a DNS nameserver.
notice	Changes to a server's state, such as being UP or DOWN or when a server is disabled. Other events at startup, such as starting proxies and loading modules, are also included. Health check logging, if enabled, also uses this level.
info	TCP connection and HTTP request details and errors.
debug	You may write custom Lua code that logs at this level.

Set conditional log levels

Sometimes, you'll want to capture a more verbose log entry for a particular request or response, such as when the server returned an error. To change the log level for individual requests or responses that match specified criteria, use one of these directives, depending on whether you're targeting the request phase or response phase, and whether your're in HTTP or TCP mode:

- `http-request set-log-level <level> [ { if | unless } <condition> ]`
- `http-response set-log-level <level> [ { if | unless } <condition> ]`
- `tcp-request content set-log-level <level> [ { if | unless } <condition> ]`
- `tcp-response content set-log-level <level> [ { if | unless } <condition> ]`

Using these directives, you can set the log level to any of the eight syslog levels or to the special level `silent`, which disables logging for this request/response. This rule is not final so the last matching rule wins.

Specify a condition using the ACL syntax:

```
frontend www
  bind :80
  acl failed_request status 400 401 403 404 405 408 429 500 503
  http-response set-log-level err if failed_request
```

Log ASAP

By default, logs are emitted when the session terminates. That is, when all log values are finalized. Capturing some log values, such as total transfer time and message size, can delay generation of the log content, particularly for a large file transfer.

To avoid such delays, you can choose to log available values as quickly as possible by specifying the `logasap` option. With this option, the load balancer creates the log message as soon as the server connection has been established in `mode tcp` or as soon as the server sends the complete headers in `mode http`.

Values not finalized at this time are total time and total size metrics, and they are prefixed with a plus sign "`+`" in the log to indicate that they are not true, final values.

Enable early logging by adding `option logasap` to the `defaults` or `frontend` section, as shown below. In this example, we also capture the `Content-Length` header in the logs so that the logs at least indicate how many bytes are expected to be transferred.

hapee-lb.cfg

```
frontend fe_main
  bind 0.0.0.0:80
  mode http
  log global
  option httplog
  option logasap
  http-request capture req.hdr(Content-Length) len 15
```

output

```
>>> Feb  6 12:14:14 localhost \
      hapee-lb[14389]: 10.0.1.2:33317 [06/Feb/2019:12:14:14.655] fe_main \
      static/srv1 9/10/7/14/+30 200 +243 - - ---- 3/1/1/1/0 1/0 \
      "GET /image.iso HTTP/1.0"
```

Configure conditional logging

This section applies to:

- HAProxy Enterprise 3.1 and newer

Using the `when()` converter, you can configure the load balancer to emit certain log messages only when certain conditions are met, for example, on error or because of another abnormal state. See when() converter for more information.

Log profiles

This section applies to:

- HAProxy Enterprise 3.1 and newer

In addition to defining log formats individually throughout a load balancer configuration, you can define them in named log profiles that you can then apply wherever needed.

You can define as many log profiles as you want, giving each a name allowing you to apply it independently. With log profiles, you can choose the one best suited for each log server and even emit logs to multiple servers at the same time, each with its own format. Log profiles also allow you to write log messages when different events occur, such as when accepting a connection, receiving a request, connecting to a backend server, and receiving a response.

Log profiles present plenty of opportunities:

- Create a log profile for emitting timing information to see how long HAProxy Enterprise took to handle a request.
- Create a log profile that emits the maximum amount of data for use in debugging.
- Switch the log format just by changing the `profile` argument on the `log` line.
- Reuse profiles across multiple frontends.
- Decide whether you want to emit messages for every step defined in a profile or for only some of them by setting the `log-steps` directive.

In the example below, we define two log profiles: one named `syslog`, which uses the syslog format, and another named `json`, which uses JSON. For `syslog`, we set the `log-tag` directive inside to change the syslog header's `tag` field, to give a hint to the syslog server about how to process the message. Notice that we also get to choose when to emit the log message. We're emitting the log message on the `close` event, when HAProxy has finalized the request-response transaction and has access to all of the data:

hapee-lb.cfg

```
log-profile syslog
  log-tag "haproxy"
  on close format "$HAPROXY_HTTP_LOG_FMT"

log-profile json
  on close format "%{+json}o %(client_ip)ci %(client_port)cp %(request_date)tr %(frontend_name)ft %(backend_name)b %
(server_name)s %(time_to_receive)TR %(time_waiting)Tw %(time_to_connect)Tc %(time_server_response)Tr %(time_active)Ta %
(status_code)ST %(bytes_read)B %(request_cookies)CC %(response_cookies)CS %(termination_state)tsc %
(process_active_connections)ac %(frontend_active_connections)fc %(backend_active_connections)bc %
(server_active_connections)sc %(retries)rc %(server_queue)sq %(backend_queue)bq %(request_headers)hr %
(response_headers)hs %(request_line)r"
```

Our frontend uses both log profiles. By setting the `profile` argument on each log line, the frontend will send `syslog` to one log server and `json` to another. It's required to also enable logging in the frontend, which we accomplish by setting `option httplog`, but `option tcplog` would also work.

hapee-lb.cfg

```
frontend mysite
  bind :80
  option httplog
  log 10.0.0.10:514  format rfc5424 profile syslog local0 info
  log 10.0.0.11:9200 format raw     profile json   local0 info
```

By default, HAProxy emits a log message when the `close` event fires, but you can emit messages on other events, too. The supported events are `accept`, `close`, `connect`, `error`, `request`, and `response`.

By tweaking the syslog profile to include more `on` lines, we have logged a message at each event in HAProxy's processing:

hapee-lb.cfg

```
log-profile syslog
  on accept format "Client connected. IP: %ci"
  on request format "Request received. %r"
  on connect format "Connected to backend server. %b / %s"
  on response format "Response received."
  on close format "$HAPROXY_HTTP_LOG_FMT"
  on error format "Error!"
```

To enable these extra messages, set the `log-steps` directive to `all` or to a comma-separated list of events:

hapee-lb.cfg

```
frontend mysite
  bind :80

  log-steps all
  log 10.0.0.10:514  format rfc5424 profile syslog local0 info
```

Custom logs any time

> **This section applies to:**
>
> - HAProxy Enterprise 3.1 and newer

The `do-log` action allows you to emit custom log messages at any point in the processing of a request or response. Add the `do-log` action where desired in your load balancer configuration. In the example below, we set a variable named `req.log_msg` and then use a `do-log` action to log the variable value.

hapee-lb.cfg

```
frontend mysite
  bind :80

  log 10.0.0.10:514  format rfc5424 profile syslog local0 info

  acl is_evil_client req.hdr(user-agent) -m sub evil
  http-request set-var(req.log_msg) str("Blocking evil client") if is_evil_client
  http-request do-log
  http-request deny if is_evil_client
```

Update your syslog `log-profile` section (see Log profiles) so that it includes the line on `http-req`, which defines the log format to use whenever `http-request do-log` is called.

The following example log format prints the value of the variable `req.log_msg`:

hapee-lb.cfg

```
log-profile syslog
  log-tag "haproxy"
  on close format "$HAPROXY_HTTP_LOG_FMT"
  on http-req format "%[var(req.log_msg)]"
```

Your log will show the custom log message:

```
Blocking evil client
```

The `do-log` action works with other directives, too. Each matches up with a step in the `log-profile` section:

Directive	Matches event
`http-response do-log`	`http-res`
`http-after-response do-log`	`http-after-res`
`quic-initial do-log`	`quic-init`
`tcp-request connection do-log`	`tcp-req-conn`
`tcp-request session do-log`	`tcp-req-sess`
`tcp-request content do-log`	`tcp-req-cont`
`tcp-response content do-log`	`tcp-res-cont`

Set the log format

By default, HAProxy Enterprise logs are sparse, capturing minimal information about each TCP connection in the access logs:

```
>>> Jan 27 21:15:06 localhost hapee-lb[2901]: Connect from 192.168.50.1:61459 to 192.168.50.25:80 \
      (fe_main/HTTP)
```

The information includes:

Field	Example value
the PID of the HAProxy Enterprise process	hapee-lb[2901]
the text “Connect from” followed by the client’s source IP address and port	Connect from 192.168.50.1:61459
the text “to” followed by the destination IP address and port	to 192.168.50.25:80
the name of the frontend that received the request and the mode (e.g. HTTP)	(fe_main/HTTP)

To get a more detailed log, use one of the directives described in the following sections.

TCP log format

The `option tcplog` directive is suitable for `frontend` sections that either specify `mode tcp` or don't set `mode` at all.

```
defaults
  mode tcp
  option tcplog
```

After reloading your configuration, you'll see the following log format:

output

```
>>> Jan 27 21:22:53 localhost hapee-lb[3042]: 192.168.50.1:61535 [27/Jan/2021:21:22:10.944] \
      fe_main be_servers/s1 1/0/42404 2895 cD 1/1/0/0/0 0/0
```

The information included is detailed in the TCP log format section of the reference guide.

The `tcplog` log format is equivalent to the following `log-format` definition, if you were to set it yourself:

```
log-format "%ci:%cp [%t] %ft %b/%s %Tw/%Tc/%Tt %B %ts %ac/%fc/%bc/%sc/%rc %sq/%bq"
```

HTTP log format

The `option httplog` directive is suitable for `frontend` sections that specify `mode http`. This option provides the same information as the `tcplog` option, but with additional HTTP details.

```
defaults
  mode http
  option httplog
```

After reloading your configuration, you'll see the following log format:

output

```
>>> Jan 27 21:52:56 localhost hapee-lb[3098]: 192.168.50.1:61818 [27/Jan/2021:21:52:56.086] \
      fe_main be_servers/s1 0/0/1/1/2 200 517 - - ---- 1/1/0/0/0 0/0 {1wt.eu} {} "GET / HTTP/1.1"
```

The information included is detailed in the HTTP log format section of the reference guide.

The `httplog` log format is equivalent to the following `log-format` definition, if you were to set it yourself:

```
log-format "%ci:%cp [%tr] %ft %b/%s %TR/%Tw/%Tc/%Tr/%Ta %ST %B %CC %CS %tsc %ac/%fc/%bc/%sc/%rc %sq/%bq %hr %hs %{+Q}r"
```

CLF log format for TCP

> **This section applies to:**
>
> - HAProxy Enterprise 3.1 and newer

If you intend to pass the log to a log analyzer that supports the CLF format, you can specify the optional `clf` argument:

```
defaults
  mode tcp
  option tcplog clf
```

The CLF format provides the same details as the `tcplog` format, but in the order required for CLF parsing.

For example, `option tcplog clf` is equivalent to the following `log-format` definition:

```
log-format "%{+Q}o %{-Q}ci - - [%trg] %r %ST %B \"\" \"\" %cp %ms %ft %b %s %TR %Tw %Tc %Tr %Ta %tsc %ac %fc  %bc %sc
%rc %sq %bq %CC %CS %hrl %hsl"
```

The CLF format is intended for HTTP data and includes fields not found in TCP data. These fields are set to static values in the log. The HTTP request will show as TCP and the response code will show as `000`.

By default, the CLF format reports timestamps with whole-second granularity. To report timestamps with millisecond granularity instead, use the `option tcplog` directive (without appending `clf`), or use a custom log format that reports `%tr`.

CLF log format for HTTP

If you intend to pass the log to a log analyzer that supports the CLF format, you can specify the optional `clf` argument:

```
defaults
  mode http
  option httplog clf
```

The CLF format provides the same details as the `httplog` format, but in the order required for CLF parsing.

For example, `option httplog clf` is equivalent to the following `log-format` definition:

```
log-format "%{+Q}o %{-Q}ci - - [%trg] %r %ST %B \"\" \"\" %cp %ms %ft %b %s %TR %Tw %Tc %Tr %Ta %tsc %ac %fc  %bc %sc
%rc %sq %bq %CC %CS %hrl %hsl"
```

By default, the CLF format reports timestamps with whole-second granularity. To report timestamps with millisecond granularity instead, use the `option httplog` directive (without appending `clf`), or use a custom log format that reports `%tr`.

HTTPS log format

> **This section applies to:**
>
> - HAProxy Enterprise 2.5r1 and newer

The `option httpslog` directive is suitable for HTTP over TLS connections. This option provides the same information as the `option httplog` directive, but with additional TLS details.

```
defaults
  mode http
  option httpslog
```

After reloading your configuration, you'll see the following log format:

output

```
>>> Feb  6 12:14:14 localhost hapee-lb[3098]: 192.168.50.1:61818 [27/Jan/2021:21:52:56.086] \
      fe_main be_servers/s1 0/0/1/1/2 200 517 - - ---- 1/1/0/0/0 0/0 {1wt.eu} {} \
      "GET / HTTP/1.1" 0/0/0/0/0 1wt.eu/TLSv1.3/TLS_AES_256_GCM_SHA384
```

The information included is detailed in the HTTPS log format section of the reference guide.

The `httpslog` log format is equivalent to the following `log-format` definition:

```
log-format "%ci:%cp [%tr] %ft %b/%s %TR/%Tw/%Tc/%Tr/%Ta %ST %B %CC %CS %tsc %ac/%fc/%bc/%sc/%rc %sq/%bq %hr %hs %{+Q}r %
[fc_err]/%[ssl_fc_err,hex]/%[ssl_c_err]/%[ssl_c_ca_err]/%[ssl_fc_is_resumed] %[ssl_fc_sni]/%sslv/%sslc"
```

Custom log format

You can define your own log format and record a custom set of information about connections or HTTP requests. Your log format can include variables and values from fetch methods.

Use the `log-format` directive to create a new log format in your `defaults` or `frontend` section. It specifies a string that contains variables referring to the fields that you'd like to capture.

Often, you will want to keep the existing information collected by the premade log format rules and append new information to them. From version 2.7r1 onward, to make that easier, you can use one of the following environment variables:

- `HAPROXY_TCP_LOG_FMT`
- `HAPROXY_HTTP_LOG_FMT`
- `HAPROXY_HTTPS_LOG_FMT`

In versions prior to 2.7r1, define the environment variables yourself:

```
global
  setenv HAPROXY_TCP_LOG_FMT "%ci:%cp [%t] %ft %b/%s %Tw/%Tc/%Tt %B %ts %ac/%fc/%bc/%sc/%rc %sq/%bq"
  setenv HAPROXY_HTTP_LOG_FMT "%ci:%cp [%tr] %ft %b/%s %TR/%Tw/%Tc/%Tr/%Ta %ST %B %CC %CS %tsc %ac/%fc/%bc/%sc/%rc %sq/%bq %hr %hs %{+Q}r"
  setenv HAPROXY_HTTPS_LOG_FMT "%ci:%cp [%tr] %ft %b/%s %TR/%Tw/%Tc/%Tr/%Ta %ST %B %CC %CS %tsc %ac/%fc/%bc/%sc/%rc %sq/%bq %hr %hs %{+Q}r %[fc_err]/%[ssl_fc_err,hex]/%[ssl_c_err]/%[ssl_c_ca_err]/%[ssl_fc_is_resumed] %[ssl_fc_sni]/%sslv/%sslc"
```

Reference the environment variable in your `log-format` line. Below, we start with the HTTP log format and then append a variable named `txn.myvariable` to the end of the original HTTP log format:

```
defaults
  log-format "$HAPROXY_HTTP_LOG_FMT %[var(txn.myvariable)]"
```

A log format variable is a string prefixed by the character `%`. You can use them to create custom log formats. A common way to use this is to record a variable in the logs by using the `var` fetch method as shown below:

```
frontend fe_main
  bind :80

  # store the 'path' in a variable named 'txn.mypath'
  http-request set-var(txn.mypath) path

  # logs will show the path value
  log-format "$HAPROXY_HTTP_LOG_FMT %[var(txn.mypath)]"
```

Avoid spaces in the variable name. A space is considered a separator. If a variable is between square brackets (`[` ... `]`), then it can be an expression, such as to call a fetch method to get a value. Below, we get the value from the `res.cache_hit` fetch method, which indicates whether the load balancer's cache was hit:

```
frontend fe_main
  log-format "$HAPROXY_HTTP_LOG_FMT %[res.cache_hit]"
```

HTTP response variables, those with a `res.` prefix, are easy because they're certain to be available. However, variables set during the request phase are not available to the `log-format` directive, which executes during the response phase. To use them, you must store them in a transaction-level variable, as shown below:

```
frontend fe_main
  http-request set-var(txn.host) req.hdr(Host)
  log-format "$HAPROXY_HTTP_LOG_FMT %[var(txn.host)]"
```

You can prefix a variable with a plus or minus sign and a flag. A plus adds the flag, while a minus removes it. There are three flags available:

- `Q`: adds double quotes around the value
- `X`: represents the value as hexadecimal
- `E`: escapes characters with a backslash, such as double quotes and backslashes

In the example below, we add double quotes around the `Host` header value:

```
frontend fe_main
  http-request set-var(txn.host) req.hdr(Host)
  log-format "$HAPROXY_HTTP_LOG_FMT %{+Q}[var(txn.host)]"
```

Use the special variable `%o` to propagate its flags to all following variables in the same format string.

To emit a verbatim `%`, it must be preceded by another `%` to result in `%%`. HAProxy Enterprise automatically merges consecutive separators.

Log to JSON

> **This section applies to:**
>
> - HAProxy Enterprise 3.0r1 and newer

To define a custom log format that returns a JSON-formatted string:

- Prefix your `log-format` string with `%{+json}o`.
- Prefix each field with a key surrounded by parentheses.

Example:

```
frontend mysite
  bind :80
  log-format "%{+json}o %(client_ip)ci %(client_port)cp %(request_date)tr %(frontend_name)ft %(backend_name)b %
(server_name)s %(time_to_receive)TR %(time_waiting)Tw %(time_to_connect)Tc %(time_server_response)Tr %(time_active)Ta %
(status_code)ST %(bytes_read)B %(request_cookies)CC %(response_cookies)CS %(termination_state)tsc %
(process_active_connections)ac %(frontend_active_connections)fc %(backend_active_connections)bc %
(server_active_connections)sc %(retries)rc %(server_queue)sq %(backend_queue)bq %(request_headers)hr %
(response_headers)hs %(request_line)r"
  use_backend webservers
```

Entries in the log will look like this:

```
{"client_ip": "172.21.0.1", "client_port": 57526, "request_date": "05/Jun/
2024:20:59:57.698", "frontend_name": "mysite", "backend_name": "webservers", "server_name": "web1", "time_to_receive": 0, "
HTTP/1.1"}
```

Log to CBOR

> **This section applies to:**
>
> - HAProxy Enterprise 3.0r1 and newer

To define a custom log format that returns a CBOR-formatted string:

- Prefix your `log-format` string with `%{+cbor}o`.
- Prefix each field with a key surrounded by parentheses.

Example:

```
frontend mysite
  bind :80
  log-format "%{+cbor}o %(client_ip)ci %(client_port)cp %(request_date)tr %(frontend_name)ft %(backend_name)b %
(server_name)s %(time_to_receive)TR %(time_waiting)Tw %(time_to_connect)Tc %(time_server_response)Tr %(time_active)Ta %
(status_code)ST %(bytes_read)B %(request_cookies)CC %(response_cookies)CS %(termination_state)tsc %
(process_active_connections)ac %(frontend_active_connections)fc %(backend_active_connections)bc %
(server_active_connections)sc %(retries)rc %(server_queue)sq %(backend_queue)bq %(request_headers)hr %
(response_headers)hs %(request_line)r"
  use_backend webservers
```

Entries in the log will look like this (line breaks added for clarity):

```
BF69636C69656E745F69706A3137322E32312E302E316B636C69656E745F706F727419E030
6C726571756573745F64617465781830352F4A756E2F323032343A32313A30323A33392E33
34346D66726F6E74656E645F6E616D657F666D7973697465FF6C6261636B656E645F6E616D
656A776562736572766572736B7365727665725F6E616D6564776562316F74696D655F746F
5F72656365697665006C74696D655F77616974696E67006F74696D655F746F5F636F6E6E65
6374007474696D655F7365727665725F726573706F6E7365006B74696D655F616374697665
006B7374617475735F636F646518C86A62797465735F726561641903E16F72657175657374
5F636F6F6B696573F670726573706F6E73655F636F6F6B696573F6717465726D696E617469
6F6E5F7374617465642D2D2D2D781A70726F636573735F6163746976655F636F6E6E656374
696F6E7301781B66726F6E74656E645F6163746976655F636F6E6E656374696F6E7301781A
6261636B656E645F6163746976655F636F6E6E656374696F6E730078197365727665725F61
63746976655F636F6E6E656374696F6E730067726574726965730 6C7365727665725F7175
657565006D6261636B656E645F7175657565006F726571756573745F68656164657273F670
726573706F6E73655F68656164657273F66C726571756573745F6C696E657
```

Log with Docker

When HAProxy Enterprise runs as a Docker container, you can collect logs in the following ways:

- Forward logs to standard out, allowing you to capture them using a log aggregation tool.
- Forward logs to a remote Syslog server, which can run in a separate container.

Log to standard out

This method publishes logs to standard out, allowing you to offload the collection of logs to an external tool, such as logspout.

1. Place a `log` directive into the `global` section of your configuration file:
   - use `stdout` as the address
   - use `format raw` for a shortened log format compatible with a variety of tools
   - use `local0` as the facility code

   ```
   global
     log stdout format raw local0
   ```

2. Add a `log global` directive to a `defaults` section to enable the global logging rule in all subsequent `listen`, `frontend`, and `backend` sections:

```
defaults
  log global
```

3. View logs by querying the HAProxy Enterprise container using the `docker logs` command:

```
sudo docker logs hapee
```

Log to a syslog container

This method allows you to forward logs to a container running a Syslog server, such as Rsyslog.

1. Place a `log` directive into the `global` section of your configuration file:

   - use the IP address or name of your Syslog container, with an optional port number
   - use `local0` as the facility code

```
global
  log rsyslog:514 local0
```

2. Add a `log global` directive to a `defaults` section to enable the global logging rule in all subsequent `listen`, `frontend`, and `backend` sections:

```
defaults
  log global
```

   The full configuration might look like this:

```
global
  log rsyslog:514 local0

defaults
  log global

frontend fe_main
  bind :80
  default_backend be_main

backend be_main
  server web1 web1:8080 check
```

3. Launch containers:

   - Create the Docker network:

```
sudo docker network create -d bridge my-network
```

   - Run the web server container:

```
sudo docker run -d --network my-network --name web1 \
  --restart unless-stopped jmalloc/echo-server
```

   - Run the Rsyslog container:

```
sudo docker run -d --network my-network --name rsyslog \
  --restart unless-stopped jumanjiman/rsyslog
```

   - Run the HAProxy Enterprise container:

```
sudo docker run -d --network my-network --name hapee \
  -p 80:80 -p 443:443 -p 5555:5555 --restart unless-stopped \
  -v (pwd):/etc/hapee-3.1 \
  hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
```

4. View logs by querying the rsyslog container using the `docker logs` command:

```
sudo docker logs rsyslog
```

See also

- For complete information about logging configuration, see Logging reference.

Manage the HAProxy Enterprise service

This page applies to:

- HAProxy Enterprise - all versions

This section describes how to manage the load balancer service on Linux.

Get the HAProxy Enterprise version

To see the installed version of the software:

```
/opt/hapee-3.1/sbin/hapee-lb version
```

Start and stop the service

HAProxy Enterprise runs as a service, which you can start or stop by calling `systemctl`:

Start the service:

```
sudo systemctl start hapee-3.1-lb
```

Stop the service:

```
sudo systemctl stop hapee-3.1-lb
```

Use the `systemctl status` command to check whether the service is running:

```
sudo systemctl status hapee-3.1-lb
```

output

```
hapee-3.1-lb.service - HAPEE Load Balancer
  Loaded: loaded (/lib/systemd/system/hapee-3.1-lb.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2020-12-02 15:56:43 UTC; 1min 35s ago
  Main PID: 12977 (hapee-lb)
  Tasks: 3 (limit: 1152)
  CGroup: /system.slice/hapee-3.1-lb.service
          ├─12977 /opt/hapee-3.1/sbin/hapee-lb -Ws -f /etc/hapee-3.1/hapee-lb.cfg -p /ruhapee-3.1-lb.pid
          └─12992 /opt/hapee-3.1/sbin/hapee-lb -Ws -f /etc/hapee-3.1/hapee-lb.cfg -p /run/ hapee-3.1-lb.pid

Dec 02 15:56:43 lb1 systemd[1]: Starting HAPEE Load Balancer...
Dec 02 15:56:43 lb1 hapee-lb[12977]: [NOTICE] 336/155643 (12977) : New worker #1 (12992) forked
Dec 02 15:56:43 lb1 systemd[1]: Started HAPEE Load Balancer.
```

Reload the configuration

To reload the configuration without restarting the process, call `systemctl reload`:

```
sudo systemctl reload hapee-3.1-lb
```

This sends the `SIGUSR2` signal, which causes the load balancer's main process to re-execute itself. It reloads the configuration file and sends the `SIGUSR1` signal to workers to perform a soft stop and create new workers. No connections will be dropped.

In versions prior to 2.6r1, you must add the `expose-fd listeners` parameter on the `stats socket` line in the `global` section of your configuration for this to work. Versions 2.6r1 and later do not require this.

```
stats socket  /var/run/hapee-3.1/hapee-lb.sock user hapee-lb group hapee mode 660 level admin expose-fd listeners
```

More tuning options:

- Add hard-stop-after to your configuration to limit how long to wait for workers to finish active connections before forcing them to stop. It sets a maximum time to wait. You could use this to prevent long-running sessions from blocking the reload.
- Add grace to your configuration to pause for a period of time before initiating the soft stop of workers. During this delay, you could send a notification of the pending reload.

Validate the configuration file

Use the `hapee-lb` program's `-c` flag to validate the configuration file.

Depending on your version, the command's output indicates that the file is valid in these ways:

- In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
- In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

> **Multiple configuration files**
>
> If you have multiple configuration files in your application, be sure to check them all in the correct order.

Example:

```
sudo /opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg -V
```

Alerts and warnings that appear in the console output explain the errors encountered. An alert is a fatal condition and prohibits the service from starting. A warning indicates a condition that is not fatal now but may become fatal in a future release. Address all the errors until you receive the successful output: `Configuration file is valid`.

Manage SSL certificates

This page applies to:

- HAProxy Enterprise - all versions

There are two ways to manage certificates on a load balancer.

- Runtime API. The Runtime API provides commands that you can issue directly to the running load balancer. Changes take effect without requiring reload or restart, but if you want changes to persist the next time the load balancer is reloaded or restarted, you have to change files on the load balancer node.
- Files residing on the load balancer node. These include certificate files, CRT list files, and the load balancer configuration file. This method of managing certificates requires that you reload or restart the load balancer. Changes are persistent. If your application uses a large number of certificates, a restart or reload can have a significant impact on memory usage.

In practice, you may choose to use both methods: make changes using the Runtime API, and then change the files on the load balancer node to make the changes persistent.

Update an SSL certificate using the Runtime API

You can update an SSL certificate that was loaded into memory at startup. Use the Runtime API command set ssl cert. The workflow to update a certificate is:

1. Use `set ssl cert` to start a transaction that replaces the application's certificate with one that you have on your local workstation.

   In this example, we specify the `crt` argument on the `bind` line to indicate the location of our TLS certificates. In that directory, we've stored `site.pem`.

   ```
   global
     stats socket :9999 level admin expose-fd listeners

   frontend fe_main
     mode http
     bind :80
     bind :443 ssl crt /etc/hapee-3.1/certs/
   ```

   We update `site.pem` with the new certificate `./new_certificate.pem` from the local workstation:

```
echo -e "set ssl cert /etc/hapee-3.1/certs/site.pem <<\n$(cat ./new_certificate.pem)\n" | \
socat tcp-connect:172.25.0.10:9999 -
```

2. Commit the transaction using the Runtime API command commit ssl cert.

```
echo "commit ssl cert /etc/hapee-3.1/certs/site.pem" | \
socat tcp-connect:172.25.0.10:9999 -
```

When you use the Runtime API, your changes take effect in the memory of the running load balancer, but are not stored on disk. They will therefore be lost when the load balancer stops. To make the changes persistent, modify certificate files on the load balancer node.

Add an SSL certificate to a CRT list using the Runtime API

You can add an SSL certificate to a CRT list using the Runtime API command add ssl crt-list. A CRT list is a text file listing certificates, specified in the load balancer configuration with the `bind` directive's `crt-list` argument.

> **Important**
>
> Adding a new certificate to a CRT list does not add the certificate itself. To add the certificate, call new ssl crt-list, set ssl crt-list, and commit ssl crt-list. Then call add ssl crt-list.

The workflow to add a certificate to a CRT list is:

1. Use `add ssl crt-list` to upload the local certificate file into a CRT list in memory.

   In this example, the CRT list, `/etc/hapee-3.1/certificate-list.txt`, is specified in the `bind` directive's `crt-list` argument.

   ```
   frontend fe_main
     mode http
     bind :80
     bind :443 ssl crt-list /etc/hapee-3.1/certificate-list.txt
   ```

   We will add a certificate residing on the local workstation, `new_certificate.pem`, to the load balancer CRT list. The command also sets ALPN attributes and the SNI value for the certificate

   ```
   echo -e "add ssl crt-list /etc/hapee-3.1/certificate-list.txt <<\n/etc/hapee-3.1/certs/new_certificate.pem [alpn h2]
   mysite.local\n" | \
   sudo socat stdio tcp4-connect:127.0.0.1:9999
   ```

output

```
Inserting certificate '/etc/hapee-3.1/certs/new_certificate.pem' in crt-list '/etc/hapee-3.1/certificate-list.txt'.
Success!
```

When you use the Runtime API, your changes take effect in the memory of the running load balancer, but are not stored on disk. They will therefore be lost when the load balancer stops. To make the changes persistent, modify certificate files on the load balancer node.

Modify certificate files on the load balancer node

To make certificate changes persistent, modify certificate files on the load balancer node.

1. Upload the new certificate to the load balancer node.
2. On the load balancer node, confirm the location of existing certificates. Suggested places to inspect:
   - Load balancer configuration file: `/etc/hapee-3.1/hapee-lb.cfg`
   - CRT lists, for example: `/etc/hapee-3.1/certificate-list.txt`
3. Move the new certificate to the certificate directory and give it the desired name.

   In this example, the application's TLS certificate file, `site.pem`, is located in directory `/certs`, which is specified by the `bind` directive's `crt` argument:

   ```
   global
     stats socket :9999 level admin expose-fd listeners

   frontend fe_main
     mode http
     bind :80
     bind :443 ssl crt /certs/
   ```

   Move the new certificate to the certificate directory:

   ```
   sudo mv /home/user/example.pem /certs/site.pem
   ```

4. If you're adding a certificate with a new name, edit the load balancer configuration file or CRT list file and make modifications to specify the new certificate.
5. Reload the load balancer to ensure the old certificate is no longer loaded in memory.

```
sudo systemctl reload hapee-3.1-lb
```

See also

- To add an entry to an SSL CRT list without interrupting the load balancer, see add ssl crt-list.
- To commit a transaction of changes to SSL certificates without interrupting the load balancer, see commit ssl cert.
- To add a certificate to a transaction of SSL certificate changes to be applied to a running HAProxy process, see set ssl cert.

Troubleshooting

Common questions

This page applies to:

- HAProxy Enterprise - all versions

This guide isn't intended to solve problems directly, but rather to lead you from a general problem to a specific one to be solved. This isn't to replace the ability to ask our support team for help directly, just to give a direction to the questions, especially if you aren't fully familiar with HAProxy Enterprise.

You receive a connection timeout response

Possible causes include:

- HAProxy Enterprise has reached its global `maxconn` value of connections.

  Check the HAProxy Enterprise Stats page or Real Time Dashboard and see if the frontend section's `cur` column, which indicates the number of current connections, is equal to the `max` column (or the same as the global settings in the top left of the page). This would indicate that the load balancer or its frontend are at the maximum connection limit (`maxconn` in the configuration) and that number needs to be raised (or the reason it is at the limit otherwise investigated).

- HAProxy Enterprise can't be reached on the network

  If the logs don't show anything, try using `tcpdump` on the load balancer server to see if it is receiving any SYN packets at all. Make this filter as specific as needed to keep it quiet enough to read. For example, `host 192.168.122.14 and port 443` to restrict to a specific inbound IP address.

  ```
  tcpdump -vv -i any "port 80"
  ```

- Other causes can be more complicated to troubleshoot, but may be discovered by checking the access logs in `/var/log/hapee-3.1/`.

You receive a connection refused response

Possible causes include:

- HAProxy Enterprise is not running. Check the service with `sudo systemctl status hapee-3.1-lb`.
- The connection reached the wrong server. Check whether traffic was received on another server.

You receive an empty response

- Check the Stats page or Real Time Dashboard to see if the backend application may be down.
- Check the HAProxy Enterprise logs to see the termination state code, which shows the reason that the connection was aborted.

You receive a 503 Service Unavailable response

- Check the Stats page or Real Time Dashboard to see if the backend application may be marked as down.

  On the Stats page, check for any backends with all servers colored red. If you see any, you can mouse over the `LastChk` column value with its dotted underline to get a more specific reason for the failure. Alternatively, you can grep for `is DOWN` in the access or admin logs to find a message with the same information. You can also grep for `503`.

- If the logs show the backend name being the same as the frontend name, check if the frontend has a `default_backend` line. If it doesn't, it's possible that none of the `use_backend` lines matched the request. Requests that fall out of a frontend without matching a `use_backend` or `default_backend` rule return a 503 response.

  Once you've found the line in the logs, look at the termination state code to see why the request failed.

You receive a 504 Gateway Timeout response

- Check the access logs to see why HAProxy Enterprise timed out while waiting for the server to respond. If `curl` works fine but problems still exist, the next step is to find out what the difference is. For example, SSL cipher/protocol mismatches.

HAProxy Enterprise fails to restart or reload

- The load balancer configuration may have a syntax error. Check the status of the service:

```
systemctl status hapee-3.1-lb
```

Or display the recent log entries:

```
tail -n50 /var/log/messages
```

Look for lines that begin with `[ALERT]`, such as:

```
Mar 31 13:04:09 rhel8vm hapee-lb[3055]: [ALERT] 090/130409 (3055) : parsing [/etc/hapee-3.1/hapee-lb.cfg:123] :
unknown keyword 'hxtp-request' in 'frontend' section
```

There may be multiple alert lines, but the first one is the most urgent.

- If the error mentions not being able to bind to a socket and this is a passive node in a VRRP cluster and HAProxy Enterprise is configured to bind to a specific IP address, check that the sysctl option `net.ipv4.ip_nonlocal_bind` is set to 1. Usually, you can do this by uncommenting the line in the sysctl file HAProxy Enterprise ships with in `/etc/sysctl.d/30-hapee-3.1.conf`.

Contact Us

To get help with your HAProxy Enterprise load balancer:

1. Log into the Customer Portal.
2. Create a new ticket under the **Support** tab.

Not an HAProxy Enterprise customer? Talk to us.

Decrypt TLS traffic

This page applies to:

- HAProxy Enterprise 3.0r1 and newer

When diagnosing network issues, you may need to analyze TLS-encrypted traffic to see the underlying application-layer protocol messages. You can use `tcpdump` to capture packets and save them to a `.pcap`, or packet capture file. You can then import such a file to Wireshark for analysis, but you must provide additional information to Wireshark so that it can decipher the traffic. You can enable the logging of TLS keys in HAProxy Enterprise, which you can then import into Wireshark. Wireshark will use these secrets to decipher the encrypted packets in your `.pcap` file. As of version 3.0, you can produce a keylog file for both traffic between clients and the load balancer and traffic between the load balancer and backend servers.

Use for troubleshooting only

The following procedures will have you enable logging for TLS keys. There are both security and performance implications to consider when enabling logging for TLS keys. When you enable logging for keys, TLS secrets are logged in plaintext, which depending on your system may potentially be unsecure. Also, the load balancer will consume more memory per SSL session when this logging is enabled. Enable this behavior only while troubleshooting and be sure to secure your load balancer access logs.

Decrypt traffic between the load balancer and clients

To analyze TLS traffic between the load balancer and clients:

1. In your load balancer configuration, set `tune.ssl.keylog` to `on` in the `global` section. This activates the retrieval of the TLS keys you will use for decryption in Wireshark.

2. Force the load balancers and clients to use TLS 1.3 by adding the `ssl-min-ver` argument to your TLS `bind` line. TLS 1.3 is required for logging the TLS keys and for allowing you to decrypt the traffic in Wireshark:

```
frontend  fe_main
  bind *:443 ssl crt /etc/hapee-3.1/certs/cert.pem ssl-min-ver TLSv1.3
```

> **Tip**
>
> You can also set the `ssl-min-ver` globally using the option `ssl-default-bind-options`. For example:
>
> ```
> global
>   ssl-default-bind-options ssl-min-ver TLSv1.3
> ```

3. Define a custom log format in your frontend that writes TLS session secrets to the access log. The log format uses sample fetches to retrieve the keys. We are using the frontend fetches here, as indicated by `fc` in the fetch names:

```
frontend fe_main
  log-format "$HAPROXY_HTTP_LOG_FMT CLIENT_EARLY_TRAFFIC_SECRET %[ssl_fc_client_random,hex] %
[ssl_fc_client_early_traffic_secret]\nCLIENT_HANDSHAKE_TRAFFIC_SECRET %[ssl_fc_client_random,hex] %
[ssl_fc_client_handshake_traffic_secret]\nSERVER_HANDSHAKE_TRAFFIC_SECRET %[ssl_fc_client_random,hex] %
[ssl_fc_server_handshake_traffic_secret]\nCLIENT_TRAFFIC_SECRET_0 %[ssl_fc_client_random,hex] %
[ssl_fc_client_traffic_secret_0]\nSERVER_TRAFFIC_SECRET_0 %[ssl_fc_client_random,hex] %
[ssl_fc_server_traffic_secret_0]\nEXPORTER_SECRET %[ssl_fc_client_random,hex] %[ssl_fc_exporter_secret]
\nEARLY_EXPORTER_SECRET %[ssl_fc_client_random,hex] %[ssl_fc_early_exporter_secret]"
```

4. Reload the load balancer to apply the configuration changes:

```
sudo systemctl reload hapee-3.1-lb
```

5. Initiate a packet capture between the load balancer and clients using `tcpdump` to capture the traffic. For example, to save packets to a `.pcap` file on the load balancer instance named `mycap.pcap`, you could use the following command. Note that you may need to change the `port` and network interface (`-i`) depending on your settings. The `port` is the port on which clients make TLS connection to your load balancer.

```
sudo tcpdump -s 0 port 443 -i eth0 -w mycap.pcap
```

> **Tip**
>
> You can list your network interfaces using a command such as `ifconfig -a` or `ip link show`, depending on your OS.

6. While your capture is running, and after a client connects to the load balancer, the access log will contain lines like this that are the keys for the TLS session:

```
CLIENT_EARLY_TRAFFIC_SECRET 0007A9877A21DAAA12156C5230F69D219A95DB00F0595F54E7C87C27AE91E1BA
CLIENT_HANDSHAKE_TRAFFIC_SECRET 0007A9877A21DAAA12156C5230F69D219A95DB00F0595F54E7C87C27AE91E1BA
fa3eb968fcb530d416e33cb25e377038ffbf7b4fb943fcf28b4b283e780e02cdc4171a6c7285f972a26828c6747460a3
SERVER_HANDSHAKE_TRAFFIC_SECRET 0007A9877A21DAAA12156C5230F69D219A95DB00F0595F54E7C87C27AE91E1BA
cc6bbfa6e770685b61fbe86b51863678fbbfc3688d55bf4aaff351553bbcb4788460f8e85048cda257d4e6df547fe6d7
CLIENT_TRAFFIC_SECRET_0 0007A9877A21DAAA12156C5230F69D219A95DB00F0595F54E7C87C27AE91E1BA
3cb0bb3bc836ef9b3b98984bb7f76a1b0d36b5d28acc94b91c8bde7052b17e112afd83078f39edb3eefc8cdcaac06f21
SERVER_TRAFFIC_SECRET_0 0007A9877A21DAAA12156C5230F69D219A95DB00F0595F54E7C87C27AE91E1BA
c11e1a31231ff2561c4c37d0d82132c263a070fb13c897008a4539cf38f1e3ff27a16d9b73efe2d1dc0c9e5df3fed84e
EXPORTER_SECRET 0007A9877A21DAAA12156C5230F69D219A95DB00F0595F54E7C87C27AE91E1BA 393e485a78edb1c09be95a335d67fa6b82
```

Note that each TLS session (connection) will generate its own keys.

7. Save the lines from the access logs containing the secrets to a text file. Import the file into Wireshark via **Preferences > Protocols > TLS > (Pre)-Master-Secret log filename**.

8. Open the `.pcap` file with your captured traffic in Wireshark to see the deciphered traffic.

Once you have finished troubleshooting:

1. In your load balancer configuration, set `tune.ssl.keylog` to `off` in the `global` section, or delete the line entirely. This disables the logging of TLS keys.

2. Remove the log format line that retrieves the TLS keys from your configuration.

3. Reload the load balancer to apply the configuration changes:

```
sudo systemctl reload hapee-3.1-lb
```

Decrypt traffic between the load balancer and backend servers

To analyze TLS traffic between the load balancer and backend servers:

1. In your load balancer configuration, set `tune.ssl.keylog` to `on` in the `global` section. This activates the retrieval of the TLS keys you will use for decryption in Wireshark.

2. Force the load balancer and the backend servers to use TLS 1.3 by adding the `ssl-min-ver` argument to the servers. TLS 1.3 is required for logging the TLS keys and for allowing you to decrypt the traffic in Wireshark:

```
backend servers
  server s1 192.168.56.50:443 ssl verify required ca-file /etc/haproxy/certs/ca.crt ssl-min-ver TLSv1.3
```

Note that here we have also added `verify required` to our server line and have provided the CA certificate using `ca-file`. This enforces a check where the load balancer will verify the server certificate. For more information see verify reference.

> **Tip**
>
> You can also set the `ssl-min-ver` globally using the global option `ssl-default-bind-options`. For example:
>
> ```
> global
>   ssl-default-bind-options ssl-min-ver TLSv1.3
> ```

3. Define a custom log format in your frontend that writes TLS session secrets to the access log. The log format uses sample fetches to retrieve the keys. We are using the backend fetches here, as indicated by `bc` in the fetch names:

```
frontend fe_main
  log-format "$HAPROXY_HTTP_LOG_FMT CLIENT_EARLY_TRAFFIC_SECRET %[ssl_bc_client_random,hex] %
[ssl_bc_client_early_traffic_secret]\nCLIENT_HANDSHAKE_TRAFFIC_SECRET %[ssl_bc_client_random,hex] %
[ssl_bc_client_handshake_traffic_secret]\nSERVER_HANDSHAKE_TRAFFIC_SECRET %[ssl_bc_client_random,hex] %
[ssl_bc_server_handshake_traffic_secret]\nCLIENT_TRAFFIC_SECRET_0 %[ssl_bc_client_random,hex] %
[ssl_bc_client_traffic_secret_0]\nSERVER_TRAFFIC_SECRET_0 %[ssl_bc_client_random,hex] %
[ssl_bc_server_traffic_secret_0]\nEXPORTER_SECRET %[ssl_bc_client_random,hex] %[ssl_bc_exporter_secret]
\nEARLY_EXPORTER_SECRET %[ssl_bc_client_random,hex] %[ssl_bc_early_exporter_secret]"
```

4. Reload the load balancer to apply the configuration changes:

```
sudo systemctl reload hapee-3.1-lb
```

5. Initiate a packet capture between the load balancer and the backend servers using `tcpdump` to capture the traffic. For example, to save packets to a `.pcap` file on the load balancer instance named `mycap.pcap`, you could use the following command. Note that you may need to change the `port` and network interface (`-i`) depending on your settings. The `port` is the port on which your load balancer connects to your backend servers.

```
sudo tcpdump -s 0 port 443 -i eth0 -w mycap.pcap
```

> **Tip**
>
> You can list your network interfaces using a command such as `ifconfig -a` or `ip link show`, depending on your OS.

6. While your capture is running, and after the load balancer connects to a backend server, the access log will contain lines like this that are the keys for the TLS session:

```
CLIENT_EARLY_TRAFFIC_SECRET C030AF8EAEE688F1F3A360E5D53E260DEAB346F93CE594153D95E33E4BFD5F80 -
CLIENT_HANDSHAKE_TRAFFIC_SECRET C030AF8EAEE688F1F3A360E5D53E260DEAB346F93CE594153D95E33E4BFD5F80
15ab9abf57145fe49c73d9a617eca9b918d5c4dd455c4bb923c04a936475241facbac21f66bca7c459f5179f753f4afa
SERVER_HANDSHAKE_TRAFFIC_SECRET C030AF8EAEE688F1F3A360E5D53E260DEAB346F93CE594153D95E33E4BFD5F80
09bded135c6b85959d0c2eaf09d177cc4fb9e2d9777cbda5a234d0894ef84b64bbd346cc331a16111d4273d639090d5b
CLIENT_TRAFFIC_SECRET_0 C030AF8EAEE688F1F3A360E5D53E260DEAB346F93CE594153D95E33E4BFD5F80
155b07c8fcef945cbad456f6b11e216fde42f9ac1cdc8c6eff4bed845caf520a2a490ccba3ae06ffe3d9091904674c41
SERVER_TRAFFIC_SECRET_0 C030AF8EAEE688F1F3A360E5D53E260DEAB346F93CE594153D95E33E4BFD5F80
18ed3dc1188b7ed1085cbdf41b0f0388b80904f6f21b8962f57cdf460d5694f2b2d99f7055ac44f0e6afefc9e790626b
EXPORTER_SECRET C030AF8EAEE688F1F3A360E5D53E260DEAB346F93CE594153D95E33E4BFD5F80
9479651fd91e38d549b284ecae7c6430743ae56cc4e8fb899eaf0a4016891d3991b01691c1c4c787d95a10c
```

   Note that each TLS session (connection) will generate its own keys.

7. Save the lines from the access logs containing the secrets to a text file. Import the file into Wireshark via **Preferences > Protocols > TLS > (Pre)-Master-Secret log filename**.

8. Open the `.pcap` file with your captured traffic in Wireshark to see the deciphered traffic.

Once you have finished troubleshooting:

1. In your load balancer configuration, set `tune.ssl.keylog` to `off` in the `global` section, or delete the line entirely. This disables the logging of TLS keys.

2. Remove the log format line that retrieves the TLS keys from your configuration.

3. Reload the load balancer to apply the configuration changes:

```
sudo systemctl reload hapee-3.1-lb
```

See also

- To enable logging of TLS keys, see the tune.ssl.keylog reference.
- To specify the default SSL bind options, see the ssl-default-bind-options reference.
- To verify the server certificate, see verify reference.
- To specify the PEM file containing CA certificates, see ca-file reference.

Enable core dumps

This page applies to:

- HAProxy Enterprise - all versions

In the rare event that an HAProxy Enterprise process crashes or behaves abnormally, you can capture a core dump (also known as a crash dump) that you can send to the Support team. A core dump is a file that encapsulates the state of an application when it crashes and is useful in diagnosing and fixing potential issues. Core dumps are not enabled by default, so you must configure your OS settings to allow the collection of these files.

Enable core dumps

1. Enable the core dump handler. This sets the core dump handler inside the default HAProxy Enterprise change root environment.

   Find the chroot directory

   The default chroot environment for HAProxy Enterprise is `/var/empty`. You can find this value in the configuration file `/etc/hapee-3.1/hapee-lb.cfg` in the `global` section.

   The default chroot environment is `/var/empty`. We want core dumps to be saved in `/var/empty/tmp`. The kernel setting `kernel.core_pattern` sets this value.

   ```
   sudo sysctl -w fs.suid_dumpable=1
   sudo sysctl -w kernel.core_pattern=/tmp/core.%P.%u.%g.%s.%t
   ```

2. Optional: Persist the configuration so that core dumps are still enabled after reboot. Add the following lines to `/etc/sysctl.d/99-sysctl.conf`. This again sets the directory for saving core dumps to `/tmp` inside of the chroot environment, `/var/empty`.

   99-sysctl.conf

   ```
   # add these lines to the end of the file
   fs.suid_dumpable=1
   kernel.core_pattern=/tmp/core.%P.%u.%g.%s.%t
   ```

3. Create a subdirectory inside of the chroot environment with permissions that allow the `hapee-lb` user to write to it. This subdirectory should be the same as the directory you specified for `kernel.core_pattern` in the previous step. We will create a `/tmp` directory inside of `/var/empty` and set its permissions:

```
sudo mkdir /var/empty/tmp
sudo chmod 0777 /var/empty/tmp
```

4. To set the maximum size of a core dump file, add the `DefaultLimitCORE` setting to the file `/etc/systemd/system.conf`. Below, we set the value to `infinity`.

system.conf

```
# add this line to the end of the file
DefaultLimitCORE=infinity
```

5. Restart the Systemd daemon. Processes running under Systemd will not be affected by this restart.

```
sudo systemctl daemon-reexec
```

6. Edit the HAProxy Enterprise configuration so that it includes `set-dumpable` in the `global` section:

```
global
  set-dumpable
```

7. Restart HAProxy Enterprise.

```
sudo systemctl restart hapee-3.1-lb
```

Retrieve core dumps

After a crash in HAProxy Enterprise, the system will generate a core dump file and place it in one of two locations:

- If the fault occurred in HAProxy Enterprise's master process, the core dump file will be in `/tmp`.
- If it occurred in a worker process, it will be in the location you configured as your `kernel.core_pattern` (probably `/var/empty/tmp`).

In one of those locations will be a file that starts with `core`. This file is the core dump. The core file will look like `core.17442.997.994.6.1689180587.17442`.

The core dump file name has significance. If you configure your `kernel.core_pattern` to name files with the pattern `core.%P.%u.%g.%s.%t`, the resulting file name will include:

Variable	Description
`%P`	Process ID of the dumped process (as it appears in the initial PID namespace).
`%u`	UID of the dumped process.
`%g`	GID of the dumped process.
`%s`	Number of the signal that caused the dump.
`%t`	Unix time of the dump.

The last part of the filename is also the process ID.

Produce a core dump for a running process

It is possible to retrieve a core dump from HAProxy Enterprise without adjusting resource limits, changing kernel settings, or restarting HAProxy Enterprise. This is possible with a utility named `gcore`. Retrieving the core dump in this way may be useful when it is not possible to complete those steps, or in the case for retrieving process state information when a process may be stuck but has not crashed. The downside to this approach is that unlike the previous procedures which enable core dumps for any future crashes, using `gcore` is a manual procedure.

1. To produce a core dump for a running HAProxy Enterprise process, first find the process ID, or PID, using `ps`. The `ps` command will produce two process IDs. The first column of the output shows the user. The second column is the PID.

   In this output, the master process, the process run under `root`, has a process ID of `19973`. The worker process, the process run under user `hapee-lb`, has a process ID of `19975`.

   ```
   ps -ef | grep hapee
   ```

   output

   ```
   root     19973     1  0 17:39 ? 00:00:00 /opt/hapee-3.1/sbin/hapee-lb
     -Ws -f /etc/hapee-3.1/hapee-lb.cfg -p /run/hapee-3.1-lb.pid

   hapee-lb 19975 19973  0 17:39 ? 00:00:00 /opt/hapee-3.1/sbin/hapee-lb
     -Ws -f /etc/hapee-3.1/hapee-lb.cfg -p /run/hapee-3.1-lb.pid
   ```

2. If the `gcore` utility is not installed, you can install it using your package manager. It is packaged with `gdb`.

Apt:

```
sudo apt-get install gdb
```

Yum:

```
sudo yum gdb
```

3. Use the `gcore` command with a process ID to produce a core dump file in your current working directory.

```
sudo gcore 19973
```

output

```
Using host libthread_db library "/lib64/libthread_db.so.1".
0x00007ffb49cfdea3 in epoll_wait () from /lib64/libc.so.6
Saved corefile core.19973
[Inferior 1 (process 19973) detached]
```

Produce a core dump for a stuck process

If the `gcore` utility is not available, `prlimit` can be used to produce a core dump for a running process that is stuck. `prlimit` is used to set resource limits dynamically in the current session for running processes.

> **Warning**
>
> Proceed with this method only as a last resort, and only if the application is totally nonresponsive. The process will be stopped abruptly, and this may result in unexpected behavior (such as unsaved changes).

1. To produce a core file, find the process ID, or PID, using `ps`. The `ps` command will produce two process IDs. The first column of the output shows the user. The second column is the PID.

   In this output, the master process, the process run under `root`, has a process ID of `19973`. The worker process, the process run under user `hapee-lb`, has a process ID of `19975`.

```
ps -ef | grep hapee
```

output

```
root     19973     1  0 17:39 ? 00:00:00 /opt/hapee-3.1/sbin/hapee-lb
  -Ws -f /etc/hapee-3.1/hapee-lb.cfg -p /run/hapee-3.1-lb.pid

hapee-lb 19975 19973  0 17:39 ? 00:00:00 /opt/hapee-3.1/sbin/hapee-lb
  -Ws -f /etc/hapee-3.1/hapee-lb.cfg -p /run/hapee-3.1-lb.pid
```

2. Set the core file size limit to `unlimited` for the process. This example sets the core file size limit for the process with ID `19973` to `unlimited`:

```
sudo prlimit --core=unlimited:unlimited --pid=19973
```

You can check the limits for a process using the `prlimit` command:

```
sudo prlimit --pid=19973
```

output

```
RESOURCE   DESCRIPTION                             SOFT      HARD UNITS
AS         address space limit                unlimited unlimited bytes
CORE       max core file size                 unlimited unlimited blocks
CPU        CPU time                           unlimited unlimited seconds
DATA       max data size                      unlimited unlimited bytes
FSIZE      max file size                      unlimited unlimited blocks
LOCKS      max number of file locks held      unlimited unlimited
MEMLOCK    max locked-in-memory address space     65536     65536 bytes
MSGQUEUE   max bytes in POSIX mqueues            819200    819200 bytes
NICE       max nice prio allowed to raise           0         0
NOFILE     max number of open files             20027     20027
NPROC      max number of processes               7163      7163
RSS        max resident set size              unlimited unlimited pages
RTPRIO     max real-time priority                   0         0
RTTIME     timeout for real-time tasks        unlimited unlimited microsecs
SIGPENDING max number of pending signals          7163      7163
STACK      max stack size                     8388608 unlimited bytes
```

3. Enable the core dump handler. This sets the core dump handler inside the default HAProxy Enterprise change root environment.

> **Tip**
>
> The default chroot environment for HAProxy Enterprise is `/var/empty`. You can find this value in the configuration file `/etc/hapee-3.1/hapee-lb.cfg` in the `global` section.

The default chroot environment is `/var/empty`. We want core dumps to be saved in `/var/empty/tmp`. The kernel setting `kernel.core_pattern` sets this value.

```
sudo sysctl -w fs.suid_dumpable=1
sudo sysctl -w kernel.core_pattern=/tmp/core.%P.%u.%g.%s.%t
```

4. Create a subdirectory inside of the chroot environment with permissions that allow the `hapee-lb` user to write to it. This subdirectory should be the same as the directory you specified for `kernel.core_pattern` in the previous step. This is required to generate core dumps for HAProxy Enterprise's worker processes.

   We will create a `/tmp` directory inside of `/var/empty` and set its permissions:

   ```
   sudo mkdir /var/empty/tmp
   sudo chmod o+w /var/empty/tmp
   ```

5. For the process that is stuck, force a crash. This command will abruptly stop the process with PID `19973`:

   ```
   sudo kill -SIGABRT 19973
   ```

   After forcing HAProxy Enterprise to stop abruptly, you may need to restart the service for it to resume processing.

   ```
   systemctl restart hapee-3.1-lb
   ```

6. After forcing the crash, the system will generate a core dump file and place it in one of two locations:

   - If you issued the `kill` command on HAProxy Enterprise's master process, the core dump file will be in `/tmp`.
   - If you issued the `kill` command on one of HAProxy Enterprise's worker processes, it will be in the location you configured as your `kernel.core_pattern` (probably `/var/empty/tmp`).

Enable core dumps for Docker

You can enable core dumps when running HAProxy Enterprise as a Docker container. To enable core dumps in Docker:

1. Configure the kernel settings on your host instance (the instance running Docker) to specify the location for saving core dumps. This location is communicated to all Docker containers running on the instance.

   This sets the kernel setting for `core_pattern` to specify that core dump files should be saved to `/tmp`. Make sure that the directory you specify for `core_pattern` is a directory that exists.

   ```
   echo '/tmp/core.%P.%u.%g.%s.%t' | sudo tee /proc/sys/kernel/core_pattern
   ```

2. Start the HAProxy Enterprise Docker container with the following arguments. These are similar to the arguments provided for starting the container normally, without enabling core dumps (see: Install HAProxy Enterprise on Docker), with a few additional arguments added to enable core dumps within the container.

   We are providing three additional parameters to the `docker run` command: `init`, `ulimit`, and `mount`.

   ```
   sudo docker run \
     --name hapee-3.1 \
     --init \
     --ulimit core=-1 \
     --mount type=bind,source=/tmp/,target=/tmp/ \
     -d \
     -p 80:80 \
     -p 443:443 \
     -p 5555:5555 \
     -v $(pwd):/etc/hapee-3.1  \
     --restart=unless-stopped \
     hapee-registry.haproxy.com/haproxy-enterprise:3.1r1
   ```

   Be sure to specify the directory containing your configuration files using `-v`.

   - `--init` tells Docker to implement signal handling for the container. This is required to catch an application crash.
   - `--ulimit core=-1` sets the core dump file size limit to `unlimited`.
   - `--mount type=bind,source=/tmp/,target=/tmp/` tells Docker to mount the `/tmp` directory on the host instance into the container.

   The Docker container is read-only, and as such, the core dump files cannot be saved inside of the container. Specifying this `mount` argument guarantees that the core files still exist on your host system after the container is stopped or deleted.

   In the previous step, we set the location for saving core dump files to `/tmp`, so we must provide two additional parameters for `mount`, `source` and `target`, each also set to `/tmp`.

The Docker container inherits the kernel settings from the host instance, so we expect the Docker container to write core dump files to the `/tmp` directory.

3. Core dump files produced by crashes in both HAProxy Enterprise's master process and its worker processes will be placed in `/tmp` on the host instance.

Enable diagnostic archive

This page applies to:

- HAProxy Enterprise 2.9r1 and newer

The diagnostic archive or `hapee-tech-support` tool allows you to collect diagnostic data for your system running HAProxy Enterprise. The `hapee-tech-support` tool creates a `.tar.gz` file containing information about the state of the system that you can provide to the HAProxy Technologies Support team. The `.tar.gz` diagnostic archive file is named with the date and time the file was created. For example:

- `tech-support-20240226-194158.tar.gz`

The diagnostic archive contains the following:

- Information about the system, running processes, and installed packages
- Operating system information, including kernel parameters
- Information about hardware resources such as disk, CPU, and memory
- Networking and firewall information
- HAProxy Enterprise logs and statistics
- HAProxy Enterprise configuration files

Enable diagnostic archive

To produce a diagnostic archive, issue the following command to run the `hapee-tech-support` tool. Note that the tool must be run as `sudo`:

```
sudo /opt/hapee-3.1/bin/hapee-tech-support
```

Example output

```
tech-support temporary folder is /tmp/ts/tech-support-20240226-200057
Collecting system status information
Collecting platform-specific information
Collecting hardware information
Dumping 5 seconds of system activity...
Collecting networking information
Collecting firewalling information
Found HAPEE logs.
Collection last 1000 lines of hapee logs
Collecting Debian based information
Collecting HAPEE or HAProxy config files
tech-support finished

Report saved to /tmp/ts/tech-support-20240226-200057.tar.gz
Please attach this file to e-mail correspondence with HAProxy's support.
Please remove any prior tech-support files that you no longer want.
```

The output will indicate the location and filename of the diagnostic archive. In the example above, it is `/tmp/ts/tech-support-20240226-200057.tar.gz`. This `.tar.gz` file is what you will provide to the HAProxy Technologies Support team. By default, the diagnostic archives will be saved to `/tmp/ts`, though you can also configure a different location.

Options

The following options are available for the `hapee-tech-support` tool:

Option	Description
`--stdout (-)`	Send the output from the operation to stdout. This same output is also sent to a file packaged in the diagnostic archive named `tech-support-<date>-<time>-<pid>.log`.
`--output-dir (-o) OUTPUT_DIR`	Set the output file location. The default is `/tmp/ts`.
`--save-config (-s)`	Save HAProxy Enterprise configuration files.
`--hapee-log-lines (-l)`	Save `n` HAProxy Enterprise log lines. By default, the `hapee-tech-support` tool saves the last 1000 lines. If set to `0`, the `hapee-tech-support` tool will save all log lines, otherwise it will save the last `n` lines.
`--multi-file-out (-m)`	Split each diagnostic item's output into individual `.txt` files. Otherwise, all of the output will be in a single file named `tech-support-<date>-<time>-<pid>.log` inside the diagnostic archive.
`--keep (-k)`	Keep the directory the `hapee-tech-support` tool creates. Usually this is deleted after it has been packaged into the `.tar.gz`. This will reside in the same location as the diagnostic archive file.
`--plugins PLUGINS_DIR`	Set the plugins directory. The default is `/opt/hapee-3.1/bin/plugins`. The plugins each collect information about different parts of the system.
`--help (-h)`	Display the help text.

Enterprise modules

- Bot and crawlers
- Device detection
- Dynamic updates
- Geolocation
- Global Profiling Engine
- GSLB
- Real-time Dashboard
- Response body injection
- Route health injection
- Send metrics
- Single sign-on
- SNMP
- SSL-CRL
- Traffic mirroring
- UDP load balancing
- Web Application Firewall

Bots and crawlers

- Bot management
- Captcha
- Client fingerprinting
- Javascript Challenge
- Verify crawlers

Device detection

- 51Degrees
- DeviceAtlas
- ScientiaMobile

51Degrees

This page applies to:

- HAProxy Enterprise 1.7r2 and newer

The 51Degrees module provides device detection services using the 51Degrees database.

Install the 51Degrees module

1. Go to the 51Degrees Downloads page to get the 51Degrees device detection data file.
   - For HAProxy Enterprise 3.0r1 and newer, use the V4 `.hash` data file.
   - For HAProxy Enterprise 2.9r1 and older, use the V3 `.trie` data file.
2. Copy the data file to your HAProxy Enterprise server (for example `/etc/hapee-3.1/51Degrees-LiteV4.1.hash`).
3. Install the 51Degrees module according to your platform:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-51d
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-51d
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-51d
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-51d
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-51d
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-51d
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-51d
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-51d
```

4. In the `global` section of your configuration, add the following lines. Change the `51degrees-property-name-list` depending on the 51Degrees device properties you want to use. Note that some properties become available depending on your 51Degrees pricing tier.

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-51d.so
  51degrees-data-file /etc/hapee-3.1/51Degrees-LiteV4.1.hash
  51degrees-property-name-list DeviceType IsMobile ScreenPixelsHeight ScreenPixelsWidth
```

5. Reload the configuration to apply the changes.

```
sudo systemctl reload hapee-3.1-lb
```

Use 51Degrees

After installing the module, use the following directives to perform database lookups.

51d.all

Use the fetch method `51d.all` to perform a lookup in the database that returns the values of the specified properties. The function can be passed up to five property names and if a property name can't be found, it returns the value `NoData`.

Syntax

```
51d.all(<prop>[,<prop>*])
```

In this example, we create an HTTP request header that contains device information:

```
frontend www
  bind :80
  mode http
  http-request set-header X-DeviceInfo %[51d.all(DeviceType,IsMobile,ScreenPixelsHeight,ScreenPixelsWidth)]
```

In the next example, we define an ACL named `is_mobile` and then use it when choosing a backend:

```
frontend www
  bind :80
  mode http
  acl is_mobile 51d.all(IsMobile) "True"
  use_backend mobile_site if is_mobile
  default_backend desktop_site
```

51d.single

Use the converter `51d.single` to perform a lookup in the database that returns the values of the specified properties. It takes the `User-Agent` header as an input parameter. The function can be passed up to five property names and if a property name can't be found, it returns the value `NoData`.

Syntax:

```
51d.single(<prop>[,<prop>*])
```

In this example, we create an HTTP request header that contains device information:

```
frontend www
  bind :80
  mode http
  http-request set-header X-DeviceInfo %[req.hdr(user-
agent),51d.single(DeviceType,IsMobile,ScreenPixelsHeight,ScreenPixelsWidth)]
```

Update the database during runtime

Use the 51Degrees Update module to keep the contents of the device detection database current. This allows you to keep multiple load balancers synced with the latest data.

1. Install a web server of your choice and host the database file at a URL the load balancer can access. For example, host the file at `http://192.168.0.1:8000/51Degrees-LiteV4.1.hash`.

2. Install the package `hapee-3.1r1-lb-51d-update`.

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-51d-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-51d-update
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-51d-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-51d-update
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-51d-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-51d-update
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-51d-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-51d-update
```

3. Add the following lines to the `global` section of your configuration:

```
global
  module-load hapee-lb-51d-update.so
  51degrees-update url http://192.168.0.1:8000/51Degrees-LiteV4.1.hash delay 24h log
```

Be sure to specify the port number where your file is hosted, for example `8000` as in the example above.

With this configuration, HAProxy Enterprise downloads the database every 24 hours and prints a message in the logs when it succeeds or if it encounters errors during the update.

Runtime API

It is possible to manage the 51Degrees module via the Runtime API.

51d-update force-update

> **Prerequisites**
>
> This command becomes available after you have installed both the 51Degrees and 51Degrees Update modules.

The 51Degrees update module will update the database based on the interval you specified in the configuration. Use the `51d-update force-update` command to force an immediate update of the 51Degrees database.

In this example, we force an update of the database.

```
echo "51d-update force-update" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
51Degrees: forcing update now
```

51d-update status

> **Prerequisites**
>
> This command becomes available after you have installed both the 51Degrees and 51Degrees Update modules.

You can check the status of the update using the `51d-update status` command. Note that while the update is processing, the status will show the progress:

```
echo "51d-update status" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
51Degrees module status
--------------------------------------------------------------------------------
 initialized:          yes

 Database update
  configuration:      /etc/hapee-2.7/hapee-lb.cfg:43
  url:                http://192.168.64.1/51Degrees-LiteV4.1.hash
  http status count:  0 0 0 0 7 / 0
  period/delay:       1d / 5s 10s 5s
  use cksum/hash/mod: no / no / no
  reload/retry count: 0 0 2 / 7 2
  reload time:        <NEVER> / <NEVER> / 2023-08-08 14:25:37
  currently updating: 51Degrees-LiteV4.1.hash
    status/retry:     0 / 1
    data size:        0 / 0 (0.0%)
    dur/time left:    1m49s / 3s
```

51d-update show

> **Prerequisites**
>
> This command becomes available after you have installed both the 51Degrees and 51Degrees Update modules.

Use the `51d-update show` command to display the configuration of the 51Degrees update module. The output of this command includes information about each configured database file and its update status.

Below, we retrieve the status information for our configured databases:

```
echo "51d-update show" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
# 51d-update configuration
# url: next update
http://192.168.0.1/51Degrees-LiteV4.1.hash: 23h58m
```

Troubleshooting

This section covers troubleshooting steps.

Error reading 51Degrees data file

You get the following error during startup:

```
51Degrees Setup - Error reading 51Degrees data file. The data (file: '/etc/hapee-3.1/51Degrees-EnterpriseV3.4.trie') is an unsupported version. Check you have the latest data and API.
```

The data file you are trying to load is not compatible with the version of the HAProxy Enterprise 51Degrees module. For example, you tried to load a `.trie` data file, but the module expects a `.hash` data file.

- For HAProxy Enterprise 3.0r1 and newer, use the V4 `.hash` data file.
- For HAProxy Enterprise 2.9r1 and older, use the V3 `.trie` data file.

DeviceAtlas

This page applies to:

- HAProxy Enterprise 1.7r2 and newer

The DeviceAtlas module provides device detection services using the DeviceAtlas database.

Install the DeviceAtlas module

1. Go to the DeviceAtlas website and sign up for the DeviceAtlas for Web product. Then sign in to access the API data file. Depending on which version of HAProxy Enterprise you're running, download the compatible data file. You can set this on the DeviceAtlas Data File Options page.

   - For HAProxy Enterprise 3.0r1 and newer, use a 3.x API data file.
   - For HAProxy Enterprise 2.9r1 and older, use a 2.x API data file.

2. Download the API data file as a zip file and extract the zip file to get the `.json` data file. The file has a name like `72277_20241104.json`, but you can rename it to `deviceatlas.json`. Copy the JSON data file to your HAProxy Enterprise server. Example: `/etc/hapee-3.1/deviceatlas.json`.

3. Download the Enterprise API for C library as a zip file. You'll need the 3.x or 2.x library, depending on which version of HAProxy Enterprise you're using.

   To compile the library, follow the guide on the DeviceAtlas website, DeviceAtlas Device API Usage C. This step requires you to transfer the zip file to Linux, FreeBSD, or Mac, since those operating systems have the necessary build tools. For example, on Debian/Ubuntu, run:

   ```
   sudo apt update
   sudo apt install -y --no-install-recommends build-essential cmake unzip

   unzip deviceatlas-enterprise-c-3.2.3.zip
   cd deviceatlas-enterprise-c-3.2.3
   cmake -DNO_CURL=ON .
   sudo make
   ```

4. Copy the compiled C library from `Src/libda.so.3` to your HAProxy Enterprise server.

   - For the 3.x library, copy it as `/lib/x86_64-linux-gnu/libda.so.3` on Ubuntu/Debian or `/lib/lib64/libda.so.3` on RedHat Enterprise Linux.
   - For the 2.x library, copy it as `/lib/x86_64-linux-gnu/libda.so.2` on Ubuntu/Debian or `/lib/lib64/libda.so.2` on RedHat Enterprise Linux.

5. On your HAProxy Enterprise server, install the DeviceAtlas module according to your platform:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-da
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-da
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-da
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-da
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-da
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-da
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-da
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-da
```

6. In the `global` section of your configuration, add the following lines:

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-da.so
  deviceatlas-json-file /etc/hapee-3.1/deviceatlas.json
```

   The `deviceatlas-json-file` directive loads a DeviceAtlas database.

7. Optional: Add the `deviceatlas-cache-size` directive to set the DeviceAtlas property `atlas.config.cache_size`, which sets the number of cache entries. It defaults to 0.

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-da.so
  deviceatlas-json-file /etc/hapee-3.1/deviceatlas.json
  deviceatlas-cache-size 10000
```

8. Optional: Set the `deviceatlas-property-separator` directive to change the separator to use between values returned from the database. By default, it uses a pipe symbol (`|`).

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-da.so
  deviceatlas-json-file /etc/hapee-3.1/deviceatlas.json
  deviceatlas-property-separator :
```

9. Optional: Change the log level, which you can set to a number between 0 and 3, where 3 is the most verbose. The default is 0.

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-da.so
  deviceatlas-json-file /etc/hapee-3.1/deviceatlas.json
  deviceatlas-log-level 3
```

10. Optional: Change the name of the DeviceAtlas Client-side Component cookie, if using client-side properties. It defaults to `DAPROPS`.

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-da.so
  deviceatlas-json-file /etc/hapee-3.1/deviceatlas.json
  deviceatlas-properties-cookie mycookiename
```

11. Reload the configuration to apply the changes.

```
sudo systemctl reload hapee-3.1-lb
```

Use DeviceAtlas

After installing the module, use the following directives to perform database lookups.

da-csv-fetch

Use the fetch method `da-csv-fetch` to perform a lookup in the database that returns the values of the specified properties.

Syntax:

```
da-csv-fetch(<prop>[,<prop>*])
```

In this example we create an HTTP request header that contains device information:

```
frontend www
  bind :80
  mode http
  http-request set-header X-DeviceInfo %[da-csv-
fetch(primaryHardwareType,osName,osVersion,browserName,browserVersion,browserRenderingEngine)]
```

In the next example, we define an ACL named `is_mobile` and then use it when choosing a backend:

```
frontend www
  bind :80
  mode http
  acl is_mobile da-csv-fetch(mobileDevice) 1
  use_backend mobile_site if is_mobile
  default_backend desktop_site
```

da-csv-conv

Use the converter `da-csv-conv` to perform a lookup in the database that returns the values of the specified properties. It takes the `User-Agent` header as an input parameter:

Syntax:

```
da-csv-conv(<prop>[,<prop>*])
```

In this example we create an HTTP request header that contains device information:

```
frontend www
  bind :80
  mode http
  http-request set-header X-DeviceInfo %[req.hdr(user-agent),da-csv-
conv(primaryHardwareType,osName,osVersion,browserName,browserVersion,browserRenderingEngine)]
```

Update the database during runtime

> **This section applies to:**
>
> - HAProxy Enterprise 1.9r1 and newer

Use the DeviceAtlas Update module to keep the contents of the device detection database current. This allows you to keep multiple load balancers synced with the latest data.

1. Install a web server of your choice and host the database file at a URL HAProxy Enterprise can access. For example, host the file at `http://192.168.0.1:8000/deviceatlas.json`.

2. Install the package `hapee-3.1r1-lb-da-update`:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-da-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-da-update
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-da-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-da-update
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-da-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-da-update
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-da-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-da-update
```

3. Add the following lines to the `global` section of your configuration file:

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-da-update.so
  deviceatlas-update url http://192.168.0.1:8000/deviceatlas.json delay 24h log
```

Be sure to specify the port number where your file is hosted, for example `8000` as in the example above. With this configuration, HAProxy Enterprise downloads the database every 24 hours and displays a message in the logs when it succeeds or if it encountered errors during the update.

The `deviceatlas-update` directive enables updating the database over HTTP from a specified URL. Updating a database with a newer version invalidates any cached lookups (if caching is used), unless you enable `checksum` and the new and old database contents are identical.

4. Learn more about deviceatlas-update

   The directive `deviceatlas-update` supports the following syntax:

   ```
   deviceatlas-update url <url> [delay <u> | xdelay <u s b r>] [timeout <t>] [retries <n>] [checksum] [hash]
   [modified] [source <addr>[:<port>]] [log] [dontlog-normal] [param*]
   ```

   where:

Argument	Description
`url <url>`	**Required**. Specifies the database update URL. The updated data can be either JSON or precompiled JSON.
`delay <u>`	Specifies the period between each attempt to download a new database version. The delay is a simplified version of the `xdelay` keyword.
`xdelay <u s b r>`	`<u>` specifies the period between each attempt to download a new database version; `<s>` specifies the initial (first) download delay; `<b>` specifies the delay between the download of each element of the database; If the download fails, `<r>` determines the delay for the next attempt; Default values are: u = 5m, s = 5s, b = 10s, and r = 30s.
`timeout <t>`	Specifies the HTTP connection timeout for attempts to download a new database version. The value is set in milliseconds by default, but you can set it to any other unit if you add a unit suffix to the number. Defaults to 5 seconds.
`retries <n>`	Specifies the number of retries to download a new DeviceAtlas database version. If not set, the global `retries` value applies (defaults to 3).
`checksum`	If set, determines the use of the `SHA1` control sum to verify that the contents of the recently downloaded database is identical to the current one. If they are identical, then live-reload of the database does not occur, thereby preserving the cached contents (if using caching).
`hash`	If set, enables authentication of the downloaded database. Each file undergoing upgrade must have the associated file with `SHA1` checksum. A `SHA1` checksum file has the extension `.sha1`. The typical way of creating a `SHA1` checksum file is: `sha1sum file > file.sh`.
`modified`	Specifies the use of the time from the `Last-Modified` response HTTP header. Example: checks whether to update the data using the `If-Modified-Since` request HTTP header.
`source <addr>[:<port>]`	Sets the source address for outgoing connection. `<addr>` is the IPv4 address the load balancer binds to before it connects to a server; The default value is 0.0.0.0 to let the system select the most optimal address to reach its destination; `<port>` is optional; The default value of zero means that the system selects a free port; Does not support port ranges.
`log`	Specifies whether to log operation errors.
`dontlog-normal`	Deactivates logging of successful updates.
`param*`	Lists other server parameters that are useful for configuring SSL features.

The options `checksum` and `modified` are mutually exclusive. If you define them at the same time, the option `modified` automatically switches off and a warning message prints when the load balancer starts.

Runtime API

It is possible to manage the DeviceAtlas module via the Runtime API.

da-update update

> **Prerequisites**
>
> This command becomes available after you have installed both the DeviceAtlas and DeviceAtlas Update modules.

The DeviceAtlas update module will update its database based on the interval you specified in the configuration. You can also force an update by specifying the time you would like the update to run, for example, 5 minutes from now. If you don't specify a time, or set the update delay to zero, the update will run immediately. The delay time you specify cannot exceed the time until the next regular update.

In this example, we want the update to run 5 minutes from now so we specify `5m` as the delay time.

```
echo "da-update update 5m" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
DeviceAtlas: forcing update in 5m
```

da-update status

> **Prerequisites**
>
> This command becomes available after you have installed both the DeviceAtlas and DeviceAtlas Update modules.

You can check the status of the update using the `da-update status` command. Note that while the update is processing, the status will show the progress:

```
echo "da-update status" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
DeviceAtlas module status
-------------------------------------------------------------------------------
 initialized:          yes

 Data update
  configuration:      /etc/hapee-3.1/hapee-lb.cfg:43
  url:                http://192.168.64.1:8000
  http status count:  0 0 0 0 7 / 0
  period/delay:       1d / 5s 10s 5s
  use cksum/hash/mod: no / no / no
  reload/retry count: 0 0 2 / 7 2
  reload time:        <NEVER> / <NEVER> / 2023-08-08 14:25:37'
  download time:      <NEVER> / <NEVER>
  currently updating: deviceatlas.json
     status/retry:      0 / 2
     data size:         0 / 0 (0.0%)
     dur/time left:     8s / 1s
```

da-update show

> **Prerequisites**
>
> This command becomes available after you have installed both the DeviceAtlas and DeviceAtlas Update modules.

Use the `da-update show` command to display the configuration of the DeviceAtlas Update module. The output of this command includes information about the configured database file and its update status.

Below, we retrieve the status information for our configured database file:

```
echo "da-update show" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
# da-update configuration
# url: next update
http://192.168.0.1:8000/deviceatlas.json: 23h58m
```

ScientiaMobile WURFL InFuze

This page applies to:

- HAProxy Enterprise 1.7r2 and newer

The WURFL module provides device detection services using the ScientiaMobile InFuze database.

Install the WURFL InFuze module

1. Log into your account at the ScientiaMobile website and subscribe to WURFL InFuze. Follow the installation instructions in the WURFL InFuze for C: User Guide to install the API on your HAProxy Enterprise server.
2. Copy the WURFL InFuze device detection database (XML file) to your HAProxy Enterprise server (for example `/etc/hapee-3.1/wurfl.xml`).
3. Install the WURFL module according to your platform:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-wurfl
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-wurfl
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-wurfl
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-wurfl
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-wurfl
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-wurfl
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-wurfl
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-wurfl
```

4. In the `global` section of your configuration, add the following lines:

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-wurfl.so
  wurfl-data-file /etc/hapee-3.1/wurfl.xml
  wurfl-information-list wurfl_id model_name
  wurfl-cache-size 100000
```

In this example:

- `wurfl-data-file` sets the path to the WURFL data file.
- `wurfl-information-list` list of WURFL capabilities, virtual capabilities, and property names to use in injected HTTP headers. Separate each value with a space. See the WURFL InFuze documentation for values that you can use.
- `wurfl-cache-size` sets the number of entries to keep in the LRU (least recently used) cache. The LRU cache speeds up lookup operations on User-Agent strings by storing previously processed lookups.

5. Optional: Set the `wurfl-information-list-separator` directive to change the separator to use between values returned from the database. By default, it uses a comma (`,`).

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-wurfl.so
  wurfl-data-file /etc/hapee-3.1/wurfl.xml
  wurfl-information-list wurfl_id model_name
  wurfl-cache-size 100000
  wurfl-information-list-separator :
```

6. Optional: Set the `wurfl-patch-file` directive to the file paths to WURFL patch definitions. You can have as many as necessary, and the API applies them in the order they appear in the configuration file. Separate each value with a space.
7. Optional: Set the `wurfl-engine-mode` directive to target a WURFL engine. It can be either `performance`, which is the default, or `accuracy`.
8. Optional: Set the `wurfl-useragent-priority` directive to either `plain` to prioritize using the plain User-Agent or `sideloaded_browser` to prioritize the sideloaded browser User-Agent. The default is `sideloaded_browser`.
9. Reload the configuration to apply the changes.

```
sudo systemctl reload hapee-3.1-lb
```

Use WURFL InFuze

After installing the module, use the following directives to perform database lookups.

wurfl-get

Use the fetch method `wurfl-get` to perform a lookup in the database that returns the values of the specified properties. You must declare these properties beforehand with the `wurfl-information-list` global directive.

Syntax:

```
wurfl-get(<prop>[,<prop>*])
```

In this example, we create an HTTP request header that contains device information.

```
frontend www
  bind :80
  mode http
  http-request set-header X-WURFL-Properties %[wurfl-get(wurfl_id,model_name)]
```

wurfl-get-all

Use the fetch method `wurfl-get-all` to perform a lookup in the database that returns values for all properties that the `wurfl-information-list` directive declared.

Syntax:

```
wurfl-get-all()
```

In this example we create an HTTP request header that contains all the device properties.

```
frontend www
  bind :80
  mode http
  http-request set-header X-WURFL-All %[wurfl-get-all()]
```

Update the database during runtime

> **This section applies to:**
>
> - HAProxy Enterprise 1.9r1 and newer

Use the WURFL Update module to keep contents of the device detection database current. This allows you to keep multiple load balancers synced with the latest data.

1. Install a web server of your choice and host the database file at a URL HAProxy Enterprise can access. For example, host the file at `http://192.168.0.1:8000/wurfl.xml.gz`.

2. Install the package `hapee-3.1r1-lb-wurfl-update`:

   Apt:

   ```
   sudo apt-get install hapee-<VERSION>-lb-wurfl-update
   ```

   Example for HAProxy Enterprise 3.1r1:

   ```
   sudo apt-get install hapee-3.1r1-lb-wurfl-update
   ```

Yum:

```
sudo yum install hapee-<VERSION>-lb-wurfl-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-wurfl-update
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-wurfl-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-wurfl-update
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-wurfl-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-wurfl-update
```

3. In the `global` section of your configuration, add the following directives:

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-wurfl-update.so
  wurfl-update url http://192.168.0.1:8000/wurfl.xml.gz delay 24h log
```

Be sure to specify the port number where your file is hosted, for example `8000` as in the example above. With this configuration, HAProxy Enterprise downloads the database every 24 hours and displays a message in the logs when it succeeds, or when it encounters an error during the update.

4. Learn more about wurfl-update

   The directive `wurfl-update` supports the following syntax:

```
wurfl-update url <url> [delay <u> | xdelay <u s b r>] [timeout <t>] [retries <n>] [checksum | modified] [hash]
[source <addr>[:<port>]] [log] [dontlog-normal] [param*]
```

where:

Argument	Description
`url <url>`	**Required**. Specifies the database update URL.
`delay <u>`	Specifies the period between each attempt to download a new database version. The delay is a simplified version of the `xdelay` keyword.
`xdelay <u s b r>`	`<u>` specifies the period between each attempt to download a new database version; `<s>` specifies the initial (first) download delay; `<b>` specifies the delay between the download of each element of the database; If the download fails, `<r>` determines the delay for the next attempt; Default values are: u = 5m, s = 5s, b = 10s, and r = 30s; If the load balancer cannot download the new version after three attempts, it cancels and discards the download until the next time interval defined by `<u>`.
`timeout <t>`	Specifies the HTTP connection timeout in milliseconds (default) for attempts to download a new database version. The value can be any other unit if you add a unit suffix. It defaults to 5000 milliseconds.
`retries <n>`	Specifies the number of retries to download a new WURFL database version. If not set, the global `retries` value applies (defaults to 3).
`checksum`	If set, this determines the use of the `SHA1` control sum to verify that the recently downloaded database is identical to the current one. If it is, then live-reload of the database does not occur, thereby preserving the cached contents (if using caching).
`hash`	If set, enables authentication of the downloaded data; Each file undergoing upgrade must have the associated file with `SHA1` checksum; A `SHA1` checksum file has the extension `.sha1`; The typical way of creating a `SHA1` checksum file is: `sha1sum file > file.sha1`.
`modified`	Specifies the use of the time from the `Last-Modified` response HTTP header. Example: checks whether to update the data using the `If-Modified-Since` request HTTP header.
`source <addr>[:<port>]`	Sets the source address for outgoing connections. `<addr>` is the IPv4 address the load balancer binds to before it connects to a server; The default value is 0.0.0.0 to let the system select the most optimal address to reach its destination; `<port>` is optional; The default value of zero means that the system selects a free port; It does not support port ranges.
`log`	Specifies whether to log operation errors.
`dontlog-normal`	Deactivates logging of successful updates.
`param*`	Lists other server parameters that are useful for configuring SSL features.

The options `checksum` and `modified` are mutually exclusive. If you define them at the same time, the option `modified` automatically switches off and a warning message prints when HAProxy Enterprise starts.

Runtime API

It is possible to manage the WURFL InFuze module via the Runtime API.

wurfl-update force-update

Prerequisites

This command becomes available after you have installed both the WURFL and WURFL Update modules.

The WURFL InFuze update module will update its database based on the interval you specified in the configuration. Use `wurfl-update force-update` to force an immediate update of the WURFL InFuze database.

In this example, we force an update of the database.

```
echo "wurfl-update force-update" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
WURFL: forcing update in 5m
```

wurfl-update status

Prerequisites

This command becomes available after you have installed both the WURFL and WURFL Update modules.

You can check the status of the update using the `wurfl-update status` command. Note that while the update is processing, the status will show the progress:

```
echo "wurfl-update status" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
WURFL InFuze module status
------------------------------------------------------------------------------
 initialized:          yes

 Data update
  configuration:      /etc/hapee-2.7/hapee-lb.cfg:43
  url:                http://192.168.64.1:8000/
  http status count:  0 0 0 0 7 / 0
  period/delay:       1d / 5s 10s 5s
  use cksum/hash/mod: no / no / no
  reload/retry count: 0 0 2 / 7 2
  reload time:        <NEVER> / <NEVER> / 2023-08-08 14:25:37
  currently updating: wurfl.xml.gz
     status/retry:      0 / 2
     data size:         0 / 0 (0.0%)
     dur/time left:     8s / 1s
```

wurfl-update show

> **Prerequisites**
>
> This command becomes available after you have installed both the WURFL and WURFL Update modules.

Use the `wurfl-update show` command to display the configuration of the WURFL Update module. The output of this command includes information about the configured database file and its update status.

Below, we retrieve the status information for our configured database file:

```
echo "wurfl-update show" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
# wurfl-update configuration
# url: next update
http://192.168.0.1:8000/wurfl.xml.gz: 23h58m
```

Dynamic updates

The Update module allows you to update the contents of ACL files, Map files, and TLS ticket key files periodically without reloading the load balancer. At startup, the process loads the contents of the files. Then, at a set interval, it downloads updates to the files from a specified URL. The content of the downloaded files replace the existing content.

This mechanism allows you to keep ACL and Map files up to date across multiple HAProxy Enterprise servers. Each server will check for changes independently.

Install the Update module

1. Install the Update module according to your platform:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-update
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-update
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-update
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-update
```

2. In the `global` section of your configuration file, add a `module-load` directive to load the Update module:

```
global
  module-path /opt/hapee-3.1/modules
  module-load hapee-lb-update.so
```

Update map files

To update map files with the module:

1. Below the `global` section, add a new configuration section named `dynamic-update`. Inside, add one or more `update` lines that specify from where and how often to download new content. Here is an example that updates the contents of a file at `/etc/hapee-3.1/redirects.map` from the URL `http://192.168.0.1:80/redirects.map` every 5 minutes:

```
dynamic-update
  update id /etc/hapee-3.1/redirects.map map url http://192.168.0.1:80/redirects.map delay 5m
```

As an example, the contents of this file below lists URL paths to redirect clients to during maintenance periods, based on the client's IP address:

redirects.map

```
10.0.0.0/8     /pages/maintenance-internal.html
192.168.0.0/16 /pages/maintenance-external.html
0.0.0.0/0      /pages/maintenance-catchall.html
```

2. Learn more about update

   The directive `update` supports the following syntax:

```
update id <id> url <url> [delay <delay> | xdelay <delay start next retry>] [timeout <timeout>] [retries <nb>]
[modified | xmodified ] [source <addr>[:<port>]] [log] [dontlog-normal] [map] [purge <count> [<interval>
[<id>]]] [tls-ticket-keys] [param*]
```

where:

Argument	Description
`id <url>`	**Required**. The file name. The module uses the absolute file path. The file must exist when HAProxy Enterprise starts.
`url <url>`	**Required**. Specifies the update URL.
`delay <u>`	Specifies the period between each attempt to download a new file version. The delay is a simplified version of the `xdelay` keyword.
`xdelay <u s b r>`	`<u>` specifies the period between each attempt to download a new file version; `<s>` specifies the initial (first) download delay; `<b>` is not used in this module and its value is not important. However, its value must be present. Default: 10s; If the download fails, `<r>` determines the delay for the next attempt; Default values are: u = 5m, s = 5s, b = 10s, and r = 30s.
`timeout <t>`	Specifies the HTTP connection timeout for attempts to download a new file version. The value is set in milliseconds by default, but you can set it to any other unit if you add a unit suffix to the number. Defaults to 5 seconds.
`retries <n>`	Specifies the number of retries to download a new file. If not set, the global `retries` value applies (defaults to 3).
`modified`	Specifies the use of the time from the `Last-Modified` response HTTP header. Example: checks whether to update the data using the `If-Modified-Since` request HTTP header.
`xmodified`	Available since version 2.4r1. The same as `modified` except that the file modification time is set immediately after reading the file status.
`source <addr>[:<port>]`	Sets the source address for outgoing connections. `<addr>` is the IPv4 address the load balancer binds to before it connects to a server; The default value is 0.0.0.0 to let the system select the most optimal address to reach its destination; `<port>` is optional; The default value of zero means that the system selects a free port; Does not support port ranges.
`log`	Specifies whether to log operation errors.
`dontlog-normal`	Deactivates logging of successful updates.
`map`	Informs that the downloaded file must be interpreted as a `map` file. By default, the file is interpreted as an `acl` file.
`purge <count> [<interval> [<id>]]`	This is available only for `map` and `acl` data types. Use the `<count>` argument to indicate that the module should delete old patterns when it downloads the next set of data. Otherwise, use `<interval>` and `<id>` to run a separate purge task. `<count>` is the number of old and/or invalid patterns that can be purged at once. Allowed values for that parameter are between 1 and 100000. The default is 1000; `<interval>` is the frequency for calling the pattern purging task. The value is set in milliseconds (if not specified by another time unit). If you specify the interval as 0, the task will be executed as quickly as possible. When setting the interval to 0, it is not recommended to purge a large group at once (a `<count>` of 100 is the optimal number for interval 0); `<interval>` has no default, so a value must be specified. `<id>` assigns a specific id to the purge task. You can use `<id>` to group certain purging tasks, or you can assign each operation its own task. The default is 1000, which means that all purging operations will be performed in the common task which has an id of 1000.

Argument	Description
`tls-ticket-keys`	Specifies that the downloaded file is a TLS ticket keys file (instead of an ACL or Map file).
`param*`	Lists other server parameters that are useful for configuring SSL features.

The options `checksum` and `modified` are mutually exclusive. If you define them at the same time, the option `modified` automatically switches off and a warning message prints when the load balancer starts.

3. In your `frontend`, use one of the `map` converters to read a value from the map file.

   For example, the `map_ip` converter will select the first matching row and cast it as an IP address. You can map a value to `0.0.0.0/0` as the last entry in the `.map` file to act as a catch all for IP addresses that do not match any other patterns in the file.

   In the following snippet, a `frontend` references this data using an `acl` directive and an `http-request redirect` directive:

```
frontend fe_main
  mode http
  bind :80
  acl redirect_needed src,map_ip(/etc/hapee-3.1/redirects.map) -m found
  acl redirect_performed path_beg /maint/
  http-request redirect location %[src,map_ip(/etc/hapee-3.1/redirects.map)] if redirect_needed !redirect_performed
```

   If you specify a file for download via `update` in a `dynamic-update` block, you must make reference to this file somewhere else in your configuration (for example, in a `frontend` block, as in the snippet above shows).

   If you do not reference the file you specify in `dynamic-update` for use in another configuration section, the service will log an error similar to the following:

```
[ALERT] (1525) : config : 'update': id '/redirects.map' not found in file ./haproxy.cfg at line 22
[ALERT] (1525) : config : Fatal errors found in configuration.
```

Update TLS session ticket keys

TLS compatibility

TLS session tickets are supported up to TLS version 1.2.

When the load balancer and a client exchange messages over TLS, they share a secret key to encrypt their communication. This key is called a session key and it lasts only as long as the TLS session. Each new TLS session uses a unique key, which makes it difficult for an attacker to exploit.

Generating a session key costs the load balancer CPU time, which can add up when managing many clients. By asking each client to store their session key locally and reuse it between visits, the load balancer offloads some work and speeds up the time it takes to establish a TLS connection. The client can send their old session key to the load balancer to resume their session, negating the need to create a new one.

To make this secure, the load balancer uses a secret key known only to it to encrypt the session key in a package called a session ticket before sending it to the client. Only the load balancer can decrypt the ticket. The load balancer's secret key should be changed at least every 24 hours in case an attacker steals it.

When you have multiple load balancers terminating TLS, you should ensure that TLS ticket keys are distributed across all of the load balancers. Otherwise, if a client makes a new connection with another HAProxy Enterprise server in the same cluster, a new key will need to be exchanged (with the associated CPU work).

In this section, we describe how to use the Update module to distribute the keys to all load balancers.

Configure your webserver

Perform these steps on a web server that's accessible to your HAProxy Enterprise servers:

1. Create a file named `update_hapee_tls_tickets.sh` and add the following contents to it. This script replaces the ticket keys stored in the file `hapee_ticket_keys.txt`. In this example, we use `/var/www` as the web server's directory, but you can change this depending on your web server's folder structure.

   ```
   #!/bin/bash
   openssl rand 80 -base64 >> /var/www/hapee_ticket_keys.txt
   new_keys=$(tail -n3 /var/www/hapee_ticket_keys.txt)
   echo "$new_keys" > /var/www/hapee_ticket_keys.txt
   ```

2. Run `sudo crontab -e` and add the following cron job to call the script every five minutes, changing the `/path/to/file/` for your environment:

   ```
   */5 * * * * /bin/bash /path/to/file/update_hapee_tls_tickets.sh >>/dev/null 2>&1
   ```

3. If possible, configure your web server so that only the HAProxy Enterprise servers can access the URL for `hapee_ticket_keys.txt`. If anyone else can get the contents of this file, they will be able to launch man-in-the-middle attacks against TLS connections to your load balancer servers.

Configure HAProxy Enterprise

Perform these steps on your HAProxy Enterprise servers:

1. To give your load balancers a file with keys, which the Update module will later replace, create the file `/etc/ssl/hapee_ticket_keys.txt`. Add three ticket keys to it by running the following command three times:

```
openssl rand -base64 48 >> /etc/ssl/hapee_ticket_keys.txt
```

2. Add a `tls-ticket-keys` argument to your HTTPS `bind` line:

```
frontend ft
  bind :443 ssl crt /etc/ssl/site.pem  tls-ticket-keys /etc/ssl/hapee_ticket_keys.txt
```

3. Below the `global` section, add a new configuration section named `dynamic-update`. Inside, add an `update` line that specifies from where and how often to download new content. Here is an example that updates the contents of a file at `/etc/ssl/hapee_ticket_keys.txt` from the URL `http://192.168.0.1/hapee_ticket_keys.txt` every 2 minutes:

```
dynamic-update
  update tls-ticket-keys id /etc/ssl/hapee_ticket_keys.txt url http://192.168.0.1/hapee_ticket_keys.txt delay 2m
```

   After updating each load balancer server, each will query the web server at an interval and update the three keys in memory.

   As HAProxy Enterprise will use the middle of the three keys for encryption, slight variances in timing are acceptable, as the other HAProxy Enterprise servers will still be able to decrypt tickets from the other servers unless they miss two updates (since it's running every two minutes, even in unfavorable conditions this should not happen).

   If you are transferring these keys over an untrusted network, the Update module supports HTTPS. Simply add the `ssl` and `ca-file /etc/ssl/certs/ca-bundle.trust` arguments to the `update` line, where `ca-file` points to your CA file bundle.

Runtime API

It is possible to manage the Update module via the Runtime API.

lb-update list

The `lb-update list` command returns the list of `update` lines in the `dynamic-update` section of the configuration file. Use `lb-update list` to display the files that the Update module manages.

```
echo "lb-update list" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
# lb-update configured entries
# id: next update
/etc/hapee-3.1/redirects.map: 4m49s
```

lb-update force-update

The `lb-update force-update` command launches an immediate update for the specified file.

```
echo "lb-update force-update /etc/hapee-3.1/redirects.map" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
lb-update: force-update `/etc/hapee-3.1/redirects.map`.
```

lb-update status

The `lb-update status` command shows the module status.

```
echo "lb-update status" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
lb-update module status
------------------------------------------------------------------------------
 initialized:          yes

 SMP update
   configuration:      /etc/hapee-3.1/hapee-lb.cfg:30
   entry:              /etc/hapee-3.1/redirects.map
   generation id:      256
   count:              3 / 6
   url:                http://127.0.0.1:3000/redirects.map
   http status count:  0 44 0 0 0 / 0
   period/delay:       10s / 5s 10s 5s
   use modified:       no
   reload/retry count: 44 0 0 / 0 0
   reload time:        2023-03-29 19:13:49 / 2023-03-29 19:13:40 / <NEVER>
   download time:      2023-03-29 19:13:49 / <NEVER>
   next update in:     4s
   purging id(s):      #1000 0:20 1:23 2:23 4:23 6:22 8:20 11:20 16:17 253:16
      block size:       1
      interval:         1h
```

The fields of the `lb-update status` command are described below:

Field	Description
initialized	Whether the Update module was initialized. Initialization in this case does not mean that the module read the configuration (if any), but only that HAProxy Enterprise called the module initialization function.
configuration	The configuration file and the line number where the `update` directive is specified.
entry	The value of the `update` directive's `id` parameter you used when calling `lb-update force-update`.
generation id	The current pattern generation identifier. The set of patterns from the next download will be associated with this ID.
count	The number of patterns from the currently active generation / total number of patterns (curr / total). In case new data is being downloaded, the number of currently loaded patterns is also displayed (curr new / total).
url	The item's download URL. HTTP and HTTPS web protocols are supported.
http status count	The download HTTP status counter, sorted by status classes 1xx, 2xx, 3xx, 4xx, and 5xx. The final number after the slash is a count of responses that had a non-standard HTTP status code outside of the 100-599 range.
period	The download period. Every so often there will be an attempt to download the data. In case the download cannot be completed within that period, it will be aborted and the data discarded.
delay	Indicates three values: start/next/retry. The `<start>` value specifies the initial (first) download delay. The `<next>` value is not used in this module and its value is not important. It is present only because of the compatibility of delay parameters with other lb-update-like modules. If the download fails, `<retry>` determines the delay for the next attempt. The number of consecutive downloads is 3 (after that the download is canceled until the next time specified by the download period).
use modified	Indicates whether the module uses the time from the `Last-Modified` response HTTP header to check whether to update the data using the `If-Modified-Since` request HTTP header.
reload count	Indicates three numbers. The first indicates the number of times the data reload was performed. The second indicates the number of times the data was not reloaded because the data did not differ from the old data. The third indicates the number of times the data reload failed because the data was incorrect.
retry count	Indicates two numbers: the number of unsuccessful downloads and the number of final unsuccessful downloads. For example, if each download is tried 3 times (the first number) and if that fails, then the second number is increased by 1. After that, the subsequent download attempt is aborted for the respective cycle and the time for the next cycle is awaited.
reload time	Indicates the date a reload was last performed, the date when data was not reloaded because the data did not differ from the old data, and the date when the data reload failed because the data was incorrect.
download time	Indicates the date of the last successful download and the date of the last unsuccessful download.
next update in	Indicates the time until the download of new data will be started. In cases where the download has not yet finished, you will see a `currently updating` section instead.
purging id(s)	The number of purgeable patterns specified by their pattern generation id. If the purge operation is configured to run on an interval (instead of on the next download), the task IDs for each purge are listed as well (only one task id is listed if all purge operations run under the common task #1000). Each set of patterns and purgeable tasks are listed by `<pattern generation id>:<count of patterns to purge>`. Two additional fields may appear under `purging id(s)`:

Field	Description
	`block size` indicates the maximum number of patterns to purge at one time; `interval` indicates how often the purge task will run. This is present only if the purge operation is configured to run on an interval.

lb-update purge

> **This section applies to:**
>
> - HAProxy Enterprise 2.4r1 and newer

Deletes data from the list of old/invalid patterns for the specified file. Use `lb-update purge` to run a purge task immediately.

```
echo "lb-update purge /etc/hapee-3.1/redirects.map 13 1" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
lb-update: purge  `/etc/hapee-3.1/redirects.map` 13 : all pattern(s) purged!
```

Two additional parameters are optional for `purge`:

- `<gen_id>`: the pattern generation id
- `<count>`: the number of records to purge

If no `<gen_id>` is specified, unnecessary patterns are deleted in the order in which they were added to the list. The default value for `<count>` is 1000 and the allowable range is between 1 and 100000.

Use the `lb-update status` command to find the pattern generation id (`<gen_id>`). These ids may be listed under `purging id(s)` where generation ids are listed with the number of patterns ready to purge per id.

```
purging id(s): #1000 0:20 1:23 2:23 4:23 6:22 8:20 11:20 16:17 253:17
```

If, for a particular generation id, there remain additional patterns to purge, the number of remaining patterns is shown:

```
echo "lb-update purge /etc/hapee-3.1/redirects.map 13 100" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
lb-update: purge  `/etc/hapee-3.1/redirects.map` 13 : 100 pattern(s) purged, 200 left!
```

Geolocation

- Digital Element
- MaxMind

Digital Element NetAcuity

This page applies to:

- HAProxy Enterprise 1.8r1 and newer

The NetAcuity module provides geolocation lookups using Digital Element's GeoIP databases.

Install the NetAcuity module

1. Log into your account at the Digital Element website and download the NetAcuity database `.tar.gz` file. Extract the contents of the file and place them in a directory on your HAProxy Enterprise server (e.g. `/etc/hapee-3.1/netacuity/`).

2. Install the NetAcuity module according to your platform:

Apt:

```
sudo apt-get install hapee-<VERSION>-b-netacuity
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-b-netacuity
```

Yum:

```
sudo yum install hapee-<VERSION>-b-netacuity
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-b-netacuity
```

Zypper:

```
sudo zypper install hapee-<VERSION>-b-netacuity
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-b-netacuity
```

Pkg:

```
sudo pkg install hapee-<VERSION>-b-netacuity
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-b-netacuity
```

3. In the `global` section of HAProxy Enterprise configuration, add the following lines:

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-netacuity.so
  netacuity-load 04 /etc/hapee-3.1/netacuity
  netacuity-cache-size 2000000
```

4. Reload the HAProxy Enterprise configuration to apply the changes.

Configure the NetAcuity module

The module adds the following `global` directives:

Directive	Description
`netacuity-load <feature_code> <directory>`	**Required**. Specifies the local directory where you store NetAcuity files. The `<feature_code>` depends on the type of database. For example, if you name your NetAcuity files `na_04_01.db`, `na_04_02.db`, etc., then set it to `04`.
`netacuity-cache-size <number>`	Specifies the size of the LRU cache used for lookups. The minimum size of 0 disables the cache. The maximum cache size is 10000000. Default: 0

Use converters to perform database lookups

netacuity-lookup-ipv4

Use the converter `netacuity-lookup-ipv4` to perform a lookup in the IPv4 database that returns the values of the specified properties. It can return several properties by specifying each successively; In that case, the returned values are separated by commas.

Syntax:

```
netacuity-lookup-ipv4(<prop>[,<prop>*])
```

The maximum number of properties in one lookup is eight. Valid property types are:

- `src-ip` (Note: This property shows the IP address of the client in IPv6 format.)
- `pulse-area-codes`
- `pulse-city`
- `pulse-city-code`
- `pulse-city-conf`
- `pulse-conn-speed`
- `pulse-conn-type`
- `pulse-continent-code`
- `pulse-country`
- `pulse-country-code`
- `pulse-country-conf`
- `pulse-gmt-offset`
- `pulse-in-dst`
- `pulse-internal-code`
- `pulse-latitude`
- `pulse-longitude`
- `pulse-metro-code`
- `pulse-postal-code`
- `pulse-postal-conf`
- `pulse-region`
- `pulse-region-code`
- `pulse-region-conf`
- `pulse-timezone-name`
- `pulse-two-letter-country`

In this example we set HTTP request headers that contain geolocation properties based on client's source IP address.

```
frontend www
  bind :80
  mode http
  http-request add-header X-NetAcuity-IPv4-1 %[src,netacuity-lookup-ipv4("src-ip","pulse-area-codes","pulse-city","pulse-city-code","pulse-city-conf","pulse-conn-speed","pulse-conn-type","pulse-continent-code")]
  http-request add-header X-NetAcuity-IPv4-2 %[src,netacuity-lookup-ipv4("pulse-country","pulse-country-code","pulse-country-conf","pulse-gmt-offset","pulse-in-dst","pulse-internal-code","pulse-latitude","pulse-longitude")]
  http-request add-header X-NetAcuity-IPv4-3 %[src,netacuity-lookup-ipv4("pulse-metro-code","pulse-postal-code","pulse-postal-conf","pulse-region","pulse-region-code","pulse-region-conf","pulse-timezone-name","pulse-two-letter-country")]
```

netacuity-lookup-ipv6

Use the converter **`netacuity-lookup-ipv6`** to perform a lookup in the IPv6 database that returns the values of the specified properties. It can return several properties by specifying each successively; In that case, the returned values are separated by commas.

Syntax:

```
netacuity-lookup-ipv6(<prop>[,<prop>*])
```

The maximum number of properties in one lookup is eight. Valid property types are:

- `src-ip` (Note: This property shows the IP address of the client in IPv6 format.)
- `pulse-area-codes`
- `pulse-city`
- `pulse-city-code`
- `pulse-city-conf`
- `pulse-conn-speed`
- `pulse-conn-type`
- `pulse-continent-code`
- `pulse-country`
- `pulse-country-code`
- `pulse-country-conf`
- `pulse-gmt-offset`
- `pulse-in-dst`
- `pulse-internal-code`
- `pulse-latitude`
- `pulse-longitude`
- `pulse-metro-code`
- `pulse-postal-code`
- `pulse-postal-conf`
- `pulse-region`
- `pulse-region-code`
- `pulse-region-conf`
- `pulse-timezone-name`
- `pulse-two-letter-country`

In this example we set HTTP request headers that contain geolocation properties based on client's source IP address.

```
frontend www
  bind :80
  mode http
  http-request add-header X-NetAcuity-IPv6-1 %[src,netacuity-lookup-ipv6("src-ip","pulse-area-codes","pulse-city","pulse-city-code","pulse-city-conf","pulse-conn-speed","pulse-conn-type","pulse-continent-code")]
  http-request add-header X-NetAcuity-IPv6-2 %[src,netacuity-lookup-ipv6("pulse-country","pulse-country-code","pulse-country-conf","pulse-gmt-offset","pulse-in-dst","pulse-internal-code","pulse-latitude","pulse-longitude")]
  http-request add-header X-NetAcuity-IPv6-3 %[src,netacuity-lookup-ipv6("pulse-metro-code","pulse-postal-code","pulse-postal-conf","pulse-region","pulse-region-code","pulse-region-conf","pulse-timezone-name","pulse-two-letter-country")]
```

Update the database during runtime

Use the NetAcuity Update feature to keep the contents of the geolocation database current. This allows you to keep multiple HAProxy Enterprise nodes synced with the latest data.

1. Install a web server of your choice and host the database file at a URL where HAProxy Enterprise can access. For example, host the files at `http://192.168.122.1:8000/netacuity.tar.gz`. We recommend that you host the unzipped file directory at this URL.

2. Add the following lines to the `global` section of your configuration file, where the URL hosts an updated version of the file. Be sure to specify the port number where your file is hosted, for example, `8000` as in the example below.

   ```
   global
     # ... other global settings
     netacuity-update url 04 http://192.168.122.1:8000/netacuity.tar.gz delay 24h  timeout 100ms  retries 3  checksum hash  log
   ```

With this configuration, HAProxy Enterprise downloads the database every 24 hours and displays a message in the logs when it succeeds or if it encountered errors during the update.

netacuity-update

The `netacuity-update` directive enables updating the database over HTTP from a specified URL. Updating a database with a newer version invalidates any cached lookups (if using cache), unless you enable the checksum setting and the new and old database contents are identical.

The directive supports the following syntax:

```
netacuity-update url <feature_code url>
  [delay <u> | xdelay <u s b r>]
  [timeout <t>]
  [retries <n>]
  [checksum]
  [hash]
  [log]
  [dontlog-normal]
  [param\*]
```

where:

Argument	Description
`url <feature_code url>`	**Required**. Specifies URL for the database update. We recommend that you host the unzipped file directory at this URL. The feature code depends on the type of database. For example, if you name your NetAcuity files `na_04_01.db`, `na_04_02.db`, etc., then set the feature code to `04`.
`delay <u>`	`<u>` specifies the period between each attempt to download a new database version. The delay is a simplified version of the `xdelay` keyword.
`xdelay <u s b r>`	`xdelay` settings are defined as follows: `<u>` specifies the period between each attempt to download a new database version. Default: 5m. `<s>` specifies the initial (first) download delay. Default: 5s. `<b>` specifies the delay between the download of each element of the database. Default: 10s. If the download fails, `<r>` determines the delay for the next attempt. Default: 30s. If the new version of the database fails to download after three attempts, the module cancels the download until the next time interval specified by `<u>`. In this case, it discards the downloaded data.
`timeout <t>`	Specifies the HTTP connection timeout for attempts to download a new database version. The value is in milliseconds by default, but you can set it to any other unit if you add it as a suffix to the number. Default: 5s.
`retries <n>`	Specifies the number of retries to download a new NetAcuity database version. If unspecified, the global `retries` value applies. Default: 3
`checksum`	If present, determines the use of the SHA1 control sum to verify that the content of the recently downloaded database is identical to the one already used. If they are identical, the module does not do a live-reload of the database, thereby preserving cache contents (if you use caching).
`hash`	If present, enables authentication of downloaded data. Each upgraded file must have the associated file with a SHA1 check The SHA1 checksum file has the extension `.sha1`. The typical way to create the SHA1 checksum file is: `sha1sum file > file.sha1`
`log`	Specifies whether to log operation errors.
`dontlog-normal`	Deactivates logging for successful updates.
`param*`	Lists other server parameters; useful to configure special SSL features.

Runtime API

It is possible to manage the NetAcuity module via the Runtime API. For general information on using the runtime API, see HAProxy Enterprise Runtime API.

lb-netacuity cache disable

Disable the NetAcuity lookup cache.

If the NetAcuity cache is enabled, each lookup is cached so that subsequent requests are loaded from the cache instead of the NetAcuity database. The NetAcuity module will perform all lookups in its database when the cache is disabled.

Below, we disable the cache:

```
echo "lb-netacuity cache disable" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
NetAcuity: cache disabled
```

lb-netacuity cache enable

Enable the NetAcuity lookup cache.

If the NetAcuity cache is enabled, each lookup is cached so that subsequent requests are loaded from the cache instead of the NetAcuity database.

Below, we enable the cache:

```
echo "lb-netacuity cache enable" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
NetAcuity: cache enabled
```

lb-netacuity cache invalidate

Invalidate the NetAcuity lookup cache.

If the NetAcuity cache is enabled, each lookup is cached so that subsequent requests are loaded from the cache instead of the NetAcuity database. Invalidating the cache marks all entries in the cache invalid. After invalidating the cache, the NetAcuity module will perform lookups in its database as it rebuilds its cache for subsequent requests.

Below, we invalidate the cache:

```
echo "lb-netacuity cache invalidate" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
NetAcuity: LRU cache invalidated
```

lb-netacuity disable

Disable the NetAcuity database lookup engine.

When disabled, the NetAcuity module does not perform lookups.

Below, we disable NetAcuity lookups:

```
echo "lb-netacuity disable" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
NetAcuity: data lookup disabled
```

lb-netacuity enable

Enable the NetAcuity database lookup engine.

The NetAcuity database lookup engine is enabled automatically upon installation. You may need to re-enable it if you have manually disabled it.

Below, we enable NetAcuity lookups:

```
echo "lb-netacuity enable" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
NetAcuity: data lookup enabled
```

lb-netacuity get

Display all the data associated with the specified IP address.

You can retrieve all of the data associated with a specified IP address. This query works even if the NetAcuity module is disabled and the result of the query is not cached. The query is performed against the database and not the cache. You can specify the IP address in IPv4 or IPv6 format.

Retrieve the NetAcuity database information for the IP address `192.168.64.1` by specifying that IP address with the `lb-netacuity get` command:

```
echo "lb-netacuity get 192.168.64.1" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
NetAcuity 192.168.64.1 data
------------------------------------------------------------
edge-area-codes: (null)
edge-city: (null)
edge-city-code: (null)
edge-city-conf: (null)
edge-conn-speed: (null)
edge-continent-code: (null)
edge-country: (null)
edge-country-code: (null)
edge-country-conf: (null)
edge-gmt-offset: (null)
edge-in-dst: (null)
edge-internal-code: (null)
edge-latitude: (null)
edge-longitude: (null)
edge-metro-code: (null)
edge-postal-code: (null)
edge-postal-conf: (null)
edge-region: (null)
edge-region-code: (null)
edge-region-conf: (null)
edge-timezone-name: (null)
edge-two-letter-country: (null)
keys: 22/32 slots, 378/1024 byte(s) used; data: 0/187 byte(s) used
```

lb-netacuity status

Display the status of the NetAcuity module.

The NetAcuity module maintains some statistics about its operations. The `lb-netacuity status` command provides information about its configuration and cache.

The results of the `lb-netacuity status` command show information about the current state of its cache, as well as information about updates to its database.

```
echo "lb-netacuity status" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
NetAcuity module status
-------------------------------------------------------------------------------
 initialized:           yes

 Database
   configuration:       /etc/hapee-2.7/hapee-lb.cfg:34
   enabled:             yes
   invalid:             no
   feature code:        4
   directory:           /etc/hapee-2.7/netacuity
   lookup separator:    ","

 LRU cache
   enabled:             yes
   serial:              0 1
   usage/size:          0/2000000

 Database update
   configuration:       /etc/hapee-2.7/hapee-lb.cfg:37
   feature code:        4
   url:                 http://192.168.122.1:8000
   http status count:   0 4 0 0 0 / 0
   period/delay:        1d / 5s 10s 5s
   use checksum/hash:   no / no
   reload/retry count:  1 0 0 / 0 0
   reload time:         2023-08-09 09:53:45 / <NEVER> / <NEVER>
   next update:         na_04_01.db in 23h49m
```

lb-netacuity update

Force an update of the NetAcuity database.

The NetAcuity update module will update the database based on the interval you specified in the configuration. You can also force an update by specifying the time you would like the update to run, for example, 5 minutes from now. If you don't specify a time, or set the update delay to zero, the update will run immediately. The delay time you specify cannot exceed the time until the next regular update, otherwise you will receive an error message similar to "NetAcuity: the specified delay time is further than the time of the next update (3s), cancelling request".

In this example, we want the update to run 5 minutes from now so we specify `5m` as the delay time.

```
echo "lb-netacuity update 5m" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
NetAcuity: forcing update in 5m
```

You can check the status of the update using the `lb-netacuity status` command. Note that while the update is processing, the status will show the progress:

```
echo "lb-netacuity status" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
      NetAcuity module status
      ------------------------------------------------------------------------------
      initialized:           yes

      Database
        configuration:       /etc/hapee-2.7/hapee-lb.cfg:34
        enabled:             yes
        invalid:             no
        feature code:        4
        directory:           /etc/hapee-2.7/netacuity
        lookup separator:    ","

      LRU cache
        enabled:             yes
        serial:              0 1
        usage/size:          0/2000000

       Database update
        configuration:       /etc/hapee-2.7/hapee-lb.cfg:37
        feature code:        4
        url:                 http://192.168.64.1:8000
        http status count:   0 0 0 2 0 / 0
        period/delay:        1d / 5s 10s 5s
        use checksum/hash:   yes / yes
        reload/retry count:  0 0 0 / 2 0
        reload time:         <NEVER> / <NEVER> / <NEVER>
        currently updating:  na_04_01.db
            status/retry:      0 / 2
            data size:         0 / 0 (0.0%)
            dur/time left:     8s / 1s
```

MaxMind

This page applies to:

- HAProxy Enterprise 1.7r2 and newer

Using HAProxy Fusion?

If you're using HAProxy Fusion, then see the HAProxy Fusion - MaxMind topic instead.

The MaxMind module provides geolocation lookups using MaxMind's GeoIP2 databases.

Install the MaxMind module

1. Log into your account at the MaxMind website and download the GeoIP databases. Copy them to your HAProxy Enterprise server (e.g. `/etc/hapee-3.1/GeoIP2-City.mmdb`, `/etc/hapee-3.1/GeoIP2-ASN.mmdb`).

2. Install the MaxMind module according to your platform:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-maxmind
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-maxmind
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-maxmind
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-maxmind
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-maxmind
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-maxmind
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-maxmind
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-maxmind
```

3. In the `global` section of your configuration, add the following lines:

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-maxmind.so
  maxmind-load mlock_max 512000000  CITY /etc/hapee-3.1/GeoIP2-City.mmdb  ASN /etc/hapee-3.1/GeoIP2-ASN.mmdb
  maxmind-cache-size 200000
```

4. Reload the HAProxy Enterprise configuration to apply the changes.

Discover properties

1. Install the mmdblookup utility. This utility enables you to perform look ups for IP addresses in a MaxMind database file and learn the structure of the data.

2. Look up an IP address. For this exercise, the IP value can be any routable address:

```
mmdblookup --file /etc/hapee-3.1/GeoIP2-City.mmdb --ip 40.121.152.233
```

This returns a JSON document, as shown below. Use the document's structure to find a property to use with the `maxmind-lookup` converter. For example, to have HAProxy Enterprise look up the English language city name for a client's IP address, use the keys city, names, and en as represented in the JSON returned from `mmdblookup`:

```
{
   "city":
   {
      "geoname_id": 4792307 <uint32>
      "names":
      {
         "en": "Washington" <utf8_string>
      }
   }
  // data continues...
}
```

Alternatively, use the mmdbinspect utility:

```
mmdbinspect -db /etc/hapee-3.1/GeoIP2-City.mmdb 40.121.152.233
```

Configure the MaxMind module

The module adds the following `global` directives:

maxmind-load

Required. Loads a MaxMind database.

Syntax:

```
maxmind-load [mlock_max <number>] <db_type> <db_path> [<db_type> <db_path>]>*]
```

Argument	Description
`mlock_max <number>`	Affects unprivileged HAProxy Enterprise invocations and sets the maximum locked memory in bytes. If you run HAProxy Enterprise in Docker, add the `IPC_LOCK` Linux capability when calling `docker run` in addition to setting `mlock_max`. `sudo docker run --cap-add IPC_LOCK ...`
`<db_type>`	It can be one of the following: ASN, ANONYMOUS, CITY, CONNTYPE, COUNTRY, DOMAIN, ISP, ANY.
`<db_path>`	Sets a path and filename from which to load the database type of `<db_type>`.

maxmind-cache-size

Sets the size of the LRU cache used for lookups, defaults to 0. Setting to 0 disables cache.

Syntax:

```
maxmind-cache-size <number>
```

Use converters to perform database lookups

Use the converter `maxmind-lookup` to perform a lookup in the database that returns the values of the specified property. Properties in the MaxMind database are stored hierarchically. For example, you can find the name of a city in English at `city > names > en`.

Syntax:

```
maxmind-lookup(<db_type>,<prop-level1>[,<prop-level2>*])
```

In this example we set HTTP request headers that contain CITY and ASN properties based on client's source IP address.

```
frontend www
  bind :80
  mode http
  http-request add-header X-CityName  %[src,maxmind-lookup("CITY","city","names","en")]
  http-request add-header X-ISOCode   %[src,maxmind-lookup("CITY","country","iso_code")]
  http-request add-header X-ASN       %[src,maxmind-lookup("ASN","autonomous_system_number")]
  http-request add-header X-ASNOrg    %[src,maxmind-lookup("ASN","autonomous_system_organization")]
```

Update the database during runtime

Use the MaxMind Update feature to keep the contents of the geolocation database current. This allows you to keep multiple HAProxy Enterprise nodes synced with the latest data.

1. Install a web server of your choice and host the database file(s) at a URL where HAProxy Enterprise can access. For example, host the files at `http://192.168.122.1:8000/GeoIP2-City.mmdb` and `http://192.168.122.1:8000/GeoIP2-ASN.mmdb`.

2. Add the following lines to the `global` section of your configuration file, where the URL hosts an updated version of the file. Be sure to specify the port number where your file is hosted, for example, `8000` as in the example below.

   ```
   global
     # ... other global settings
     maxmind-update url CITY http://192.168.122.1:8000/GeoIP2-City.mmdb  url ASN http://192.168.122.1:8000/GeoIP2-
   ASN.mmdb  delay 24h  checksum  log
   ```

With this configuration, HAProxy Enterprise downloads the database every 24 hours and displays a message in the logs when it succeeds or if it encountered errors during the update.

maxmind-update

The `maxmind-update` directive enables updating the database over HTTP from a specified URL. You can specify multiple database types and their respective URLs. If there are multiple database types specified, they will download sequentially with a delay between each download. Updating a database with a newer version invalidates any cached lookups (if caching is used), unless you enable `checksum` and the new and old database contents are identical.

The directive supports the following syntax:

```
maxmind-update url <db_type> <db_url>
  [url <db_type> <db_url>]*
  [delay <number>]
  [timeout <number>]
  [retries <number>]
  [checksum]
  [log]
  [dontlog-normal]
```

Argument	Description
`<db_type>`	**Required**. Can be any of the following: ASN, ANONYMOUS, CITY, CONNTYPE, COUNTRY, DOMAIN, ISP, ANY. You must have already used the <db_type> with the `maxmind-load` global keyword.
`<db_url>`	**Required**. URL to connect to and download a new version of the database of type `<db_type>`.
`delay <time value>`	Specifies the delay between each attempt to download a new database version.
`timeout <time value>`	Specifies the HTTP connect timeout for attempts to download a new database version. The default value is in milliseconds, but you can specify any other unit if you add it as a suffix to the number (default: 5 milliseconds).
`retries <number>`	Specifies the number of retries to download a new database version. If unspecified, the global retries value applies (default: 3).
`checksum`	If present, it specifies to use a SHA1 checksum to verify that a newly downloaded database is identical to the current one. If they are identical, then a live-reload of the database does not take place, thereby preserving cache contents (if using caching).
`log`	Specifies whether to log operation errors.
`dontlog-normal`	Deactivates logging for successful updates.

Runtime API

It is possible to manage the MaxMind module via the Runtime API. For general information on using the runtime API, see HAProxy Enterprise Runtime API.

The supported commands are listed below.

maxmind-update cache disable

Disable the MaxMind LRU lookup cache.

If the MaxMind cache is enabled, each lookup is cached so that subsequent requests are loaded from the cache instead of the MaxMind database. The MaxMind module will perform all lookups in its database when the cache is disabled.

Below, we disable the cache:

```
echo "maxmind-update cache disable" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
MaxMindDb: cache disabled
```

maxmind-update cache enable

Enable the MaxMind LRU lookup cache.

If the MaxMind cache is enabled, each lookup is cached so that subsequent requests are loaded from the cache instead of the MaxMind database.

Below, we enable the cache:

```
echo "maxmind-update cache enable" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
MaxMindDb: cache enabled
```

maxmind-update cache invalidate

Invalidate the MaxMind lookup cache.

If the MaxMind cache is enabled, each lookup is cached so that subsequent requests are loaded from the cache instead of the MaxMind database. Invalidating the cache marks all entries in the cache invalid. After invalidating the cache, the MaxMind module will perform lookups in its database as it rebuilds its cache for subsequent requests.

Below, we invalidate the cache:

```
echo "maxmind-update cache invalidate" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
MaxMindDb: LRU cache invalidated
```

maxmind-update disable

Disable the MaxMind database lookup engine. When disabled, the MaxMind module does not perform lookups.

Below, we disable MaxMind lookups:

```
echo "maxmind-update disable" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
MaxMind: data lookup disabled
```

maxmind-update enable

Enable the MaxMind database lookup engine.

The MaxMind database lookup engine is enabled automatically upon installation. You may need to re-enable it if you have manually disabled it.

Below, we enable MaxMind lookups:

```
echo "maxmind-update enable" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
MaxMind: data lookup enabled
```

maxmind-update update

Force an update of the MaxMind database.

The MaxMind update module will update the database based on the interval you specified in the configuration. You can also force an update that will run immediately.

In this example, we want to force an update of the database.

```
echo "maxmind-update force-update" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
MaxMindDb: forcing update now
```

You can check the status of the update using the `maxmind-update status` command. Note that while the update is processing, the status will show the progress.

```
echo "maxmind-update status" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
MaxMindDb module status
------------------------------------------------------------------------------
 initialized:             yes

 Database
   configuration:         /etc/hapee-2.7/hapee-lb.cfg:31
   enabled:               yes
   invalid:               no
   CITY:                /etc/hapee-2.7/GeoIP2-City.mmdb

 LRU cache
   enabled:               yes
   serial:                0 1
   usage/size:            0/200000

 Database update
   configuration:         /etc/hapee-2.7/hapee-lb.cfg:32
     CITY:                http://192.168.64.1/GeoIP2-City.mmdb
       period/next:       1d / 23h59m
       http status count: 0 0 0 1 0 / 0
   use cksum/hash/mod:    yes / yes / no
   period/delay:          1d / 5s 10s 5s
   reload/retry count:    0 0 0 / 1 0
   reload time:           <NEVER> / <NEVER> / <NEVER>
   currently updating:    GeoIP2-City.mmdb
     status/retry:        0 / 0
     data size:           0 / 0 (0.0%)
     dur/time left:       0.837s / 4s
```

maxmind-update show

Display the configuration of the MaxMind module.

You can dump the configuration for the MaxMind module using the `maxmind-update show` command. The output of this command includes information about each configured database file and its update status.

Below, we retrieve the status information for our configured databases:

```
echo "maxmind-update show" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
# maxmind-update configuration
# <db_type> url: <configured_url>
"CITY" url: http://192.168.64.1/GeoIP2-City.mmdb
# Next update is "CITY" in 23h42m
```

maxmind-update status

Display the status of the MaxMind module.

The MaxMind module maintains some statistics about its operations. The `maxmind-update status` command provides information about its configuration and cache.

The results of the `maxmind-update status` command show information about the current state of its cache, as well as information about updates to its database.

```
echo "maxmind-update status" | \
  sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
MaxMindDb module status
------------------------------------------------------------------------------
 initialized:             yes

 Database
   configuration:         /etc/hapee-2.7/hapee-lb.cfg:31
   enabled:               yes
   invalid:               no
    CITY:                /etc/hapee-2.7/GeoIP2-City.mmdb

 LRU cache
   enabled:               yes
   serial:                -1 -1
   usage/size:            0/200000

 Database update
   configuration:         /etc/hapee-2.7/hapee-lb.cfg:32
     CITY:                http://192.168.64.1:8000/GeoIP2-City.mmdb
   period/next:       1d / 23h58m
   http status count: 0 1 0 0 0 / 0
   use cksum/hash/mod:    no / no / no
   period/delay:          1d / 5s 10s 5s
   reload/retry count:    1 0 0 / 0 0
   reload time:           2023-08-09 10:13:21 / <NEVER> / <NEVER>
```

Global Profiling Engine

Overview of the Global Profiling Engine

This page applies to:

- HAProxy Enterprise - all versions

The Global Profiling Engine collects stick table data from all HAProxy Enterprise nodes in a cluster. It can perform calculations on both current and historical data before pushing the data back to the load balancers, giving them all a comprehensive and up-to-date view of client behavior across the cluster.

For example, in an active-active load balancer configuration where traffic is relayed to two or more load balancers in a round-robin rotation, the GPE ensures that each load balancer receives the sum of requests that a client has made, even when some of those requests were routed to the other load balancer.

It calculates statistics for current, real-time traffic, which you can use to take action based on an immediate threat, such as a web scraper. It also calculates historical statistics, which you can use to make decisions based on traffic trends from the past. For example, you can factor in the request rate at the 90th percentile from yesterday.

The Global Profiling Engine should be installed on a separate server from your HAProxy Enterprise nodes, from where it can collect and aggregate data from stick tables from across your load balancer cluster.

When would you use the profiling engine? The table below outlines various, common ways to set up your load-balancing cluster and explains which benefit from using the Global Profiling Engine:

Standalone load balancer

If you operate a single HAProxy Enterprise node, collecting data with stick tables can help improve security and observability by monitoring client activity in real time. However, **you do not need to use the Global Profiling Engine** because there aren't multiple load balancers collecting data that needs to be aggregated, unless you want to use its historical aggregations feature.

Active-standby cluster

In an active-standby cluster, you operate two HAProxy Enterprise nodes as peers, with one peer actively receiving traffic while the other is on standby. If you define a `peers` section in your configuration, then the stick tables of the active node are propagated to the passive node, where they overwrite existing values. Overwriting would be bad if both peers were receiving traffic and collecting data, but in an active-passive scenario, it's perfectly fine. The metrics stay accurate on all nodes. In this case, **you do not need the Global Profiling Engine**, which aggregates data from multiple, active peers, unless you want to use its historical aggregations feature.

Active-active cluster with Global Profiling Engine

In an active-active cluster, you operate two HAProxy Enterprise nodes as peers, with both receiving traffic at the same time. To ensure that stick table counters are accurate cluster-wide, **you need the Global Profiling Engine**, which runs on a separate server, collecting stick table data from both peers and aggregating it. Once aggregated, the data is sent back to the peers so that they can make decisions based on it.

Install the Global Profiling Engine on HAProxy Enterprise

This page applies to:

- HAProxy Enterprise - all versions

The following section gives detailed information on how to install the Global Profiling Engine.

Install steps

To install the Global Profiling Engine:

1. Provision a server on which you will install the Global Profiling Engine.
2. On the new server, add package repositories for your operating system.

Debian:

In a terminal window, set a variable named `key` to your HAProxy Enterprise license key.

```
key=<HAProxy Enterprise Key>
```

Use the `echo` and `tee` commands to add the HAProxy Technologies extras repository location into the file `/etc/apt/sources.list.d/haproxy-tech.list`, as shown below.

```
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/HAPEE-key-extras.asc] https://www.haproxy.com/download/hapee/key/${key}-plus/extras/debian-$(. /etc/os-release && echo "$VERSION_CODENAME")/amd64/ $(. /etc/os-release && echo "$VERSION_CODENAME") main" | sudo tee -a /etc/apt/sources.list.d/haproxy-tech.list
```

The packages that HAProxy Technologies provides are signed. To install them, import the public key:

```
sudo apt-get update
sudo apt-get install --yes apt-transport-https curl dirmngr gnupg-agent grep
sudo mkdir -p /etc/apt/keyrings
sudo curl -s -L "https://pks.haproxy.com/linux/enterprise/HAPEE-key-extras.asc" -o /etc/apt/keyrings/HAPEE-key-extras.asc
```

Ubuntu:

In a terminal window, set a variable named `key` to your HAProxy Enterprise license key.

```
key=<HAProxy Enterprise Key>
```

Use the `echo` and `tee` commands to add the HAProxy Technologies extras repository location into the file `/etc/apt/sources.list.d/haproxy-tech.list`, as shown below.

```
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/HAPEE-key-extras.asc] https://www.haproxy.com/download/hapee/key/${key}-plus/extras/ubuntu-$(. /etc/os-release && echo "$VERSION_CODENAME")/amd64/ $(. /etc/os-release && echo "$VERSION_CODENAME") main" | sudo tee -a /etc/apt/sources.list.d/haproxy-tech.list
```

The packages that HAProxy Technologies provides are signed. To install them, import the public key:

```
sudo apt-get update
sudo apt-get install --yes apt-transport-https curl dirmngr gnupg-agent grep
sudo mkdir -p /etc/apt/keyrings
sudo curl -s -L "https://pks.haproxy.com/linux/enterprise/HAPEE-key-extras.asc" -o /etc/apt/keyrings/HAPEE-key-extras.asc
```

RHEL:

Create a new file `/etc/yum.repos.d/haproxy-tech.repo` if it does not exist and add the contents below. Replace `<HAProxy Enterprise Key>` with the key you were given when you registered. Also replace `rhel-8` with your RHEL version number.

haproxy-tech.repo

```
[hapee-plus-extras]
name=hapee-plus-extras
enabled=1
baseurl=https://www.haproxy.com/download/hapee/key/<HAProxy Enterprise Key>-plus/extras/rhel-8/$basearch/bin/
gpgcheck=1
```

The packages that HAProxy Technologies provides are signed. To install them, import the public key:

```
rpm --import https://pks.haproxy.com/linux/enterprise/HAPEE-key-extras.asc
```

3. Install packages:

Debian:

Update the repository cache and install required dependencies:

```
sudo apt-get install --yes hapee-extras-gpe10
```

Ubuntu:

Update the repository cache and install required dependencies:

```
sudo apt-get install --yes hapee-extras-gpe10
```

RHEL:

```
yum makecache
yum install -y hapee-extras-gpe10
```

4. To start the Global Profiling Engine, run:

```
sudo systemctl enable hapee-extras-gpe
sudo systemctl start hapee-extras-gpe
```

Check your current version

To check which version of GPE you are running, call the `hapee-gpe` command with the `-v` flag:

```
/opt/hapee-extras/sbin/hapee-gpe -v
```

Manage the service

Start, stop and check the status of the Global Profiling Engine service by using the corresponding `systemctl` commands.

Start the service:

```
sudo systemctl start hapee-extras-gpe
```

Restart the service:

```
sudo systemctl restart hapee-extras-gpe
```

Show current status:

```
sudo systemctl status hapee-extras-gpe
```

Stop the service:

```
sudo systemctl stop hapee-extras-gpe
```

Configure real-time aggregation of stick table data

This page applies to:

- HAProxy Enterprise - all versions

The Global Profiling Engine collects real-time stick table data from all HAProxy Enterprise nodes in the cluster. It then aggregates that data and pushes it back to all of the nodes.

For example, if LoadBalancer1 receives two requests and LoadBalancer2 receives three requests, the Global Profiling Engine will sum those numbers to get a total of five, then push that to both LoadBalancer1 and LoadBalancer2. This is helpful for an active/active load balancer configuration wherein the nodes need to share client request information to understand activity across the cluster accurately.

The aggregated data doesn't overwrite the data on the load balancer nodes. Instead, it is pushed to secondary stick tables with, for example, a suffix of `.agg`. You would use a fetch method to retrieve the aggregated data and perform an action, like rate limiting.

Stick table data is transferred between the HAProxy Enterprise servers and the Global Profiling Engine server using the peers protocol, a protocol created specifically for this purpose. You must configure which servers should participate, both on the Global Profiling Engine server and on each HAProxy Enterprise node.

Configure HAProxy Enterprise nodes

An HAProxy Enterprise node must be configured to share its stick table data with the Global Profiling Engine server. Once aggregated, the profiling engine sends the data back to each node, which is stored in a new stick table.

Follow these steps on each load balancer:

1. Edit the file `/etc/hapee-3.1/hapee-lb.cfg`.

   Add a `peers` section.

hapee-lb.cfg

```
global
  [...]
  # By setting this, you are directing HAProxy Enterprise to use the server line
  # that specifies this name as the local node.
  localpeer enterprise1
  [...]

peers mypeers

  # This is the address and port that the load balancer will receive aggregated data from the GPE server
  bind 0.0.0.0:10000

  # The local HAProxy Enterprise node hostname defined by one of the following:
  # 1) the value provided when the load balancer process is started with the -L argument
  # 2) the localpeer name from the global section of the load balancer configuration (suggested method)
  # 3) the hostname as returned by the system hostname command (default)
  server enterprise1

  # The Global Profiling Engine
  # If you run GPE on the same server, use a different port here
  server gpe 192.168.50.40:10000

  # stick tables definitions
  table request_rates     type ip size 100k expire 30s store http_req_rate(10s)
  table request_rates.agg type ip size 100k expire 30s store http_req_rate(10s)
```

Inside it:

- Define a `bind` line to set the IP address and port at which this node should receive data back from the Global Profiling Engine server. In this example, the `bind` directive listens on all IP addresses at port 10000 and receives aggregated data.
- Define a `server` line for the current load balancer server. The server name value is important because it must match the name you set in the Global Profiling Engine server's configuration for the corresponding `peer` line. The hostname may be one of the following, in order of precedence:
  - the value provided with the `-L` argument specified on the command line used to start the load balancer process
  - the `localpeer` name specified in the `global` section of the load balancer configuration (this method is used in this example)
  - the host name returned by the system `hostname` command. This is the default, but we recommend using one of the other two methods

  In this example, the local HAProxy Enterprise node is listed with only its hostname, `enterprise1`. It isn't necessary to specify its IP address and port.
- Define a `server` line for the Global Profiling Engine server. Set its IP address and port. The name you set here is also important; it must match the corresponding `peer` line in the Global Profiling Engine server's configuration.
- Define stick tables. For each one, add a duplicate line where the table name has the suffix `.agg`. In this example, the non-aggregated stick table `request_rates` will store current HTTP request rates. The stick tables record the rate at which clients make requests over 10 seconds. We clear out stale records after 30 seconds by setting the `expire` parameter on the stick table. The `type` parameter sets the key for the table, which in this case is an IP address. The stick table `request_rates.agg` receives its data from the Global Profiling Engine. Its suffix, `.agg`, will match the profiling engine's configuration.

localpeer definition

If you receive an error for your load balancer configuration that looks like the following after specifying the name for your load balancer on the `server` line:

```
[WARNING]  (6125) : config : Removing incomplete section 'peers mypeers' (no peer named 'enterprise1')
```

Specify your hostname value for `localpeer` in your `global` section:

hapee-lb.cfg

```
global
  localpeer enterprise1
```

This global setting is required in cases where your hostname (retrieved using the system `hostname` command) is different from your desired peer name. Be sure to update your GPE configuration to use the name you specify as the `localpeer` name, as well as update your load balancer configuration to reference that name on the `server` line for your load balancer in your `peers` section.

2. Add directives to your frontend, backend, or listen sections that populate the non-aggregated stick tables with data.

   Below, the `http-request track-sc0` line adds request rate information for each client that connects to the load balancer, using the client's source IP address (`src`) as the key in the stick table.

   hapee-lb.cfg

   ```
   frontend fe_main
     bind :80
     default_backend webservers

     # add records to the stick table using the client's
     # source IP address as the table key
     http-request track-sc0 src table mypeers/request_rates
   ```

3. Add directives that read the aggregated data returned from the Global Profiling Engine server. That data is stored in the table with the suffix `.agg`.

   Below, the `http-request deny` line rejects clients that have a request rate greater than 1000. The client's request rate is an aggregate amount calculated from all active load balancers. Note that this line reads data from the `request_rates.agg` table.

hapee-lb.cfg

```
# perform actions like rate limiting
http-request deny deny_status 429 if { sc_http_req_rate(0,mypeers/request_rates.agg) gt 1000 }
```

4. Restart HAProxy Enterprise.

```
sudo systemctl restart hapee-3.1-lb
```

Configure the Global Profiling Engine

The Global Profiling Engine server collects stick table data from HAProxy Enterprise load balancers in your cluster, but you must set which load balancers will be allowed to participate by listing them in the configuration file.

Use dynamic configuration

> **This section applies to:**
>
> - HAProxy Enterprise - GPE version 1.0 (`hapee-extras-gpe10` package or newer)

Load balancers can connect to the GPE server without you adding them explicitly to the GPE configuration file. Include the `dynamic-peers` directive in either:

- the `aggregations` section to enable it for only that section.
- the `global` section to enable it for multiple `aggregations` sections.

For example, to set `dynamic-peers` in an `aggregations` section:

1. On the Global Profiling Engine server, edit the file `/etc/hapee-extras/hapee-gpe-stktagg.cfg`. Add an `aggregations` section that includes `dynamic-peers`:

hapee-gpe-stktagg.cfg

```
global
  # Enables the Global Profiling Engine API
  stats socket /var/run/hapee-extras/gpe-api.sock

aggregations data
  # set how to map non-aggregated to aggregated stick tables
  from any to .agg

  # the profiling engine listens at this address
  peer gpe 0.0.0.0:10000 local

  # register load balancer on the fly
  dynamic-peers
```

2. Optional: If you have multiple `aggregations` sections, which is useful for serving multiple clusters of load balancers, then you can simplify your setup by setting a `bind` directive in the `global` section instead of setting a `peer` line with the `local` keyword in each `aggregations` section. This sets the address at which to listen for incoming stick table data.

hapee-gpe-stktagg.cfg

```
global
  # Enables the Global Profiling Engine API
  stats socket /var/run/hapee-extras/gpe-api.sock

  bind 0.0.0.0:10000

aggregations data
  # set how to map non-aggregated to aggregated stick tables
  from any to .agg

  # register load balancer on the fly
  dynamic-peers
```

If you do this, then on the load balancers the `peer` line for the GPE server must use the same name as the `aggregations` section. Here, the name is `data`.

hapee-lb.cfg

```
peers mypeers
  peer data 192.168.56.26:10000
  peer enterprise1 192.168.50.41:10000
  peer enterprise2 192.168.50.42:10000
  ...
```

3. Restart the Global Profiling Engine service:

```
sudo systemctl restart hapee-extras-gpe
```

> **Restart after modifying tables**
>
> Whenever you make a change to your stick table definitions, such as to add new counters to the `store` argument or change the `expire` argument, restart the Global Profiling Engine service for it to take effect.

Use static configuration

You can specify the IP address of each load balancer that is allowed to connect:

1. On the Global Profiling Engine server, edit the file `/etc/hapee-extras/hapee-gpe-stktagg.cfg`.

   In the `aggregations` section, add a `peer` line for the Global Profiling Engine itself and for each HAProxy Enterprise node. Each peer's name (e.g. `enterprise1`) should match the name you set in the HAProxy Enterprise configuration, since that is how the profiling engine validates the peer.

> **Peer names**
>
> Be sure that the peer names you specify in the GPE server's configuration match exactly the names you specified in your load balancer configuration. For example, the following load balancer configuration sets the load balancer's `localpeer` name to `enterprise1` and we reference this name again in the `peers` section:
>
> **hapee-lb.cfg**
>
> ```
> global
>   localpeer enterprise1
>   ...
>
> peers mypeers
>   bind 0.0.0.0:10000
>   server enterprise1
>   ...
> ```
>
> As such, it must appear in the GPE server's configuration as `enterprise1` in order for GPE to make connection to the load balancer.

hapee-gpe-stktagg.cfg

```
global
  # Enables the Global Profiling Engine API
  stats socket /var/run/hapee-extras/gpe-api.sock

aggregations data
  # set how to map non-aggregated to aggregated stick tables
  from any to .agg

  # the profiling engine listens at this address
  peer gpe 0.0.0.0:10000 local

  # the load balancers listen at these addresses
  peer enterprise1 192.168.50.41:10000
  peer enterprise2 192.168.50.42:10000
```

In this example:

- The Global Profiling Engine API provides a programmable API, which listens at the socket `/var/run/hapee-extras/gpe-api.sock`. The `stats socket` directive enables a CLI that lets you view data that the aggregator has stored.
- In the `aggregations` section, the `from` line defines how non-aggregated stick tables map to aggregated stick tables, and what the suffix for the aggregated stick tables should be. The keyword `any` means that any stick table found will be aggregated. Aggregated data is pushed to tables with the same name, but ending with the suffix `.agg`. In the example, the engine expects stick tables to be named like `request_rates` and it will push aggregated data to `request_rates.agg`.

  You can also use a more specific mapping. In the example below, the engine expects stick tables to be named like `request_rates.nonagg` and it will push aggregated data to `request_rates.agg`. Stick tables without the `.nonagg` suffix will be ignored.

  hapee-gpe-stktagg.cfg

  ```
  from .nonagg to .agg
  ```

- The `peer` line with the `local` argument indicates the local GPE server.
- HAProxy Enterprise `peer` lines must use the same name you set on the `server` line in the HAProxy Enterprise configuration (e.g. `enterprise1`), and they must specify the IP addresses and ports where the load balancers are receiving aggregated data.

2. Restart the Global Profiling Engine service:

```
sudo systemctl restart hapee-extras-gpe
```

Restart after modifying tables

Whenever you make a change to your stick table definitions, such as to add new counters to the `store` argument or change the `expire` argument, restart the Global Profiling Engine service for it to take effect.

Verify your setup

Check that the Global Profiling Engine and load balancers are setup correctly by utilizing their APIs.

1. On the load balancer, call the Runtime API function `show peers` to check that the Global Profiling Engine is listed and that its `last_status` is `ESTA` (established):

   Below, the `show peers` command lists connected peers:

   ```
   echo "show peers" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock | head -2
   ```

   output

   ```
   0x5651d4a03010: [07/Jul/2021:17:02:30] id=mypeers disabled=0 flags=0x2213 resync_timeout=<PAST> task_calls=92
   0x5651d4a06540: id=gpe(remote,active) addr=192.168.50.40:10000 last_status=ESTA last_hdshk=2m17s
   ```

2. Call the Runtime API function `show table` to see data in non-aggregated and aggregated stick tables.

   Below, we view data in the stick table named `request_rates.agg`:

   ```
   echo "show table mypeers/request_rates.agg" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
   ```

   output

   ```
   # table: mypeers/request_rates.agg, type: ip, size:102400, used:1
   0x7fc0e401fb80: key=192.168.50.1 use=0 exp=28056 http_req_rate(10000)=5
   ```

3. On the Global Profiling Engine server, call the `show aggrs` function to see load balancers that are registered as peers. A `state` of `0x7` means a successful connection. If you see a state of `0xffffffff`, that means that a connection was not successful. Often, this is caused by the peer names not matching between the Global Profiling Engine's configuration and the HAProxy Enterprise configuration.

   Below, the `show aggrs` command shows that the peer named `enterprise1` has connected:

```
echo "show aggrs" | sudo socat stdio /var/run/hapee-extras/gpe-api.sock
```

output

```
aggregations data
  peer 'enterprise1'(0) sync_ok: 1 accept: 1(last: 6080) connect: 1(last: 16086) state: 0x7 sync_state: 0x3
         sync_req_cnt: 0 sync_fin_cnt: 0 sync_cfm_cnt: 0
```

Optional: Bind outgoing connections to an interface

If the server where you are running the Global Profiling Engine has multiple network interfaces, you can configure the engine to bind to a specific one for outgoing data sent to HAProxy Enterprise servers.

To bind outgoing connections to a specific address, use the `source` directive in the `global` section.

IPv4 examples

hapee-gpe-stktagg.cfg

```
global
  source 126.123.10.12:12345
```

The port is optional. It defaults to 0 for random ports.

hapee-gpe-stktagg.cfg

```
global
  source 126.123.10.12
```

IPv6 examples

hapee-gpe-stktagg.cfg

```
global
  source [2607:f8b0:400e:c00::ef]:12345
```

The port is optional. It defaults to 0 for random ports.

hapee-gpe-stktagg.cfg

```
global
  source [2607:f8b0:400e:c00::ef]
```

GPE with session persistence

> **This section applies to:**
>
> - HAProxy Enterprise 2.9r1

A special situation arises when you want to use the Global Profiling Engine to sync session persistence data across load balancers. Session persistence uses a stick table to track which server a client was routed to initially and from then on continues to route that client to the same server.

1. For example, consider the `backend` below that enables session persistence, but without GPE:

   hapee-lb.cfg

   ```
   backend servers
     stick-table type ip  size 1m  expire 30m
     stick on src
     server s1 192.168.0.10:80 check
     server s2 192.168.0.11:80 check
   ```

2. We need to make the following changes to the `backend`:

   - Remove the `stick-table` line.
   - Make the `stick on` directive reference the `sessions` table in the `peers` section named `mypeers`.
   - By default, each load balancer can arrange the servers differently. However, we need to ensure consistent server IDs across all load balancers, so we use the `id` argument to set the IDs explicitly.

   hapee-lb.cfg

   ```
   backend servers
     stick on src table mypeers/sessions
     server s1 192.168.0.10:80 check id 1
     server s2 192.168.0.11:80 check id 2
   ```

3. Move the `stick-table` definition to the `peers` section:

hapee-lb.cfg

```
peers mypeers
  bind 0.0.0.0:10000
  server enterprise1
  server gpe 192.168.50.40:10000

  table sessions     type ip  size 1m  expire 30m store server_id,server_key
  table sessions.agg type ip  size 1m  expire 30m store server_id,server_key write-to mypeers/sessions
```

In this example:

- We have moved the stick table to the `peers` section and named it `sessions`. You must set its `store` argument to `server_id,server_key`.
- A table named `sessions.agg` syncs session persistence data to GPE, which then syncs it to all load balancers. The aggregate table must set the `write-to` argument so that the data is written back to the `sessions` table. The `write-to` parameter allows remote load balancers to update the local `sessions` table with session persistence data.

4. Restart HAProxy Enterprise.

```
sudo systemctl restart hapee-3.1-lb
```

5. Make this exact change on the other HAProxy Enterprise server, then restart it.

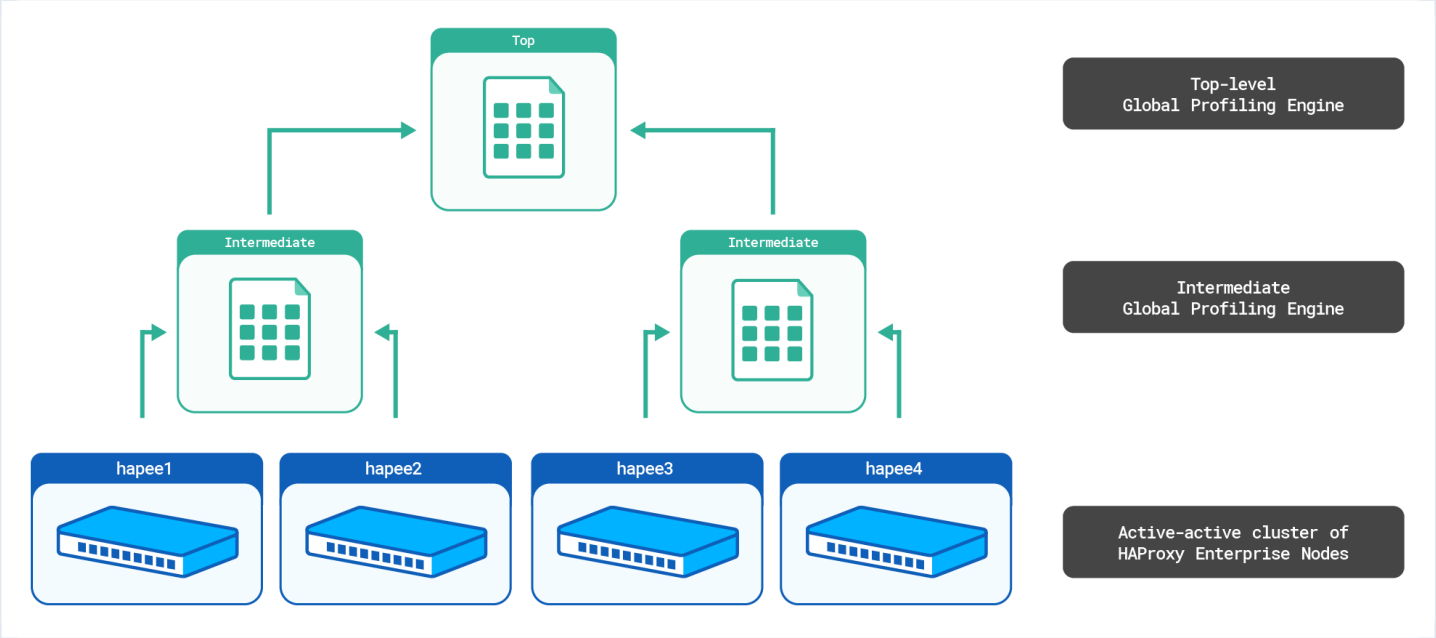
6. On the GPE server, restart the Global Profiling Engine service:

```
sudo systemctl restart hapee-extras-gpe
```

Multi-level setup

You can aggregate stick tables from other Global Profiling Engines, which allows you to aggregate stick tables across different data centers, for example.

We will consider the following setup:



The top-level aggr3 Global Profiling Engine will sum the counters from the intermediate aggr1 and aggr2 aggregate stick tables. It will then send the top-level aggregate stick table to all HAProxy Enterprise nodes.

You can also host multiple top-level servers for high availability. In that case, intermediate servers simply push their data to both. See below for details.

Configure the top-level Global Profiling Engine

Follow these steps on the server you wish to be the top-level Global Profiling Engine.

1. Edit the file `/etc/hapee-extras/hapee-gpe-stktagg.cfg`.

hapee-gpe-stktagg.cfg

```
global
  stats socket /var/run/hapee-extras/gpe-api.sock

aggregations toplevel
  from .intermediate to .agg
  peer top-gpe  0.0.0.0:10000 local
  peer intermediate-gpe1  192.168.56.111:10000 down
  peer intermediate-gpe2  192.168.56.112:10000 down
```

- The current server has the `local` keyword set on its `peer` line.
- In this example, two other Global Profiling Engine servers, intermediate-gpe1 and intermediate-gpe2, are listed with the `down` keyword, which means that they are one level down from the top.
- The top-level Global Profiling Engine will aggregate stick table data from the intermediate servers. Their stick tables should have the `.intermediate` suffix.
- The top-level Global Profiling Engine will push aggregated data back to the intermediate servers. The globally aggregated stick tables should have the `.agg` suffix.

Configure the intermediate Global Profiling Engines

Follow these steps on the servers you wish to be the intermediate-level Global Profiling Engines.

1. Edit the file `/etc/hapee-extras/hapee-gpe-stktagg.cfg`.

intermediate-gpe1

hapee-gpe-stktagg.cfg

```
global
  stats socket /var/run/hapee-extras/gpe-api.sock

aggregations myaggr
  from any to .intermediate
  forward .agg
  peer intermediate-gpe1  0.0.0.0:10000 local
  peer top-gpe  192.168.56.113:10000 up
  peer enterprise1 192.168.50.41:10000
  peer enterprise2 192.168.50.42:10000
```

intermediate-gpe2

hapee-gpe-stktagg.cfg

```
global
  stats socket /var/run/hapee-extras/gpe-api.sock

aggregations myaggr
  from any to .intermediate
  forward .agg
  peer intermediate-gpe2  0.0.0.0:10000 local
  peer top-gpe  192.168.56.113:10000 up
  peer enterprise3 192.168.50.51:10000
  peer enterprise4 192.168.50.52:10000
```

- The `from` line aggregates stick table data to tables with the suffix `.intermediate`. You can disable sending data to the top-level aggregators by adding `no-ascend` to this line. Disable sending aggregated data to the load balancers downstream from the intermediate aggregators by adding `no-feedback`.
- The current server has the `local` keyword set on its `peer` line.
- The upper-level Global Profiling Engine peer is denoted by the `up` keyword.
- Each intermediate Global Profiling Engine is aware of only the HAProxy Enterprise nodes it manages and of the top-level Global Profiling Engine.
- The intermediate-level Global Profiling Engines will aggregate stick table data from the HAProxy Enterprise servers.
- The `forward` line relays the top-level server's `.agg` stick tables to the HAProxy Enterprise servers.
- The intermediate-level Global Profiling Engines will push aggregated data back to the HAProxy Enterprise servers. The aggregated stick tables should have the `.agg` suffix.

Configure for high availability

To create a highly available setup, you can have multiple top-tier servers. In this configuration, the top-tier servers use the `group` parameter to link the intermediate servers together into a single entity. If either top-tier server goes down, the other still receives aggregated results from the intermediate group entity. If either intermediate server goes down, the other intermediate server continues to compute and aggregate results to send to the top-tier servers.

Info

Grouping the intermediate aggregators requires that the aggregators be active at the same time. Consequently, the high availability configuration imposes greater network traffic and CPU demands on the peers, whether serving as sending or receiving peers, and on the load balancer nodes.

1. The top-tier servers should each have the same configuration.

2. In the top-tier configurations, add the `group` parameter to the intermediate peers `peer` directives.

   top-tier servers

   Edit the file `/etc/hapee-extras/hapee-gpe-stktagg.cfg`.

   hapee-gpe-stktagg.cfg

   ```
   global
     stats socket /var/run/hapee-extras/gpe-api.sock

   aggregations toplevel
     from .intermediate to .agg
     peer top-gpe  0.0.0.0:10000 local
     peer intermediate-gpe1  192.168.56.111:10000 down  group 1
     peer intermediate-gpe2  192.168.56.112:10000 down  group 1
   ```

3. On the intermediate peers, add the entry for the additional top-level server.

   Edit the file `/etc/hapee-extras/hapee-gpe-stktagg.cfg`.

   intermediate-gpe1

   hapee-gpe-stktagg.cfg

   ```
   global
     stats socket /var/run/hapee-extras/gpe-api.sock

   aggregations myaggr
     from any to .intermediate
     forward .agg
     peer intermediate-gpe1  0.0.0.0:10000 local
     peer top-gpe1  192.168.56.113:10000 up
     peer top-gpe2  192.168.56.114:10000 up
     peer enterprise1 192.168.50.41:10000
     peer enterprise2 192.168.50.42:10000
   ```

See also

- For complete information on the peers section syntax and usage, see Peers configuration reference.
- To set the local instance's peer name, see localpeer reference.

Configure historical aggregation of stick table data

This page applies to:

- HAProxy Enterprise - all versions

In addition to aggregating stick table data from multiple HAProxy Enterprise nodes in real time and pushing that data back to each node, the profiling engine also stores historical data. For example, you can configure it to tell you what the average HTTP request rate was at the same time of day yesterday. Or, you can check what the average rate was at this same time a week ago, and adjust rate limiting to match.

Historical data allows you to perform dynamic decisions in your load balancer based on data from the past, such as to set rate limits that change depending on the hour.

Configure the Global Profiling Engine

Follow these steps to configure historical aggregation of stick table data.

1. On the Global Profiling Engine server, as shown for real-time aggregation, configure the list of peers in the `/etc/hapee-extras/hapee-gpe-stktagg.cfg` file.

   hapee-gpe-stktagg.cfg

   ```
   global
     stats socket /var/run/hapee-extras/gpe-api.sock

   aggregations data
      from any to .agg
      peer gpe 0.0.0.0:10000 local
      peer enterprise1 192.168.50.41:10000

      # list more 'peer' lines for other load balancers in the cluster
      # e.g. peer enterprise2 192.168.50.42:10000
   ```

2. Edit the file `/etc/hapee-extras/hapee-gpe.json` to configure data retention policies for storing historical data.

   Data is stored in buckets of time. For example, you might keep 12 1-minute buckets, 24 1-hour buckets, and 2 week-buckets, as shown below, which would allow you to compare a client's current request rate to the average request rate during the same hour yesterday, for example.

hapee-gpe.json

```
{
    "worker_thread_count": 4,
    "inter_worker_queue_size": 1024,
    "collector_queue_size": 64,
    "httpd_port": 9888,
    "datadir": "/var/cache/hapee-extras/hct_datadir",
    "default_stick_table_handling": 1,
    "prometheus_exporter": 1,
    "ignore_tables": [],
    "detail_tables": [],
    "aggregate_tables": [],
    "stat_retentions": [
        {
          "duration": 300,
          "retention": 12
        },
        {
          "duration": 3600,
          "retention": 24
        },
        {
          "duration": 86400,
          "retention": 14
        }
    ]
}
```

In this example:

- The `httpd_port` field sets the port on which to publish historical data, which HAProxy Enterprise servers poll for updates. Here, it hosts the data at port 9888. The default IP address is `0.0.0.0` and can be changed with the `httpd_addr` option.
- The `aggregate_tables`, `detail_tables`, and `ignore_tables` fields are all empty since we set `default_stick_table_handling` to `1` which will aggregate all tables.
- The `prometheus_exporter` field enables the generation of profiling engine data in Prometheus format from the profiling engine's `/metrics` endpoint.
- The `stat_retentions` section lists data retention policies. Each policy sets a `duration` in seconds, which is the size of the data bucket, and a `retention`, which is the number of buckets to keep. A bucket stores counters from your stick tables. For example, if your stick table tracks the HTTP request rate over 10 seconds, a 1-hour bucket might store many thousands of these 10-second request rate data points.

For each bucket, the server calculates statistics and serves them on the configured port.

3. Restart the Global Profiling Engine:

```
sudo systemctl restart hapee-extras-gpe
```

Configure HAProxy Enterprise

In this section, you will see examples of how to configure HAProxy Enterprise for historical aggregation.

Example: Use the Global Profiling Engine to enforce rate limits

One use case for historical aggregation is to compare a client's current request rates against the request rates over time and to then make rate-limiting decisions based on the current rate.

Configure each HAProxy Enterprise server to download and use the historical data.

1. Create an empty file at `/etc/hapee-3.1/historical.map`.

   Although an in-memory representation of this file will hold historical values received from the profiling engine, the file must exist on the filesystem when HAProxy Enterprise starts.

   HAProxy Enterprise updates a representation of this file in memory only. You will not see the contents of the file itself updated and it will remain empty, but you can see the in-memory values by calling the Runtime API `show map` method.

2. Install the Update module, which polls the profiling engine for new data to load into the map file.

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-update
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-update
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-update
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-update
```

3. Edit the file `/etc/hapee-3.1/hapee-lb.cfg`.

   In the `global` section of the file, add a `module-load` directive to load the Update module:

hapee-lb.cfg

```
global
  module-load hapee-lb-update.so
```

4. Configure the Update module to poll the profiling engine's `/aggs` endpoint for data by adding a `dynamic-update` section that contains an `update` directive.

   The `url` parameter should use the profiling engine's IP address.

hapee-lb.cfg

```
dynamic-update
  update id /etc/hapee-3.1/historical.map map url http://192.168.50.40:9888/aggs delay 3600s log
```

In this example:

- The `dynamic-update` section configures HAProxy Enterprise to poll the profiling engine for historical data updates.
- The `update` line's `id` parameter sets the local file to update (remember, this file will not be updated on disk, only in HAProxy Enterprise's runtime memory).
- The `map` parameter switches the Update module into map file mode.
- The `url` parameter specifies the IP and port of the profiling engine. It specifies the `/aggs` URL path.
- The `delay` parameter sets the polling interval to 3600 seconds. Since our smallest `stat_retentions` duration is 3600 seconds, we can poll GPE hourly.
- The `log` parameter enables logging to the HAProxy Enterprise access log.

5. As you would for real-time aggregation, add a `peers` section that lists the local node and the profiling engine on `server` lines.

   Here you will also define stick tables with their `.agg` clones.

hapee-lb.cfg

```
peers mypeers
  bind :10000

  # The local HAProxy Enterprise node hostname defined by one of the following:
  # 1) the value provided when the load balancer process is started with the -L argument
  # 2) the localpeer name from the global section of the load balancer configuration (suggested method)
  # 3) the hostname as returned by the system hostname command (default)
  server enterprise1

  # The Global Profiling Engine
  server gpe 192.168.50.40:10000

  # stick tables definitions
  table request_rates           type ip size 100k expire 30s store http_req_rate(10s)
  table request_rates.agg type ip size 100k expire 30s store http_req_rate(10s)
```

In this example:

- Define a `bind` line to set the IP address and port at which this node should receive data back from the Global Profiling Engine server. In this example, the `bind` directive listens on all IP addresses at port 10000 and receives aggregated data.
- Define a `server` line for the current load balancer server. The server name value is important because it must match the name you set in the Global Profiling Engine server's configuration for the corresponding `peer` line. The hostname may be one of the following, in order of precedence:
  - the value provided with the `-L` argument specified on the command line used to start the load balancer process
  - the `localpeer` name specified in the `global` section of the load balancer configuration (this method is used in this example)
  - the host name returned by the system `hostname` command. This is the default, but we recommend using one of the other two methods
- Define a `server` line for the Global Profiling Engine server. Set its IP address and port. The name you set here is also important. It must match the corresponding `peer` line in the Global Profiling Engine server's configuration.

6. Use map fetch methods in your `frontend` section to read information from the local map file and make traffic routing decisions.

   In the example below, we deny clients that have a request rate higher than the 99th percentile of requests from the same hour (3600 seconds) yesterday (86400 seconds ago).

   hapee-lb.cfg

```
frontend fe_main
  bind :80

  # add records to the stick table using the client's IP address as the table key
  http-request track-sc0 src table mypeers/request_rates

  # store the 99th percentile rate and the client's current rate in variables
  http-request set-var(req.rate_99percentile) str(/request_rates.http_req_rate.3600sec.86400sec_ago.99p),map(/etc/
hapee-3.1/historical.map,1000)
  http-request set-var(req.client_rate) sc_http_req_rate(0,mypeers/request_rates.agg)

  # set ACL expressions
  acl historical_rate_greater_than_zero var(req.rate_99percentile) -m int gt 0
  acl client_rate_exceeds_historical_rate var(req.rate_99percentile),sub(req.client_rate) -m int lt 0

  # deny the request if it exceeds the historical rate
  http-request deny deny_status 429 if historical_rate_greater_than_zero client_rate_exceeds_historical_rate

  default_backend webservers
```

In this example:

- The `http-request track-sc0` line adds the current client to the stick table, using their source IP address as the primary key.
- The `http-request set-var(req.rate_99percentile)` line reads the value of the `/request_rates.http_req_rate.3600sec.86400sec_ago.99p` statistic from the `historical.map` data. If that statistic does not exist or has no data (which happens if there was no traffic during that hour), a value of 1000 is used instead. See the Reference guide to learn how these statistics are named.
- The `http-request set-var(req.client_rate)` line retrieves the current client's request rate from the `mypeers/request_rates.agg` table, which uses real-time aggregation to collect data from all load balancers in your cluster.
- The `http-request deny` line rejects requests if the client's current request rate (aggregated across load balancers) exceeds the 99th percentile rate for all users from the same hour yesterday. (note: If the historical rate is zero, then it defaults to a value of 1000).

7. Restart HAProxy Enterprise.

```
sudo systemctl restart hapee-3.1-lb
```

Verify the setup. First, check that the HAProxy Enterprise admin logs show that the Update module is downloading the map file successfully. If there was an error, it will be written there. If everything worked, there will be no output (no errors).

Also, verify that data is being published by calling the `/aggs` URL with `curl` on the aggregation server. You will need to wait until the first bucket has been populated with data, though, which depends on the size of the bucket, before you will see data.

```
curl http://localhost:9888/aggs
```

output

```
/request_rates.http_req_rate.3600sec.3600sec_ago.cnt 13
/request_rates.http_req_rate.3600sec.3600sec_ago.sum 29
/request_rates.http_req_rate.3600sec.3600sec_ago.avg 0
/request_rates.http_req_rate.3600sec.3600sec_ago.per_sec_avg 0
/request_rates.http_req_rate.3600sec.3600sec_ago.burst_avg 2
/request_rates.http_req_rate.3600sec.3600sec_ago.min 1
/request_rates.http_req_rate.3600sec.3600sec_ago.max 7
/request_rates.http_req_rate.3600sec.3600sec_ago.50p 2
/request_rates.http_req_rate.3600sec.3600sec_ago.75p 3
/request_rates.http_req_rate.3600sec.3600sec_ago.90p 3
/request_rates.http_req_rate.3600sec.3600sec_ago.95p 3
/request_rates.http_req_rate.3600sec.3600sec_ago.99p 3
/request_rates.http_req_rate.3600sec.3600sec_ago.99.9p 3
```

You can also call the Runtime API's `show map` function to see the data stored in the map file.

Example: Use the Global Profiling Engine to calculate response time percentiles

You can use the Global Profiling Engine to track response time percentiles across your HAProxy Enterprise cluster. You can record these response time percentiles on some interval and then use the data for analysis.

Configure each HAProxy Enterprise server to download and use the historical data.

1. Create an empty file at `/etc/hapee-3.1/historical.map`.

   Although an in-memory representation of this file will hold historical values received from the profiling engine, the file must exist on the filesystem when HAProxy Enterprise starts. HAProxy Enterprise updates a representation of this file in memory only. You will not see the contents of the file itself updated and it will remain empty, but you can see the in-memory values by calling the Runtime API `show map` method.

2. Install the Update module, which polls the profiling engine for new data to load into the map file.

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-update
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-update
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-update
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-update
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-update
```

3. Edit the file `/etc/hapee-3.1/hapee-lb.cfg`.

   In the `global` section of the file, add a `module-load` directive to load the Update module:

hapee-lb.cfg

```
global
  module-load hapee-lb-update.so
```

4. Configure the Update module to poll the profiling engine's `/aggs` endpoint for data by adding a `dynamic-update` section that contains an `update` directive.

   The `url` parameter should use the profiling engine's IP address.

hapee-lb.cfg

```
dynamic-update
  update id /etc/hapee-3.1/historical.map map url http://192.168.50.40:9888/aggs delay 10s log
```

In this example:

- The `dynamic-update` section configures HAProxy Enterprise to poll the profiling engine for historical data updates.
- The `update` line's `id` parameter sets the local file to update (remember, this file will not be updated on disk, only in HAProxy Enterprise's runtime memory).
- The `map` parameter switches the Update module into map file mode.
- The `url` parameter specifies the IP and port of the profiling engine. It specifies the `/aggs` URL path.
- The `delay` parameter sets the polling interval to 10 seconds.
- The `log` parameter enables logging to the HAProxy Enterprise access log.

5. As you would for real-time aggregation, add a `peers` section that lists the local node and the profiling engine on `server` lines.

   Here you will also define stick tables with their `.agg` clones.

   hapee-lb.cfg

   ```
   peers mypeers
     bind :10000
     server enterprise1
     server gpe 192.168.50.40:10000
     table st_responsetime type string len 64 size 100k expire 1h store gpt0
     table st_responsetime.agg type string len 64 size 100k expire 1h store gpt0
   ```

   Be sure that the hostname of the HAProxy Enterprise node and the hostname of the Global Profiling Engine instance that you specify are the configured hostnames of those instances. Use the `hostname` command on each instance to retrieve the names.

   ```
   hostname
   ```

   output

   ```
   enterprise1
   ```

6. In your frontend, track the total response time of each request in the stick table mypeers/st_responsetime. We use the general purpose tag (gpt0) to store the response time value in the stick table. Each record uses a unique ID as its key, where the unique ID's format is a combination of the client's IP address, client's port, frontend IP address, frontend port, a timestamp, a request counter, and the process ID.

hapee-lb.cfg

```
frontend fe_main
  bind :80

  # generate a unique ID
  unique-id-format %{+X}o\ %ci:%cp_%fi:%fp_%Ts_%rt:%pid
  http-request set-var(txn.path) path
  # add records to the stick table using the unique ID as table key
  http-request track-sc0 unique-id table mypeers/st_responsetime

  # prepare and perform the calculation for response times
  http-response set-var-fmt(txn.response_time) %Tr
  http-response set-var-fmt(txn.connect_time) %Tc
  http-response set-var-fmt(txn.queue_time) %Tw
  http-response sc-set-gpt0(0) var(txn.response_time),add(txn.queue_time),add(txn.connect_time)

  # store the 99th percentile rate in variables
  http-request set-var(req.response_time_99percentile) str(/st_responsetime.gpt0.3600sec.3600sec_ago.99p),map(/etc/
hapee-3.1/historical.map,1000)

  default_backend webservers
```

7. Restart HAProxy Enterprise.

```
sudo systemctl restart hapee-3.1-lb
```

Verify the setup. First, check that the HAProxy Enterprise admin logs show that the Update module is downloading the map file successfully. If there was an error, it will be written there. If everything worked, there will be no output (no errors).

Also, verify that data is being published by calling the `/aggs` URL with `curl` on the aggregation server. You will need to wait until the first bucket has been populated with data, though, which depends on the size of the bucket, before you will see data.

```
curl http://localhost:9888/aggs
```

output

```
/st_responsetime.gpt0.3600sec.3600sec_ago.cnt 2
/st_responsetime.gpt0.3600sec.3600sec_ago.sum 24153
/st_responsetime.gpt0.3600sec.3600sec_ago.avg 12076
/st_responsetime.gpt0.3600sec.3600sec_ago.per_sec_avg 0
/st_responsetime.gpt0.3600sec.3600sec_ago.burst_avg 0
/st_responsetime.gpt0.3600sec.3600sec_ago.min 9769
/st_responsetime.gpt0.3600sec.3600sec_ago.max 14384
/st_responsetime.gpt0.3600sec.3600sec_ago.50p 9775
/st_responsetime.gpt0.3600sec.3600sec_ago.75p 14391
/st_responsetime.gpt0.3600sec.3600sec_ago.90p 14391
/st_responsetime.gpt0.3600sec.3600sec_ago.95p 14391
/st_responsetime.gpt0.3600sec.3600sec_ago.99p 14391
/st_responsetime.gpt0.3600sec.3600sec_ago.99.9p 14391
```

You can also call the Runtime API's `show map` function to see the data stored in the map file.

Configure logging for the Global Profiling Engine

This page applies to:

- HAProxy Enterprise - all versions

You have several options for logging Global Profiling Engine events.

- Write each syslog priority level to a separate file.
- Log to a UNIX domain socket (local syslog server).
- Log to a TCP/UDP inet socket (remote syslog server).
- (legacy) Redirect the stdout/stderr stream output to a file.

Write syslog priority levels to files

This section applies to:

- HAProxy Enterprise - GPE version 1.0 (`hapee-extras-gpe10` package or newer)

To write logged messages with different priority levels to a file:

1. Create the directory where you will save log files, such as `/var/log/haproxy-gpe/`.

```
sudo mkdir /var/log/haproxy-gpe/
```

2. Edit the file `/etc/hapee-extras/hapee-gpe.json`. Add `log_format`, `log_stderr`, and `log_files` fields:

hapee-gpe.json

```
{
  "worker_thread_count": 4,
  "inter_worker_queue_size": 1024,
  "collector_queue_size": 64,
  "httpd_port": 9888,
  "datadir": "/var/cache/hapee-extras/hct_datadir",
  "default_stick_table_handling": 1,
  "prometheus_exporter": 1,
  "ignore_tables": [],
  "detail_tables": [],
  "aggregate_tables": [],
  "stat_retentions": [
    {
      "duration": 300,
      "retention": 12
    },
    {
      "duration": 3600,
      "retention": 24
    },
    {
      "duration": 86400,
      "retention": 14
    }
  ],
  "log_format": "file",
  "log_stderr": "local3",
  "log_files": [
    { "priority": "all",     "path": "/var/log/haproxy-gpe/common.log"  },
    { "priority": "err",     "path": "/var/log/haproxy-gpe/err.log"     },
    { "priority": "warning", "path": "/var/log/haproxy-gpe/warning.log" }
  ]
}
```

where:

- `log_format` is `file`.
- `log_stderr` redirects stderr messages to the `local3` syslog facility. You can choose any facility value, but you must configure your local syslog service to handle these messages.
- `log_files` is an array of files to write to for each syslog priority. Use any of `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, `info`, or `debug`. Use `all` as a catch-all for all other priorities. If the file doesn't exist, it will be created automatically, which gives you a way to archive log files by changing the file's name, since the file will be recreated.

For more details, see Reference.

3. Restart the GPE service:

```
sudo systemctl restart hapee-extras-gpe
```

Log to UNIX domain socket

> **This section applies to:**
>
> - HAProxy Enterprise - GPE version 1.0 (`hapee-extras-gpe10` package or newer)

To write logged messages to a syslog UNIX domain socket on the GPE server:

1. Edit the file `/etc/hapee-extras/hapee-gpe.json`. Add `log_format`, `log_stderr`, and `log_socket` fields:

hapee-gpe.json

```
{
  "worker_thread_count": 4,
  "inter_worker_queue_size": 1024,
  "collector_queue_size": 64,
  "httpd_port": 9888,
  "datadir": "/var/cache/hapee-extras/hct_datadir",
  "default_stick_table_handling": 1,
  "prometheus_exporter": 1,
  "ignore_tables": [],
  "detail_tables": [],
  "aggregate_tables": [],
  "stat_retentions": [
    {
      "duration": 300,
      "retention": 12
    },
    {
      "duration": 3600,
      "retention": 24
    },
    {
      "duration": 86400,
      "retention": 14
    }
  ],
  "log_format": "rfc3164",
  "log_stderr": "local3",
  "log_socket": "/dev/log"
}
```

where:

- `log_format` is `rfc3164`.
- `log_stderr` redirects stderr messages to the `local3` syslog facility. You can choose any facility value, but you must configure your local syslog service to handle these messages.
- `log_socket` is the socket's file path.

For more details, see Reference.

The GPE server will send messages to the local syslog server with a predefined set of syslog facility codes that map to severity levels, which cannot be overridden. Configure your syslog server to expect these facility codes:

GPE Priority	Facility	Syslog Priority
emerg	local0	local0.debug
alert	local1	local1.debug
crit	local2	local2.debug
err	local3	local3.debug
warning	local4	local4.debug
notice	local5	local5.debug
info	local6	local6.debug
debug	local7	local7.debug

The exception is logging the `stderr` stream to syslog, where any syslog facility can be specified as output. The syslog priority used in that case is `.debug`.

2. Restart the GPE service:

```
sudo systemctl restart hapee-extras-gpe
```

Log to TCP/UDP inet socket

This section applies to:

- HAProxy Enterprise - GPE version 1.0 (`hapee-extras-gpe10` package or newer)

To write logged messages to a remote syslog server:

1. Edit the file `/etc/hapee-extras/hapee-gpe.json`. Add `log_format`, `log_stderr`, and `log_proto`, `log_nodename`, and `log_servname` fields:

hapee-gpe.json

```
{
  "worker_thread_count": 4,
  "inter_worker_queue_size": 1024,
  "collector_queue_size": 64,
  "httpd_port": 9888,
  "datadir": "/var/cache/hapee-extras/hct_datadir",
  "default_stick_table_handling": 1,
  "prometheus_exporter": 1,
  "ignore_tables": [],
  "detail_tables": [],
  "aggregate_tables": [],
  "stat_retentions": [
    {
      "duration": 300,
      "retention": 12
    },
    {
      "duration": 3600,
      "retention": 24
    },
    {
      "duration": 86400,
      "retention": 14
    }
  ],
  "log_format": "rfc5424",
  "log_stderr": "local3",
  "log_proto": "tcp",
  "log_nodename": "192.168.56.10",
  "log_servname": "514"
}
```

where:

- `log_format` is `rfc5424`.
- `log_stderr` redirects stderr messages to the `local3` syslog facility. You can choose any facility value, but you must configure your target syslog service to handle these messages.
- `log_proto` is either `tcp` or `udp`.
- `log_nodename` is the remote syslog server's address.
- `log_servname` is the remote syslog server's service name or a decimal port number like `514`.

For more details, see Reference.

The GPE server will send messages to the remote syslog server with a predefined set of syslog facility codes that map to severity levels, which cannot be overridden. Configure your syslog server to expect these facility codes:

GPE Priority	Facility	Syslog Priority
emerg	local0	local0.debug
alert	local1	local1.debug
crit	local2	local2.debug
err	local3	local3.debug
warning	local4	local4.debug
notice	local5	local5.debug
info	local6	local6.debug
debug	local7	local7.debug

The exception is logging the `stderr` stream to syslog, where any syslog facility can be specified as output. The syslog priority used in that case is `.debug`.

2. Restart the GPE service:

```
sudo systemctl restart hapee-extras-gpe
```

Redirect stdout/stderr to a file

> **Legacy notice**
>
> This is the legacy way to log to a file. It's better to write syslog priority levels to files.

To redirect all logged messages from stdout / stderr to a file:

1. Edit the file `/etc/default/hapee-extras-gpe` and set the `-l` startup argument to a file path. Prefix the path with either `A:` for append or `W:` for truncate. For details, see Startup arguments.

hapee-extras-gpe

```
GPE_OPTIONS="-f /etc/hapee-extras/hapee-gpe-stktagg.cfg -c /etc/hapee-extras/hapee-gpe.json -l A:/var/log/haproxy-gpe/log-file.txt"
```

2. Restart the GPE service:

```
sudo systemctl restart hapee-extras-gpe
```

Reference

This section describes the available log setting fields.

Field	Type	Description
log_appname	string	rfc5424 `APP-NAME` field (default: `-`).
log_files	array	Each array item specifies: • priority: `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, `info`, `debug`. Use `all` to indicate others not otherwise handled. • Full path of the log file, such as `/var/log/gpe-debug.txt`.
log_format	string	Set to `rfc3164` to use the rfc3164 protocol, `rfc5424` to use the rfc5424 protocol, or to the path of a file.
log_hostname	string	rfc3164/rfc5424 `HOSTNAME` field (default: hostname where the program is running).
log_msgid	string	rfc5424 `MSGID` field (default: `-`).
log_nodename	string	Node name or an IP address of the syslog server.
log_procid	string	rfc5424 `PROCID` field (default: `-`, process ID if `PID` is entered for the value).
log_proto	string	Protocol used when connecting to the syslog server: `tcp` or `udp`.
log_servname	string	A service name or TCP/UDP port number of the syslog server. Default: `514`.
log_socket	string	Full path of the UNIX domain syslog socket.
log_stderr	string	Syslog facility to use for log messages sent from `stderr` stream (default: not used).
log_rec_len	number	The size of the circular buffer for log messages (the number of syslog messages) (2 - 8192, default: 256).
log_rec_size	number	The maximum size of a syslog message (in bytes) (256 - 2048, default: 512).

See also

- For a description of syslog usage and history, see RFC 3164: The BSD syslog protocol at RFC.org.
- For the official syslog protocol, which prescribes how it shall be used, see RFC 5424: The syslog protocol at RFC.org.
- For a shorter article on syslog, which includes some historical background as well as an overview of the protocol, see Syslog at Wikipedia.org.

Configure Prometheus metrics for the Global Profiling Engine

This page applies to:

- Global Profiling Engine 1.0 and newer

The Global Profiling Engine supports Prometheus metrics that reveal characteristics of your stick table data and the health of the GPE service. You could import this data into dashboarding software, such as Grafana, to see trends over time.

Enable metrics

To enable Prometheus metrics:

1. On the Global Profiling Engine server, edit the file `/etc/hapee-extras/hapee-gpe.json`.
2. Set the `prometheus_exporter` field to one of the following values:

Value	Description
0	Disable metrics. (default)
1	Enable only general metrics.
2	Enable only health metrics.
3	Enable both general and health metrics.

In this example, we enable both general and health metrics:

hapee-gpe.json

```
{
    "worker_thread_count": 4,
    "inter_worker_queue_size": 1024,
    "collector_queue_size": 64,
    "httpd_port": 9888,
    "datadir": "/var/cache/hapee-extras/hct_datadir",
    "default_stick_table_handling": 1,
    "prometheus_exporter": 3,
    ...
```

The metrics are available at the port set by `httpd_port` at `/metrics` for general metrics and `/health` for health metrics.

Metrics reference

This section describes the available metrics.

General metrics

General metrics record the count, sum, average, minimum, maximum, and percentiles of your stick table data.

Metric	Type	Description
gpe_metrics_cnt	gauge	Count.
gpe_metrics_sum	gauge	Sum.
gpe_metrics_avg	gauge	Average.
gpe_metrics_per_sec_avg	gauge	Per second average.
gpe_metrics_burst_avg	gauge	Burst average.
gpe_metrics_min	gauge	Minimum.
gpe_metrics_max	gauge	Maximum.
gpe_metrics_50p	gauge	Histogram 50 percentile.
gpe_metrics_75p	gauge	Histogram 75 percentile.
gpe_metrics_90p	gauge	Histogram 90 percentile.
gpe_metrics_95p	gauge	Histogram 95 percentile.
gpe_metrics_99p	gauge	Histogram 99 percentile.
gpe_metrics_99_9p	gauge	Histogram 99.9 percentile.

Health metrics

Health metrics reveal the health of your Global Profiling Engine service.

Metric	Type	Description
gpe_coll_q_usage	gauge	Usage of the collector queues (%).
gpe_dropped_since_start	counter	Number of messages dropped since start.
gpe_open_fds	gauge	Open file descriptors.
gpe_raw_q_usage	gauge	Usage of the RAW worker queues (%).
gpe_resident_memory_btyes	gauge	Resident memory size in bytes.
gpe_stat_q_usage	gauge	Usage of the STAT worker queues (%).
gpe_stktagg_connections	gauge	Number of connections (stktagg).
gpe_stktagg_tables	counter	Number of stkt tables (stktagg).
gpe_stktagg_updates_since_start	counter	Number of update messages received (stktagg) since start.
gpe_total_cpu_time	gauge	Total user and system CPU time spent in seconds.
gpe_uptime	gauge	Uptime in seconds.
gpe_virtual_memory_bytes	gauge	Virtual memory size in bytes.
gpe_writes_since_start	counter	Number of messages written to worker queues since start.

Global Profiling Engine reference

This page applies to:

- HAProxy Enterprise - all versions

This section describes all configuration options for the Global Profiling Engine.

Startup arguments

The Global Profiling Engine supports the following startup arguments, which you can set in the file `/etc/default/hapee-extras-gpe`:

Argument	Description
`-f <stktagg_config_file>`	Sets the path to the GPE peers configuration file.
`-c <gpe_config_file>`	Sets the path to the GPE settings configuration file.
`-d`	Enables debug mode.
`-l [<flag>:]<log_file>`	Enables logging to a file and sets the path to the target file. Supported flags are: `A`: append `W`: truncate
`-v`	Displays the version.

GPE peers configuration

The following fields can be set in the `/etc/hapee-extras/hapee-gpe-stktagg.cfg` file:

global section

The `global` section supports the following fields:

Field	Description
`bind`	**Available since GPE 1.0** Adds a listener to which aggregations sections can be attached. This allows GPE to listen for several `aggregations` sections on the same IP/port. You can set multiple `bind` lines, and more than one can serve the same `aggregations` section if they use the same `use_aggrs` values. See the `bind` syntax section below.
`cpu-map`	This has the same meaning as the `cpu-map` directive for HAProxy. It configures the CPU affinity of the GPE processes.
`dynamic-peers`	**Available since GPE 1.0** Enables the GPE to accept up to 64 concurrent connections from peers that are not defined in the configuration file by `aggregations` sections. This limit of 64 concurrents peers includes the ones set up by the configuration file.
`hash-table`	**Available since GPE 1.0** Configures global hash-table settings that are common for all stick tables managed by the process.
`source`	Binds an IP/port for outgoing connections. Set it as `source <ipv4 or ipv6>[:<port>]`.
`stats socket`	Creates a listener with which you can interact through a TCP or UNIX domain socket with the aggregator at runtime.
`trash-batch`	Sets the number of incoming update messages on a table to perform a lookup of expired entries to trash. A maximum of the same number of entries is trashed. Default: 10000.

The `bind` line has the following syntax:

```
bind <ip:port> [ssl] [crt <path>] [ca-file <path] [verify none | optional | required] [use_aggrs (all | <aggregation ID>[,aggregation ID>...)]
```

where:

Argument	Description
`use_aggrs`	This option must be followed by the special value `all` or by a list of `aggregations` section identifiers separated by commas. If some sections are not defined in the configuration file, they will be ignored. The `all` special value may be used to specify that you want the listener to accept incoming connections for any of the aggregation configurations defined by `aggregations` sections. This is also the default value when `use_aggrs` is not specified. **NOTE**: As a listener configured on a `bind` line has no peer name, a peer configured on the load balancer side to connect to such a listener must set a peer with the `aggregations` section identifier as the peer name.

The `hash-table` line has the following syntax:

```
hash-table [load-factor <lf>] [low <low>] [high <high>] [steps <steps>]
```

where:

Argument	Description
`load-factor <lf>`	Sets the optimal load factor that the GPE will try its best to enforce on the hash-table. Load factor is the ratio of the current number of entries in the table divided by the number of buckets. When load factor is reached, the GPE will automatically try to grow (and rehash) the hash-table on the fly. • It expects values between 1 and 255. • Default: 3. The default value is known to offer a good performance/memory ratio, but if memory is not a concern and performance should prevail, setting this value to 2 could have a noticeable impact. • In some setups setting load-factor to 1 may also be relevant.
`low <low>`	Sets the pow2 (power of 2) exponent that will be used to compute the number of buckets. • It expects values between 0 and 31. • All hash-tables will be initialized with this value, which means it will directly affect the memory footprint for all stick-tables handled by the GPE. A large value means higher performance (less resizing involved), and a lower value means a smaller upfront memory cost. • Default: 12. The default value means the hash-table will start with 4096 buckets (pow2(12) = 4096).
`high <high>`	Sets the pow2 (power of 2) exponent that will be used to compute the buckets upper limit (maximum number of buckets that the hash-table may allocate upon resizes). • It expects values between 0 and 31. • Default: 20. The default value means the hash-table may allocate up to 1m buckets (pow2(20) = 1048576) upon resize.
`steps <steps>`	Takes a list of intermediary pow2 exponents (separated by coma) that will be used for hash-table resizing. Values outside of `<low>...<high>` range will be ignored: if no valid step is specified between `<low>` and `<high>`, then the hash-table will directly jump from `<low>` to `<high>` pow2 upon resize. Set `<step>` to `all` to use all possible pow2 exponents between `<low>` and `<high>`. • Default: 14,16,18,19,20.

aggregations section

You can have one or more `aggregations` sections, where the following fields are supported:

Field	Description
`dynamic-peers`	**Available since GPE 1.0** This field has exactly the same meaning as `dynamic-peers` in the `global` section. It enables the dynamic peers feature. But contrary to the option for `global` section, the one for `aggregations` sections limits the feature to that `aggregations` section.
`forward`	This field supports the syntax `forward <suffix1,suffix2...>`. The stick tables with those suffixes are considered to be forwarded from upward to downward servers. Note that updates to one of those tables coming from a down server will be ignored.
`from`	Gives some information to the GPE about how to name the destination stick tables (the aggregated stick-tables). You can set multiple `from` lines to define multiple source-to-destination mappings. It accepts two forms of syntax: • `from any to <suffix>` • `from <suffix1,suffix2,...> to <suffix> [accept-no-suffix] [no-feedback] [no-ascend] [global-exp]` A suffix string must have `.` as the first character. Specify multiple suffixes by separating them with commas with no spaces between. Use `any` to aggregate all stick tables. Or specify which tables to aggregate by their suffix, which requires that you add suffixes to your stick table names in the load balancer configuration. • If `accept-no-suffix` is specified, then aggregate tables that have no suffix at all in addition to those with suffixes specified in the list. Be careful when using the `any` input suffix or `accept-no-suffix` keyword with multiple `from` lines, since the first matching `from` line is used, and `any` or `accept-no-suffix` will cause the `from` line to always match. Therefore, they should be placed on the last `from` rule. • If `no-feedback` is specified, then only "up" peers will receive aggregation results. • If `no-ascend` is set, "up" peers won't receive aggregation results. You can combine it with `no-feedback` to ensure the aggregator is the only host that can access the aggregation results. • If `global-exp` is specified, then all connected peers for a given table entry will have their expire tracker refreshed each time one of the peers pushes an update to the entry.
`peer`	The `peer` lines support the syntax: • `peer <peer name> <ip:port> ['local' or 'up' or 'down'] [group <group_id>] [ssl] [crt <path>] [ca-file <path>] ['verify none' or 'verify optional' or 'verify required']` They are made of two mandatory settings, their name and IP/port, followed by optional, exclusive keywords `local`, `up` or `down`, and possibly the optional `group` setting for multilevel setups. • `ssl` enables TLS/SSL. • `crt` is the certificate to present to the peer. • `ca-file` is the file containing the trusted CAs in PEM format. • `verify` has three possible values: `none` means that verify is disabled; `optional` means that a client certificate is requested but verify is performed only if the peer provide a certificate in response; `required` means that the peer certificate is mandatory in the response and the verification is always performed.

Field	Description
	Note that there can only be at most 64 peers handled by an `aggregations` section regardless of the fact if they are dynamic or declared inside an `aggregations` section. So, there can be at most 64 `peer` lines, the local peer being not included. If there is no listener configured by a `bind` line in the `global` section to handle the `aggregations` section, one peer must be local and followed by the `local` keyword.

Historical stats configuration

The following fields can be set in the `/etc/hapee-extras/hapee-gpe.json` file:

Field	Description
`aggregate_tables`	An array of stick table names that should be processed. One set of aggregates will be created for each stick table as a whole.
`collector_queue_size`	The size of the message queue between workers and the collector. It must be greater than 2 and a power of 2. Default: `64`.
`datadir`	The directory in which to store historical data files.
`default_stick_table_handling`	Indicates how the server should process stick tables that are not listed in the `ignore_tables`, `detail_tables`, or `aggregate_tables` arrays. Values: `0` = ignore, `1` = aggregate, `2` = detailed processing (Experimental).
`default_toplist_table_handling`	**Available since GPE 1.0** Indicates whether to enable the generation of toplist statistics for tables not included in `enable_toplist_tables`. Enabled when `1`, disabled when `0`. Default: `0`.
`detail_tables` (Experimental)	An array of stick table names that should be processed. One set of aggregates will be created for every value in the stick table.
`disable_toplist_tables`	**Available since GPE 1.0** A list of stick tables to not generate toplists for.
`enable_toplist_tables`	**Available since GPE 1.0** A list of stick tables to generate toplists for.
`httpd_addr`	**Available since GPE 1.0** The IP address on which to publish historical statistics data. Can be IPv4 or IPv6, with or without brackets. IPv4 wildcard is `*`. Default: `0.0.0.0`.
`httpd_port`	The TCP port on which to publish historical statistics data. Default: `9888`.
`ignore_tables`	An array of stick table names that should be skipped during processing.
`inter_worker_queue_size`	The size of the message queue that handles communication between workers. It must be greater than 2 and a power of 2. Default: `1024`.
`prometheus_exporter`	**Available since GPE 1.0** Generate profiling engine data in Prometheus format at the profiling engine's `/metrics` and `/health` endpoints. It accepts these values: • 0 = disable • 1 = enable general metrics • 2 = enable health metrics • 3 = enable general and health metrics Default: `0` (disable).
`stat_retentions`	An array of data retention policies. Each policy should have: `duration`: an integer value in seconds indicating the size of the bucket (i.e. time period) to aggregate data for. `retention`: the number of buckets to keep.
`toplist_element_count`	**Available since GPE 1.0** The number of elements in a toplist. Default: `10`. Max: `50`.

Field	Description
`worker_thread_count`	The number of worker threads to start. Default: `2`.

Historical statistics reference

Calling the `/aggs` endpoint on port 9888 returns a list of available statistics. For example, if you set the following retention policy in the `stat_retentions` field:

```
// 24 1-hour buckets
"duration": 3600,
"retention": 24
```

The `/aggs` endpoint would return data for each bucket. Each bucket contains one hour of data (3600-seconds), representing one of the hours during the last 24 hours. For example, the following statistics are recorded for the hour that happened one hour ago:

```
/request_rates.http_req_rate.3600sec.3600sec_ago.cnt 362
/request_rates.http_req_rate.3600sec.3600sec_ago.sum 3547
/request_rates.http_req_rate.3600sec.3600sec_ago.avg 10
/request_rates.http_req_rate.3600sec.3600sec_ago.per_sec_avg 0
/request_rates.http_req_rate.3600sec.3600sec_ago.burst_avg 10
/request_rates.http_req_rate.3600sec.3600sec_ago.min 8
/request_rates.http_req_rate.3600sec.3600sec_ago.max 11
/request_rates.http_req_rate.3600sec.3600sec_ago.50p 9
/request_rates.http_req_rate.3600sec.3600sec_ago.75p 9
/request_rates.http_req_rate.3600sec.3600sec_ago.90p 9
/request_rates.http_req_rate.3600sec.3600sec_ago.95p 9
/request_rates.http_req_rate.3600sec.3600sec_ago.99p 9
/request_rates.http_req_rate.3600sec.3600sec_ago.99.9p 9
```

Each line is a key and value. The key has this format:

`/name-of-stick-table . name-of-counter . bucket-duration . time since bucket occurred . statistic`

For example:

```
/request_rates.http_req_rate.3600sec.86400sec_ago.99p
```

Some lines have a negative number in them:

```
/request_rates.http_req_rate.3600sec.-3600sec_ago.cnt
```

This indicates a sliding time window that has a begin and end time that changes at a regular interval (i.e. an hour ago from now, or more realistically, at the next time the smallest bucket is calculated). In contrast, the following metric would be updated only at the top of every hour:

```
/request_rates.http_req_rate.3600sec.3600sec_ago.cnt
```

The table below describes each statistic:

Field	Description
`cnt`	The count of data points of the counter (e.g. HTTP rate limit) recorded in the bucket.
`sum`	The sum of all data point values in the bucket.
`avg`	An average of all data points in the bucket that preserves the time period of the stick table counter, which makes it easy to work with when comparing it to current request rates in HAProxy Enterprise; the sum of all data points (e.g. 3547) is multiplied by the stick table counter period (e.g. 10 for `http_req_rate(10s)`), then divided by the duration of the bucket (e.g. 3600).
`persec_avg`	An average of all data points in the bucket, converted to a 1-second average (discards the period of the stick table counter).
`burst_avg`	A traditional, mathematical average; the sum of all data points is divided by the count.
`min`	The minimum data point value in the bucket.
`max`	The maximum data point value in the bucket.
`50p`	The 50th percentile.
`75p`	The 75th percentile.
`90p`	The 90th percentile.
`95p`	The 95th percentile.
`99p`	The 99th percentile.
`99.9p`	The 99.9th percentile.

Troubleshoot the Global Profiling Engine

This page applies to:

- HAProxy Enterprise - all versions

This section describes troubleshooting steps for the Global Profiling Engine.

Troubleshooting: peer connections

If calling the `/aggs` URL with `curl` on the aggregation server does not produce any data even after adequate time has passed for the first bucket to populate, it may be the case that the Global Profiling Engine does not have connection to its configured HAProxy Enterprise peers.

To verify peer connections:

1. Use the `show peers` Runtime API command on the HAProxy Enterprise server:

```
echo "show peers" | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

output

```
0xaaaad88dbc50: [06/Sep/2023:10:07:50] id=mypeers disabled=0 flags=0x6213 resync_timeout=<PAST> task_calls=166688
  0xaaaad88e2750: id=gpe(remote,active) addr=192.168.64.50:10000 last_status=ESTA last_hdshk=0s
        reconnect=4s heartbeat=2s confirm=0 tx_hbt=10896 rx_hbt=0 no_hbt=10882 new_conn=10894 proto_err=0 coll=0
        flags=0x0 appctx:0xaaaad8d064b0 st0=7 st1=0 task_calls=4 state=EST
        shared tables:
          0xaaaad88fdd40 local_id=2 remote_id=2049 flags=0x0 remote_data=0x2
              last_acked=0 last_pushed=24 last_get=0 teaching_origin=24 update=24
              table:0xaaaad88e3860 id=mypeers/st_responsetime update=24 localupdate=24 commitupdate=24 refcnt=1
        Dictionary cache not dumped (use "show peers dict")
          0xaaaad88fdaf0 local_id=1 remote_id=2050 flags=0x0 remote_data=0x2
              last_acked=0 last_pushed=0 last_get=0 teaching_origin=0 update=0
              table:0xaaaad88e3bc0 id=mypeers/st_responsetime.agg update=17 localupdate=0 commitupdate=0 refcnt=1
        Dictionary cache not dumped (use "show peers dict")
  0xaaaad88e16d0: id=hapee(local,inactive) addr=0.0.0.0:10000 last_status=NONE last_hdshk=<NEVER>
        reconnect=<NEVER> heartbeat=<NEVER> confirm=0 tx_hbt=0 rx_hbt=0 no_hbt=0 new_conn=0 proto_err=0 coll=0
        flags=0x0
        shared tables:
          0xaaaad88fde10 local_id=2 remote_id=0 flags=0x0 remote_data=0x0
              last_acked=0 last_pushed=0 last_get=0 teaching_origin=0 update=0
              table:0xaaaad88e3860 id=mypeers/st_responsetime update=24 localupdate=24 commitupdate=24 refcnt=1
        Dictionary cache not dumped (use "show peers dict")
          0xaaaad88fdbc0 local_id=1 remote_id=0 flags=0x0 remote_data=0x0
              last_acked=0 last_pushed=0 last_get=0 teaching_origin=0 update=0
              table:0xaaaad88e3bc0 id=mypeers/st_responsetime.agg update=17 localupdate=0 commitupdate=0 refcnt=1
        Dictionary cache not dumped (use "show peers dict")
```

There should be an entry for the Global Profiling engine, as well as all connected HAProxy Enterprise peers. If no data is returned or if peers are missing, there may be an issue with the configuration.

2. If entries are missing, check both the `peers` section of `/etc/hapee-3.1/hapee-lb.cfg` and the peers entries in the Global Profiling Engine configuration file `/etc/hapee-extras/hapee-gpe-stktagg.cfg`. Check the following:

   - The hostname for the Global Profiling Engine server should be its configured hostname. Use the `hostname` command to retrieve its name.

     ```
     hostname
     ```

     output

     ```
     gpe
     ```

   - In each of the configuration files, `/etc/hapee-3.1/hapee-lb.cfg`, for the load balancer configuration, and `/etc/hapee-extras/hapee-gpe-stktagg.cfg`, for the Global Profiling Engine configuration, verify that the hostnames for your load balancer servers are identical everywhere they are referenced.

     The hostname value you provide for the load balancers may be one of three things:

     - the value provided with the `-L` argument specified on the command line used to start the load balancer process
     - the `localpeer` name specified in the global section of the load balancer configuration
     - the host name returned by the system `hostname` command.

     Be sure that this value is the one you specify for as the hostname of the load balancer on the `server` line in the `peers` section of your load balancer configuration and on the corresponding `peer` entry in the `aggregations` section of the GPE configuration.

Global server load balancing

This page applies to:

- HAProxy Enterprise - all versions

HAProxy Enterprise can serve as an authoritative Domain Name System (DNS) server in a limited capacity, specifically for implementing global server load balancing (GSLB). This lets you respond to DNS queries with the IP address(es) assigned to a datacenter that is the best match for the end user, such as the one that is geographically closest to them. Or, you can configure DNS to return the address of a secondary datacenter if the primary becomes inaccessible.

By providing GSLB, you can fulfill the following use cases:

- DNS round-robin: Distributes traffic between all datacenters in multiple locations.
- Failover: Send all traffic to a primary datacenter by returning its IP address(es) in DNS responses, but direct traffic to a secondary datacenter if the primary becomes inaccessible.
- Geolocation-based DNS: GSLB enhances functionalities of the DNS naming system by distributing network traffic across servers located in multiple locations. It can detect users' locations and route traffic to the nearest datacenter to lower latency.

The load balancer continuously monitors the health of your datacenter IP addresses so that it can remove them from the DNS responses if they become unavailable. It reroutes the traffic to another available datacenter by changing DNS records dynamically.

How global server load balancing works

First, consider how DNS typically works. DNS servers translate human-readable domain names (e.g. www.example.com) to numeric IP addresses (e.g. 10.10.0.5).

1. A client's web browser queries the DNS server to get the IP address of a website.
2. The DNS server returns an IP addresses.
3. The browser connects to the website through its IP address.

Global server load balancing offers a DNS server a smarter way to choose which IP address it should return. It can take into account where the client is located in the world and the health of each datacenter before selecting the IP address to return in a DNS response. This allows it to send a client to the best match.

1. A client's web browser queries the DNS server to get the IP address of a website.

2. HAProxy Enterprise, acting as the DNS server returns an IP address, but one based on the geographic IP location of the client and/or the health of the datacenter.
3. The client gets the best possible user experience by connecting to the website through the IP address of the datacenter that is the best match for them.

HAProxy Enterprise polls the servers to make sure they remain responsive. If they stop responding, then the affected IP addresses will be removed from the list of valid responses HAProxy Enterprise will return to clients.

One caveat: GSLB uses DNS to route clients, and DNS responses are often cached. If a datacenter becomes unavailable, clients will continue to use the cached IP address returned in the original DNS response until the cached response expires. However, it remains an effective strategy overall for distributing traffic across datacenters.

Install the GSLB module

1. Install the GSLB package according to your platform:

Apt:

```
sudo apt-get install hapee-extras-gslb
```

Yum:

```
sudo yum install hapee-extras-gslb
```

Zypper:

```
sudo zypper install hapee-extras-gslb
```

Pkg:

```
sudo pkg install hapee-extras-gslb
```

2. Optional: By default, the GSLB service listens for DNS queries at the addresses `0.0.0.0:53` and `127.0.0.1:153`. To change this, edit the configuration file:
   - On Debian/Ubuntu, `/etc/default/hapee-extras-gslb`
   - On Alma/Oracle/Redhat/Rocky, `/etc/sysconfig/hapee-extras-gslb`

Change the `GSLB_LISTEN` directive to set the IP addresses and ports at which the GSLB service should listen for DNS queries.

hapee-extras-gslb

```
# Options for hapee-extras-gslb.

GSLB_LISTEN="0.0.0.0:53 127.0.0.1:153"

GSLB_CONFIGFILE="/etc/hapee-extras/hapee-gslb.conf"

GSLB_CONFIGPATH="/var/run/hapee-extras/gslb"

GSLB_RUNPATH="/var/lib/gslb"
```

3. Edit your configuration file. You can copy over the example template to get started, or use it as a reference.

```
sudo cp /etc/hapee-extras/hapee-gslb-example.conf /etc/hapee-extras/hapee-gslb.conf
```

4. Save your configuration, then enable and start the GSLB service.

```
sudo systemctl enable hapee-extras-gslb
sudo systemctl start hapee-extras-gslb
```

Scenarios

In the following sections, we describe how to configure the GSLB service for several scenarios.

DNS round-robin load-balancing

You can make the GSLB service return several IP addresses from healthy datacenters in a round-robin weighted fashion.

1. Edit `/etc/hapee-extras/hapee-gslb.conf`.
   - Replace the domain `example.com` with your domain name and record entries.
   - Add a new `list` record to the `zone` section. A `record` of type `list` is a dynamic record followed by list of space-separated `answer-list` names.
   - Create `answer-list` sections that set IP addresses to return in a round-robin rotation. Optionally, add different weights to the `answer-record` lines.

hapee-gslb.conf

```
zone example.com
  ttl 10
  # ORIGIN records
  record @ SOA ns1.example.com hostmaster.example.com 1 86400 3600 3600 3600 60
  record @ NS  ns1.example.com
  record @ ttl 3600 MX 100 mail1.example.com

  # Static records
  record ns1   ttl 20 A 10.0.0.1
  record mail1 ttl 20 A 10.0.0.2

  # Dynamic records
  record www   ttl 20 list dc1

answer-list dc1
  method single-rr
  option httpchk
  http-check connect
  http-check send uri /health.html hdr host www.example.com
  answer-record srv1 20.0.0.1 weight 10
  answer-record srv2 20.0.0.2 weight 20
```

The `answer-list` section syntax is as follows:

Directive	Description
method	Set the `single-rr` parameter.
option	Specify `httpchk` to monitor the health of servers.
http-check	Set any relevant health check parameters.
answer-record	Enter any number of `answer-record` directives along with the corresponding IP addresses. Weights determine how often a particular IP address will be returned, with higher weights being chosen more often. By specifying `method single-rr`, HAProxy Enterprise alternates which IP for a datacenter it sends to clients in order to distribute traffic across all servers. In that case, the odds of a server's IP being returned is server weight / sum of all server weights.

2. Save your configuration and then restart the GSLB service:

```
sudo systemctl restart hapee-extras-gslb
```

Geolocation-based load balancing

You can deliver content to users based on their geographic location. HAProxy Enterprise with global server load balancing enabled returns the IP address from the closest healthy datacenter or server. To use this feature, you will need to download a GeoIP database from MaxMind.

For example, you can:

- comply with regulations governing the location of data storage.
- reduce latency.
- deliver content that is tailored to users' country and native language.

To enable geolocation-based load balancing:

1. Create your account through the MaxMind website and download the GeoIP databases.

2. What is a geolocation database?

   You can store GeoIP database files, typically in a specific format like MaxMind's GeoIP2 or GeoLite2. GeoIP data is information about the geographical location of IP addresses. This data is used in GSLB to determine the optimal routing of client requests based on their geographic location. These database files contain mappings between IP addresses and their corresponding geographical information, such as country, region, city, and latitude/longitude coordinates.

   When a client makes a request to the GSLB system, the system can analyze the client's IP address and consult the GeoIP database to determine the client's location. Based on this information, the GSLB system can make intelligent routing decisions to direct the client's request to the most appropriate server or data center that can serve the request efficiently and optimize network performance.

3. Create a directory on the load balancer to store the geolocation databases. Copy the MaxMind GeoIP files to the directory. For example, City and ISP data: `/data/GeoIP2-City.mmdb`, `/data/GeoIP2-ISP.mmdb`.

4. Edit `/etc/hapee-extras/hapee-gslb.conf`.

   - Replace the domain `example.com` with your domain name and record entries.
   - Add a new `map` record to the `zone` section. A `record` of type `map` is a dynamic record followed by a `geoip-map` name.
   - Create `geoip-map` sections that set `answer-list` sections to use depending on the client's location.
   - Create `answer-list` sections. Below, the `answer-list` for DC1 contains a list of IP addresses for the datacenter in Europe, while the DC2 `answer-list` contains a list of IP addresses for the datacenter in North America.

hapee-gslb.conf

```
zone example.com
  ttl 84600
  record @ ttl 900  SOA   ns1 hostmaster 1 7200 30M 3D 900
  record @          NS    ns1.example.com.
  record ns1        A     203.0.113.1   # nameserver: Load balancer IP address
  record alias      CNAME www
  record www        map   mymap

geoip-map mymap
  location-base /data/geoip/GeoLite2-City.mmdb
  location EU/FR/Paris DC1 DC2
  location NA/US/Chicago DC2 DC1
  network 198.51.100.0/24 DC1 DC2
  network 203.0.113.0/24 DC2 DC1

answer-list DC1
  up_threshold 0.5
  method single-rr
  option tcpchk fall 10 rise 10
  tcp-check connect port 80
  answer-record srv1 198.51.100.1 weight 20
  answer-record srv2 198.51.100.2 weight 20
  answer-record srv3 198.51.100.3 weight 10
  answer-record srv4 2001:db8::4001 weight 20
  answer-record srv5 2001:db8::4002 weight 20
  answer-record srv6 2001:db8::4003 weight 10

answer-list DC2
  up_threshold 0.5
  method single-rr
  option httpchk
  http-check connect
  http-check send uri /health.html hdr host www.example.com
  http-check expect status 200,301,302
  answer-record srv1 203.0.113.10  weight 20
  answer-record srv2 203.0.113.11  weight 20
  answer-record srv3 203.0.113.12  weight 10
```

The `geoip-map` section syntax is as follows:

Directive	Description	Example
location-base	Absolute path to the geolocation database. You can supply several geolocation database names separated by spaces.	location-base /data/geoip/GeoLite2-City.mmdb
location	The first parameter is a hierarchical path to a geographic region in the order of the continent code, a country ISO code, then more specific regions like state and city name. Refer to the MaxMind reference guide and ISO-3166 for these codes. Note that GSLB will search deeper into the hierarchy if a match is not found at the current layer. For example, you could specify country and city name, but omit the state name between them. The second parameter is a space-separated list of `answer-list` section names (e.g. DC2). GSLB directs client requests sent from this location to the first healthy datacenter in the list.	location NA/US/NY DC2
network	As an alternative to using `location`, which uses geolocation data to choose the datacenter, you can also specify a client IP range. Set a subnet value in CIDR notation followed by an ordered list of datacenters (separated by spaces). The second parameter is a space-separated list of `answer-list` section names (e.g. DC2). GSLB directs client requests sent from this subnet to the first healthy datacenter in the list.	network 198.51.100.0/24 DC1

About the `answer-list` sections:

- GSLB will send DNS responses based on the location of the client. It will only send either IPv4 or IPv6 addresses, depending on the type of IP addresses the client requests.
- Weights determine how often a particular IP address will be returned, with higher weights being chosen more often. The weight values apply only to the IPv4 or IPv6 pool of IP addresses. In the example, the IPV4 pool of servers and the IPv6 pool of servers have their own total weight sums.
- By specifying `method single-rr`, HAProxy Enterprise alternates which IP for a datacenter it sends to clients in order to distribute traffic across all servers. In that case, the odds of a server's IP being returned is server weight / sum of all server weights.
- You can set `method multi-rr` to return multiple IP addresses to the client. In that case, the odds of a server's IP being returned is server weight / max weight value.
- The `up_threshold` directive determines the percentage of servers that must be up. Otherwise, traffic is routed to a different datacenter altogether.

5. Save your configuration and then restart the GSLB service:

```
sudo systemctl restart hapee-extras-gslb
```

Datacenter failover

While you can use geolocation-based load balancing to route traffic to the datacenter nearest to the client, you can also use HAProxy Enterprise for basic failover and failback between datacenters without the geolocation component. If a critical resource fails and service is disrupted, traffic will be automatically redirected to healthy datacenters. This minimizes impact and avoids manual intervention.

1. Edit `/etc/hapee-extras/hapee-gslb.conf`.

   - Replace the domain `example.com` with your domain name and record entries.
   - Add a new `list` record to the `zone` section. A `record` of type `list` is a dynamic record followed by list of space-separated `answer-list` names. Essentially, you are specifying datacenters in order of preference, with fallback datacenters following primary datacenters.
   - Create `answer-list` sections that match the names you listed on the `list` record in the `zone` section.

   Below, the list record `www` enumerates two `answer-list` sections, DC1 and DC2, where DC1 is the primary datacenter and DC2 is the fallback. You could list additional fallback datacenters too.

hapee-gslb.conf

```
zone example.com
  ttl 10

  # ORIGIN records
  record @ SOA ns1.example.com hostmaster.example.com 1 86400 3600 3600 3600 60
  record @ NS  ns1.example.com
  record @ ttl 3600 MX 100  mail1.example.com

  # static records
  record ns1   ttl 10 A 10.0.0.1
  record mail1 ttl 10 A 10.0.0.2

  # dynamic records - DC1 is primary, DC2 is a fallback
  record www ttl 30 list DC1 DC2

answer-list DC1
  up_threshold 1
  method multi-up
  option httpchk
  http-check connect
  http-check send uri /health.html hdr host www.example.com
  http-check expect status 200,301,302
  answer-record srv1 20.0.0.1

answer-list DC2
  up_threshold 1
  method multi-up
  option httpchk
  http-check connect
  http-check send uri /health.html hdr host www.example.com
  answer-record srv1 30.0.0.1
```

The `answer-list` section syntax is as follows:

Directive	Description
up_threshold	Determines the percentage of servers that must be up. Otherwise, traffic is routed to a different datacenter altogether. A threshold of 1 means that all servers in an `answer-list` must be healthy for the corresponding datacenter to be regarded as active. If you specify 0.1, then 10% of the total weighted number of servers must be up.
method	Determines which IP addresses to return. As with geolocation-based load balancing, you can return a single IP in a round-robin rotation (`single-rr`), multiple addresses in a round-robin rotation (`method multi-rr`), all servers that are up (`method multi-up`), or all servers, even if they are down (`method multi-all`).
option	Specify `httpchk` to monitor the health of servers. If the servers are HAProxy Enterprise load balancers, you can use monitor URI as the health check endpoint.
http-check	Set any relevant health check parameters.
answer-record	Enter any number of `answer-record` directives, which denote IP addresses assigned to the datacenter.

2. Save your configuration and then restart the GSLB service:

```
sudo systemctl restart hapee-extras-gslb
```

Testing

You can use the `dig` command to test implementation. For example, if your domain is `example.com` you can use `dig www.example.com @127.0.0.1 -p 153` to test.

```
; <<>> DiG 9.10.6 <<>> A @127.0.0.1 -p 153 example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4343
;; flags: qr aa rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            139   IN      A       20.0.0.1

;; Query time: 0 msec
;; SERVER: 127.0.0.1#153(127.0.0.1)
;; WHEN: Tue Jul 03 23:27:15 UTC 2023
;; MSG SIZE  rcvd: 74
```

Logs and status

View logs related to GSLB:

```
sudo journalctl -u hapee-extras-gslb -b -0 -f
```

output

```
-- Logs begin at Mon 2023-06-26 12:34:56 UTC, end at Tue 2023-06-27 08:30:00 UTC. --
  Jun 27 08:29:30 hostname haproxy[12345]: Proxy backend_1 started.
  Jun 27 08:29:35 hostname haproxy[12345]: Server backend_1/srv1 is UP.
  Jun 27 08:29:35 hostname haproxy[12345]: Server backend_1/srv2 is UP.
  Jun 27 08:29:45 hostname haproxy[12345]: Proxy backend_2 started.
  Jun 27 08:29:50 hostname haproxy[12345]: Server backend_2/srv1 is UP.
  Jun 27 08:29:50 hostname haproxy[12345]: Server backend_2/srv2 is UP.
```

To verify if the service is active:

```
sudo systemctl status hapee-extras-gslb
```

output

```
● hapee-extras-gslb.service - HAPEE GSLB
    Loaded: loaded (/lib/systemd/system/hapee-extras-gslb.service; enabled; vendor preset: enabled)
    Active: active (running) since Tue 2023-06-13 01:59:44 UTC; 1 day 15h ago
  Main PID: 1234 (hapee-extras-gs)
      Tasks: 1 (limit: 4915)
    Memory: 10.0M
    CGroup: /system.slice/hapee-extras-gslb.service
            └─1234 /usr/sbin/hapee-extras-gslb --config /etc/hapee-extras/gslb.cfg
```

Reference guide

This section describes the syntax of the zonefile, `/etc/hapee-extras/hapee-gslb.conf`.

Domain zone

A `zone` section defines a domain zone, a distinct part of the domain namespace. It contains one or more `record` directives.

Syntax:

```
zone <ZONE_NAME>
  ttl <TTL>
  record <DOMAIN_NAME string> ttl <seconds> <RECORD_TYPE> <FIELDS>
```

This example defines a zone for `example.com`.

```
zone example.com
```

In the following sections we describe directives found within the `zone` section.

Time to Live

The minimum TTL for records in this zone: `TTL <num>`. This is the default used for records unless specified otherwise in individual records.

```
zone example.com
  ttl 84600
```

The TTL here is 86,400 seconds, or 24 hours. DNS resolvers and clients that retrieve this DNS record will be allowed to cache it for up to 24 hours before checking for updates by querying the authoritative DNS server again.

Records

Records declared using the `record` directive provide `zone` responses.

Syntax:

```
record <DOMAIN_NAME string> ttl <seconds> <RECORD_TYPE> <FIELDS>
```

Arguments include:

Argument	Description
DOMAIN_NAME string	Domain name as string or use `@` to indicate root domain given on `zone` line.
ttl	Time to Live in seconds for this specific record.
RECORD_TYPE	Each record type can be one of the following: SOA, A, AAAA, CNAME, NS, MX, map, list.
FIELDS	Fields are unique to each `RECORD_TYPE`.

SOA records

Start of Authority (SOA) records provide administrative information about the zone, like the primary name server, the email of the domain administrator, and some configuration parameters. A zone must have exactly one SOA record.

```
zone example.com
  ttl 86400
  record @ ttl 900 SOA ns1.nameserver.com. admin.example.com. 2023090501 3600 1800 604800 86400
```

The SOA record's arguments include:

Argument	Description
ns1.nameserver.com	The primary authoritative name server for the domain where DNS queries for this domain are initially directed.
admin.example.com	The email address of the responsible party or administrator for the domain. In DNS zone files, it's common to use an email address with a dot (.) replaced by an @ symbol to obfuscate the email and avoid spam. So, admin.example.com typically represents an email address like admin@example.com.
2023090501	The serial number, which is incremented each time the zone is updated. This helps in tracking changes to the zone.
3600	Refresh time, which indicates how often secondary name servers should check for updates to this zone. This means that any DNS queries for the alias domain will be redirected to the target domain specified in the CNAME record.
1800	Retry time, which indicates how often secondary name servers should retry if they fail to refresh the zone.
604800	Expire time, which sets a limit on how long secondary servers can continue to use the zone data if they cannot refresh it.
86400	The minimum TTL for records in this zone. This is the default used for records unless specified otherwise in individual records.

A records

Map a domain name to an IPv4 address.

```
record www A 203.0.113.1
```

AAAA records

Map a domain name to an IPv6 address.

```
example.com. AAAA 2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

CNAME records

Creates an alias or nickname for one domain that points to another domain's canonical (primary) name.

```
www.example.com. CNAME example.com
```

MX records

An MX (mail exchange) record represents the mailing record for the domain. `MX <Priority number>` indicates the preference or priority of this mail server. Lower values represent higher priority.

Here the `ttl` is set to `30` seconds, and `10` is the priority value for the mail server:

```
record example.com ttl 30 MX 10 mail.example.com
```

NS records

Specifies the authoritative name servers for a domain. These servers are responsible for providing DNS information for the domain.

```
example.com. NS ns1.examplehosting.com
example.com. NS ns2.examplehosting.com
```

List records

A list of `answer-list` names separated by spaces associated with the domain name, for example:

```
record www2 list london paris amsterdam
```

Map records

A list of `geoip-map` names separated by spaces associated with the domain name, for example:

```
record www3 map mymap
```

Answer list

An `answer-list` configuration is a set of parameters that dictate how to select and serve the best server or resource to a client based on certain conditions such as server health, load, or geographical location. It specifies the method for server selection, health check options, and other variables.

Syntax:

```
answer-list <ANSWER_LIST_NAME string>
  up_threshold <THRESHOLD number>
  method multi-up|multi-all|multi-rr|single-rr
  option httpchk|tcpchk [fall <FALL_COUNT number>] [rise <RISE_COUNT number>]
  http-check connect
  http-check send uri <URI string> hdr <HEADER_NAME string>
  http-check expect status <STATUS number>
  answer-record <NAME> [<IP>] [weight <WEIGHT number>]
```

Directive	Description
up_threshold	Determines the percentage of servers that must be up. Otherwise, traffic is routed to a different datacenter altogether. A threshold of 1 means that all servers in an `answer-list` must be healthy for the corresponding datacenter to be regarded as active. If you specify 0.1, then 10% of the total weighted number of servers must be up.
method	Determines which IP addresses to return. As with geolocation-based load balancing, you can: return a single IP in a round-robin rotation (`single-rr`), return multiple addresses in a round-robin rotation (`method multi-rr`), return all servers that are up (`method multi-up`), or return all servers, even if they are down (`method multi-all`)
option	Specify `httpchk` or `tcpchk` to monitor the health of servers. If the servers are HAProxy Enterprise load balancers, you can use monitor URI as an HTTP health check endpoint. The health check options `httpchk` and `tcpchk` are mutually exclusive.
http-check	Set any relevant health check parameters.
answer-record	Enter any number of `answer-record` directives, which denote IP addresses assigned to the datacenter.

GeoIP map

A `geoip-map` is a configuration setting that allows you to perform geolocation-based routing using your geolocation database. This will return different IP addresses in response to client DNS queries based on their geographical location.

Syntax:

```
geoip-map <MAP_NAME string>
  location-base <PATH string>
  location <GEO string|default> <ANSWER-LISTS>
  network <SUBNET cidr> <ANSWER-LISTS>
```

Directive	Description	Example
location-base	Absolute path to the geolocation database. You can supply several geolocation database names separated by spaces.	location-base / data/geoip/ GeoLite2-City.mmdb
location	The first parameter is a hierarchical path to a geographic region in the order of the continent code, a country ISO code, then more specific regions like state and city name. Refer to the MaxMind reference guide and ISO-3166 for these codes. Note that GSLB will search deeper into the hierarchy if a match is not found at the current layer. For example, you could specify country and city name, but omit the state name between them. The second parameter is a space-separated list of `answer-list` section names (e.g. DC2). GSLB directs client requests sent from this location to the first healthy datacenter in the list.	location NA/US/ NY DC2
network	As an alternative to using `location`, which uses geolocation data to choose the datacenter, you can also specify a client IP range. Set a subnet value in CIDR notation followed by an ordered list of datacenters (separated by spaces). The second parameter is a space-separated list of `answer-list` section names (e.g. DC2). GSLB directs client requests sent from this subnet to the first healthy datacenter in the list.	network 198.51.100.0/24 DC1

See also

- The GSLB module is a wrapper around the GDNSD software. The syntax we use is different, but you may still find the GDNSD documentation useful.

Real-time Dashboard

Deprecation notice

The Real-time Dashboard has been marked as deprecated and is scheduled for removal. HAProxy Enterprise 2.8 will be the last version to support it. Commands and examples on this page specify 2.8 as the version number, but they equally apply to earlier versions.

If you are looking for a replacement for the Real-time Dashboard, look no further than the HAProxy Fusion Control Plane. Manage your HAProxy Enterprise load balancers from a single graphical dashboard or a REST-based API.

The Real-time Dashboard is a web application that collects live metrics from your HAProxy Enterprise load balancers and displays them. You can also perform actions like disabling servers.

Although you can install the dashboard onto your own web server, we host it for you at `https://dashboard.haproxy.com/`.

Installation

The Real-time Dashboard is a web application that collects live metrics from your load balancers and displays them. You can also perform actions like disabling servers.

1. Use your system's package manager to install the dashboard package onto your HAProxy Enterprise server:

Apt:

```
sudo apt-get install hapee-2.8r1-lb-dashboard
```

Yum:

```
sudo yum install hapee-2.8r1-lb-dashboard
```

Zypper:

```
sudo zypper install hapee-2.8r1-lb-dashboard
```

Pkg:

```
sudo pkg install hapee-2.8r1-lb-dashboard
```

This installs the following components:

Component	Description
`/etc/hapee-2.8/dashboard-module.cfg`	Configuration that exposes a new endpoint within HAProxy Enterprise at port 9022. When a request arrives at this port, HAProxy Enterprise dispatches it to a Lua module that returns statistics for the dashboard to collect.
`/opt/hapee-2.8/dashboard/dashboard-module/`	Folder that contains the Lua module that generates HAProxy Enterprise statistics. It contains `dashboard.luac`, the Lua module for statistics and `dashboard-config.lua`, the configuration file that defines a cluster of HAProxy Enterprise nodes from which to gather statistics.
`/opt/hapee-2.8/dashboard/dashboard/`	Folder that contains the web application for the dashboard.

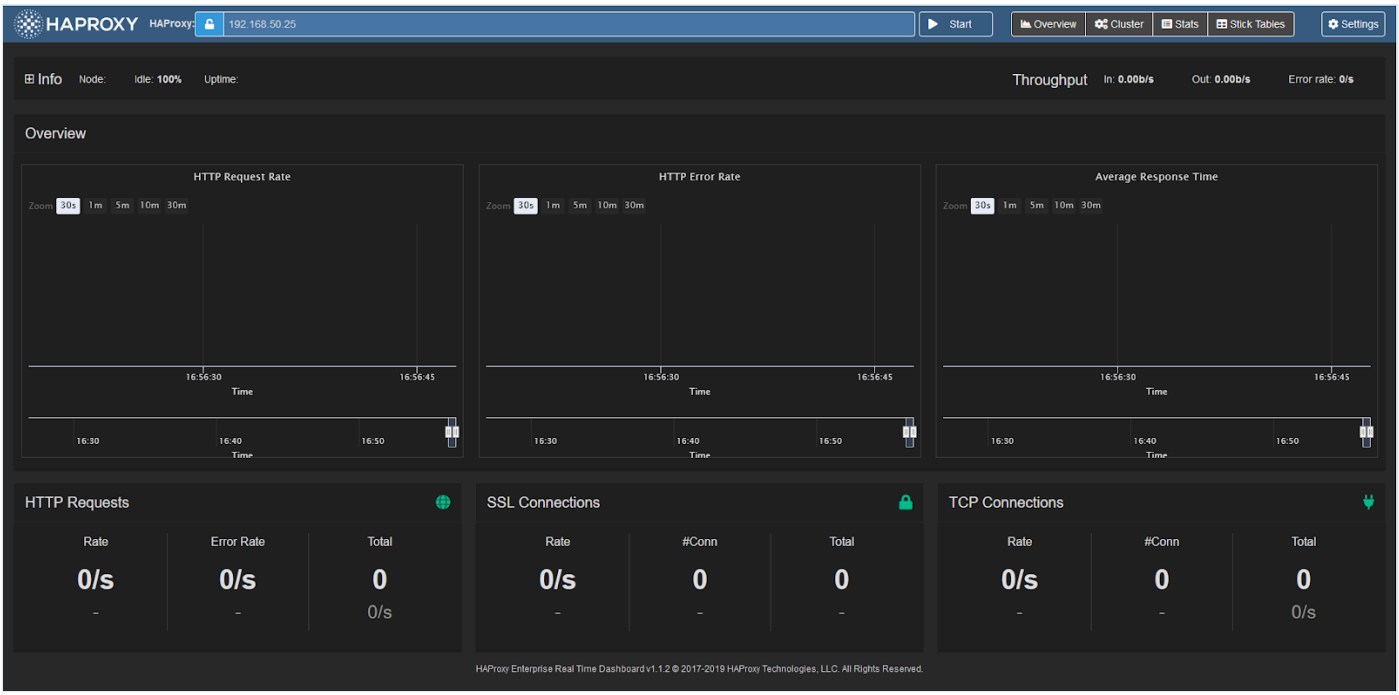
2. Copy the folder `/opt/hapee-2.8/dashboard/dashboard` to your web server to host it on an IP address and port of your choosing. Consult your web server's instructions for hosting static files.

> **Info**
>
> Because the dashboard starts without any data, you can use the one we host on haproxy.com. Go to `https://dashboard.haproxy.com` to access your hosted dashboard.

To give you an example of hosting the dashboard yourself, you can use the following Python commands to start a rudimentary web server directly from the dashboard directory:

```
cd /opt/hapee-2.8/dashboard/dashboard
python3 -m http.server 8000
```

Open the Real-time Dashboard at port 8000 using your web browser.



3. You must configure the dashboard to fetch metrics data from HAProxy Enterprise. You have two options:

   - Configure the Lua module
   - Configure the Dashboard Gateway

Lua module

The dashboard package you used to install the dashboard UI also includes a Lua module that returns HAProxy Enterprise statistics. It allows the Real-time Dashboard to display live data.

Enable metrics for a single node

1. Edit the configuration file:

   - On Debian/Ubuntu, `/etc/default/hapee-2.8-lb`
   - On Alma/Oracle/Redhat/Rocky, `/etc/sysconfig/hapee-2.8-lb`

   Change the OPTIONS line so that it contains the following `-f` argument, which loads the `dashboard-module.cfg` file when the HAProxy Enterprise service starts:

   hapee-VERSION-lb

   ```
   OPTIONS="-f /etc/hapee-2.8/dashboard-module.cfg"
   ```

The `dashboard-module.cfg` file starts a new listener at port 9022 that exposes metrics data. This data comes from an applet defined in the Dashboard Lua module located in `/opt/hapee-2.8/dashboard/dashboard-module`.

2. Restart the HAProxy Enterprise service for the change to take effect:

```
sudo systemctl restart hapee-2.8-lb
```

3. Open the dashboard in your browser. Click the **Settings** button to display the **Settings** window.

4. Set **Data Source** to **Real-time Dashboard Module**.

5. Set **Data Source Mode** to **Single Node** and close the window.

6. At the top of the Real-time Dashboard is the header bar, which display the address from where it fetches statistics. Change the URL to be the address of your HAProxy Enterprise node. You do not need to specify a port. The port defaults to 9022, so you do not need to set it here. If the address is already correct, you do not need to change it.

7. Click the **Start** button to complete setup.

Enable metrics for multiple nodes

Optionally, you can connect to data feeds from multiple HAProxy Enterprise nodes running on different servers:

1. On each HAProxy Enterprise server, install the Real-time Dashboard.

2. Edit the configuration file to enable metrics. See Enable metrics for a single node.

3. Choose one server to act as the “aggregation primary” node. On that server, update the file `/opt/hapee-2.8/dashboard/dashboard-module/dashboard-config.lua`. Initially, it contains the following empty configuration:

dashboard-config.lua

```
conf = {
    cluster = {
        nodes = nil,
        master = false
    }
}
```

4. Add the IP address and port of each HAProxy Enterprise node to the `nodes` property. Also, set the `master` property to `true`:

dashboard-config.lua

```
conf = {
    cluster = {
        nodes = {
            "192.168.50.25:9022",
            "192.168.50.26:9022",
            "192.168.50.27:9022",
        },
        master = true
    }
}
```

5. Restart the HAProxy Enterprise service:

```
sudo systemctl restart hapee-2.8-lb
```

6. Open the Real-time Dashboard in your browser, then click the **Settings** button to display the **Settings** window.

7. Set **Data Source** to **Real-time Dashboard Module**.

8. Set **Data Source Mode** to **Cluster Aggregation** and close the window.

9. Click the **Start** button to complete setup.

Set HTTP basic authentication

When you enable HTTP basic authentication in the file `dashboard-module.cfg`, the Real-time Dashboard Lua module's API is restricted to certain users. By default, the configuration defines a user named "dashboard" with the password "test". Change the username and password, and update the Real-time Dashboard's settings to match.

1. Use `mkpasswd` to create an SHA-2 password:

```
mkpasswd -m SHA-256
```

> **Info**
>
> If `mkpasswd` is not present on your OS, it can be installed by downloading the `whois` package on most Linux distributions; on RedHat you may have to explicitly install it via `sudo yum install mkpasswd`.

2. Edit the configuration file `/etc/hapee-2.8/dashboard-module.cfg2` to change the **dashboard_users** `userlist` section so that it contains one `user` line to define a username and encrypted password.

3. Add the password to the `user` line, and restart the HAProxy Enterprise service:

```
sudo systemctl restart hapee-2.8-lb
```

4. Open the Real-time Dashboard in your browser and click the **Settings** button to display the **Settings** window. Set the **Username** and **Password** values to match.

Enable HTTPS between the Dashboard UI and HAProxy Enterprise

The UI fetches metrics from HAProxy Enterprise. To encrypt this communication, you must enable HTTPS.

1. Obtain a valid TLS certificate for the node where you are running HAProxy Enterprise. This is the aggregation primary node when you run multiple HAProxy Enterprise nodes.

2. Save it to `/etc/hapee-2.8/certs`.

3. Edit the file `/etc/hapee-2.8/dashboard-module.cfg` to remove the following line from the `listen dashboard` section:

```
bind *:9022
```

4. Replace it with the following line, changing `site.pem` to be name of your TLS certificate:

```
bind *:9023 ssl crt /etc/hapee-2.8/certs/site.pem
```

5. Restart the HAProxy Enterprise service:

```
sudo systemctl restart hapee-2.8-lb
```

6. Open the Real-time Dashboard in your browser and change the address in the header bar to be the hostname that matches the TLS certificate's CN value. Click the padlock icon to turn it green.

7. Click the **Start** button to complete setup.

> **Info**
>
> If you use a self-signed certificate, you must either import it into your workstation's trusted certificate store, or accept it as an untrusted certificate in your browser. To do so, open a new tab and enter the URL to the API, such as `https://lb1.test.com:9023/dashboard/info`, and accept the certificate.

Dashboard Gateway

The Dashboard Gateway is an alternative to the Real-time Dashboard Lua module and works with all versions of HAProxy. It collects metrics data from one or several HAProxy Enterprise nodes and exposes it for the Dashboard Gateway to use. By default, it runs an API at port `9020`.

Before proceeding, install the **dashboard-gateway** package depending on your operating system:

Apt:

```
sudo apt-get install hapee-2.8r1-lb-dashboard-gateway
```

Yum:

```
sudo yum install hapee-2.8r1-lb-dashboard-gateway
```

Zypper:

```
sudo zypper install hapee-2.8r1-lb-dashboard-gateway
```

Pkg:

```
sudo pkg install hapee-2.8r1-lb-dashboard-gateway
```

Enable metrics for a single node

1. Edit your HAProxy Enterprise configuration file to add a `stats socket` directive in the `global` section.

   This enables the HAProxy Runtime API used to fetch metrics. The IP address is `127.0.0.1` and the port is `9024`. You must set the `level` to `admin` so that the Dashboard Gateway can manage the HAProxy Enterprise node, as follows:

   ```
   global
     stats socket ipv4@127.0.0.1:9024 level admin
   ```

2. Restart HAProxy:

```
sudo systemctl restart hapee-2.8-lb
```

3. Copy the `config.json.example` configuration file and rename it `config.json`.

   See Configure the Dashboard Gateway for more details.

   ```
   cd /opt/hapee-2.8/dashboard/dashboard-gateway/conf
   sudo cp config.json.example config.json
   ```

   By default, this file contains the following:

   config.json

   ```
   {
     "isLogging": true,
     "logFile": "access.log",
     "defaultHandler": "socket",
     "defaultHostSSL": false,
     "nodes": [
     {
       "host": "127.0.0.1",
       "port": "9024"
     }
     ]
   }
   ```

4. Start the Dashboard Gateway:

   ```
   sudo systemctl enable hapee-2.8-lb-dashboard-gateway
   sudo systemctl start hapee-2.8-lb-dashboard-gateway
   ```

5. Open the Real-time Dashboard in your browser and click the **Settings** button to display the **Settings** window.

6. Set **Data Source** to **Real-time Dashboard Gateway**.

7. Set **Data Source Mode** to **Single Node**.

8. Set **Real-time Dashboard Gateway URL** to the URL where the Dashboard Gateway is running. The port defaults to 9020. If you have configured it to use HTTPS, click the padlock icon to make it green, and close the window.

9. Change the URL in the header bar to the address of your HAProxy Enterprise node's socket. The port defaults to 9024, so you do not need to set it here.

10. Click the **Start** button to complete setup.

Enable metrics for multiple nodes

Optionally, you can connect to data feeds from multiple HAProxy Enterprise nodes running on different servers:

1. On each HAProxy Enterprise server, edit the HAProxy Enterprise configuration file to add a `stats socket` directive in the `global` section.

2. Give an IP address that the server running Dashboard Gateway can reach, or an asterisk (*) to listen on all IP addresses:

```
global
  stats socket ipv4@*:9024 level admin
```

3. Restart HAProxy:

```
sudo systemctl restart hapee-2.8-lb
```

4. Choose one server to be the “aggregation primary” node. On that server, ensure that the `dashboard-gateway` package is installed.

5. Make a copy of the `config.json.example` configuration file and rename it `config.json`.

```
cd /opt/hapee-2.8/dashboard/dashboard-gateway/conf
sudo cp config.json.example config.json
```

6. Update the file `config.json` so that its `nodes` section lists each HAProxy Enterprise node’s IP address and port.

   See Configure the Dashboard Gateway for more details.

config.json

```
{
   "isLogging": true,
   "logFile": "access.log",
   "defaultHandler": "socket",
   "defaultHostSSL": false,
   "nodes": [
      {
         "host": "127.0.0.1",
         "port": 9024
      },
      {
         "host": "192.168.50.26",
         "port": 9024
      }
   ]
}
```

7. Start or restart the Dashboard Gateway:

```
sudo systemctl enable hapee-2.8-lb-dashboard-gateway
sudo systemctl start hapee-2.8-lb-dashboard-gateway
```

8. Open the Real-time Dashboard in your browser and click the **Settings** button to display the **Settings** window.
9. Set **Data Source** to **Real-time Dashboard Gateway**.
10. Set **Data Source Mode** to **Cluster Aggregation**.
11. Set **Real-time Dashboard Gateway URL** to the URL where the Dashboard Gateway is running.

    If you have configured it to use HTTPS, click the padlock icon to make it green. Then, close the window.
12. Click the **Cluster** button to display the **Cluster Setup** window.

    Check that all nodes are listed. Then, click **Use Cluster** to complete setup.

Configure the Dashboard Gateway

You can enable various settings for the Dashboard Gateway in the configuration file `/opt/hapee-2.8/dashboard/dashboard-gateway/conf/config.json`. Make sure you restart the Dashboard Gateway service after making any change to this file. This section describes the available options.

Here is an example configuration:

config.json

```
{
   "apiRoot": "/",
   "authChallenge": false,
   "authRealm": "HAProxy Enterprise Real-time Dashboard Gateway",
   "defaultHandler": "socket",
   "defaultHost": "127.0.0.1",
   "defaultHostPort": 9024,
   "defaultHostSSL": false,
   "defaultModulePassword": "test",
   "defaultModuleUser": "dashboard",
   "disableHttp": false,
   "disableHttps": true,
   "httpsCertFile": "cert.pem",
   "httpsKeyFile": "key.pem",
   "httpPort": 9020,
   "httpsPort": 9021,
   "isLogging": true,
   "logFile": "access.log",
   "tlsPassphrase": "mypassphrase",
   "skipBasicAuth": false,
   "nodes": [
      {
         "host": "127.0.0.1",
         "port": 9024,
         "handler": "socket",
         "ssl": false,
         "user": "dashboard",
         "password": "test"
      }
   ]
}
```

The following table describes these options:

Option	Description
`apiRoot`	URI path under which the `/dashboard`, `/gateway`, `/cluster` and other sub-paths are available. Default: `/`
`authChallenge`	Whether to show an HTTP Basic authentication challenge when accessing the API. Note that you can enable Basic authentication by setting `skipBasicAuth` to false, but not show a prompt. Default: false
`authRealm`	Realm for HTTP Basic authentication. Default: “HAProxy Enterprise Real-time Dashboard Gateway”
`defaultHandler`	Default way for the gateway to connect to HAProxy. Either `module` or `socket`. Default: socket.
`defaultHost`	Default hostname or IP address the gateway uses to connect to HAProxy. Default: 127.0.0.1.
`defaultHostPort`	Default port the gateway uses to connect to HAProxy. Defaults to one of the following values: 9022 (HTTP), 9023 (HTTPS), 9024 (TCP), 9025 (TCP with TLS).
`defaultHostSSL`	Whether to use encryption when communicating with HAProxy. Default: false.
`defaultModulePassword`	Default Lua module password to use when connecting to nodes running the Dashboard Lua module.
`defaultModuleUser`	Default Lua module username to use when connecting to nodes running the Dashboard Lua module.
`disableHttp`	Whether to disable serving the gateway API over HTTP. Default: false.
`disableHttps`	Whether to disable serving the gateway API over HTTPS. When set to false, you must also set `httpsCertFile` and `httpsKeyFile`. Default: true.
`httpsCertFile`	Path to the SSL certificate file when serving the gateway API over HTTPS. You must set `disableHttps` to false to enable HTTPS. If you use a relative path, `./` and `./cert` will be searched. Default: `cert.pem`.
`httpsKeyFile`	Path to the SSL key file when serving the gateway API over HTTPS. You must set `disableHttps` to false to enable HTTPS. If you use a relative path, the API searches for `./` and `./cert`. Default: `key.pem`.
`httpPort`	HTTP port where the gateway listens. Default: 9020.
`httpsPort`	HTTPS port where the gateway listens. Default: 9021.
`isLogging`	Whether to log to a file in addition to stdout/stderr. Default: false.
`logFile`	Path to the log file. Default: “”.
`tlsPassphrase`	TLS passphrase for the certificate specified in `httpsCertFile` and `httpsKeyFile`. Default: “”.
`skipBasicAuth`	Disable HTTP Basic authentication when accessing the gateway API. Default: false.
`nodes`	Collection of HAProxy Enterprise nodes that the gateway communicates with to get metrics.

The `nodes` collection has the following options:

Option	Description
`host`	The hostname or IP address the gateway uses to connect to HAProxy.
`port`	The port that the gateway uses to connect to HAProxy.
`handler`	The way in which the gateway connects to HAProxy. Either `module` or `socket`.
`ssl`	Whether to use encryption when communicating with HAProxy.
`user`	To use when connecting Gateway to nodes running the Dashboard Lua module.
`password`	To use when connecting Gateway to nodes running the Dashboard Lua module.

Enable HTTP Basic authentication

When you enable HTTP Basic authentication in the file `config.json` and set `authChallenge` to true, you restrict the Dashboard Gateway API to certain users. You can set one or more usernames and passwords.

1. The configuration file `/opt/hapee-2.8/dashboard/dashboard-gateway/conf/htpasswd.json` has the following format:

   htpasswd.json

   ```
   {
      "username_1": {
         "password": "sha256_password_hash",
         "accessAll": "true"
      },
      "username_2": {
         // . . .
      },
      // . . .
   }
   ```

   For example, the default configuration is shown:

   htpasswd.json

   ```
   {
      "dashboard": {
         "password": "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08",
         "accessAll": true
      }
   }
   ```

   This adds the username dashboard with a password of test.

2. Edit the file to change the existing username and password.

   Use the following command to create the encrypted password:

   ```
   echo -n "YOUR_PASSWORD" | sha256sum
   ```

3. Open the Real-time Dashboard in your browser and click the **Settings** button to display the **Settings** window. Set the **Username** and **Password** values to match those you just configured.

Enable HTTPS between the aggregation primary and child nodes

The aggregation primary node fetches metrics from multiple HAProxy Enterprise nodes. To encrypt this communication, you must enable HTTPS.

1. Obtain a valid TLS certificate for each HAProxy Enterprise child node.

2. Make sure that the certificate is in PEM format.

   Then, combine the private key and the public certificate into a single PEM file. Copy it to the node under the path `/etc/hapee-2.8/certs`.

3. Edit the node's HAProxy Enterprise configuration file.

   In the `global` section, add an `ssl` and `crt` parameter to the `stats socket` directive. Also, change its port to 9025:

   ```
   global
     stats socket ipv4@*:9025 level admin ssl crt /etc/hapee-2.8/certs/site.pem
   ```

4. On the aggregation primary node, edit the file `/opt/hapee-2.8/dashboard/dashboard-gateway/conf/config.json`.

   For each node where you want to enable TLS, set `host` to the node's FQDN so that it matches the certificate; Set `port` to 9025; Set `ssl` set to `true`.

config.json

```
{
   "isLogging": true,
   "logFile": "access.log",
   "defaultHandler": "socket",
   "defaultHostSSL": false,
   "nodes": [
      {
         "host": "127.0.0.1",
         "port": 9024
      },
      {
         "host": "lb2.test.com",
         "port": 9025,
         "ssl": true
      }
   ]
}
```

5. Restart the Dashboard Gateway:

```
sudo systemctl restart hapee-2.8-lb-dashboard-gateway
```

Enable HTTPS between the Dashboard UI and the Dashboard Gateway

The UI fetches metrics from the Dashboard Gateway. To encrypt this communication, you must enable HTTPS.

1. Obtain a valid TLS certificate for the node where you are running the Dashboard Gateway.

   This will be the aggregation primary node if running multiple HAProxy Enterprise nodes.

2. Make sure that the certificate and private key files are in PEM format.

   Copy both the certificate and key files to the path `/opt/hapee-2.8/dashboard/dashboard-gateway`. These should be made available before restarting the Dashboard Gateway.

3. Edit the file `/opt/hapee-2.8/dashboard/dashboard-gateway/conf/config.json`.

   - Set `disableHttps` to `false`
   - Set `httpsCertFile` to the path to the certificate file
   - Set `httpsKeyFile` to the path to the key file.

config.json

```
{
   "isLogging": true,
   "logFile": "access.log",
   "defaultHandler": "socket",
   "defaultHostSSL": false,
   "disableHttps": false
   "httpsCertFile": "cert.pem",
   "httpsKeyFile": "key.pem",
   "nodes": [
      {
         "host": "127.0.0.1",
         "port": 9024
      },
      {
         "host": "lb2.test.com",
         "port": 9025,
         "ssl": true
      }
   ]
}
```

4. Restart the Dashboard Gateway:

```
sudo systemctl restart hapee-2.8-lb-dashboard-gateway
```

5. Open the Real-time Dashboard in your browser and click the **Settings** button to display the **Settings** window.

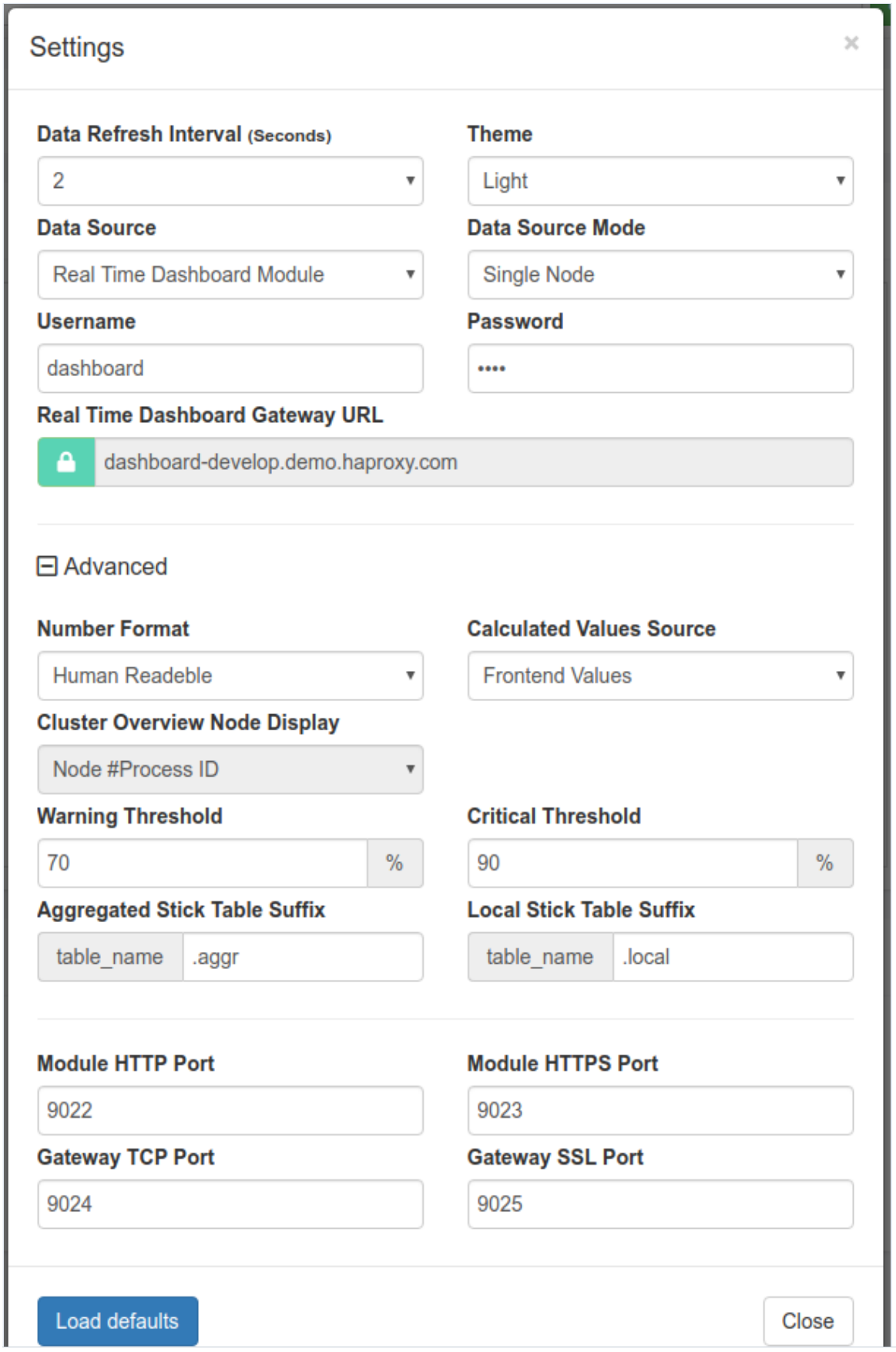
6. Set **Real-time Dashboard Gateway URL** to the FQDN of the Dashboard Gateway so that it matches the TLS certificate (e.g. `dashboard-gateway.test.com:9021`).

   Its port defaults to 9021, so you do not need to set it here.

7. Click the padlock icon in the **Real-time Dashboard Gateway URL** box so that it turns green. Then, click **Close**.

Using the dashboard

You configure the Real-time Dashboard in **Settings** at the top right of the header bar.



Real-time Dashboard saves these settings automatically in the browser's **local storage** and they persist throughout page reloads and browser restarts.

1. You can modify the following settings:

Basic setting	Description
Data Refresh Interval	Data refresh period.
Theme	GUI theme. Currently available themes are Dark (default) and Light.
Data Source	Real-time Dashboard module (default) or Real-time Dashboard Gateway.
Data Source Mode	Single Node: Real-time Dashboard requests and displays only the data from the module node it connects to, or the addressed node when the data source is Gateway. Cluster Aggregation: Real-time Dashboard requests and displays data from all HAProxy Enterprise nodes from which the module or Gateway was configured to aggregate data.
Username	Username for connecting to the Module or Gateway.
Password	Password for connecting to the Module or Gateway.
Real-time Dashboard Gateway Address	URL of the Real-time Dashboard Gateway (applicable when Data Source is set to **Real-time Dashboard Gateway**).

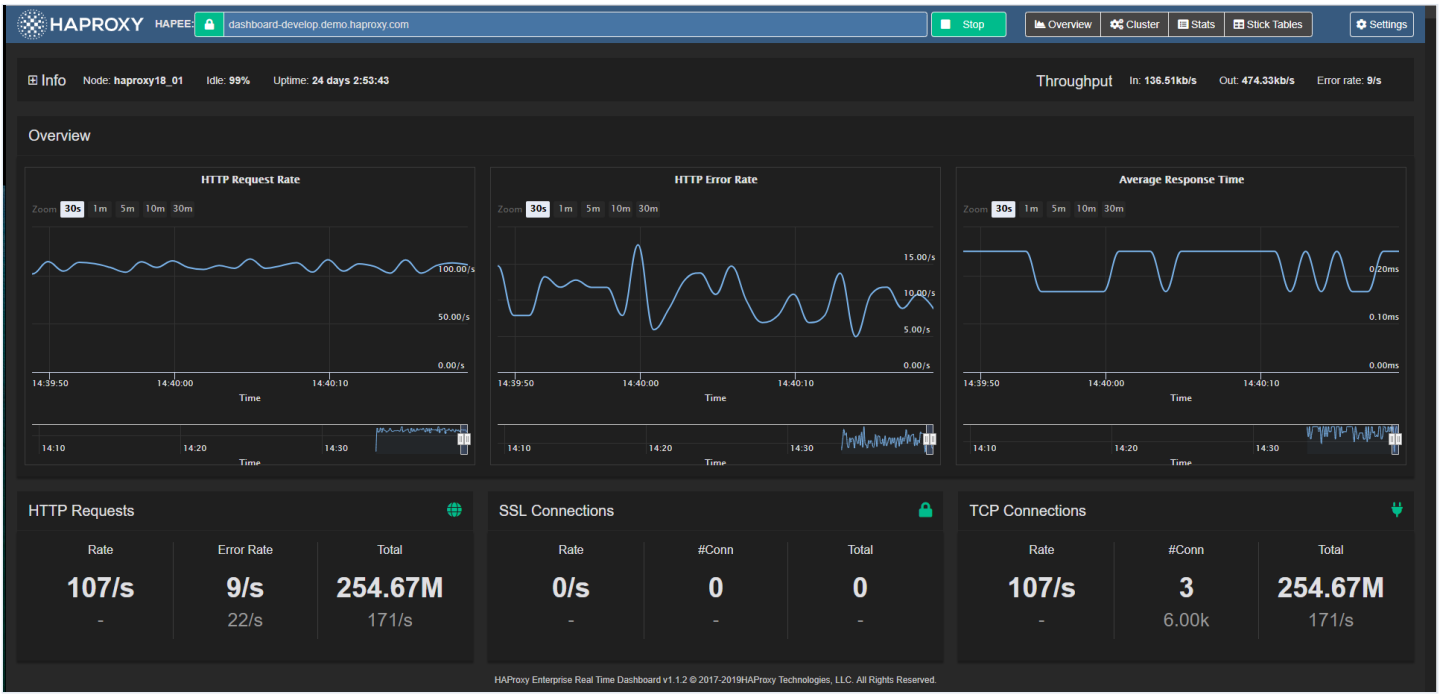
Advanced setting	Description
Number Format	Select between three different formats to display numerical values in Real-time Dashboard: Human Readable: formats values and adds SI order-of-magnitude suffixes (i.e. 1.26 kb/s). Formatted Values: inserts commas in the value and limits the number of decimal places (i.e. 1,261.00 b/s). Raw Values: displays the raw value from the statistics, except when Real-time Dashboard calculates the value, in which case it also limits the number of decimal places (i.e. 1261.00 b/s).
Calculated Values Source	Frontend Values: The values for graphs and Info Bar are summarized from the frontend block stats, errors are request errors. Backend Values: The values for graphs and Info Bar are summarized from the backend block stats, errors are response errors.
Cluster Overview Node Display	Determines the Node display format on Cluster Overview pane: `Node #Process ID`: i.e. `my_server #2`. `Host:Port`: i.e. `127.0.0.1:9022`.
Warning Threshold	For values on the Overview Pane with configured limits, this determines the percentage of the limit acting as a threshold over which values show a warning color (yellow). Default is 70%.
Critical Threshold	For values on the Overview Pane with configured limits, this determines the percentage of the limit acting as a threshold over which values show a critical color (red). Default is 90%.
Aggregated Stick Table Suffix	The suffix that the Global Profiling Engine takes on to denote the aggregated (i.e. to) stick tables. Default is `.aggr`
Local Stick Table Suffix	The suffix that the Global Profiling Engine takes on to denote the source (i.e. from) stick tables. Default is `.local`
Module HTTP Port	Port number that Real-time Dashboard listens on by default for non-TLS HTTP connections. If there is no other value specified in the Header URL Box, this is the port number that Real-time Dashboard connects to. Default: 9022.
Module HTTPS Port	Port number where the Real-time Dashboard listens on by default for HTTPS connections (when the URL Box is in **locked** state). Default: 9023.
Gateway TCP Port	Port number where the HAProxy Enterprise Admin Socket listens on by default for non-TLS socket connections. This value gets relayed to the Dashboard Gateway if there is no other value specified in the URL Box. Default: 9024.
Gateway SSL Port	Port number where the HAProxy Enterprise Admin Socket listens on by default for TLS-encrypted socket connections (when the URL Box is in **locked** state. If there is no other port value given, this value gets relayed to the Dashboard Gateway). Default: 9025.

2. Click **Close** to modify your settings.

3. Click **Load defaults** if you need to return to default settings.

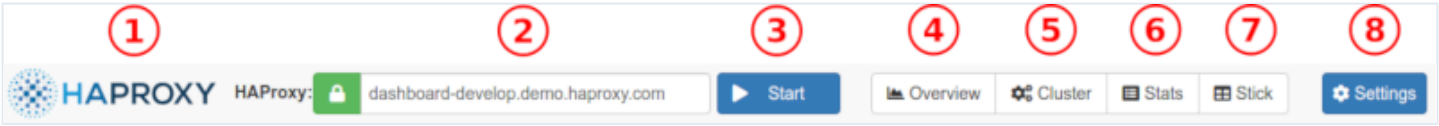
Navigate around the Real-time Dashboard

The general layout of the application is split into two main sections. Header Bar, Info Bar and a pane displaying selected views below it.



Header bar

Located at the top of the screen, the header bar allows you to perform the following tasks:



To reload the page:

- Click on the HAProxy Enterprise Logo (**1**).

To enter the address of the HAProxy Enterprise node:

- In the URL Box (**2**), use a format similar to `[<http|https>://]domain|IP[:port][/path]`.
- The mandatory information is the domain name or IP address of the HAProxy Enterprise node. It defaults to the domain where the UI is hosted.
- You can omit http or https because this information is stored
- If you do not specify a port, Real-time Dashboard uses the default port
- If you enter a path, Real-time Dashboard appends the standard API endpoints (**/dashboard** or **/cluster**) to it
- The icon indicates whether the access to the HAProxy Enterprise node uses TLS encryption. This is a toggle function.
- If you connect to Real-time Dashboard over HTTPS, you cannot connect to an HAProxy Enterprise node with an unencrypted link; hence the toggle is disabled.
- If you connect to an HAProxy Enterprise node over HTTPS, we recommend that you use a valid SSL certificate. Invalid certificates (not trusted by the browser) cause Javascript calls to fail silently and return a **Request Error** message when you click on **Servers Status**.

To turn on or off data fetching:

- Click on the Start/Stop button (**3**). If you set the data refresh interval to **Off** in **Settings**, you can display an instantaneous update by clicking **Refresh**.

To display the overview:

- Click on Overview button (**4**).

To display the cluster overview:

- Click on Cluster button (**5**) Available only in Cluster Aggregation mode. In Single Mode, it displays the Cluster Setup Dialog (see below).

To display statistics:

- Click on Stats button (**6**).

To display stick tables:

- Click on Stick button (**7**).

To configure the Real-time Dashboard:

- Click Settings (**8**) to display configuration options.

Info Bar

The Info Bar contains general information about an node. In Cluster Aggregation mode, it displays nodes to which the Real-time Dashboard is connected.

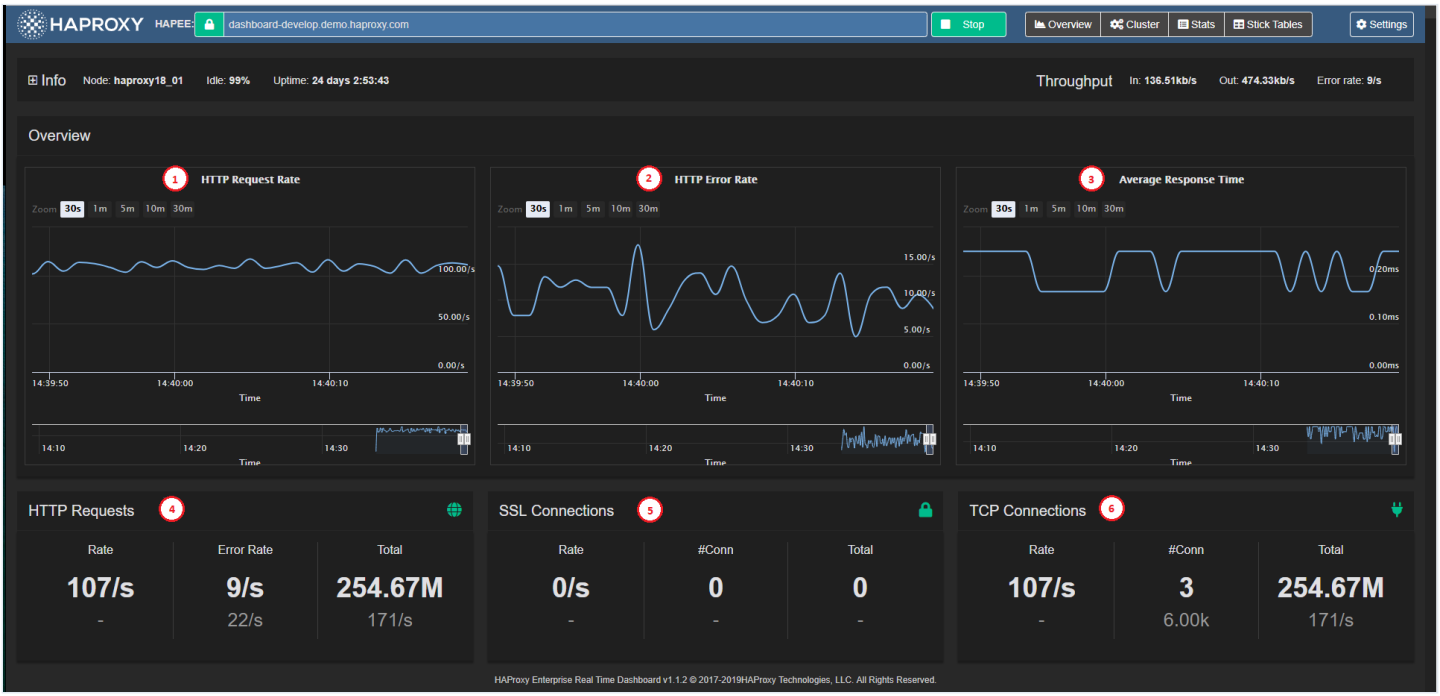
When in Cluster Aggregation mode, the main view of the Info Bar displays aggregated information; to drill down to information for each node, click the **+** icon.

The Info bar gives the following information:

Field	Description
Node	Host name of the node. A tool tip displays the PID, process number and version when hovering over **Node**.
Host	IP/Address and Port used to identify the Node in the Real-time Dashboard / Gateway. Displayed only in Cluster Aggregation drill-down.
Idle	Percentage of CPU idle time on the host. When hovering over this metric, a tool tip displays the CPU utilization of the HAProxy Enterprise node on the machine.
Uptime	HAPEE-LB service uptime to the moment in days, hours:min:secs, hours are given in 24-hour format.
In	Total Bytes In across all Frontends/Backends. You can select between Frontends or Backends as the source of this information in Settings.
Out	Total Bytes Out across all Frontends/Backends. Same as above.
Error rate	Total Errors divided by elapsed time across all Frontends/Backends. Same as above.

Read metrics in Real-time Dashboard

The Overview window is the initial view of the application. It displays the node-level (or cluster-level) summarized information, as well as key metric charts.



It contains the following elements:

HTTP Request Rate Chart (1) Charts the total HTTP request rate across the node (or across all nodes in Cluster Mode).

HTTP Error Rate Chart (2) Charts the total HTTP error rate across the node (or across all nodes in Cluster Mode).

Average Response Time Chart (3) Charts the average response time across the node (or across all nodes in Cluster Mode).

HTTP Requests Panel (4) Displays the node-level (or aggregate) HTTP Request related stats and limits.

SSL Connections Panel (5) Displays the node-level (or aggregate) SSL Connection related stats and limits.

TCP Connections Panel (6) Displays the node-level (or aggregate) TCP Connection related stats and limits.

Overview charts

The Overview graphs display information for HTTP traffic, average response time, and SSL/TCP connections.

Zoom Level Selector (1) Sets five different zoom levels for the Main Chart Display (30 seconds, 1 minute, 3 minutes, 10 minutes, and 30 minutes) The zoom level is synchronized across all three Overview Graphs.

Main Chart Display (2) A tool tip appears when hovering over any chart point to show the detailed data point (time and value). This data point is synchronized across all three Overview Charts.

Time Axis (3) Horizontal axis is the data point event time axis

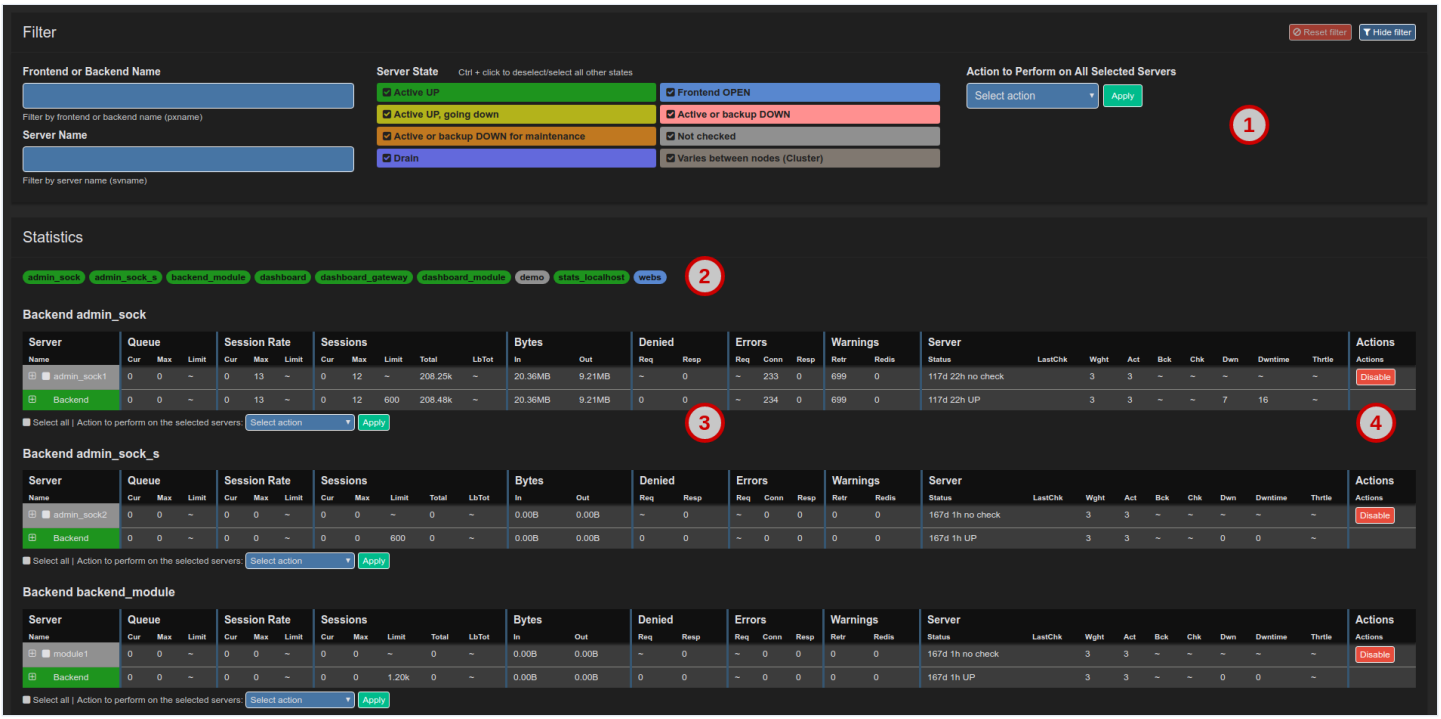
Value Axis (4) Vertical axis is the data-point values axis

Chart Navigator (5) Displays the local data history in a miniature view with its own time axis.

Navigator Handles (6) Focuses on a time period in the local history when you drag the handles. When you release the handle, the selected zoom level/period is synchronized across the three Overview Charts.

Read statistics in Real-time Dashboard

The Stats Pane presents individual Proxy (frontend, backend or listen block) statistics, as well as individual Server statistics in a table view.



The Stats Pane consists of the following major sections:

Filter Block (1) See Set filter criteria

Quick Navigation buttons (2) Gives quick access to the selected Proxy on the screen.

Stats Tables (3) Proxy and Server statistics, see Stats Pane under Data Fields for more detailed data on statistics values.

Quick Action Buttons (4) See below

Stats table

The tables operate slightly differently depending on the Data Source Mode:

- **Single** Node: Each server appears on one line, and its statistics and status come from the node to which Real-time Dashboard is connected.
- **Cluster Aggregation**: Each server appears as a header line in gray with statistics and status aggregated across all the nodes that it contains.

Depending on the mode, Stats tables display the following information:

Proxy Name Real-time Dashboard infers from available server lines whether the block is a frontend, backend, or listen (or FE/BE pair with same name, which it treats as a listen block).

Server Names Name of the server

Statistics Columns Hover over each column heading to display additional information. If the server state is not up, the entire row takes on the Server State color, as used for Filter Block labels. See **Stats Pane** in the section Understanding data from Real-time Dashboard for more detailed data on statistics values.

Drill down for more information:

- Click the **+** icons to display statistics for each node (when in cluster mode)

Perform an administrative task:

1. Select a server or all servers in the proxy.
2. Choose an action from the drop-down list.
3. Click **Apply**.

Enable or disable proxy:

The **Enable/Disable** button on the right of the screen allows you to turn off the server for maintenance or turn it back on if it is down.

Read proxy and server states

The Cluster Overview displays with a matrix/“spreadsheet” overview of states of Proxies and Servers for each connected HAProxy Enterprise Node (see Dashboard Aggregation Cluster). The columns in the matrix represent the nodes (which can be HAProxy Enterprise nodes or processes).

The key elements on the pane are:

Quick navigation buttons (1) Allows quick access to a server or proxy.

Cluster nodes (2) The rows in the matrix represent HAProxy Enterprise Nodes. The check boxes allow you to select a server or proxy on which to perform an administrative task.

Cluster Overview Matrix (3) Displays an overview of statuses for Proxies and Servers across Nodes. Servers that do not exist on specific nodes display an empty space (background color) in the matrix.

Proxy Rows (4) The rows are organized into Proxies containing Servers. The check boxes allow you to select a server or proxy on which to perform an administrative task.

Server Rows (5) The status color of the header column is medium-gray when the status for the server varies across nodes

Server Columns (6) Each cell (column) represents the Server status on that column's node.

Set filter criteria

Filtering allows you to display or to perform administrative tasks on selected proxies (i.e. frontend, backend and/or listen blocks) in both Stats and Cluster panes.

The filter criteria persist when you switch between Stats and Cluster panes.

Select filter

You can set filters in the following ways:

- **Text** - Allows filtering by Frontend or Backend Name (i.e. Proxy name) or by Server Name. There are three modes of filtering:
  - Partial String Matching: For example, when you enter **ss**, it matches up with **blessing**, **pass**, or **sserver**.
  - Multi-String Partial Matching: For example, when you enter **we se** it matches up with **webserver**, **awesome_service** and **we_base**.
  - RegEx Matching: For example, when you enter a string that begins and ends with a forward slash (`/`) it matches only those strings that a regular expression between those two slashes would match. The precise regular expression syntax used is JavaScript Regular Expressions.
- **Server Status** - Allows filtering by server status. A click on any server status toggles its state between selected and unselected.

Use filter for administrative tasks:

For a list of administrative tasks, see the section on how to Manage servers.

Reset filter:

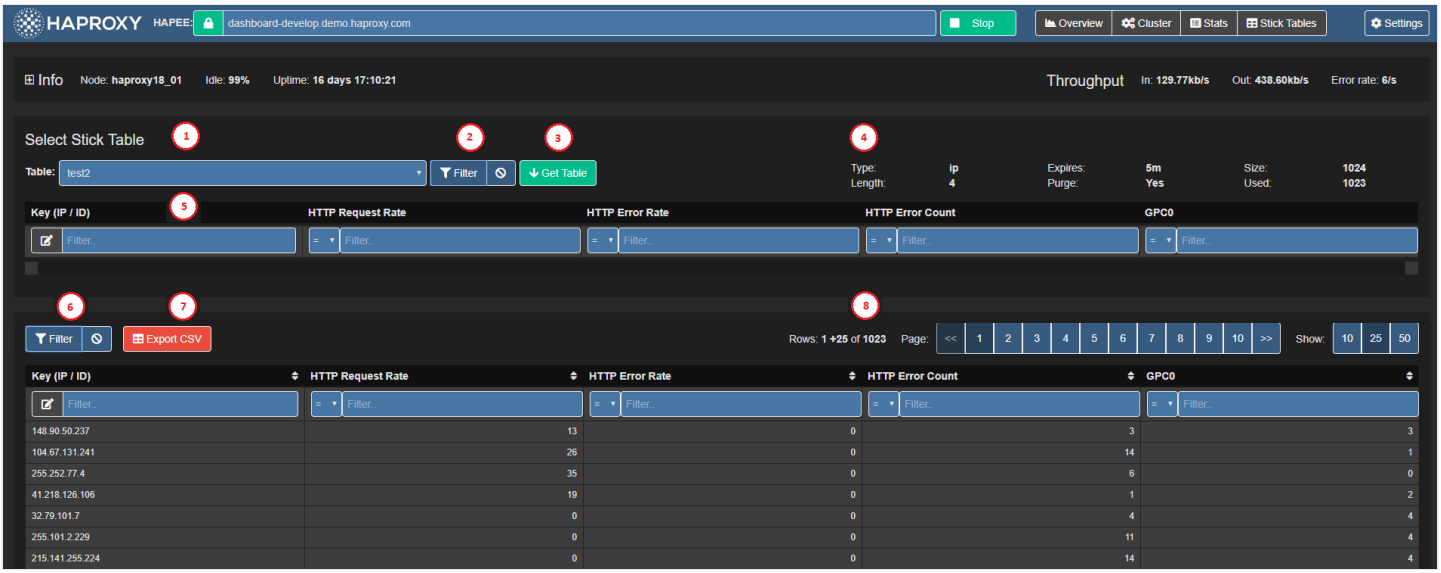
- Click **Reset filter** to restore the filter to default state (all Proxies/Servers visible); this function is enabled only when filter is active.

Hide filter:

- Click **Hide filter**to collapse the filter block (to free up vertical screen space).

Retrieve Stick Table contents

The Stick Table Explorer enables the user to retrieve the contents of stick tables from the HAProxy Enterprise node, and to sort them and filter them to gain insight into the data contained within, and even export them into Excel-friendly CSV format.



The Real-time Dashboard works with the Stick Table Explorer to enable you to explore the contents of HAProxy Enterprise Stick Tables in your node(s).

The aggregation cluster mode assumes that the Global Profiling Engine aggregates stick tables across the same set of nodes, and that the primary node uses aggregated stick tables from the Global Profiling Engine.

Local tables used for aggregation have the suffix `.local`, and resulting aggregated tables have the suffix `.aggr` (modifiable in General Settings).

> **Info**
>
> If you do not use the Global Profiling Engine in aggregation cluster mode, stick tables are only available in the single node mode from each of the HAProxy Enterprise nodes. In cluster mode, the non-aggregated tables get filtered out. The Stick Table Explorer pane is available, but the list of available stick tables is empty.

Apart from setting up the aggregation cluster and editing table suffixes as necessary, there is no additional setting to configure in Real-time Dashboard.

Retrieve a stick table

1. Click on **Stick Tables**in the header bar to display the Stick Table Explorer.
2. Select a table from a drop-down list (**1**).
3. Filter out any unwanted data by clicking **Filter** (**2**) and specifying the filter criteria. Toggling this button shows or hides the filter.

4. Click **Get** (**3**) to retrieve the following stick table information (**4**):

   - Type - type of key
   - Length - number of table columns
   - Expires - how long a row is kept in the table before it expires
   - Purge - whether the table purges old elements
   - Size - maximum number of rows
   - Used - current number of rows

Apply filters

You can filter table data in two stages:

- Globally, prior to node retrieval (**5**): Real-time Dashboard retrieves data on the HAProxy Enterprise node itself according to the filtering criteria (up to four maximum).
- Locally, in the browser (**6**): In contrast to the global filter, which occurs on the HAProxy Enterprise node itself, the local filter happens within the browser. When you remove a criterion, the table refreshes accordingly; however, it has no effect on the amount of data retrieved from the node.

You can combine the two filters to limit the data retrieved and refine the data to display within the browser.

Set a filter using the interface:

1. Enter a key (string enclosed in quotes “”).
2. Select an operator from the drop-down list and enter a value.

> **Info**
>
> **Key** does not have any operator and requires a substring for matching.

Set a filter using an expression:

1. In the filter box, click on pencil icon to turn it into a text input box.
2. Enter an expression in the following format: `[data_name operator value] AND [data_name operator value] ... AND [data_name operator value]`. The logical AND is implicit in the filter and no other logical operator is accepted.

where:

- **data_name** is one of: `key, server_id, gpc0, gpc0_rate, gpc1, gpc1_rate, conn_cnt, conn_cur, conn_rate, sess_cnt, sess_rate, http_req_cnt, http_req_rate, http_err_cnt, http_err_rate, bytes_in_cnt, bytes_in_rate, bytes_out_cnt, bytes_out_rate`. When you hover over the table header, a tool tip displays its data point name.
- **operator** is any of =, !=, <, >, <= and =>
- **value** is numerical value used for comparison in the filter.

Export CSV

- Click **CSV** (**7**) to download the entire table in CSV format.

> **Info**
> All data gets exported to CSV, regardless of any filter previously set.

Navigate between pages

1. Click on the page number (**8**) or << and >> to switch pages.
2. Choose the number of items to display: **10**, **25**, or **50O**.

Sort data

- Click on the up or down arrow next to the column header to sort data by ascending or descending order.

Understanding data from Real-time Dashboard

The Real-time Dashboard interface contains a number of input and output values. The following list documents their meaning, syntax, and, if applicable, the current method of value calculation.

Overview pane

The **Overview** pane displays general and summarized real-time information about the HAProxy Enterprise node.

Overview	Description
Overview - Throughput	Real time throughput rates for the HAProxy Enterprise process, containing 3 values: In - Rate of incoming data, in bits per second. Out - Rate of outgoing data, in bits per second. Error Rate - HTTP Error Rate, in errors per second.
Overview - HTTP Request Rate	A graph representation of HTTP request rate, in requests per second.
Overview - HTTP Error Rate	A graph representation of HTTP error rate, in errors per second.
Overview - Average Response Time	A graph representation of average HTTP response time, in milliseconds.
Overview - HAProxy Enterprise Information	HAProxy Enterprise process information, containing: Instance name - HAProxy Enterprise info field **Name**. Instance version - HAProxy Enterprise info field **Version**. Node name - HAProxy Enterprise info field **node**. Process ID - HAProxy Enterprise info field **Pid**. Instance description - HAProxy Enterprise info field **Description**. Uptime - HAProxy Enterprise info field **Uptime**. Idle% - HAProxy Enterprise info field **Idle_pct**.

HTTP Requests	Description
HTTP Requests - Rate	Data block containing two values: Current HTTP request rate, in requests per second. Maximum allowed HTTP request rate, in requests per second.
HTTP Requests - Total	Data block containing two values: Total HTTP requests since the HAProxy Enterprise node was started or since the counters were reset. Highest HTTP request rate seen, in requests per second, since the HAProxy Enterprise node was started or since the counters were reset.
HTTP Requests - Error Rate	Data block containing two values: Current HTTP error rate, in errors per second. Highest HTTP error rate seen, in requests per second, since the HAProxy Enterprise node was started or since the counters were reset.

SSL Connections	Description
SSL Connections - Rate	Data block containing two values: Current SSL connection rate, in requests per second. Maximum allowed SSL connection rate, in requests per second.
SSL Connections - #Conn	Data block containing two values: Current number of active SSL connections. Maximum allowed number of active SSL connections.
SSL Connections - Total	Data block containing two values: Total SSL connections since the HAProxy Enterprise node was started or since the counters were reset. Highest SSL connection rate seen, in requests per second, since the HAProxy Enterprise node was started or since the counters were reset.

TCP Connections	Description
TCP Connections - Rate	Data block containing two values: Current TCP connection rate, in requests per second. Maximum allowed TCP connection rate, in requests per second.
TCP Connections - #Conn	Data block containing two values: Current number of active TCP connections. Maximum allowed number of active TCP connections.
TCP Connections - Total	Data block containing two values: Total TCP connections since the HAProxy Enterprise node was started or since the counters were reset. Highest TCP connection rate seen, in requests per second, since the HAProxy Enterprise node was started or since the counters were reset.
HAProxy Enterprise Real-time Dashboard Copyright and Version	Page footer containing the version of the HAProxy Enterprise Real-time Dashboard. The most recent released version is **v1.0**.

Stats pane

The **Stats** pane displays detailed real time information about the HAProxy Enterprise frontends and backends, and allows you to perform management actions on servers.

Filter	Description
Filter - Frontend/ Backend Name	Filters the list and only displays frontend and backend sections whose name matches filter criteria.
Filter - Server Name Full or partial server name	Filters the list and only displays servers (and their corresponding frontend or backend sections) whose name matches the specified filter.
Filter - Server State List of server states	Filters the list and only displays servers (and their corresponding frontend or backend sections) whose state matches one of the selected states. Holding **Ctrl** while clicking on a particular state will deselect or select all other states for convenience.
Filter - Action to perform	Select the action to perform on all servers visible after filtering (see Manage servers)

The checkbox left of the frontend/backend/listen name selects all servers in the block for the Server Management ------ Actions at the bottom. For each server line:

Field	Description
Server name	Name of the server (svname)
Queue - Cur	Current number of requests in server's queue
Queue - Max	Maximum value of current queued requests
Queue - Limit	Configured maximum queue for the server
Session Rate - Cur	Number of sessions per second over last elapsed second
Session Rate - Max	Maximum number of new sessions per second
Session Rate - Limit	Configured limit on new sessions per second
Sessions - Cur	Current number of sessions
Sessions - Max	Maximum number of sessions
Sessions - Limit	Configured session limit
Sessions - Total	Cumulative number of sessions
Sessions - LbTot	Total number of times the server was selected
Bytes - In	Cumulative bytes in to server
Bytes - Out	Cumulative bytes out from server
Denied - Req	Requests denied due to security concerns
Denied - Resp	Responses denied due to security concerns
Error - Req	Request errors
Error - Conn	Number of requests that encountered an error while trying to connect to a backend server
Error - Resp	Response errors
Warnings - Retr	Number of times a connection to a server was retried
Warnings - Redis	Number of times a request was redispatched to another server
Server - Status	Current status and time elapsed since last UP ↔ DOWN transition
Server - LastChk	Status of last health check
Server - Wght	Total weight (backend), or server weight (server)
Server - Act	Number of active servers (backend)
Server - Bck	Number of backup servers (backend)
Server - Chk	Failed Health Checks

Field	Description
Server - Dwn	Number of UP → DOWN transitions
Server - Dwntime	Total downtime (backend)
Server - Throttle	Current throttle percentage for the server
Actions	Action on server: Enable - Set server to READY. Disable - Set server to MAINT.

Manage servers

You can perform specific management tasks on selected frontends, backends and listen blocks.

Perform an action on server(s)

1. Select a server by clicking on the checkbox next to its name. To select all servers in a block, click the checkbox next to the block name.
2. Select a task from the drop-down list (see below).
3. Click **Apply**.

Statistics	Description
Set state to READY	Set server state to READY (Enable Server)
Set state to DRAIN	Set server state to DRAIN (remove from load-balancing, but remain available for health checks and persistent connections)
Set state to MAINT	Set server state to MAINT (Disable Server)
Health: disable checks	Mark primary server health checks as temporarily stopped
Health: enable checks	Enable primary server health checks
Health: force UP	Force server health to UP (regardless of pending checks)
Health: force NOLB	Force server health to NOLB state (regardless of slow checks)
Health: force DOWN	Force server health to DOWN (regardless of pending checks)
Agent: disable checks	Mark auxiliary agent checks as temporarily stopped
Agent: enable checks	Enable auxiliary agent checks
Agent: force UP	Force server agent state to UP
Agent: force DOWN	Force server agent state to DOWN
Kill Sessions	Immediately terminate all server sessions

Response body injection

This page applies to:

- HAProxy Enterprise 1.7r2 and newer

The Response Body Injection (RBI) module allows you to insert content into an HTTP response before it is sent back to the client. It can insert content into any of the following HTML sections:

- `<html>`
- `<head>`
- `<title>`
- `<body>`

Install the RBI module

1. Install the RBI module as follows, depending on your platform:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-htmldom
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-htmldom
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-htmldom
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-htmldom
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-htmldom
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-htmldom
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-htmldom
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-htmldom
```

2. In the `global` section of your HAProxy Enterprise configuration file, add a `module-load` line:

```
global
  module-load hapee-lb-htmldom.so
```

3. Add a `filter htmldom` directive to a `listen`, `frontend`, or `backend` section to begin injecting content.

   In this example configuration, we replace all `http://` links in the page with `https://` by injecting a Javascript `<script>` tag inside the `<head>` section.

   The script does the work of updating the links on the page.

```
frontend www
  filter htmldom mode strict head-append '<script type="text/javascript"
defer="defer">window.addEventListener("load", function() {var alinks = document.getElementsByTagName("a");var acount
= alinks.length;for (var i=0; i < acount; i++) {alinks[i].href = alinks[i].href.replace("http://","https://");}})</
script>'
```

   Here is the same script, formatted for easier reading.

```
<script type="text/javascript" defer="defer">
    window.addEventListener("load", function() {
        var alinks = document.getElementsByTagName("a");
        var acount = alinks.length;

        for (var i=0; i < acount; i++) {
            alinks[i].href = alinks[i].href.replace("http://","https://");
        }
    })
</script>
```

When you insert a large amount of content into a response, consider increasing the size of the global performance tuning parameter `tune.bufsize`. See tune.bufsize.

Caveats

The RBI filter automatically deactivates under certain conditions, including:

- the request method is `HEAD`.
- both request and response are HTTP/1.1, but the `Transfer-Encoding` header value is not `chunked` and `Content-Length` is unknown.
- the response is compressed (i.e. the `Content-Encoding` header is present).
- the response contains the `Cache-Control` header set to `no-transform` (no transformations or conversions can happen).
- the `Content-Type` header is set to a value other than `text/html`.

Also, note that:

- you can only use one `htmldom` filter per listener/frontend/backend.
- due to the current limitation of the filter's implementation, we do not recommended that you use this filter in combination with other filters that modify the response on the same listener/frontend/backend. Using them can cause undefined behavior (for example, the compression filter).
- if you encounter problems with the response body manipulation, check your HAProxy Enterprise logs. The RBI filter logs problems related to the buffer operations, such as initialization and manipulation.

Changed HTTP headers

The `filter htmldom` directive modifies the response in the following ways:

- It removes all occurrences of the `Accept-Encoding` header from the incoming request to ensure that the response does not get compressed.
- It removes all occurrences of the `Content-Length` header in order to modify the response.
- If the message is HTTP/1.1 or above, it sets the `Transfer-Encoding` header to `chunked`.

Filter parameters

The `filter htmldom` directive instructs HAProxy Enterprise to inject content into an HTML document. You can declare it only once for each proxy section. Its syntax is:

```
# Used in the a frontend, listen, or backend section
filter htmldom [mode <mode>] <action> <content>
```

where:

Directive	Description
`mode <mode>`	Sets the HTML parser in either of these modes: `relaxed` (default): Faster and intended for use cases where you know what to expect in the server response (the HTML document is simple). `strict`: Uses `libxml2` and performs more parsing checks. We recommend this in cases where the document may contain many JavaScript declarations or if there may be broken tags.
`<action>`	**(required)** Identifies the HTML element in which to inject new content. It applies to the first matching element. Valid actions are: `html-prepend`: Insert content at the beginning of the `<html>` element. `html-append`: Insert content at the end of the `<html>` element. `head-prepend`: Insert content at the beginning of the `<head>` element. `head-append`: Insert content at the end of the `<head>` element. `title-prepend`: Insert content at the beginning of the `<title>` element. `title-append`: Insert content at the end of the `<title>` element. `body-prepend`: Insert content at the beginning of the `<body>` element. `body-append`: Insert content at the end of the `<body>` element.
`<content>`	**(required)** The HTML content to inject into the response from a server. It is a `log-format` string.

Route health injection

This page applies to:

- HAProxy Enterprise - all versions

The Route Health Injection (RHI) module monitors your load balancer's connectivity to backend servers and has the ability to remove the entire load balancer from duty if the load balancer can suddenly not reach those servers. The idea is that if a load balancer can't reach the servers, and you're running an active/active load balancer pair, then you can deactive the problematic load balancer and route all traffic to the other, healthy load balancer.

The RHI module is meant to work with IP routing protocols, such as BGP and OSPF, configured for Equal-cost multi-path (ECMP) routing. ECMP enables the network to route traffic to a destination over multiple paths, allowing you to relay IP packets to both of your load balancers in parallel. The ability to detect problems and route traffic away from unhealthy load balancers is important for making ECMP resilient.

Concepts

This section describes the concepts behind route health injection combined with ECMP.

How ECMP routing works

Your router attempts to send packets to their destination using the most efficient network path. When two network paths have identical costs, meaning they are equally good, then the router can load balance traffic across both paths given that it supports ECMP. With ECMP, you can configure the router to see both of your load balancers as being different, but equal, routes to the destination IP address. So the router can send traffic to both load balancers in parallel, achieving high availability.

How do the two load balancers present themselves to the router as being routes of equal costs? They present routes to the same IP address as passing through themselves and inject those routes into the router's routing table. In essence, they become gateways for reaching the same IP address.

A router on the `192.168.0.0/24` network sees:

Destination address	Route via
`192.168.1.10`	* Load balancer 1 at `192.168.0.101` or * Load balancer 2 at `192.168.0.102`

The RHI module shares routes with the router. It does this by installing BIRD Internet Routing Daemon, which is open-source software, onto the load balancer. BIRD shares routes by using either the BGP or OSPF protocol.

While to the router it looks as through the load balancers are simply the next hop towards the destination, actually the load balancers are the final hop. They own the destination IP address. Having both load balancers bound to the same IP address could cause conflicts on the network, though. Two ways to solve this problem that we'll cover are:

- Add a second address to the load balancer's loopback network interface and disable ARP so that this address isn't advertised on the network. That way, only the load balancer sees this address and essentially sends the traffic to itself.
- Intercept traffic destined for a non-locally bound address by configuring transparent proxying.

How Route Health Injection works

The RHI module adds this load balancer as a route in BIRD's configuration using a custom routing table named volatile. BIRD then broadcasts these routes to peer routers using the BGP or OSPF protocol. If either a frontend or a backend is down, then the RHI module removes the route from the volatile table, notifying BIRD to stop advertising this load balancer as a route on the network, diverting the flow of traffic to the other load balancer in the active/active cluster. You can configure ECMP on your router to load balance traffic to both load balancers via the advertised routes.

Prerequisites

Ensure that you've met the following prerequisites:

- Your router has enabled ECMP. Consult your router's documentation for details.

Install and configure RHI

In this section, you will learn how to set up route health injection.

Install the RHI module

To install the RHI module, perform these steps on each load balancer:

1. Install the RHI module using your package manager:

Apt:

```
sudo apt-get install hapee-extras-rhi
```

Yum:

```
sudo yum install hapee-extras-rhi
```

Zypper:

```
sudo zypper install hapee-extras-rhi
```

Pkg:

```
sudo pkg install hapee-extras-rhi
```

This installs the hapee-extras-route package too, which is our version of the BIRD Internet Routing Daemon. The daemon is stored as `/opt/hapee-extras/sbin/hapee-route`.

2. Create a socket for the Runtime API.

The RHI module needs to connect to the Runtime API to collect information about the health of your frontends and backends. Add a new `stats socket` line to the `global` section of your HAProxy Enterprise configuration. This exposes the Runtime API as the socket `/var/run/hapee-extras/hapee-lb.sock`:

hapee-lb.cfg

```
global
  stats socket /var/run/hapee-extras/hapee-lb.sock user hapee-lb group hapee mode 660
```

Use the correct socket path

Your load balancer configuration likely contains a `stats socket` line in the `global` section already, but if it isn't using the file path shown here, add a second `stats socket` line or else it won't work.

3. Optional: Change the socket path

The RHI module expects the Runtime API socket to be `/var/run/hapee-extras/hapee-lb.sock`. However, you can change the path that the RHI module expects by setting the variable `HAPEE_LB_SOCKET` in the following file:

- On Debian/Ubuntu: `/etc/default/hapee-extras-rhi`
- On RHEL: `/etc/sysconfig/hapee-extras-rhi`

For example:

hapee-extras-rhi

```
HAPEE_LB_SOCKET="/var/run/hapee-3.1/hapee-lb.sock"
```

Still, you must ensure that this same path is configured in your HAProxy Enterprise configuration file:

hapee-lb.cfg

```
global
  stats socket /var/run/hapee-3.1/hapee-lb.sock user hapee-lb group hapee mode 660
```

Configure route health injection

To configure the RHI module to share routes with your router:

1. Edit the file `/etc/hapee-extras/hapee-rhi.cfg`. The default configuration contains an example:

hapee-rhi.cfg

```
# Inject the 10.200.200.200/32 address into the route daemon if
# all the backends "be_static" and "be_app" are up.
10.200.200.200/32 = all(b:be_static,b:be_app)
```

This file contains a list of routes that the RHI module will add to BIRD's volatile table, but only if the given rule returns true. A rule checks the status of one or more frontends or backends to see if they are up or down. A backend is treated as down if all servers fail their health checks or if you manually disable the servers. A frontend is down if you disable it manually.

In this example, the rule uses `all` function to announce the `192.168.1.10/32` IP address only when both the `be_static` and `be_app` backend are up and running. When the condition is false, the IP address is removed from the list of advertised routes.

Let's look at another example. The following line uses the `any` function to advertise the IP if either the `be_app` or `be_app2` backends are up and running:

hapee-rhi.cfg

```
192.168.1.10/32 = any(b:be_app,b:be_app2)
```

2. After making changes to the `hapee-rhi.cfg` file, save the file. Then enable and restart the service:

```
sudo systemctl enable hapee-extras-rhi
sudo systemctl restart hapee-extras-rhi
```

3. To configure BIRD for BGP or OSPF, which are used to advertise routes to peer routers, edit the file `/etc/hapee-extras/hapee-route.cfg`.

- Add a section for either BGP or OSPF, depending on which protocol you intend to use for advertising routes to peers.

- Use BGP

  An example BGP configuration section:

  hapee-route.cfg

  ```
  protocol bgp r1 {
  local 192.168.0.101 as 65001;
  neighbor 192.168.0.1 as 65001;
  graceful restart on;
  import none;

  # advertise the IP route
  export where proto = "vol1";
  }
  ```

  In this example:

  - The `local` directive refers to the IP address assigned to this load balancer's network interface and assigns the Autonomous System Number 65001.
  - The `neighbor` directive refers to the layer 3 device, such as the gateway router, with which we are establishing a BGP session.
  - The `export` line advertises routes from the volatile table, `vol1`.

- Use OSPF

  An example OSPF configuration section:

hapee-route.cfg

```
protocol ospf anycast {
  tick 2;
  import none;

  # advertise the IP route
  export where proto = "vol1";

  area 0.0.0.0 {
    stub no;
    interface "eth0" {
       hello 10;
       retransmit 6;
       cost 10;
       transmit delay 5;
       dead count 4;
       wait 50;
       type broadcast;
    };
  };
}
```

- The `export` line advertises routes from the volatile table, `vol1`.

4. After making changes to the `hapee-route.cfg` file, save the file. Then restart the service:

```
sudo systemctl restart hapee-extras-route
```

5. Optional: To advertise IPv6 routes, repeat these steps for the `hapee-extras-route6` service.

6. Verify that RHI added a route to BIRD by calling the `show route` command. The IP should display.

```
sudo /opt/hapee-extras/bin/hapee-route-cli show route
```

output

```
BIRD 1.6.3 ready.
192.168.1.10/32    dev auto [vol1 18:54:15] * (0)
```

When you disable all servers in the backend, the command should not return this route.

Intercept traffic destined for the IP

Remember that both of your load balancers must be able to receive packets at the same IP address defined in the route shared with your router, for example `192.168.1.10/32`, but without causing a conflict on the network. Here are two ways to accomplish that:

- Add a second address to the load balancer's loopback network interface

  The IP must be handled on each server's loopback interface to accept connections, but can't be advertised on the network or it will be identified as an IP conflict by some network components. To avoid IP address conflicts, disable ARP for IP addresses managed by the loopback interface.

  Perform these steps on both load balancers:

  1. Edit the HAProxy Enterprise configuration file, `/etc/hapee-3.1/hapee-lb.cfg`:

     hapee-lb.cfg

     ```
     frontend www
       bind 192.168.1.10:80 name http
       bind 192.168.1.10:443 name https ssl crt site.pem
     ```

     - In the `frontend` section, define one or more `bind` lines that listen at the IP address you created a route for, such as `192.168.1.10`. This address should not be assigned to any network interfaces.
     - Each load balancer should be assigned the same IP address.

  2. Save the changes and then reload the service.

     ```
     sudo systemctl reload hapee-3.1-lb
     ```

  3. Manage the IP address through a loopback interface.

Debian:

Edit the file `/etc/network/interfaces`. Add a new `iface` section for the `lo` interface and add the address under it:

interfaces

```
# The loopback network interface
auto lo
iface lo inet loopback

iface lo inet static
  address 192.168.1.10/32
```

Ubuntu:

Edit the netplan YAML configuration file located in `/etc/netplan`. The configuration file is probably the one having the lowest number and has a name like `00-installer-config.yaml` or `01-netcfg.yaml`.

Edit the netplan YAML file, adding an `lo` section under the `ethernets` level:

01-netcfg.yaml

```
network:
  ethernets:
    lo:
      dhcp4: false
      addresses:
      - "192.168.1.10/32"
```

Then use `sudo netplan try` and `sudo netplan apply` before rebooting to make sure the configuration is valid. Ignore warnings about Open vSwitch.

RHEL:

To persist the IP address on RHEL 9.2 or newer, use NetworkManager. Previous versions did not support managing the loopback interface with NetworkManager.

```
sudo nmcli connection modify lo +ipv4.addresses 192.168.1.10/32
sudo nmcli con up 'lo'
```

To persist the IP address on RHEL systems older than 9.2, create a new service for loading it at boot:

```
sudo touch /etc/systemd/system/01-static-ip.service
sudo vi /etc/systemd/system/01-static-ip.service
```

Add the following lines to the service file:

01-static-ip.service

```
[Unit]
Description=Add static IP to loopback
Wants=network-online.target
After=network-online.target

[Service]
Type=oneshot
# create the address
ExecStart=-/usr/sbin/ip address add 192.168.1.10/32 dev lo

[Install]
WantedBy=multi-user.target
```

Set the service to start on boot:

```
sudo systemctl enable 01-static-ip.service
```

4. Add the following lines to `/etc/sysctl.conf`.

```
net.ipv4.conf.all.arp_ignore=1
net.ipv4.conf.all.arp_announce=2
```

5. Restart the HAProxy Enterprise server.

- Configure transparent proxying

Perform these steps on both load balancers to enable transparent proxying:

1. Edit the HAProxy Enterprise configuration file, `/etc/hapee-3.1/hapee-lb.cfg`:

   hapee-lb.cfg

   ```
   frontend www
     bind 192.168.1.10:80 name http transparent
     bind 192.168.1.10:443 name https ssl crt site.pem transparent
   ```

   - In the `frontend` section, define one or more `bind` lines that listen at the IP address you created a route for, such as `192.168.1.10`. This address should not be assigned to any network interfaces.
   - Because the IP address is not configured on the network interface, add the `transparent` argument. This indicates that the IP address should be bound even though it does not belong to the local machine. Packets targeting this address will be intercepted as if the address were locally configured. This feature uses the Linux kernel's TPROXY feature.
   - Each load balancer should be assigned the same IP address.

2. Save the changes and then reload the service.

   ```
   sudo systemctl reload hapee-3.1-lb
   ```

3. Add firewall rules that intercept packets that have a destination IP address matching a listening socket, which in this case is our transparent proxied IP address. Also add policy-based routing rules to deliver the traffic locally.

Debian/Ubuntu:

Create firewall and routing rules:

```
sudo iptables -t mangle -N DIVERT
sudo iptables -t mangle -A PREROUTING -p tcp -m socket -j DIVERT
sudo iptables -t mangle -A PREROUTING -p udp -m socket -j DIVERT
sudo iptables -t mangle -A DIVERT -j MARK --set-mark 1
sudo iptables -t mangle -A DIVERT -j ACCEPT
sudo ip rule add fwmark 1 lookup 100
sudo ip route add local 0.0.0.0/0 dev lo table 100
```

To make the `iptables` changes persistent after reboot, use the `iptables-save` command. It saves the changes and configures the system to restore them at reboot.

```
sudo apt install iptables-persistent
sudo su -c 'iptables-save > /etc/iptables/rules.v4'
```

To verify the IP tables rules, use the `iptables` command. Note that the output may show `tcp` or the number 6 and `udp` or the number 17:

```
sudo iptables -L -v -n -t mangle
```

output

```
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source               destination
1941  335K DIVERT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            socket
   0     0 DIVERT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            socket

...

Chain DIVERT (1 references)
pkts bytes target     prot opt in     out     source               destination
1941  335K MARK       all  --  *      *       0.0.0.0/0            0.0.0.0/0            MARK set 0x1
1941  335K ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

To make the policy route (`ip rule`) and route table (`ip route`) changes persist after reboot, your next step depends on whether or not your system uses netplan. Typically, Ubuntu uses it, but Debian does not.

- If your system uses netplan, persist the policy route (`ip rule`) and route table (`ip route`) changes in the netplan YAML configuration file located in `/etc/netplan`. The configuration file is probably the one having the lowest number and has a name like `00-installer-config.yaml` or `01-netcfg.yaml`.

  Edit the netplan YAML file, adding an `lo` section under the `ethernets` level:

  01-netcfg.yaml

  ```
  network:
    ethernets:
      lo:
        routing-policy:
        - to: 0.0.0.0/0
          mark: 1
          table: 100
        routes:
        - to: 0.0.0.0/0
          type: local
          table: 100
  ```

  Then use `sudo netplan try` and `sudo netplan apply` before rebooting to make sure the configuration is valid. Ignore warnings about Open vSwitch.

- If your system does not use netplan, persist the changes by creating a new service for loading them at boot. Create the `systemd` service file:

  ```
  sudo touch /etc/systemd/system/01-static-route.service
  sudo vi /etc/systemd/system/01-static-route.service
  ```

  Add the following lines to the service file:

01-static-route.service

```
[Unit]
Description=Add route table 100
Wants=network-online.target
After=network-online.target

[Service]
Type=oneshot
# create the route table and rule
ExecStart=-/usr/sbin/ip route add local 0.0.0.0/0 dev lo table 100
ExecStart=-/usr/sbin/ip rule add fwmark 1 lookup 100

[Install]
WantedBy=multi-user.target
```

Set the service to start on boot:

```
sudo systemctl enable 01-static-route.service
```

Restart the system. The saved settings will be restored after the restart.

RHEL:

Create firewall rules:

```
sudo firewall-cmd --permanent --direct --add-chain ipv4 mangle DIVERT
sudo firewall-cmd --permanent --direct --add-rule ipv4 mangle PREROUTING 0 -p tcp -m socket -j DIVERT
sudo firewall-cmd --permanent --direct --add-rule ipv4 mangle PREROUTING 0 -p udp -m socket -j DIVERT
sudo firewall-cmd --permanent --direct --add-rule ipv4 mangle DIVERT 0 -j MARK --set-mark 1
sudo firewall-cmd --permanent --direct --add-rule ipv4 mangle DIVERT 1 -j ACCEPT
```

Reload the firewall:

```
sudo firewall-cmd --reload
```

To persist the IP routing rules, create a new service for loading them at boot:

```
sudo touch /etc/systemd/system/01-static-route.service
sudo vi /etc/systemd/system/01-static-route.service
```

Add the following lines to the service file:

01-static-route.service

```
[Unit]
Description=Add route table 100
Wants=network-online.target
After=network-online.target

[Service]
Type=oneshot
# create the route table and rule
ExecStart=-/usr/sbin/ip route add local 0.0.0.0/0 dev lo table 100
ExecStart=-/usr/sbin/ip rule add fwmark 1 lookup 100

[Install]
WantedBy=multi-user.target
```

Set the service to start on boot:

```
sudo systemctl enable 01-static-route.service
```

Restart the system.

To verify the rule table, use the `ip` command:

```
sudo ip rule ls
```

output

```
...
32765:  from all fwmark 0x1 lookup 100
...
```

To verify the route table, use the `ip` command:

```
sudo ip route ls table 100
```

output

```
local default dev lo scope host
```

At this point, you have completed the setup of the Route Health Injection module.

View the BIRD volatile table

To see the volatile table definition:

1. Edit the file `/etc/hapee-extras/hapee-route.cfg`.
2. Scroll down to the `protocol volatile` section.

hapee-route.cfg

```
protocol volatile vol1 {
  #  gateway <ip>
}
```

By default, BIRD announces routes through the gateway configured on the network interface, but you can specify a different network gateway by uncommenting the `gateway` directive and typing its IP address.

For example:

hapee-route.cfg

```
protocol volatile vol1 {
  gateway 192.168.1.244
}
```

You can add more volatile tables to support advertising routes for different frontends:

hapee-route.cfg

```
protocol volatile vol1 {
}

protocol volatile vol2 {
}
```

Then, prefix each route in the RHI module's configuration, `/etc/hapee-extras/hapee-rhi.cfg`, with a table's name:

hapee-rhi.cfg

```
vol1%192.168.1.10/32 = all(b:be_static,b:be_app)
vol2%192.168.1.11/32 = any(b:k8s_servers)
```

Reference

This section describes the options available when configuring the RHI module.

RHI configuration file

This section describes the syntax of the RHI configuration file, `/etc/hapee-extras/hapee-rhi.cfg`.

```
<network>[,<network>,[...]] = <agg>(<b:|f:><name>[,<b:|f:><name>,[...]])
```

Argument	Description
`<network>`	`[%<protoname>]{<ipv4>,<ipv6>}[/<mask>]` Specify an IPv4 or IPv6 CIDR subnet or list of several comma-delimited subnets. If you do not specify any subnet mask, RHI applies the /32 mask. For advanced configuration, you can supply the name of a volatile table in the `%<protoname>` section (default is `vol1`).
`<agg>`	Aggregation function: `all` - Returns true if all listed proxies are active. `any` - Returns true if at least one of the proxies listed is active. `never` - Always false. For debugging purposes.
`<b:|f:>`	Prefix of either `b` for backend or `f` for frontend.
`<name>`	Name of the frontend or backend.

See also

- For more information on BIRD configuration, protocol support, and other technical details, see The BIRD Internet Routing Daemon official website.
- To bind to addresses even if they do not belong to the local server, see transparent reference.

Send metrics

This page applies to:

- HAProxy Enterprise 1.8r1 and newer

The Send Metrics module streams HAProxy Enterprise metrics to an external program. You can configure which fields to include in the data.

Install the Send Metrics module

1. Install the module using your package manager:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-send-metrics
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-send-metrics
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-send-metrics
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-send-metrics
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-send-metrics
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-send-metrics
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-send-metrics
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-send-metrics
```

2. Update the `global` section of your configuration to send the data:

```
global
  module-path /opt/hapee-3.1/modules
  module-load hapee-lb-send-metrics.so
  send-metrics-url POST http://192.168.0.1:8000/metrics/  xdelay 1m 5s 1s 1s  timeout 100ms  retries 3  log
  send-metrics-content-type application/json
  send-metrics-data '{ "Hostname": "%H", "Date": "%T", "connections": "%ac" }'
  send-metrics-header 'X-APIKey: abcd1234'
```

In this example, we:

- sends data at a 1 minute interval to the URL `http://192.168.0.1:8000/metrics/` using an HTTP `POST` request.
- format the data as JSON.
- sends the server's hostname, the date, and the current number of active connections.
- attaches an HTTP header named `X-APIKey` to the request.

See the section below for an explanation of each directive and its parameters.

Global directives

The Send Metrics module enables the following directives in the `global` section.

send-metrics-url

The `send-metrics-url` directive is required. It tells the load balancer to send metrics data over HTTP to a specified URL.

Syntax:

```
send-metrics-url POST <url> [delay <u> | xdelay <u s b r>] [timeout <t>] [retries <n>] [log] [dontlog-normal]
```

where:

Argument	Description
`post <url>`	**Required**. Specifies the URL to send metrics data to.
`delay <u>`	`<u>` Specifies the period between each attempt to send new data metrics. The `delay` keyword is a simplified version of the `xdelay` keyword.
`xdelay <u s b r>`	`<u>` specifies the period between each attempt to send new data metrics. If the module cannot send the metrics data after three attempts, it cancels the update until the next time interval defined by `<u>`. It defaults to 60m. `<s>` specifies the initial (first) delay to send the data. It defaults to 5s. `<b>` is not used in this module and its value is not important. It defaults to 10s. If the data fails to send, `<r>` determines the delay for the next attempt. It defaults to 5s.
`timeout <t>`	Specifies the HTTP connection timeout for attempts to send new data metrics. The value is in milliseconds by default, but you can set it to any other unit if you add it as a suffix to the number. Default: 5s
`retries <n>`	Specifies number of retries to send new data metrics. If unspecified, HAProxy Enterprise uses the global `retries` value. Default: 3s
`log`	Specifies whether to log operation errors.
`dontlog-normal`	Deactivates logging for successful updates.
`param*`	A list of other `server` line arguments. This is useful for configuring special TLS features.

send-metrics-data

The `send-metrics-data` directive is required. It sets the data to send to the selected HTTP server. You can set individual log variables within the data, prefixed with `%`.

send-metrics-content-type

- The `send-metrics-content-type` directive allows the module to set the `Content-Type` header when sending data, including and limited to the content types listed below. If unspecified, the module uses the `application/octet-stream` Content-Type.

Supported content types are:

- `application/json`
- `application/octet-stream`
- `application/x-www-form-urlencoded`
- `text/plain`

send-metrics-header

The `send-metrics-header` directive allows the module to set additional HTTP headers within the HTTP `POST` request.

send-metrics-debug

The `send-metrics-debug` directive sets the debug level. Use this only when the module runs in debug mode; in normal use, it has no significance. Default: level 7.

Runtime API

The following Runtime API commands are available:

lb-send-metrics show-data

The `lb-send-metrics show-data` command serves to check the operation of the module by printing the content the module sent to the selected HTTP server.

lb-send-metrics status

The `lb-send-metrics status` command displays the module's status.

lb-send-metrics update

The `lb-send-metrics update [delay]` command runs the update at a time specified with the `delay` setting. If unspecified (or the delay is 0), the update executes immediately. The delay cannot exceed the time until the next regular update.

lb-send-metrics debug

The `lb-send-metrics debug [level]` command sets the debug level. You can use this only when the module runs in debug mode; in normal use it has no significance. Default: level 7.

Single sign-on

Single sign-on (AD FS Web Proxy)

This page applies to:

- HAProxy Enterprise 3.1r1 and newer

This ADFSPIP module enables HAProxy Enterprise to serve as a reverse proxy in front of your Active Directory Federation Services (AD FS) federation server. AD FS authenticates users so that they can access web applications running inside your corporate network. HAProxy Enterprise gives clients outside your corporate network access to those web applications, after they've signed in via AD FS. You will typically deploy HAProxy Enterprise at the edge of your network where external clients can access it.

Using HAProxy Enterprise as a web proxy in front of AD FS serves the same job as and replaces the Web Application Proxy on Windows Server. The module uses the Active Directory Federation Services and Proxy Integration Protocol to communicate with AD FS.

Essential concepts include:

- Upon startup, HAProxy Enterprise connects to your AD FS federation server and establishes a trust relationship.
- When HAProxy Enterprise receives an HTTP requests from an unauthenticated client, it redirects the client to the federation server URL to sign in. Because external clients won't have direct access to the federation server, this request will go through HAProxy Enterprise. As such, you'll need to configure split-brain DNS so that the federation server URL points to HAProxy Enterprise for external clients, but to the true AD FS server for internal clients.
- After successful sign in through AD FS, the client returns with an authentication token to the web application URL it originally requested.
- HAProxy Enterprise validates the authentication token using the AD FS server's token signing certificate. It then relays the client's requests to the web application.
- You must publish each web application for which you want to enable AD FS authentication. Publishing entails adding the application as a Relying Party Trust in AD FS, which adds it to the list of proxied services.

Prerequisites

Please check that you've met the following prerequisites:

- You're running Windows Server 2016 or later.
- Your Windows Server has the Active Directory Domain Services role and has created an Active Directory forest. See the Windows Server guide Install Active Directory Domain Services.
- Your Windows Server has the Active Directory Federation Services (AD FS) role. See the Windows Server guide Install the AD FS Role Service.

> **Use a wildcard certificate**
>
> While configuring AD FS, you'll be asked to select an SSL certificate. To reduce the number of certificates, consider buying a wildcard certificate that will match all of your subdomains. It would have a CN value like `*.example.com`.

- Optional: If you plan to use this server as a DNS server, add the DNS role. See the Windows Server guide Installing DNS Server.

Installation and setup

In this section, you'll learn how to install and set up the module.

1. Configure split-brain DNS

To access web applications, users will sign in via the AD FS federation server's sign-in page at a URL like `https://adfs.example.com/adfs/ls?version=1.0&action=signin`. That domain name should resolve to an IP address belonging to HAProxy Enterprise so that the load balancer can proxy the sign-in requests.

Configure your DNS server for split-brain DNS so that the domain name resolves differently for internal and external clients. Learn more in the Windows Server guide Use DNS Policy for Split-Brain DNS Deployment.

- HAProxy Enterprise should resolve the federation server's domain name, for example `adfs.example.com`, to the internal IP address of the AD FS server. Later, you'll see how to configure a `resolvers` section in your HAProxy Enterprise configuration to use your DNS nameserver.
- External clients should resolve the domain name, again `adfs.example.com`, to the public IP address of HAProxy Enterprise.
- External clients should resolve the web application's external URL, such as `www.example.com` to the public IP address of HAProxy Enterprise.

For testing purposes, you can simply set the `hosts` file on the client.

2. Collect certificates

The AD FS server has certificates that you'll need to copy to your HAProxy Enterprise server before it can function as a web proxy in front of AD FS. You'll get these certificates:

Certificate	Description
Token signing certificate	The federation server uses this certificate's associated public/private key pair to digitally sign all authentication tokens that it produces. HAProxy Enterprise needs only the public key to validate tokens.
Client certificate	HAProxy Enterprise authenticates to the federation server by presenting this certificate.

On the AD FS server:

1. Get the token signing certificate, which the AD FS server uses to digitally sign authentication tokens it produces. HAProxy Enterprise will use the public key portion of the certificate to validate the authentication tokens, also known as JSON Web Tokens (JWT), that clients receive from AD FS after authenticating.

   - Open **Server Manager** and go to **Tools > AD FS Management**.
   - Expand **Service** and select **Certificates**.
   - Right click the `ADFS Signing` certificate and select **View certificate**.
   - From the **Details** tab of the **Certificate** dialog, select **Copy to File**.
   - Use the Certificate Export Wizard to export the certificate as a DER encoded binary X.509 (.CER) file. You don't need to export its private key.
   - Transfer this file to the HAProxy Enterprise server and place it in a directory such as `/etc/hapee-3.1/adfs-signing-cert.cer`.
   - Convert the file to the PEM format:

     ```
     sudo openssl x509 -inform der -in adfs-signing-cert.cer -out adfs-signing-cert.pem
     ```

   - Extract the public key from the PEM file:

     ```
     openssl x509 -pubkey -noout -in ./adfs-signing-cert.pem | sudo tee adfs-signing-pubkey.pem
     ```

   Auto-rotated certificates

   By default, AD FS rotates the token signing certificate after an interval, which means you'll need to repeat this process routinely. You can change the rotation interval or disable this feature and manage the rotation manually. To learn more, see the Windows guide Obtain and configure TS and TD certificates for AD FS.

2. Each time that HAProxy Enterprise relays a client to the AD FS sign-in page, it must attach its own client certificate so that the connection will be accepted. To generate a client certificate, first create a CA certificate for your organization that you'll use to sign client certificates. Use the `New-SelfSignedCertificate` PowerShell command to create a CA certificate for your organization. Here, we create a self-sign CA certificate, save it to the local computer's My store, then export it as a file:

```
$ca = New-SelfSignedCertificate -Type Custom `
  -KeySpec Signature `
  -Subject "CN=Example CA" `
  -KeyExportPolicy Exportable `
  -HashAlgorithm sha256 `
  -KeyLength 2048 `
  -CertStoreLocation cert:\LocalMachine\My `
  -KeyUsageProperty Sign `
  -KeyUsage CertSign `
  -NotAfter (Get-Date).AddMonths(120)

$password = ConvertTo-SecureString -String "P@ssword12345" -Force -AsPlainText
Export-PfxCertificate -Cert $ca -FilePath C:\example-ca.pfx -Password $password
```

3. Import the CA certificate into the Client Authentication Issuers certificate store:

```
$password = ConvertTo-SecureString -String "P@ssword12345" -Force -AsPlainText
Import-PfxCertificate -FilePath C:\example-ca.pfx -CertStoreLocation cert:
\LocalMachine\ClientAuthIssuer -Exportable -Password $password
```

4. Create a client certificate and sign it with the CA certificate you just created:

```
$client = New-SelfSignedCertificate -Type Custom `
  -KeyUsage DigitalSignature `
  -Subject "CN=HAProxy Enterprise" `
  -KeyExportPolicy Exportable `
  -HashAlgorithm sha256 `
  -KeyLength 2048 `
  -CertStoreLocation cert:\LocalMachine\My `
  -Signer $ca `
  -TextExtension @("2.5.29.37={text}1.3.6.1.5.5.7.3.2") `
  -NotAfter (Get-Date).AddMonths(120)

$password = ConvertTo-SecureString -String "P@ssword12345" -Force -AsPlainText
Export-PfxCertificate -Cert $client -FilePath C:\client.pfx -Password $password
```

5. Transfer the `client.pfx` file to the HAProxy Enterprise server and convert it to a PEM file:

```
sudo openssl pkcs12 -in ./client.pfx -out ./client.pem -nodes
```

3. Establish a connection

HAProxy Enterprise must connect to the AD FS server and use Active Directory credentials to form a trust relationship.

On the AD FS server:

1. If you previously deployed a Web Application Proxy, remove the relying party trust it created by opening a PowerShell terminal as an administrator and running this command:

```
Remove-AdfsWebApplicationProxyRelyingPartyTrust
```

2. Create a new web application proxy relying party trust for HAProxy Enterprise:

```
Add-AdfsWebApplicationProxyRelyingPartyTrust -Identifier haproxy-enterprise -Name haproxy-enterprise
```

On the HAProxy Enterprise server:

1. Install the ADFSPIP module according to your platform:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-adfspip
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-adfspip
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-adfspip
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-adfspip
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-adfspip
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-adfspip
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-adfspip
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-adfspip
```

2. In the `global` section of your configuration, add directives that configure the module:

hapee-lb.cfg

```
global
  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-adfspip.so
  adfspip-load crt /etc/hapee-3.1/client.pem  ca-file @system-ca  jwt-verify-pubkey /etc/hapee-3.1/adfs-signing-pubkey.pem
  adfspip-register host adfs.example.com username user@example.com password P@ssword12345
```

where:

- `module-path` sets the file path to the enterprise modules folder.
- `module-load` loads the ADFSPIP module.
- `adfspip-load` sets the TLS certificates needed for establishing a trust relationship between the load balancer and the AD FS server. See Reference for details.
- `adfspip-register` sets the host name of the AD FS server to connect to and the credentials of an Active Directory domain administrator that has access to that server. See Reference for details.

3. Unless your HAProxy Enterprise server can already resolve the hostname (for example, `adfs.example.com`), you'll need to add a `resolvers` section to your configuration to specify your DNS nameserver(s). For more information, see the topic DNS resolution.

hapee-lb.cfg

```
resolvers windows_server_dns
  nameserver ns1 10.0.0.4:53
```

Then set the global directive `httpclient.resolver.id` to use it. This line should go before the `adfspip-load` and `adfspip-register` lines.

hapee-lb.cfg

```
global
  httpclient.resolvers.id windows_server_dns

  module-path /opt/hapee-3.1/modules/
  module-load hapee-lb-adfspip.so
  adfspip-load crt /etc/hapee-3.1/client.pem  ca-file @system-ca  jwt-verify-pubkey /etc/hapee-3.1/adfs-signing-pubkey.pem
  adfspip-register host adfs.example.com username user@example.com password P@ssword12345
```

4. Save the file then reload the load balancer to apply the configuration changes:

```
sudo systemctl reload hapee-3.1-lb
```

5. Your load balancer access logs should show that the HAProxy Enterprise server is communicating with the AD FS server successfully. For example, check the log file `/var/log/hapee-3.1r1/lb-access-<DATE>.log`. The `200` HTTP status indicates success.

lb-access-DATE.log

```
<MOD-ADFSPIP> -/- 4/0/22/152/176 200 340 - - ---- 0/0/0/0/0 0/0 {10.0.0.4}
  "POST https://adfs.example.com/adfs/proxy/EstablishTrust HTTP/1.1"
<MOD-ADFSPIP> -/- 4/0/0/40/42 200 222 - - ---- 0/0/0/0/0 0/0 {10.0.0.4}
  "GET https://adfs.example.com/adfs/proxy/WebApplicationProxy/trust?api-version=1 HTTP/1.1"
<MOD-ADFSPIP> -/- 3/0/0/58/59 200 6997 - - ---- 0/0/0/0/0 0/0 {10.0.0.4}
  "GET https://adfs.example.com/adfs/proxy/GetConfiguration?api-version=2 HTTP/1.1"
<MOD-ADFSPIP> -/- 3/0/0/43/44 200 656 - - ---- 0/0/0/0/0 0/0 {10.0.0.4}
  "GET https://adfs.example.com/adfs/proxy/RelyingPartyTrusts?api-version=1 HTTP/1.1"
```

4. Configure proxied applications

For each web application that you want to proxy traffic through HAProxy Enterprise you must publish it by adding a relying party trust.

On the AD FS server:

1. Create a Relying Party Trust:
   - Open **Server Manager** and go to **Tools > AD FS Management**.
   - Right click **Relying Party Trust** and select **Add Relying Party Trust**.
   - In the **Add Relying Party Trust Wizard** dialog, select `Non claims aware` for the type of application. Click **Start**.
   - Set a display name, such as `webapp`, then click **Next**.
   - Set the relying party trust identifier, such as `https://webapp.example.com/`. It must use the `https://` scheme and end with a trailing slash. click **Add**, then **Next**. The relying party trust identifier uniquely identifies the application for which you want to enable authentication. HAProxy Enterprise will relay traffic to this address inside the corporate network.
   - Set the access control policy. Select the Active Directory users and groups that should have access to this application. For example, you can select the **Permit specific group** option to set an active directory group that contains the users that should have access, such as `EXAMPLE\\OfficeWorkers`. Or choose **Permit everyone**. When finished, click **Next**.
   - Click **Next** on the **Ready to Add Trust** screen to complete the setup. Then **Close**.

On the HAProxy Enterprise server:

1. Add an `adfspip-register-server` directive to the global section:

   hapee-lb.cfg

   ```
   global
     adfspip-register-server url https://webapp.example.com/ external_url https://www.example.com/ backend
   webapp_backend
   ```

   where:

   - `url` matches the relying party trust identifier. It must use the `https://` scheme and end in a trailing slash. HAProxy Enterprise will relay traffic to this address inside the corporate network.
   - `external_url` sets the URL at which external clients will access the application. It must use the `https://` scheme. The relying party trust in AD FS will afterwards show this on its Proxy Endpoints tab.
   - `backend` sets the backend pool of servers to use.

2. Add a `frontend` section that receives traffic from clients. This frontend will accept requests for the external URL (for example, `https://www.example.com`) and for the AD FS URL (for example, `https://adfs.example.com`).

hapee-lb.cfg

```
frontend webapp
  bind :80
  bind :443 ssl crt /etc/hapee-3.1/wildcard.example.com.pem
  http-request redirect scheme https unless { ssl_fc }
  http-request add-header X-MS-Proxy haproxy-enterprise
  http-request adfspip
  use_backend %[var(txn.adfspip.app_backend)] if { var(txn.adfspip.app_backend) -m found }
  default_backend adfs_server
```

where:

- `bind :443` receives HTTPS traffic. Note that in the example we're using a wildcard certificate, but you could also set `crt` to a directory of certificates to have the load balancer choose the correct one based on SNI.
- `http-request add-header` adds the required `X-MS-Proxy` HTTP header, which indicates the name of the web proxy.
- `http-request adfspip` checks the incoming request to route it to the sign-in page, validate clients with authentication tokens, and sets the `txn.adfspip.app_backend` variable to the value from the `adfspip-register-server` directive's `backend` argument.
- `use_backend` relays requests to the correct web application backend based on the `txn.adfspip.app_backend` variable.
- `default_backend` proxies sign-in requests to the AD FS federation server.

3. Add a backend for the web application and another for the AD FS server:

hapee-lb.cfg

```
backend webapp_backend
  balance roundrobin
  server web1 10.0.0.10:443 ssl ca-file @system-ca check

backend adfs_server
  server adfs adfs.example.com resolvers windows_server_dns ssl ca-file @system-ca sni str(adfs.example.com) crt /
etc/hapee-3.1/client.pem
```

where:

- The backend named `webapp_backend` defines web application servers. That name of the backend, in this case `webapp_backend`, should match the value from the `adfspip-register-server` directive's `backend` argument. The servers must listen using HTTPS.
- The backend named `adfs_server` defines the federation server(s). Its `server` directive has a `ca-file` argument that validates the server's TLS certificate. With a value of `@system-ca`, it uses certificate authorities defined on the system, but you can also set it to a PEM file. The `sni` argument sets the SNI value to send to the federation server, without which the server won't accept the request. The `crt` argument sets the client certificate for authenticating the HAProxy Enterprise server to the federation server.

Reference

The ADFSPIP module supports the following global directives.

adfspip-load

Loads certificates for communicating with AD FS.

Syntax:

```
adfspip-load crt <cert> [ca-file <ca-file>] [ca-verify-file <ca-file>] jwt-verify-cert <jwt_cert>
```

Argument	Description
`crt`	PEM file containing the SSL client certificate that will be used for connections with the AD FS server.
`ca-file`	(optional) Designates a PEM file from which to load CA certificates used to verify server's certificate. The `@system-ca` parameter could be used in place of the CA file in order to use the trusted CAs of your system, as done with the `server` directive.
`ca-verify-file`	(optional) Designates a PEM file from which to load CA certificates used to verify client's certificate. The `@system-ca` parameter could be used in place of the CA file in order to use the trusted CAs of your system, as done with the `server` directive.
`jwt-verify-pubkey`	Public certificate used to verify the signature of JSON Web Tokens generated by the AD FS server.

adfspip-register

Registers HAProxy Enterprise as a proxy with AD FS.

Syntax:

```
adfspip-register host <hostname> [username <admin-username>] [password <admin-password>]
```

Argument	Description
`host`	AD FS server hostname.
`username`	Username to be used to authenticate to the AD FS server. It can be set via the `HAPROXY_ADFSPIP_USERNAME` environment variable as well. If both are provided, the option from the configuration file will be taken first.
`password`	Password to be used to authenticate to the AD FS server. It can be set via the `HAPROXY_ADFSPIP_PASSWORD` environment variable as well. If both are provided, the option from the configuration file will be taken first.

adfspip-register-server

Registers a web application with AD FS that will be proxied through HAProxy Enterprise.

Syntax:

```
adfspip-register-server url <url> backend <backend> [external_url <external_url>]
```

Argument	Description
`url`	Relying party trust internal URL. This is the address inside the corporate network that HAProxy Enterprise will relay requests to.
`external_url`	Corresponding external URL to the internal `<url>`. Clients outside the corporate network will use this URL to access the web application.
`backend`	Backend towards which end-user request must be forwarded.

Troubleshooting

This section describes common troubleshooting steps.

General troubleshooting

Check the access log for errors, found in `/var/log/hapee-3.1/lb-access-<date>.log`.

AD FS server returns 503 Service Unavailable

If you receive a `503` Service Unavailable response from the AD FS server, there are several possible causes:

- In the `adfs_server` backend, check that the `server` line is setting an SNI value that matches the federation server's service communications certificate. To view this certificate on the AD FS server:
  - Open **Server Manager** and go to **Tools > AD FS Management**.
  - Expand **Service** and select **Certificates**.
  - Right click the `Service communications` certificate and select **View certificate**.
- On the AD FS server, check that the federation server's name that you set when configuring AD FS matches the server's hostname. If not, remove and reinstall the AD FS role.
- Check that you set a relying party trust identifier that uses `https://` and ends with a trailing slash.
- Check that HAProxy Enterprise reports no errors when you run `sudo systemctl status hapee-3.1-lb`.

identifier not in the right format

If you receive the error message `process_relyingpartytrust_resp: identifier not in the right format (not a uri)`, this could mean that the module is trying to parse relying party trusts that aren't non-claims aware type applications. For example, you might have SAML applications registered in AD FS.

Single sign-on (Kerberos)

This page applies to:

- HAProxy Enterprise - all versions

Kerberos is an authentication protocol. It is the default authentication protocol in Windows. The Single Sign-On (SSO) module integrates with Kerberos to let users sign in to an HTTP application with Windows Active Directory credentials. Kerberos doesn't typically go over HTTP, but an extension to the HTTP protocol, RFC 4559, adds a way to negotiate the protocol via the `Www-Authenticate` and `Authorization` HTTP headers. Then Kerberos tickets (identity tokens) are exchanged over the network by wrapping them inside Simple and Protected Negotiate (SPNEGO) tokens, defined by RFC 4178.

HAProxy Enterprise, being in front of the HTTP applications for which you want to enable single sign-on, validates the user's Kerberos ticket and grants them access. Therefore, the applications themselves do not need to implement Kerberos directly.

To implement single sign-on using Kerberos, we will complete the following steps:

1. Configure Windows Server.
2. Configure HAProxy Enterprise.
3. Configure the user's PC.

Configure Windows Server

In this section, you will configure Windows Server for Kerberos authentication.

Prerequisites

On Windows Server, please check that the following prerequisites have been met:

- You've installed the Active Directory Domain Services role.
- You've promoted the Windows Server instance to be a domain controller. A domain controller is a Windows Server where you administer Active Directory.
- You've created an Active Directory domain. In this guide, we will use the domain `example.com`.
- You've installed the DNS Server role.

Create a user account and SPN

With Kerberos, every service gets a unique Service Principal Name (SPN). An SPN identifies a service running on a host. Because HAProxy Enterprise will act as a proxy in front of the applications, we will assign the SPNs to it. In Windows Server, we will create a user account in Active Directory to represent the load balancer and then assign to it an SPN for each load balanced web app. It might seem odd to assign a user account to a load balancer, but this is simply where we will map which services the load balancer can accept authorization requests for. By doing this, we are able to generate a keytab file, which we will store on the load balancer. When a user accesses the service via the load balancer, they present a ticket that the load balancer must decrypt with its keytab to prove that it is authorized by Active Directory.

1. Log in to your AD domain controller.

2. Open PowerShell and create a new user with the `New-ADUser` command. Note that the backtick is the word-wrap operator, allowing you to split the command onto multiple lines. Change the password used in the example:

```
New-ADUser `
  -Name loadbalancer1 `
  -Description "Load balancer 1 service account" `
  -AccountPassword (ConvertTo-SecureString "P@ssword1" -AsPlainText -Force) `
  -PasswordNeverExpires $true `
  -KerberosEncryptionType AES256 `
  -Enabled $true
```

3. Create a keytab file for the account by calling the `ktpass` command from PowerShell. The command does two things:

   - Maps the Service Principal Name `HTTP/webapp.example.com@EXAMPLE.COM` to the AD user account `example\loadbalancer1`. The load balancer is then authorized to proxy requests for the service `webapp.example.com`.
   - Generates a keytab file at `C:\loadbalancer1.keytab`, which contains the service's key. After you've copied this file to the load balancer, the load balancer will be able to decrypt tickets.

```
ktpass /out C:\loadbalancer1.keytab /princ HTTP/webapp.example.com@EXAMPLE.COM /mapUser
example\loadbalancer1 /crypto AES256-SHA1 /pType KRB5_NT_PRINCIPAL /pass +rndPass
```

output

```
Targeting domain controller: windowsserver.example.com
Using legacy password setting method
Successfully mapped HTTP/webapp.example.com to loadbalancer1.
Key created.
Output keytab to C:\loadbalancer1.keytab
Keytab version: 0x502
keysize 90 HTTP/webapp.example.com@EXAMPLE.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x12 (AES256-SHA1) keylength
32
```

In this example:

Field	Description
`/out`	The keytab to produce.
`/princ`	The principal name (HTTP/FQDN@REALM).
`/mapUser`	Maps the principal to the given user account.
`/crypto`	The cryptosystem to use. `AES256-SHA1` means `AES256-CTS-HMAC-SHA1-96`.
`/pType`	The principal type. `KRB5_NT_PRINCIPAL` is the general, recommended ptype.
`/pass`	Sets the password.
`+rndPass`	Generates a random password.

View the Service Principal Name

To view a user account's Service Principal Name:

1. Open **Server Manager** and go to **Tools > Active Directory Users and Computers**.
2. In the **Active Directory Users and Computers** window, open the **View** menu and enable **Advanced Features**.
3. Expand your domain, select **Users**, and then double-click the user to view its properties. In our example, we created a user named `loadbalancer1`.
4. On the user's properties, select the **Attribute Editor** tab.
5. Scroll to the **servicePrincipalName** to see its value. In our example, the value is `HTTP/webapp.example.com`.

Add a DNS record

Add a DNS record for your web app, pointed at your load balancer.

1. On your AD domain controller, go to **Server Manager > Tools > DNS**.
2. Expand **Forward Lookup Zones** in the tree, right-click on your domain (for example, `example.com`) and select **New Host (A or AAAA)**.
3. On the **New Host** dialog, set the name. For example, `webapp`.
4. Set the IP address. For example, use your load balancer's IP address.
5. Click **Add Host**.

You should see a dialog that says “The host record was successfully created”.

Optional: Disable NTLM

By disabling NTLM altogether you can make your network safer and eliminate the chance that the user’s browser and Windows Server will try to use it.

NTLM is an older and less secure protocol than Kerberos that runs in some Windows Active Directory domains. To authenticate with Kerberos over HTTP, the user’s web browser will exchange a series of messages with Windows Server to negotiate which authentication protocol to use. This is how the `Www-Authenticate: Negotiate` HTTP header works. Though after this negotiation, the client and server may try to use NTLM instead of Kerberos.

If you see the following error in the SSO debug logs, then try disabling NTLM:

“GSS-API error gss_accept_sec_context: An unsupported mechanism was requested”

To disable NTLM:

1. On your AD domain controller, go to **Server Manager > Tools > Group Policy Management**.
2. Expand your forest and the domains folder in the Group Policy Management tree.
3. If you do not yet have a group policy object, right-click on your domain and choose **Create a GPO in this domain, and Link it here**. Assign the new GPO a name.
4. Expand your domain, right-click your new policy object, and choose **Edit**. The **Group Policy Management Editor** appears.
5. Under **Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options**, double click the setting named **Network security: Restrict NTLM: NTLM authentication in this domain**. Set its value to `Disable`.
6. Optional: To speed up propagation of the new setting, on the target PC joined to the domain run PowerShell as an administrator and execute the command `gpupdate /force`.

Configure HAProxy Enterprise

In this section, you will configure HAProxy Enterprise for Kerberos authentication.

Install and configure the module

Perform these steps on the load balancer node.

1. Install the SSO module according to your platform:

Apt:

```
sudo apt-get install hapee-extras-spoa-sso
```

Yum:

```
sudo yum install hapee-extras-spoa-sso
```

Zypper:

```
sudo zypper install hapee-extras-spoa-sso
```

Pkg:

```
sudo pkg install hapee-extras-spoa-sso
```

It installs the following files:

File	Description
`/opt/hapee-extras/bin/hapee-krb-srv`	Defines the SSO Kerberos service program.
`/etc/hapee-extras/hapee-krb-srv-spoe.cfg`	Defines how the load balancer passes data via the Stream Processing Offload Protocol to the SSO Kerberos service. Typically, you will not need to edit this file.
`/etc/hapee-extras/hapee-krb-srv.cfg`	Gives an example portion of an HAProxy Enterprise load balancer configuration that you can copy-paste.
`/etc/hapee-extras/hapee-sso-spoe.cfg`	Not used in this setup.
`/etc/hapee-extras/hapee-sso.cfg`	Not used in this setup.
`/etc/hapee-extras/sample/krb-srv.ini`	Sets the location of your keytab file for each application.
`/etc/hapee-extras/sample/krb-srv.map`	Maps HTTP `Host` headers to applications.
`/etc/hapee-extras/sample/sso.ini`	Not used in this setup.
`/etc/hapee-extras/sample/sso.map`	Not used in this setup.
`/etc/default/hapee-extras-sso`	Defines the default settings for the `hapee-extras-spoa-krb-srv` service. The file sets the `KRBSRV_OPTIONS` environment variable.

2. Copy the keytab file from Windows Server to the load balancer. For example, save it as `/etc/hapee-extras/keytabs/loadbalancer1.keytab`.

```
sudo mkdir -p /etc/hapee-extras/keytabs
sudo cp ~/loadbalancer1.keytab /etc/hapee-extras/keytabs/
```

3. Set the owner of the keytab file:

```
sudo chown hapee-sso:hapee /etc/hapee-extras/keytabs/loadbalancer1.keytab
```

4. Set the mode of the keytab file:

```
sudo chmod 600 /etc/hapee-extras/keytabs/loadbalancer1.keytab
```

5. Copy the files `krb-srv.ini` and `krb-srv.map` from the `sample` directory to `/etc/hapee-extras`:

```
sudo cp /etc/hapee-extras/sample/krb-srv.* /etc/hapee-extras/
```

6. Edit `/etc/hapee-extras/krb-srv.ini`. Add a `domain` section for your application and set the path to the keytab file. Also add an application section that references the domain section.

krb-srv.ini

```
[domain:webapp.example.com]
krb_keytab_file = /etc/hapee-extras/keytabs/loadbalancer1.keytab

[webapp]
domain = webapp.example.com
```

7. Edit `/etc/hapee-extras/krb-srv.map`. Add a line that maps the HTTP `Host` header that you expect with the domain and app.

krb-srv.map

```
# Host header        # domain / app
webapp.example.com   webapp.example.com/webapp
```

8. Edit your main load balancer configuration file, `/etc/hapee-3.1/hapee-lb.cfg`.

- Add the `HA_SSO_SCOPE` variable to the `global` section:

```
global
  setenv HA_SSO_SCOPE "sess"
```

- Add the SSO Kerberos lines to the `frontend` for which you want to enable single sign-on (be sure to update the `crt` argument specifying the certificate file path):

hapee-lb.cfg

```
frontend www
  bind :80
  bind *:443 crt /path/to/certs/cert.pem SSL
  http-request redirect scheme https unless { ssl_fc }
  http-response set-header Strict-Transport-Security max-age=63072000

  #--------------------------------
  # SSO Kerberos section
  #--------------------------------
  option http-buffer-request
  filter spoe engine spoe_krb_srv config /etc/hapee-extras/hapee-krb-srv-spoe.cfg

  # HTTP REQUEST
  # set sso_app and sso_domain variables before calling the SPOA
  http-request set-var(txn.sso_domain) req.hdr(Host),map(/etc/hapee-extras/krb-srv.map),word(1,'/')
  http-request set-var(txn.sso_app)    req.hdr(Host),map(/etc/hapee-extras/krb-srv.map),word(2,'/')

  # set sess.sso_cookie variable with the cookie set by the browser
  http-request set-var(txn.sso_cookie) req.cook(sso_cookie) if { req.cook(sso_cookie) -m found }

  # The authorization header might be sent on the first request of a session,
  # depending on the value set in "HA_SSO_SCOPE" variable we could accept a request if
  # a previous request was allowed in the current session.
  http-request return status 401 hdr www-authenticate Negotiate unless { var("$HA_SSO_SCOPE.sso.action") -m
found } || { req.hdr(authorization) -m found }

  http-request send-spoe-group spoe_krb_srv grp-sso-check-token
  http-request deny if { var("$HA_SSO_SCOPE.sso.action") -m str dny }
  http-request deny if ! { var("$HA_SSO_SCOPE.sso.action") -m str alw }

  # The agent can extract a display name from a validated Kerberos token
  http-request del-header X-SSO-LOGIN
  http-request set-header X-SSO-LOGIN %[var("$HA_SSO_SCOPE.sso.sso_login")] if
{ var("$HA_SSO_SCOPE.sso.sso_login") -m found }

  # If the SPOA has set "$HA_SSO_SCOPE.sso.www_authenticate" or "$HA_SSO_SCOPE.sso.sso_set_cookie" variables, we
set the header accordingly
  http-response add-header Set-Cookie sso_cookie=%[var("$HA_SSO_SCOPE.sso.sso_set_cookie")];\ path=/;\ domain=%
[var("$HA_SSO_SCOPE.sso.sso_set_cookie_domain")]; if { var("$HA_SSO_SCOPE.sso.sso_set_cookie") -m found }
  http-response add-header www-authenticate %[var("$HA_SSO_SCOPE.sso.www_authenticate")] if
{ var("$HA_SSO_SCOPE.sso.www_authenticate") -m found }

  # BACKENDS
  default_backend sso_err
  use_backend sso_err if { var("$HA_SSO_SCOPE.sso.action") -m str err } # we got an error from the SPOA
```

```
  use_backend %[var("$HA_SSO_SCOPE.sso.backend")] if { var("$HA_SSO_SCOPE.sso.action") -m str alw }
{ var("$HA_SSO_SCOPE.sso.backend") -m found }
```

- Ensure that the name of the `backend` for your application matches the name of the application section in `krb-srv.ini`. In our example, `krb-srv.ini` contained `[webapp]`, so we would name the backend `webapp`:

hapee-lb.cfg

```
backend webapp
  balance roundrobin
  server web1 127.0.0.1:3000 check
```

- Add the following two `backend` sections:

hapee-lb.cfg

```
backend sso_err
  mode http

backend spoa-backend
  mode tcp
  timeout server 1m
  server sso-daemon 127.0.0.1:12345 inter 1s check
```

9. Reload the load balancer:

```
sudo systemctl reload hapee-3.1-lb
```

10. Enable and start the service:

```
sudo systemctl enable hapee-extras-spoa-krb-srv
sudo systemctl start hapee-extras-spoa-krb-srv
```

11. The service's status should indicate that it is now configured for the new application.

```
sudo systemctl status hapee-extras-spoa-krb-srv
```

output

```
hapee systemd[1]: Starting hapee-extras-spoa-krb-srv.service - HAPEE SPOA-KRB-SRV: HAPEE SSO solution...
hapee hapee-krb-srv[4955]: [00] Configured domains:
hapee hapee-krb-srv[4955]: 1713465556.954921 [00] Configured domains:
hapee hapee-krb-srv[4955]: 1713465556.954951 [00] domain:webapp.example.com (1 app)
hapee hapee-krb-srv[4955]: [00] domain:webapp.example.com (1 app)
hapee hapee-krb-srv[4956]: 1713465556.956454 [00] SSO Server will start in background, pid=4956
hapee hapee-krb-srv[4956]: [00] SSO Server will start in background, pid=4956
hapee systemd[1]: Started hapee-extras-spoa-krb-srv.service - HAPEE SPOA-KRB-SRV: HAPEE SSO solution.
hapee HAProxy-sso[4956]: [00] SSO server is ready and listening on port 12345
```

Optional: Add a second application

To add another application to SSO:

1. On your Windows Server, add a DNS record for the new application.
   - On your AD domain controller, go to **Server Manager > Tools > DNS**.
   - Expand **Forward Lookup Zones** in the tree, right-click on your domain (for example, `example.com`) and select **New Host (A or AAAA)**.
   - On the **New Host** dialog, set the name. For example, `webapp2`.
   - Set the IP address. For example, use your load balancer's IP address.
   - Click **Add Host**.
2. To add more SPNs to the user account to support other web applications, use the `/in` argument when calling `ktpass` to append new SPNs:

```
ktpass /in C:\loadbalancer1.keytab /out C:\loadbalancer1.keytab /princ HTTP/webapp2.example.com@EXAMPLE.COM /mapUser
example\loadbalancer1 /crypto AES256-SHA1 /pType KRB5_NT_PRINCIPAL /pass +rndPass
```

output

```
Existing keytab:

keytab version: 0x502
keysize 90 HTTP/webapp.example.com@EXAMPLE.COM pType 1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x12 (AES256-SHA1) keylength
32
Targeting domain controller: windowsserver.example.com
Using legacy password setting method
Successfully mapped HTTP/webapp2.example.com to loadbalancer1.
Key created.
Output keytab to C:\loadbalancer1.keytab:
Keytab version: 0x502
keysize 90 HTTP/webapp.example.com@EXAMPLE.COM ptype1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x12 (AES256-SHA1) keylength 32
keysize 91 HTTP/webapp2.example.com@EXAMPLE.COM ptype1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x12 (AES256-SHA1) keylength
32
```

3. Copy the keytab file to the load balancer.

4. Edit `/etc/hapee-extras/krb-srv.ini`. Add a `domain` section for your application and set the path to the keytab file. Also add an application section that references the domain section.

krb-srv.ini

```
[domain:webapp.example.com]
krb_keytab_file = /etc/hapee-extras/keytabs/loadbalancer1.keytab

[webapp]
domain = webapp.example.com

[domain:webapp2.example.com]
krb_keytab_file = /etc/hapee-extras/keytabs/loadbalancer1.keytab

[webapp2]
domain = webapp2.example.com
```

5. Edit `/etc/hapee-extras/krb-srv.map`. Add a line that maps the HTTP `Host` header that you expect with the domain and app.

krb-srv.map

```
# Host header        # domain / app
webapp.example.com   webapp.example.com/webapp
webapp2.example.com  webapp2.example.com/webapp2
```

6. Restart the SSO Kerberos service:

```
sudo systemctl restart hapee-extras-spoa-krb-srv
```

The service's status should indicate that it is now configured for the new application.

```
sudo systemctl status hapee-extras-spoa-krb-srv
```

output

```
hapee hapee-krb-srv[6283]: 1713397718.771858 [00] Configured domains:
hapee hapee-krb-srv[6283]: 1713397718.771899 [00] domain:webapp.example.com (1 app)
hapee hapee-krb-srv[6283]: 1713397718.771905 [00] domain:webapp2.example.com (1 app)
```

7. Edit your main load balancer configuration file, `/etc/hapee-3.1/hapee-lb.cfg`. Ensure that the name of the `backend` for your application matches the name of the application section in `krb-srv.ini`. For example, `webapp2`:

hapee-lb.cfg

```
backend webapp2
  balance roundrobin
  server web1 127.0.0.1:3001 check
```

8. Reload the load balancer:

```
sudo systemctl reload hapee-3.1-lb
```

Configure user PCs

In this section, you will configure users' Windows PCs for Kerberos authentication.

Update DNS and AD settings

To allow single sign-on in the AD domain, update the user's PC to use the domain's DNS server and, optionally, join the computer to the domain.

1. Update the user's preferred DNS server to point to Windows Server. That way, they can access internal subdomains like `webapp.example.com`. On the user's PC:

   - On the Desktop, click the **Start** button. Enter `Control Panel` and press `Enter`.
   - Navigate to **Network and Internet** and bring up the properties for your network adapter:

   Windows 10:

   - Click **Network and Sharing Center**.
   - Click **Change adapter settings**.
   - Right-click on your adapter (for example, **Ethernet**).
   - Select **Properties**.

   Windows 11:

   - Click **Advanced network settings**.
   - Click on your adapter.
   - Next to **More adapter options**, click **Edit**.

   - On the adapter's **Properties** dialog, double-click **Internet Protocol Version 4 (TCP/IPv4)** to open the protocol's properties.
   - Click **Advanced**, then select the **DNS** tab.
   - Under **DNS server addresses**, click **Add**, then enter the IP address of your Windows Server. Click **Add**.
   - Click **OK** to save the change.
   - To verify, open PowerShell and use `nslookup` to retrieve the IP address for your AD domain controller server. For example:

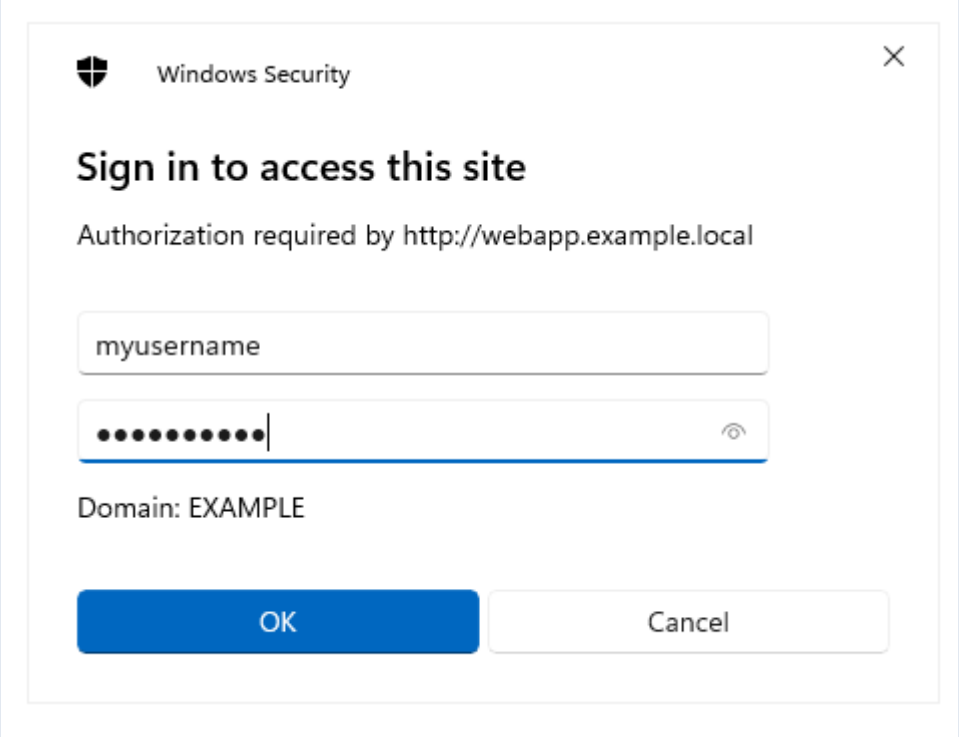
   ```
   nslookup windowsserver.example.com
   ```

   output

   ```
   Name: windowsserver.example.com
   Addresses: 192.168.56.60
   ```

2. Join the computer to the Active Directory domain.

When the user accesses the web application, for example `webapp.example.com`, through their browser, they will see a login prompt. They can enter their Active Directory credentials.



Optional: Integrated authentication

To skip the login prompt and instead enable Microsoft Edge to use the credentials of the logged-in Windows user, enable integrated authentication. This allows the browser to use the credentials from the Windows operating system and skip being prompted. You will set this as a Group Policy so that it applies to all computers joined to the Active Directory domain.

1. On the Active Directory domain controller, go to the Microsoft Edge website and click the link **Download Windows 64-bit Policy** to download the archive. Extract the contents of the archive.
2. In Windows Explorer, navigate to the policies directory for your domain. For example, for the domain `example.com`, it would be `C:\Windows\SYSVOL\sysvol\example.com\Policies`. From there, create a directory named `PolicyDefinitions`.
3. From the extracted archive, copy the contents of the `MicrosoftEdgePolicyTemplates\windows\admx` directory into the new `PolicyDefinitions` directory.
4. Open **Server Manager** and go to **Tools > Group Policy Management**. Expand your forest and domain in the Group Policy Management tree.
5. If you do not yet have a group policy object, right-click on your domain and choose **Create a GPO in this domain, and Link it here**. Assign the new GPO a name.
6. Expand your domain, right-click your new policy object, and choose **Edit**. The **Group Policy Management Editor** appears.

7. Under **Computer Configuration > Policies > Administrative Templates**, expand the new **Microsoft Edge** policy definition in the Group Policy Management Editor.
8. Within **HTTP Authentication**, edit the setting named **Configure list of allowed authentication servers**, enable it, and set its value to your load balanced application. For example, `webapp.example.com` or `*.example.com`.
9. Optional: If the setting **Supported authentication schemes** has been configured, then be sure its value includes `negotiate`.

To verify that the policy has been rolled out:

1. On the user's PC, run PowerShell and execute `gpupdate /force`.
2. Open Microsoft Edge and go to `edge://policy` to see the applied policies. You should see the policy named `AuthServerAllowlist` set to your target hostname.

Troubleshooting

This section describes how to troubleshoot the SSO Kerberos service.

Validate the configuration

To validate your configuration file:

- Call `hapee-krb-srv` with the `--check-cfg` flag:

```
/opt/hapee-extras/bin/hapee-krb-srv -f /etc/hapee-extras/krb-srv.ini --check-cfg
```

output

```
1713906912.716699 [00] Configured domains:
1713906912.717762 [00] domain:webapp.example.com (1 app)
1713906912.717880 [00] Configuration parsed successfully.
1713906912.717897 [00] Exiting
```

Debug logs

The SSO module has startup flags that enable debug-level logs, which can help when diagnosing problems.

1. Edit the file `/etc/default/hapee-extras-sso` and change the `KRBSRV_OPTIONS` line so that it includes one of the following debug flags:

Flag	Description
`--debug-spoe`	Shows information and events about the Stream Processing Offload Engine (SPOE), which is how the load balancer communicates with the SSO Kerberos service.
`--debug-spoe-variables`	Shows the SPOE variables that the load balancer passes to the SSO Kerberos service.
`--debug-sso`	Shows messages about the single sign-on flow, such as whether a client authenticated successfully.
`--debug-krb`	Show messages related to the load balancer processing the Kerberos protocol. Additional details are saved to a log file in the `/tmp` directory.
`--debug-krb-err`	Shows errors that occur while processing the Kerberos protocol.
`--debug-all`	Enables all debugging flags.

2. Restart the SSO Kerberos service:

```
sudo systemctl restart hapee-extras-spoa-krb-srv
```

In the following sections, we show examples.

debug-spoe

Show information and events about the Stream Processing Offload Engine (SPOE), which is how the load balancer communicates with the SSO Kerberos service.

hapee-extras-sso

```
KRBSRV_OPTIONS="--uid hapee-sso --gid hapee --debug-spoe"
```

Use `journalctl` to show logs:

```
sudo journalctl --follow --unit hapee-extras-spoa-krb-srv
```

output

```
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [00] <1> New Client connection accepted and assigned to worker 01
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> read_frame_cb
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> New Frame of 129 bytes received
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> Decode HAProxy HELLO frame
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> Supported versions : 2.0
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> HAProxy maximum frame size : 16380
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> HAProxy capabilities : pipelining,async
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> HAProxy supports frame pipelining
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> HAProxy supports asynchronous frame
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> HAProxy engine id : 6fb22acf-84a8-47b2-bece-bfb60a8aa88a
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> Encode Agent HELLO frame
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> Agent version : 2.0
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> Agent maximum frame size : 16380
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> Agent capabilities :
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> write_frame_cb
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> Frame of 54 bytes sent
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> read_frame_cb
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> New Frame of 2486 bytes received
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> Decode HAProxy NOTIFY frame
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> STREAM-ID=10 - FRAME-ID=1 - unfragmented frame received - frag_len=0
- len=2486 - offset=7
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] Process frame messages : STREAM-ID=10 - FRAME-ID=1 - length=2479 bytes
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] Encode Agent ACK frame
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] STREAM-ID=10 - FRAME-ID=1
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> write_frame_cb
Apr 23 19:09:14 hapee HAProxy-sso[1118]: [01] <1> Frame of 601 bytes sent
```

debug-spoe-variables

Show the SPOE variables that the load balancer passes to the SSO Kerberos service.

hapee-extras-sso

```
KRBSRV_OPTIONS="--uid hapee-sso --gid hapee --debug-spoe-variables"
```

Use `journalctl` to show logs:

```
sudo journalctl --follow --unit hapee-extras-spoa-krb-srv
```

output

```
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] SPOA argument: sso_app, size=6
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] SPOA argument: sso_domain, size=20
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] SPOA argument: sso_cookie, size=32
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] SPOA argument: method, size=3
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] SPOA argument: authorization, size=2338
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] ==> SPOE sso-check-token message: sso_domain=webapp.example.com
sso_app=webapp sso_cookie=8a1200062d3b7222432f00b47ca051dd authorization=Negotiate
YIIGzgYGKwYBBQUCoIIGwjCCBr6gMDAuBgkqhkiC9xIBAgIGCSqGSIb3EgECAgYKKwYBBAGCNwICHgYKKwYBBAGCNwICCqKCBogEggaEYIIGgAYJKoZIhvcS
AQICAQBuggZvMIIGa6ADAgEFoQMCAQ6iBwMFACAAAACjggUAYYIE/
DCCBPigAwIBBaEPGw1FWEFNUExFLkxPQ0FMoicwJaADAgECoR4wHBsESFRUUBsUd2ViYXBwLmV4YW1wbGUubG9jYWyjggS1MIIEsaADAgESoQMCAQSiggSjB
IIEny5K5yHXp3e875oOqZOeTMpZFjOygrCXZcsPVgYZ9kuaaZynUMJIVIQVzwOh0w25bmJlJXLUA3zkuFXvOD9cFTjZbvye2PVSbwyQvRcPPQul0gHd+eJCn
Giu6/DxR5UCfvtWmwCQTQ7YucFJHc5wPC1WW/2q6e3DTdTnVuFKdUL6HWDz7YL9OZKQD7Yfo7dsSYRiLc5Fd2/
Z+I4exlk5taNYYU2LpPcXS4bZEDpnGRQqPpob7ZDh03IC7ZbJpIaUeZhzv82TEV37K7Su4hXdruFthH/
jAcp0GLGC8jbALmClJyeX8KVd5cLnqYILtuZl+VBEr7ym+g64rYIk9ANbG6e9lI70zL0ZMQZBb9N78F/
5fppzvK9S3uGblu0RD+6dNKWz8uVO3Df2ExOHxewxOVktynrSbzNBg6/
m6ZN49muNXnoBTQbS4rvUj70oscLDK0MeB8ZzcE5CwjzAc1DIVKX6liDRcRXYcj1+01UFTIPhKwweh1Arcj/s925T+xFWjM+b0466p0oH1UDS1IyHxG/
sEJqPCE3cEn7RWI4TmlB0Y21oRU3d2ShrO1PRuIKo39Okm28qLq/ieSEdU3a98+FL/xUtxTy1pqwJiJHwIm7duphgxFh+C4RRj4RpsWJcfZ/
UPbEoeqlo6xisYXgIxL8IY+0O5gWOkSVHyn6HYdP3iJ9Q30eODenfbVQZpzo7qenYXnWlaOgMlV7PtZXU905Gt8NokVR022qU2IyIe+eF0afkyt1rkfJCDeK
te4n47nC66lvVXsOsvWAtwHFHtvQOyiG3f78O4m2ZMv2SxHKA2hRmwF7Pyk34dAAK0RUnOhQvO675W1qY97w7NgYcHPivPv2kn7pMt/
OzAdmU7TNUNlARDSRcuwUHZWEYG1LhCEHSWUDhDkYqkP286OEikIkfISu743kiz9fiLl2ojZot6J9wLy1KQlV9wnXG9koYkli9bHP832mswyXUU6VYoEPcZW
jqmZwpd9EH9rGxgvkXTR6OY+7c9rFzr61iZTdR0tf42tnXqyzMNU6Bew6qm2xitZlB6saMOlUmllF4hvo56rz3tEk8K+Mu+dgaI3CrEwtnbJOcxaZeWMfnCv
i3R6rs6Na0PbREECpveOx94W+LgtRjcFlF/dYjCZV4361P0XKPbLjhaxE8mu/lSWa0eEci7Yrn4izF9QeTM/
okeFv+7gg4BB7HVRnK4tV62yoiJIcLleGgB2vH9Dgy1Qw4y6B0isB8hUzV+7OJaD+yQoCAp4EaDddYsjJn7EDJgqx1xZv1XfKZdBJEW1j9dZ/
uJLHj2EaJZHSdyzTvZUI441cWBO3W1OEBmOiR7v0lGqpDYcRoUcRzIEimxjHYHhsEWOGkfAD952PvXIGFGMQRmsvMeudhlhqr9BNgnrOedNtaGN8kZAr6N3B
REIFw4iS+Oa32/6uKm5YcRo2jZLgrrHzDIEf+VNllmFlLiFRm45kWYA0fvzxB23cXAvffM4A9J6LpjNwFeAMfApNYy7B4jSTHx5/+HFfEWNkFUjb5IPvUr5j
ll68Bw6y50A+7cW+vp2rlUPaQA7sZjnXLrkP4MHCkggFQMIIBTKADAgESooIBQwSCAT9zUk/
z1QRXULlWoaR3DD+t3PLrN0UJOPW0YVKDRpOkNy5e5ZJ071RPVueC9GgcEnkqlGLillrddUpXnp8TgTwG+0Z847UwiwELmCxydxr2+XVM1hCYvLSvByJOfKJ
PqIcvvld4Xjpa7dt3Knon0x96GKw0go+aJhORM6tz4j/9V6p7Govd0WPE3tQAMyUeyoXu15CdcMAVHoa6Qbf0a5h/HB0nF0fr+yedNciBxJhkID//
Ra48HjA4PJxZSf/WVEHIYpSouJqk4cZyzEzrWbOxjQMe6kHoFvEnV/RAF9O0pgv6WWvVg7TMzVWZD9Kn/
Rl+ki59ZRcCEYGv7e7sueZuOmAvRuvq6fd+NEpv5RSFyxJtDxqEPUULierxTLZtBB7jnv+x22FbaFEyq0jg0y52s4nq6qlqp7ky6zXrvxGW
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var str scope=1
sso_set_cookie=2a9dd981820179a19cf5b8d4bae0862b; HttpOnly; Expires=Wed, 24 Apr 2024 05:14:48 GMT
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var str scope=1 sso_set_cookie_domain=
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var str scope=1
sso_cookie=2a9dd981820179a19cf5b8d4bae0862b
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var str scope=2 backend=webapp
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var str scope=2 sso_app=webapp
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var str scope=2 sso_domain=webapp.example.com
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var str scope=2 www_authenticate=Negotiate
oYG1MIGyoAMKAQChCwYJKoZIgvcSAQICooGdBIGaYIGXBgkqhkiG9xIBAgICAG+BhzCBhKADAgEFoQMCAQ+ieDB2oAMCARKibwRtIZy7J4L1xwE98XQeAjHP
qDg1V2qWDMvIyhh3x8abTBbgV9l8b69nm9XQ+3a0iVxDWB0Wuzdrm14ff2SeAmRI3WRcWV
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var str scope=2 sso_login=myuser@example.com
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var str scope=2 action=alw
Apr 23 19:14:48 hapee HAProxy-sso[1151]: [01] Encode SPOE set-var int scope=2 status=0
```

debug-sso

Show messages about the single sign-on flow, such as whether a client authenticated successfully.

hapee-extras-sso

```
KRBSRV_OPTIONS="--uid hapee-sso --gid hapee --debug-sso"
```

Use `journalctl` to show logs:

```
sudo journalctl --follow --unit hapee-extras-spoa-krb-srv
```

output

```
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] Process SPOE Message 'sso-check-token'
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] ==> Process SPOE Message 'sso-check-token' with 5 args. len left: 2462
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] ==> SPOE sso-check-token message
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] Unexpected SPOA argument: method
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [00] Cleaning cookie contexts that have expired...
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [00] Purged 0/0 cookie contexts
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] In handle_authorization()
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] Generating a new sso cookie...
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] Created new cookie 23dfd3e13b171bc5310bb885db30e0e1
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] Adding new domain context for domain webapp.example.com for cookie
23dfd3e13b171bc5310bb885db30e0e1. Expires=2024-04-24 05:17:57
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] Storing SSO cookie 23dfd3e13b171bc5310bb885db30e0e1, expires on
2024-04-24 05:17:57
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] AUTH_RESULT=SUCCESS
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] Setting action to SSO_ACTION_ALLOW
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] Next action=alw msg=(null)
Apr 23 19:17:57 hapee HAProxy-sso[1177]: [01] ======================= response frame encoded, 594 bytes
```

debug-krb

Show messages related to the load balancer processing the Kerberos protocol. Additional details are saved to a log file in the `/tmp` directory. In the example below, the log file is named `/tmp/krb_debug_webapp.example.com.log`.

hapee-extras-sso

```
KRBSRV_OPTIONS="--uid hapee-sso --gid hapee --debug-krb"
```

Use `journalctl` to show logs:

```
sudo journalctl --follow --unit hapee-extras-spoa-krb-srv
```

output

```
Apr 23 19:19:43 hapee HAProxy-sso[1207]: [00] Logging Kerberos operations to /tmp/krb_debug_webapp.example.com.log
Apr 23 19:19:43 hapee HAProxy-sso[1207]: [00] Token validated for client "myuser@example.com"
```

Then the file `krb_debug_webapp.example.com.log` in the `/tmp` directory contains additional details:

krb_debug_webapp.example.com.log

```
[1207] 1713899983.885398: Decrypted AP-REQ with server principal HTTP/webapp.example.com@EXAMPLE.COM: aes256-cts/5511
[1207] 1713899983.885399: AP-REQ ticket: myuser@EXAMPLE.COM -> HTTP/webapp.example.com@EXAMPLE.COM, session key aes256-cts/C099
[1207] 1713899983.885400: Negotiated enctype based on authenticator: aes256-cts
[1207] 1713899983.885401: Authenticator contains subkey: aes256-cts/A116
[1207] 1713899983.885402: Creating AP-REP, time 1713899983.12, subkey aes256-cts/D7DE, seqnum 255461910
[1257] 1713900224.105711: Decrypted AP-REQ with server principal HTTP/webapp.example.com@EXAMPLE.COM: aes256-cts/5511
[1257] 1713900224.105712: AP-REQ ticket: myuser@EXAMPLE.COM -> HTTP/webapp.example.com@EXAMPLE.COM, session key aes256-cts/D0FA
[1257] 1713900224.105713: Negotiated enctype based on authenticator: aes256-cts
[1257] 1713900224.105714: Authenticator contains subkey: aes256-cts/03DC
[1257] 1713900224.105715: Creating AP-REP, time 1713900223.18, subkey aes256-cts/9481, seqnum 1008188495
```

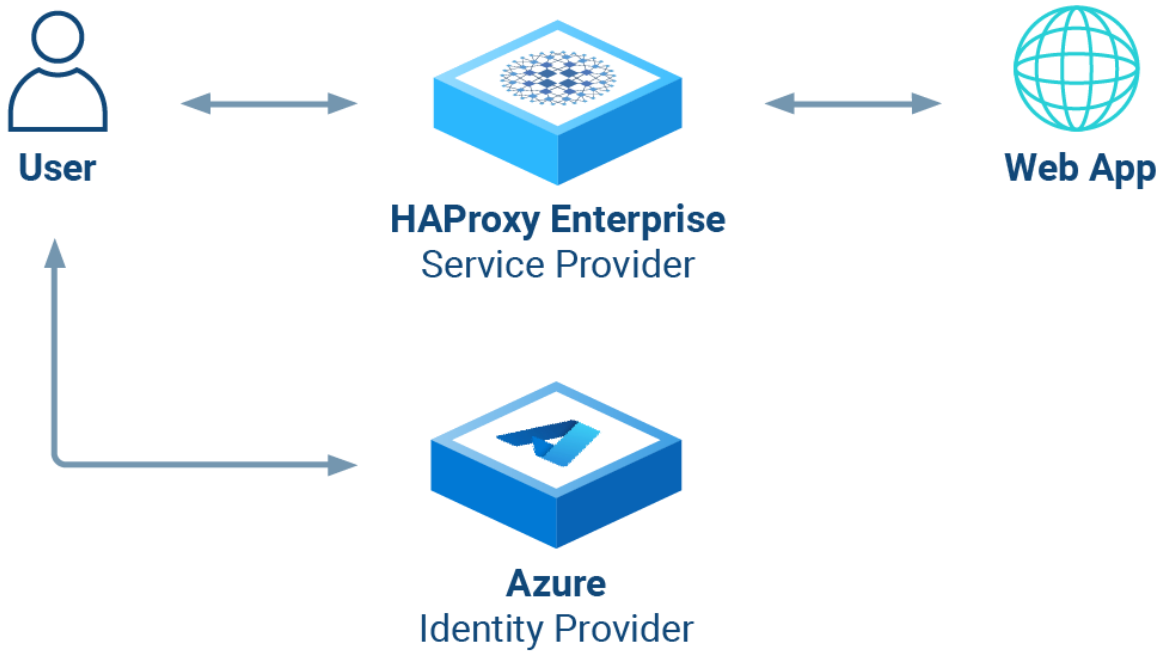
Single sign-on (SAML)

This page applies to:

- HAProxy Enterprise 3.0r1 and newer

HAProxy Enterprise's Security Assertion Markup Language (SAML) module acts as a SAML Service Provider, providing single-sign-on (SSO) to any web application located behind an HAProxy Enterprise server.

This section provides an overview of the SAML module.



About SAML 2.0 and the module

The XML-based Security Assertion Markup Language (SAML) 2.0 open-standard transfers identity data (assertions) between an Identity Provider (IdP) and a Service Provider (SP).

Term	Definition
Identity Provider	Performs authentication on the Service Provider's behalf.
Service Provider	Authorizes users to access the requested resource once they are authenticated by a trusted Identity Provider.

The SAML module acts as a SAML Service Provider. It provides Service Provider-initiated, cross-domain, web-based single-sign-on (SSO) to any web application located behind the load balancer. You then don't have to implement SAML directly in your application.

It works like this:

1. A user visits a web application at its load balanced IP address.
2. The SAML module creates an Authentication Request (AuthnRequest) and redirects the user's browser to the IdP (for example, to Microsoft Entra ID).
3. The user signs in via the IdP's login page, and the IdP validates the credentials. Then the IdP redirects the user back to the load balancer.
4. The SAML module verifies the SAML response from the IdP and then allows the user to proceed to the web application. An HTTP cookie ensures that the user will not need to log in again during their session.

Features:

- Implement SSO seamlessly, even for legacy web applications.
- Configure logging and grant access using ACLs.
- Check SAML assertions or attributes with XPath (via the `saml.ini` file).
- Retrieve SAML assertions and use them as variables in your load balancer configuration. For example, you can then enhance logs and pass user information to the application via HTTP headers.

You may find the following resources helpful for learning SAML concepts:

- Microsoft's guide: SAML authentication with Microsoft Entra ID
- Okta's SAML guide

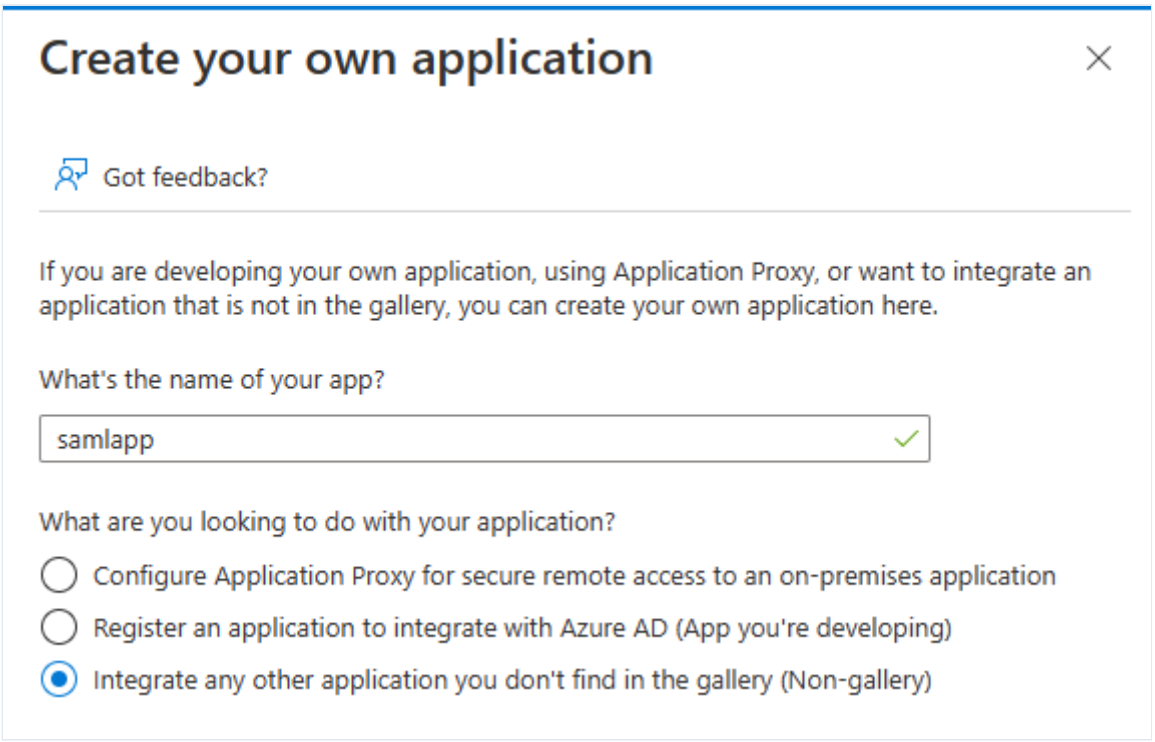
Configure the Identity Provider

The Identity Provider (IdP) manages the user accounts and passwords, provides a login page, and validates credentials. The SAML module relies on you using a third-party IdP. In this section, we'll describe how to set this up.

Use Microsoft Entra ID

This section describes how to configure Microsoft Entra ID (formerly Azure Active Directory) as the IdP.

1. Sign in to the Azure portal.
2. Search for and select **Microsoft Entra ID**. If you manage multiple tenants, then choose **Manage tenants**, select the tenant with which you would like to enable single sign-on, and click **Switch**.
3. From the **Microsoft Entra ID Overview** page, add a new **Enterprise application**.
4. On the **Browse Microsoft Entra Gallery** screen, click **Create your own application**.
   - Give the app a name, such as `samlapp`.
   - Choose **Integrate any other application you don't find in the gallery (Non-gallery)**.
   - Then click **Create**.



5. Click **Assign users and groups**. Choose users and groups that should get sign-on access to your application. Then return to the **Overview** screen.
6. Click **Set up single sign on**, then choose **SAML**.

   The **Set up Single Sign-On with SAML** page opens. Edit the **Basic SAML Configuration**:

Field	Description
Identifier (Entity ID)	Choose a unique identifier for your application, such as `samlapp`.
Reply URL (Assertion Consumer Service URL)	Set the URL at which HAProxy Enterprise will receive the SAML authentication token. For example, `https://example.com/saml/reply`.
Logout URL	Set the URL at which HAProxy Enterprise will receive a logout message from Microsoft Entra ID. For example, `https://example.com/saml/logout`.

Save and then close the basic SAML configuration panel.

7. Still on the **Set up Single Sign-On with SAML** page, edit the **SAML Certificates**.
   - For the **Signing option** choose **Sign SAML response and assertion** and set **Signing Algorithm** to `SHA-256`.
   - Save and close the SAML Signing Certificate panel.
8. You will need several properties of your Microsoft Entra ID enterprise application later when you configure HAProxy Enterprise. Save the following property values:

Property	Where to find it
Name	On the enterprise application's Overview page.
Application ID	On the enterprise application's Overview page.
Tenant ID	On the Microsoft Entra ID Overview page.

Configure the SAML module

This section describes how to configure the SAML module on the load balancer.

1. On your HAProxy Enterprise load balancer, install the `hapee-<VERSION>-lb-saml-sso` package via your system's package manager:

Apt:

```
sudo apt-get install hapee-3.1r1-lb-saml-sso
```

Yum:

```
sudo yum install hapee-3.1r1-lb-saml-sso
```

Zypper:

```
sudo zypper install hapee-3.1r1-lb-saml-sso
```

Pkg:

```
sudo pkg install hapee-3.1r1-lb-saml-sso
```

2. This creates the folder `/etc/hapee-<VERSION>/saml_examples`. For Azure, copy the files from the `saml_examples/azure` to the parent directory, `/etc/hapee-<VERSION>/`. The files include:

   - `authn_request.xml`
   - `logout_request.xml`
   - `saml.ini`

3. Edit the copied file `saml.ini`. At the top of the file, change the values of the following fields to match your Microsoft Entra ID enterprise application's properties:

Property	Set it to
`{{ID_APP_NAME}}`	Your Microsoft Entra ID enterprise application's Name, from the enterprise application's Overview page.
`{{IDP_APP_ID}}`	Your Microsoft Entra ID enterprise application's Application ID, from the enterprise application's Overview page.
`{{IDP_TENANT_ID}}`	Your Microsoft Entra ID Tenant ID, from the Microsoft Entra ID Overview page.
`{{APP_FQDN}}`	The fully qualified domain name where HAProxy Enterprise listens for requests, such as `example.com`. This should match the FQDN you used when setting the Reply URL and Logout URL in Azure. Be sure to add this record in your DNS server so that users can access your application at this domain. Note that if this uses HTTPS, then you should configure your `bind` line in the load balancer configuration to also listen on HTTPS.

An example portion of the `saml.ini` configuration:

saml.ini

```
[MySAMLApp]
{{ID_APP_NAME}} = samlapp
{{IDP_APP_ID}} = 0fbb284d-39ea-4fc6-9639-114b46b8dcb3
{{IDP_TENANT_ID}} = abcdefg-1234-5678-abcd-efgh12345678
{{CLAIM_PREFIX}} = http://schemas.xmlsoap.org/ws/2005/05/identity/claims
{{APP_FQDN}} = example.com
{{APP_LOGIN_URL}} = https://{{APP_FQDN}}/saml/reply
{{APP_LOGOUT_URL}} = https://{{APP_FQDN}}/saml/logout
{{APP_BACKEND}} = bk-{{APP_NAME}}
```

Running multiple applications

Although the configuration in `saml.ini` defines only one SAML app named `MySAMLApp`, you can define multiple single sign-on apps. Copy all lines, except for the `config_version` line, and paste them below the existing lines, but replace `MySAMLApp` with a new name such as `App2`. Then change the properties to match your second application.

4. Also in the `saml.ini` file, add the following lines, changing `mysecret` to be your own, secret string. The module will use this to encrypt the sign-on cookie. If you ever need to change the secret, then store the new secret in `saml_secret_2` and set `current_saml_secret` to `2`.

saml.ini

```
saml_secret_1 = mysecret
current_saml_secret = 1
```

5. Edit the HAProxy Enterprise configuration file, `/etc/hapee-3.1/hapee-lb.cfg`.

- In the `global` section of your configuration, add the following lines. The `saml-sso-load` directive takes two arguments. The first is the path to the directory where you've stored the `authn_request.xml` and `logout_request.xml` files. The second is the path to your `saml.ini` file.

hapee-lb.cfg

```
global
  module-path /opt/hapee-3.1/modules
  module-load hapee-lb-saml-sso.so
  saml-sso-load /etc/hapee-2.9 /etc/hapee-2.9/saml.ini
```

- Copy the SAML section below into the `frontend` section for your load balanced application. Change the arguments given to the `http-request saml-sso` and `http-response saml-sso` directives to match the app name in `saml.ini` (for example, `MySAMLApp`). Also change the `if` statement to use your fully qualified domain name.

  Below, we add the directives to enable SAML single sign-on in the frontend:

hapee-lb.cfg

```
frontend fe_main
  bind :80
  bind :443 ssl crt /etc/hapee-3.1/ssl/cert.pem
  mode http
  option http-buffer-request
  tcp-request inspect-delay 5s

  # --------------------
  # ----SAML section----
  # --------------------
  http-request saml-sso MySAMLApp if { hdr(host) -i example.com }
  http-response saml-sso MySAMLApp
  default_backend bk-err
  use_backend %[var(txn.saml.saml_app_backend)] if { var(txn.saml.saml_app_backend) -m found }
  # ------------------------
  # ----end SAML section----
  # ------------------------
```

6. Add two `backend` sections:

   - A backend named `bk-err` for when there are errors:

     hapee-lb.cfg

     ```
     backend bk-err
       mode http
     ```

   - A backend for your load balanced servers. The name should begin with `bk-` and end with the `saml.ini` section name (for example, `MySAMLApp`):

     hapee-lb.cfg

     ```
     backend bk-MySAMLApp
       option forwardfor
       server app1 127.0.0.1:8000
     ```

7. Restart the HAProxy Enterprise service.

   ```
   sudo systemctl restart hapee-3.1-lb
   ```

   You can then make requests to your application at the FQDN you configured and you will be redirected to the IdP login page. Be sure to use HTTPS if that is what you set for the Reply URL.

Verify the signature of the SAML Response

When Azure sends its SAML response that contains the information HAProxy Enterprise needs to authorize a user to access an application, it is sending an XML token. To prove that it is the trusted issuer of that token, it digitally signs it with its secret key. We recommend that you verify that key using the key's associated public X.509 certificate.

To enable signature verification:

1. Sign in to the Azure portal.
2. Search for and select **Microsoft Entra ID**. If you manage multiple tenants, then choose **Manage tenants** and then select the tenant with which you would like to enable single sign-on.
3. From the **Microsoft Entra ID Overview** page, choose **Enterprise applications**, select your application, then select **Single sign-on** in the left-hand menu.
4. From the **Set up Single Sign-On with SAML** screen, edit the SAML Certificates. Ensure that **Signing Option** is set to **Sign SAML response and assertion** and that **Signing Algorithm** is set to `SHA-256`.

5. From the SAML Certificates section, download the certificate (Base64 format). This downloads a file with a `.cer` extension. Copy this file to your HAProxy Enterprise load balancer.

6. On the HAProxy Enterprise server, edit the file `saml.ini`.

   Uncomment (remove the preceding semicolon) the lines `idp_public_cert` and `verify_signature`. Set `idp_public_cert` to the path of the certificate from your Microsoft Entra ID enterprise application.

   Below, we enable signature verification using your application's certificate:

   saml.ini

   ```
   idp_public_cert = /etc/hapee-3.1/samlapp.cer
   verify_signature=1
   ```

7. Restart the HAProxy Enterprise service.

   ```
   sudo systemctl restart hapee-3.1-lb
   ```

Set headers and variables from attributes

If the user authenticates successfully via the IdP, the IdP sends a SAMLResponse back to the SP. Part of the response is the AttributeStatement, which holds attributes like the user's given name, surname, and email address. To see those attributes, you can set response headers that contain them. The SAML module will create the response headers automatically if you use the `on_saml_response check_attr` action with the `set_var` option.

1. Start by getting a list of assertions that are available in the AttributeStatement. Use `set_var` to get all of the names. Add this to your `saml.ini` file:

   saml.ini

   ```
   ; Get all the SAML attributes names
   on_saml_response check_attr set_var=all_attributes_names  set_var_sep=| xpath=/samlp:Response/saml:Assertion/
   saml:AttributeStatement/saml:Attribute/@Name
   ```

2. Use your browser's Dev tools to inspect the response after logging in successfully. Below, we see that a response header named `All_attributes_names` names has been added and contains all of the attributes (line breaks added for readability):

```
All_attributes_names: http://schemas.microsoft.com/identity/claims/tenantid|
  http://schemas.microsoft.com/identity/claims/objectidentifier|
  http://schemas.microsoft.com/identity/claims/displayname|
  http://schemas.microsoft.com/identity/claims/identityprovider|
  http://schemas.microsoft.com/claims/authnmethodsreferences|
  http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname|
  http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname|
  http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress|
  http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name|
```

3. Add `on_saml_response check_attr set_var` lines for the attributes you want. Below, we get the attribute `displayname`:

   saml.ini

   ```
   on_saml_response check_attr set_var=display_name xpath=/samlp:Response/saml:Assertion/saml:AttributeStatement/
   saml:Attribute[@Name='http://schemas.microsoft.com/identity/claims/displayname']/saml:AttributeValue/text()
   ```

   The response will then include a header named `display_name`:

   ```
   Display_name: Joe Example
   ```

4. Attributes you capture with the `set_var` option are also made available as variables in your load balancer configuration. For example, the `display_name` variable becomes available as `txn.saml.display_name`. Below, we add it to the access log:

   hapee-lb.cfg

   ```
   frontend fe_main
     log-format "$HAPROXY_HTTP_LOG_FMT display_name=%[var(txn.saml.display_name)]"
   ```

Log out

Users can log out of your application by visiting the `/saml/logout` URL path, such as `https://example.com/saml/logout`. This will send a `LogoutRequest` to the IdP, and then the IdP will send the user back to the application for local logout of the app. Successful logouts will set the variable `txn.saml.saml_logout_ok` to `1`.

Considerations:

- If your browser blocks the logout cookie, which will often come from a different domain, then you can set the `saml_cookie_samesite` directive to `None`.

SAML configuration reference

This section describes the syntax of the file `saml.ini`.

Directives

The `saml.ini` file configures how the SAML module integrates with the SAML identity provider. It supports the following configuration directives.

Directive	Description	Type
`idp_login_url`	URL of the Web authentication portal of the Identity Provider. On Microsoft Azure, `https://login.microsoftonline.com/{{IDP_TENANT_ID}}/saml2`.	string
`config_version`	The version of the configuration file. Maintains compatibility with future versions.	string
`idp_logout_url`	Single Logout URL: Endpoint which initiates the SAML Logout for all applications. On Microsoft Azure, `https://login.microsoftonline.com/{{IDP_TENANT_ID}}/saml2`.	string
`idp_referer_url`	HTTP Referer value to check when receiving HTTP data from the Identity Provider. On Microsoft Azure, `https://login.microsoftonline.com/`.	string
`app_login_url`	URL where the application expects to receive the SAML Response from the Identity Provider. The reply URL is also referred to as the Assertion Consumer Service (ACS).	string
`app_logout_url`	When the user browses this URL, initiate a `LogoutRequest` to the Identity Provider.	string
`signing_algo`	Cryptographic algorithm used to sign the requests we send. Specify one of: `ecdsa-sha1`, `ecdsa-sha256`, `ecdsa-sha384`, `ecdsa-sha512`, `rsa-sha1`, `rsa-sha256`, `rsa-sha384`, or `rsa-sha512`.	string
`idp_public_cert`	X509 public cert of the Identity Provider (in base64 form, `.pem`) used to verify SAML Response `Response` and `Assertion` attributes.	string
`verify_signature`	Verify the signature of incoming SAML requests. Default: 0	boolean
`require_signed_response`	Fail if the XML response is not signed. Default: 0	boolean
`require_signed_assertion`	Fail if the XML assertion is not signed. Default: 0	boolean
`signing_key`	Private key used to sign requests we send.	string
`sign_authn_requests`	Set to 1 if you want to sign Authn Requests. Default: 0	boolean
`sign_logout_requests`	Set to 1 if you want to sign `LogoutRequest` requests. Default: 0	boolean
`saml_app_backend`	Backend name for this application. Default: `bk-{{APP_NAME}}`	string
`saml_cookie_secure`	Set to 1 if you want cookies to be used for HTTPS connections only (not HTTP). For more information, see the Wikipedia page about Secure cookie. For a related troubleshooting tip, see Redirect loops. Default: 0	boolean
`saml_cookie_samesite`	`SameSite` cookie attribute value. For more information, see the Wikipedia page about Same-site cookie. For a related troubleshooting tip, see Redirect loops.	string
`saml_cookie_httponly`	`HttpOnly` cookie attribute value. For more information, see the Wikipedia page about HTTP-only cookie. Default: 1	boolean
`saml_cookie_time_offset`	Cookie time offset in seconds (used to build `Expires` cookie attribute). Default: 0	integer
`saml_cookie_lifetime`	Cookie lifetime in seconds (used to build `Expires` cookie attribute). Default: 36000	integer

Directive	Description	Type
`saml_cookie_domain`	`Domain` cookie attribute value. For a related troubleshooting tip, see Redirect loops.	string
`saml_secret_1`	Secret string used to cipher the SAML cookies. Required if `current_saml_secret` is 1.	string
`saml_secret_2`	Secret string used to cipher the SAML cookies. Required if `current_saml_secret` is 2.	string
`current_saml_secret`	The current secret number ID used to cipher the SAML cookies. Select 1 to use `saml_secret_1` or 2 to `use saml_secret_2`. Defaults to 1.	integer
`authn_request_template_filename`	`Authn Request` template filename.	string
`logout_request_template_filename`	`LogoutRequest` template filename.	string

Actions

In the `saml.ini` file, actions let you validate the schema, set variables, and other tasks.

Action	Description
`on_saml_response check_attr`	Check an arbitrary attribute in a SAML Response. To store the attribute value in a load balancer variable `txn.my_var_name`, use `XPath= set_var=`. To set per application variables, use `set_var={{APP_NAME}}.my_var_name`. `{{APP_NAME}}` is replaced with the application name (that is, the section name in `saml.ini`). The SAML Response validation fails if an attribute is not present unless you set the `optional` flag.
`on_saml_response check_attr entity_id`	Check that the `audience` attribute exists. The specific `entity_id` value to check in required.
`on_saml_response check_attr version`	Check that SAML protocol version is 2.0.
`on_saml_response check_attr status_code`	Check the `SAMLResponse` status code. The status to match, `status_code`, is optional. Otherwise, compare to `urn:oasis:names:tc:SAML:2.0:status:Success`.
`on_saml_response check_attr destination`	Check that the `SAMLResponse Destination` value matches this item. `destination` is required. To match the configured `app_login_url`, use `<APP_LOGIN_URL>`. On Microsoft Azure, it must match the URL Assertion Consumer Service (ACS).
`on_saml_response check_attr issuer`	Check the `Issuer` attribute of the SAMLResponse. The specific issuer value to check is optional.
`on_saml_response check_attr issue_instant`	Check that the `IssueInstant` attribute exists. To store it in a variable, use `set_var`. To store it in a timestamp variable, use `set_var_as_timestamp`.
`on_saml_response check_attr assertion`	Check that the `Assertion` attribute exists.
`on_saml_response check_schema`	Validate the SAML response against the SAML 2.0 `xsd` schema.
`on_saml_response check_conditions`	Check the XML attribute, including `NotBefore` and `NotOnOrAfter` values.
`on_saml_response check_subject_confirmation_data`	Check the XML attribute, including `NotBefore` and `NotOnOrAfter` values.
`on_logout_request check_attr`	Check an arbitrary attribute in a `LogoutRequest`. To put it in a load balancer variable, use `XPath= set_var=`.
`on_logout_request check_attr issuer`	Check the `issuer` attribute of the `LogoutRequest`. The specific issuer value to check is optional.
`on_logout_request check_attr name_id`	Check that the `nameId` attribute exists in the `LogoutRequest`.
`on_logout_request check_attr destination`	Check that the `LogoutRequest Destination` value matches this item. The destination to match is required. To match the configured `app_logout_url`, use `<APP_LOGOUT_URL>`.

Action flags

Actions accept the following flags:

Flag	Description
`optional`	This argument is not required.
`required`	This argument is mandatory.
`nofail`	For testing purposes. This action never fails, even if it returns an error.
`required_count=<count>`	Fail if the number of searched elements is different from `<count>`.
`xpath=<XPath expression>`	XPath expression to look for. Use with `check_attr` actions.
`expected=<expected_value>`	Fail if the result is different from this expression.
`set_var=<var_name>`	When one or more XPath results are found, store its value in this specific variable. The variable name is prefixed with the application name, then with a dot.
`set_var_once`	The variable is set only after the POST from the SAML Identity Provider. Otherwise, it is set each time we see the cookie again.
`set_var_cnt`	The number of XPath results is stored in the variable `<APP_NAME>.<VAR_NAME>_cnt`.
`set_var_as_timestamp`	When used with a value in ISO 8601 date and time format (for instance `2020-01-28T15:25:14.884Z`), the variable is converted to a timestamp.
`set_var_sep=<separator>`	When multiple results are returned from the XPath query, separate them with this character.
`set_var_default=<default_value>`	Default value used if the XPath expression does not match.

Troubleshooting

This section covers ways to troubleshoot the SAML module.

Start a debug trace

To get a verbose log of the authentication process, which includes the full SAMLResponse, enable these traces:

```
(echo "trace saml-sso sink buf0; trace saml-sso level developer; trace saml-sso verbosity complete; trace saml-sso start now; show events buf0 -w" ; cat ) | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

To stop the traces, call this command:

```
(echo "trace saml-sso stop now") | sudo socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock
```

Redirect loops

If you try to access a resource and get redirected back and forth between your application and the IdP, it could be that your browser is not sending back the cookie that the SAML component just set.

- Check your `saml_cookie_secure` setting. If it is set to `1`, your browser will only use the cookie when using HTTPS, not plain HTTP.
- Check your `saml_cookie_domain` and make sure it matches the domain of your application.
- In some situations, excessively strict security settings of your browsers can also prevent the cookie from being sent to your application. Also, you might need to set the `saml_cookie_samesite` value to `None` (which may also require setting `saml_cookie_secure` to `1` on browsers).

Single sign-on (SAML) - legacy module

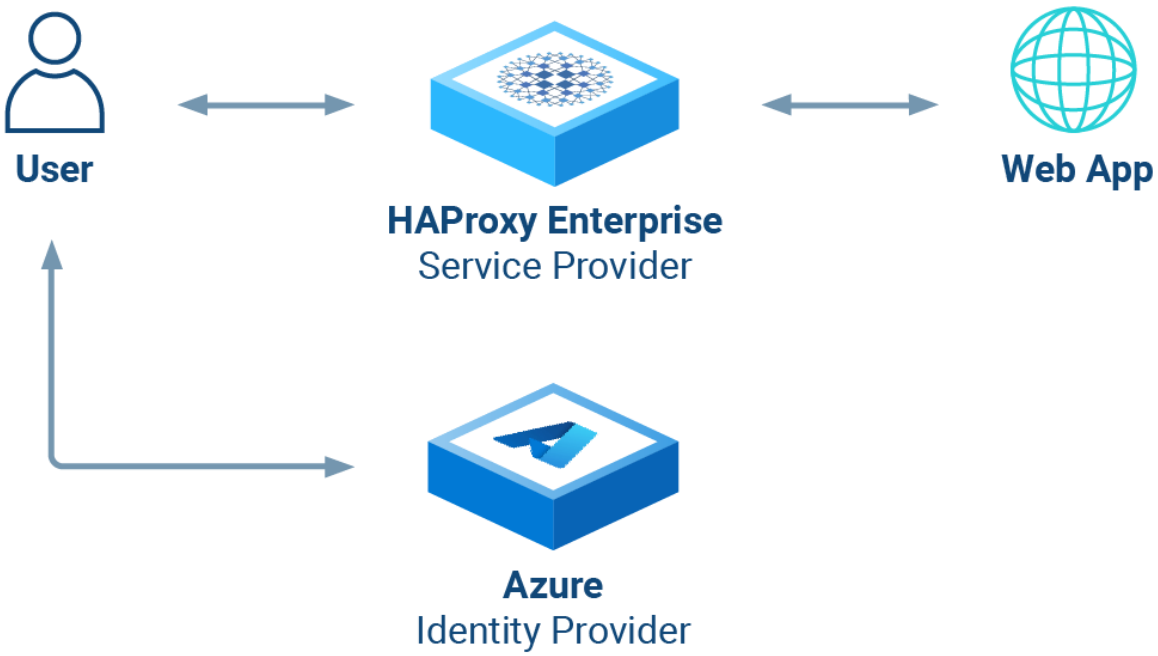
Deprecation notice

This module has been marked as deprecated and is scheduled for removal. HAProxy Enterprise 3.0 will be the last version to support it.

Commands and examples on this page specify 3.0 as the version number. However, the commands and examples apply equally to earlier versions.

HAProxy Enterprise's Security Assertion Markup Language (SAML) module acts as a SAML Service Provider, providing single-sign-on (SSO) to any web application located behind an HAProxy Enterprise server.

This section provides an overview of the SAML module.



About SAML 2.0 and the module

The XML-based Security Assertion Markup Language (SAML) 2.0 open-standard transfers identity data (assertions) between an Identity Provider (IdP) and a Service Provider (SP).

Term	Definition
Identity Provider	Performs authentication on the Service Provider's behalf.
Service Provider	Authorizes users to access the requested resource once they are authenticated by a trusted Identity Provider.

The SAML module acts as a SAML Service Provider. It provides Service Provider-initiated, cross-domain, web-based single-sign-on (SSO) to any web application located behind the load balancer. You then don't have to implement SAML directly in your application.

It works like this:

1. A user visits a web application at its load balanced IP address.
2. The SAML module creates an Authentication Request (AuthnRequest) and redirects the user's browser to the IdP (for example, to Microsoft Entra ID).
3. The user signs in via the IdP's login page, and the IdP validates the credentials. Then the IdP redirects the user back to the load balancer.
4. The SAML module verifies the SAML response from the IdP and then allows the user to proceed to the web application. An HTTP cookie ensures that the user will not need to log in again during their session.

Features:

- Implement SSO seamlessly, even for legacy web applications.
- Configure logging and grant access using ACLs.
- Check SAML assertions or attributes with XPath (via the `saml.ini` file).
- Retrieve SAML assertions and use them as variables in your load balancer configuration. For example, you can then enhance logs and pass user information to the application via HTTP headers.

You may find the following resources helpful for learning SAML concepts:

- Microsoft's guide: SAML authentication with Microsoft Entra ID
- Okta's SAML guide

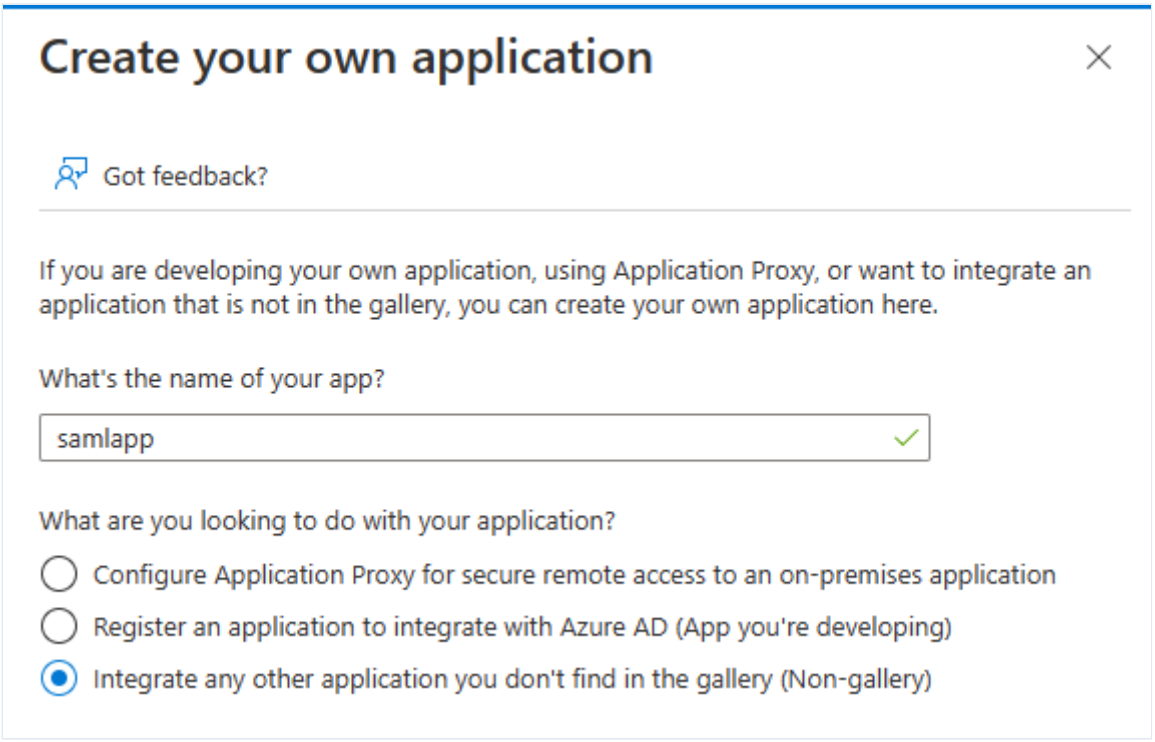
Configure the Identity Provider

The Identity Provider (IdP) manages the user accounts and passwords, provides a login page, and validates credentials. The SAML module relies on you using a third-party IdP. In this section, we'll describe how to set this up.

Use Microsoft Entra ID

This section describes how to configure Microsoft Entra ID (formerly Azure Active Directory) as the IdP.

1. Sign in to the Azure portal.
2. Search for and select **Microsoft Entra ID**. If you manage multiple tenants, then choose **Manage tenants**, select the tenant with which you would like to enable single sign-on, and click **Switch**.
3. From the **Microsoft Entra ID Overview** page, add a new **Enterprise application**.
4. On the **Browse Microsoft Entra Gallery** screen, click **Create your own application**.
   - Give the app a name, such as `samlapp`.
   - Choose **Integrate any other application you don't find in the gallery (Non-gallery)**.
   - Then click **Create**.



5. Click **Assign users and groups**. Choose users and groups that should get sign-on access to your application. Then return to the **Overview** screen.

6. Click **Set up single sign on**, then choose **SAML**.

   The **Set up Single Sign-On with SAML** page opens. Edit the **Basic SAML Configuration**:

Field	Description
Identifier (Entity ID)	Choose a unique identifier for your application, such as `samlapp`.
Reply URL (Assertion Consumer Service URL)	Set the URL at which HAProxy Enterprise will receive the SAML authentication token. For example, `https://example.com/saml/reply`.
Logout URL	Set the URL at which HAProxy Enterprise will receive a logout message from Microsoft Entra ID. For example, `https://example.com/saml/logout`.

   Save and then close the basic SAML configuration panel.

7. Still on the **Set up Single Sign-On with SAML** page, edit the **SAML Certificates**.

   - For the **Signing option** choose **Sign SAML response and assertion** and set **Signing Algorithm** to `SHA-256`.
   - Save and close the SAML Signing Certificate panel.

8. You will need several properties of your Microsoft Entra ID enterprise application later when you configure HAProxy Enterprise. Save the following property values:

Property	Where to find it
Name	On the enterprise application's Overview page.
Application ID	On the enterprise application's Overview page.
Tenant ID	On the Microsoft Entra ID Overview page.

Configure the SAML module

This section describes how to configure the SAML module on the load balancer.

1. On your HAProxy Enterprise load balancer, install the SAML module via your system's package manager.

Apt:

```
sudo apt-get install hapee-extras-spoa-saml20
```

Yum:

```
sudo yum install hapee-extras-spoa-saml20
```

Zypper:

```
sudo zypper install hapee-extras-spoa-saml20
```

Pkg:

```
sudo pkg install hapee-extras-spoa-saml20
```

2. This creates the folder `/etc/hapee-extras/saml_examples/azure`. Copy the files from that folder to `/etc/hapee-extras/`. The files include:

   - `authn_request.xml`
   - `logout_request.xml`
   - `saml.ini`

3. Edit the copied file `saml.ini`. At the top of the file, change the values of the following fields to match your Microsoft Entra ID enterprise application's properties:

Property	Set it to
`{{ID_APP_NAME}}`	Your Microsoft Entra ID enterprise application's Name.
`{{IDP_APP_ID}}`	Your Microsoft Entra ID enterprise application's Application ID.
`{{IDP_TENANT_ID}}`	Your Microsoft Entra ID Tenant ID.
`{{APP_FQDN}}`	The fully qualified domain name where HAProxy Enterprise listens for requests, such as `example.com`. This should match the FQDN you used when setting the Reply URL and Logout URL in Azure. Be sure to add this record in your DNS server so that users can access your application at this domain. Note that if this uses HTTPS, then you should configure your `bind` line in the load balancer configuration to also listen on HTTPS.

An example portion of the `saml.ini` configuration:

saml.ini

```
[MySAMLApp]
{{ID_APP_NAME}} = samlapp
{{IDP_APP_ID}} = 0fbb284d-39ea-4fc6-9639-114b46b8dcb3
{{IDP_TENANT_ID}} = abcdefg-1234-5678-abcd-efgh12345678
{{CLAIM_PREFIX}} = http://schemas.xmlsoap.org/ws/2005/05/identity/claims
{{APP_FQDN}} = example.com
{{APP_LOGIN_URL}} = https://{{APP_FQDN}}/saml/reply
{{APP_LOGOUT_URL}} = https://{{APP_FQDN}}/saml/logout
{{APP_BACKEND}} = bk-{{APP_NAME}}
```

Running multiple applications

Although the configuration in `saml.ini` defines only one SAML app named `MySAMLApp`, you can define multiple single sign-on apps. Copy all lines, except for the `config_version` line, and paste them below the existing lines, but replace `MySAMLApp` with a new name such as `App2`. Then change the properties to match your second application.

4. Also in the `saml.ini` file, add the following lines, changing `mysecret` to be your own, secret string. The module will use this to encrypt the sign-on cookie. If you ever need to change the secret, then store the new secret in `saml_secret_2` and set `current_saml_secret` to `2`.

saml.ini

```
saml_secret_1 = mysecret
current_saml_secret = 1
```

5. Edit the HAProxy Enterprise configuration file, `/etc/hapee-3.0/hapee-lb.cfg`.

   Copy the SAML section below into the `frontend` section for your load balanced application. Modify the line that sets the variable `sess.saml_app` to match the app name in `saml.ini` (for example, `MySAMLApp`), and change the `if` statement to use your fully qualified domain name. The rest does not need to be changed.

   Below, we add the directives to enable SAML single sign-on in the frontend:

hapee-lb.cfg

```
frontend fe_main
  bind :80
  bind :443 ssl crt /etc/hapee-3.0/ssl/cert.pem
  mode http
  option http-buffer-request
  tcp-request inspect-delay 5s

  # --------------------
  # ----SAML section----
  # --------------------
  # here, replace MySAMLApp and the if statement with correct FQDN, depending on your setup
  http-request set-var(sess.saml_app) str(MySAMLApp) if { hdr(host) -i example.com }

  filter spoe engine spoe-saml config /etc/hapee-extras/hapee-saml-spoe.cfg

  http-request send-spoe-group spoe-saml spoe-group-req
  http-request redirect location %[var(txn.saml.redirect_to)] code 302 if { var(txn.saml.redirect_to) -m found }
  http-request deny if ! { var(txn.saml.saml_auth_ok) -m bool } ! { var(txn.saml.saml_logout_ok) -m bool }

  http-response set-header Set-Cookie %[var(txn.saml.set_cookie)] if { var(txn.saml.set_cookie) -m found }
  http-response set-header saml-auth-error-text %[var(txn.saml.saml_auth_error_text)] if
{ var(txn.saml.saml_auth_error_text) -m found }
  http-response set-header location %[var(txn.saml.redirect_after_auth)] if { var(txn.saml.redirect_after_auth)  -m
found }
  http-response set-status 303 if { var(txn.saml.redirect_after_auth)  -m found }
  # ------------------------
  # ----end SAML section----
  # ------------------------

  # The backend to send authenticated requests to
  default_backend webservers
```

6. Add a `backend` section for the SAML module service. HAProxy Enterprise uses this when authenticating requests.

   Below, we add a backend for the SAML module service:

hapee-lb.cfg

```
backend bk-spoe
  mode tcp
  timeout connect 5s
  timeout server 30s
  server auth-server-spoe 127.0.0.1:12345
```

7. Enable and start the SAML module service.

```
sudo systemctl enable hapee-extras-spoa-saml
sudo systemctl start hapee-extras-spoa-saml
```

8. Restart the HAProxy Enterprise service.

```
sudo systemctl restart hapee-3.0-lb
```

You can then make requests to your application at the FQDN you configured and you will be redirected to the IdP login page. Be sure to use HTTPS if that is what you set for the Reply URL.

Optional: Verify the signature of the SAML Response

When Azure sends its SAML response that contains the information HAProxy Enterprise needs to authorize a user to access an application, it is sending an XML token. To prove that it is the trusted issuer of that token, it digitally signs it with its secret key. You can verify that key using the key's associated public X.509 certificate.

To enable signature verification:

1. Sign in to the Azure portal.
2. Search for and select **Microsoft Entra ID**. If you manage multiple tenants, then choose **Manage tenants** and then select the tenant with which you would like to enable single sign-on.
3. From the **Microsoft Entra ID Overview** page, choose **Enterprise applications**, select your application, then select **Single sign-on** in the left-hand menu.
4. From the **Set up Single Sign-On with SAML** screen, edit the SAML Certificates. Ensure that **Signing Option** is set to **Sign SAML response and assertion** and that **Signing Algorithm** is set to `SHA-256`.
5. From the SAML Certificates section, download the certificate (Base64 format). This downloads a file with a `.cer` extension. Copy this file to your HAProxy Enterprise load balancer, such as to `/etc/hapee-extras/`.
6. On the HAProxy Enterprise server, edit the file `/etc/hapee-extras/saml.ini`.

   Uncomment (remove the preceding semicolon) the lines `idp_public_cert` and `verify_signature`. Set `idp_public_cert` to the path of the certificate from your Microsoft Entra ID enterprise application.

   Below, we enable signature verification using your application's certificate:

saml.ini

```
idp_public_cert = /etc/hapee-extras/samlapp.cer
verify_signature=1
```

7. Restart the SAML service.

```
sudo systemctl restart hapee-extras-spoa-saml
```

Log out

Users can log out of your application by visiting the `/saml/logout` URL path, such as `https://example.com/saml/logout`. This will send a `LogoutRequest` to the IdP, and then the IdP will send the user back to the application for local logout of the app. Successful logouts will set the variable `txn.saml.saml_logout_ok` to `1`.

Considerations:

- If your browser blocks the logout cookie, which will often come from a different domain, then you can set the `saml_cookie_samesite` directive to `None`.

SAML configuration reference

This section describes the syntax of the file `saml.ini`.

Directives

The `saml.ini` file configures how the SAML module integrates with the SAML identity provider. It supports the following configuration directives.

Directive	Description	Type
`idp_login_url`	URL of the Web authentication portal of the Identity Provider. On Microsoft Azure, `https://login.microsoftonline.com/{{IDP_TENANT_ID}}/saml2`.	string
`config_version`	The version of the configuration file. Maintains compatibility with future versions.	string
`idp_logout_url`	Single Logout URL: Endpoint which initiates the SAML Logout for all applications. On Microsoft Azure, `https://login.microsoftonline.com/{{IDP_TENANT_ID}}/saml2`.	string
`idp_referer_url`	HTTP Referer value to check when receiving HTTP data from the Identity Provider. On Microsoft Azure, `https://login.microsoftonline.com/`.	string
`app_login_url`	URL where the application expects to receive the SAML Response from the Identity Provider. The reply URL is also referred to as the Assertion Consumer Service (ACS).	string
`app_logout_url`	When the user browses this URL, initiate a `LogoutRequest` to the Identity Provider.	string
`signing_algo`	Cryptographic algorithm used to sign the requests we send. Specify one of: `ecdsa-sha1`, `ecdsa-sha256`, `ecdsa-sha384`, `ecdsa-sha512`, `rsa-sha1`, `rsa-sha256`, `rsa-sha384`, or `rsa-sha512`.	string
`idp_public_cert`	X509 public cert of the Identity Provider (in base64 form, `.pem`) used to verify SAML Response `Response` and `Assertion` attributes.	string
`verify_signature`	Verify the signature of incoming SAML requests. Default: 0	boolean
`require_signed_response`	Fail if the XML response is not signed. Default: 0	boolean
`require_signed_assertion`	Fail if the XML assertion is not signed. Default: 0	boolean
`signing_key`	Private key used to sign requests we send.	string
`sign_authn_requests`	Set to 1 if you want to sign Authn Requests. Default: 0	boolean
`sign_logout_requests`	Set to 1 if you want to sign `LogoutRequest` requests. Default: 0	boolean
`saml_app_backend`	Backend name for this application. Default: `bk-{{APP_NAME}}`	string
`saml_cookie_secure`	Set to 1 if you want cookies to be used for HTTPS connections only (not HTTP). For more information, see the Wikipedia page about Secure cookie. For a related troubleshooting tip, see Redirect loops. Default: 0	boolean
`saml_cookie_samesite`	`SameSite` cookie attribute value. For more information, see the Wikipedia page about Same-site cookie. For a related troubleshooting tip, see Redirect loops.	string
`saml_cookie_httponly`	`HttpOnly` cookie attribute value. For more information, see the Wikipedia page about HTTP-only cookie. Default: 1	boolean
`saml_cookie_time_offset`	Cookie time offset in seconds (used to build `Expires` cookie attribute). Default: 0	integer
`saml_cookie_lifetime`	Cookie lifetime in seconds (used to build `Expires` cookie attribute). Default: 36000	integer

Directive	Description	Type
`saml_cookie_domain`	`Domain` cookie attribute value. For a related troubleshooting tip, see Redirect loops.	string
`saml_secret_1`	Secret string used to cipher the SAML cookies. Required if `current_saml_secret` is 1.	string
`saml_secret_2`	Secret string used to cipher the SAML cookies. Required if `current_saml_secret` is 2.	string
`current_saml_secret`	The current secret number ID used to cipher the SAML cookies. Select 1 to use `saml_secret_1` or 2 to `use saml_secret_2`. Defaults to 1.	integer
`authn_request_template_filename`	`Authn Request` template filename.	string
`logout_request_template_filename`	`LogoutRequest` template filename.	string

Actions

In the `saml.ini` file, actions let you validate the schema, set variables, and other tasks.

Action	Description
`on_saml_response check_attr`	Check an arbitrary attribute in a SAML Response. To store the attribute value in a load balancer variable `txn.my_var_name`, use `XPath= set_var=`. To set per application variables, use `set_var={{APP_NAME}}.my_var_name`. `{{APP_NAME}}` is replaced with the application name (that is, the section name in `saml.ini`). The SAML Response validation fails if an attribute is not present unless you set the `optional` flag.
`on_saml_response check_attr entity_id`	Check that the `audience` attribute exists. The specific `entity_id` value to check in required.
`on_saml_response check_attr version`	Check that SAML protocol version is 2.0.
`on_saml_response check_attr status_code`	Check the `SAMLResponse` status code. The status to match, `status_code`, is optional. Otherwise, compare to `urn:oasis:names:tc:SAML:2.0:status:Success`.
`on_saml_response check_attr destination`	Check that the `SAMLResponse Destination` value matches this item. `destination` is required. To match the configured `app_login_url`, use `<APP_LOGIN_URL>`. On Microsoft Azure, it must match the URL Assertion Consumer Service (ACS).
`on_saml_response check_attr issuer`	Check the `Issuer` attribute of the SAMLResponse. The specific issuer value to check is optional.
`on_saml_response check_attr issue_instant`	Check that the `IssueInstant` attribute exists. To store it in a variable, use `set_var`. To store it in a timestamp variable, use `set_var_as_timestamp`.
`on_saml_response check_attr assertion`	Check that the `Assertion` attribute exists.
`on_saml_response check_schema`	Validate the SAML response against the SAML 2.0 `xsd` schema.
`on_saml_response check_conditions`	Check the XML attribute, including `NotBefore` and `NotOnOrAfter` values.
`on_saml_response check_subject_confirmation_data`	Check the XML attribute, including `NotBefore` and `NotOnOrAfter` values.
`on_logout_request check_attr`	Check an arbitrary attribute in a `LogoutRequest`. To put it in a load balancer variable, use `XPath= set_var=`.
`on_logout_request check_attr issuer`	Check the `issuer` attribute of the `LogoutRequest`. The specific issuer value to check is optional.
`on_logout_request check_attr name_id`	Check that the `nameId` attribute exists in the `LogoutRequest`.
`on_logout_request check_attr destination`	Check that the `LogoutRequest Destination` value matches this item. The destination to match is required. To match the configured `app_logout_url`, use `<APP_LOGOUT_URL>`.

Action flags

Actions accept the following flags:

Flag	Description
`optional`	This argument is not required.
`required`	This argument is mandatory.
`nofail`	For testing purposes. This action never fails, even if it returns an error.
`required_count=<count>`	Fail if the number of searched elements is different from `<count>`.
`xpath=<XPath expression>`	XPath expression to look for. Use with `check_attr` actions.
`expected=<expected_value>`	Fail if the result is different from this expression.
`set_var=<var_name>`	When one or more XPath results are found, store its value in this specific variable. The variable name is prefixed with the application name, then with a dot.
`set_var_once`	The variable is set only after the POST from the SAML Identity Provider. Otherwise, it is set each time we see the cookie again.
`set_var_cnt`	The number of XPath results is stored in the variable `<APP_NAME>.<VAR_NAME>_cnt`.
`set_var_as_timestamp`	When used with a value in ISO 8601 date and time format (for instance `2020-01-28T15:25:14.884Z`), the variable is converted to a timestamp.
`set_var_sep=<separator>`	When multiple results are returned from the XPath query, separate them with this character.
`set_var_default=<default_value>`	Default value used if the XPath expression does not match.

Troubleshooting

Show status of service

Check the status of the SAML module service, which will show any errors it encountered while validating SAML responses.

```
sudo systemctl status hapee-extras-spoa-saml.service
```

Cannot redirect to the IdP form, we already come from it...

- There might be a protocol mismatch. If you have an HTTPS proxy in front of the load balancer, but the load balancer itself is not listening on HTTPS, then change it so that the load balancer listens on HTTPS.
- The URI sent to the SAML service may be wrong, causing validation of the SAML response to fail. In older versions of the module, the configuration file `/etc/hapee-extras/hapee-saml-spoe.cfg` passed the `uri` argument with the wrong value It looked like this:

```
uri=url
```

Change it to:

```
uri=pathq
```

Cannot find config for application "MySAMLApp"

- Ensure that your changes to the `saml.ini` file took effect by restarting the **saml** service.

Debug mode

The SAML module can be launched in full debug mode:

```
systemctl stop hapee-extras-spoa-saml.service
/opt/hapee-extras/bin/hapee-saml -F -f /etc/hapee-extras/saml.ini --debug-all
```

When you are in debug mode, the SAMLResponse is printed in clear.

> **Caution**
>
> Be careful not to expose confidential information that may be included in debug output. SAML responses can include confidential information.

Testing XPath expressions

When you are in debug mode, you can save SAML responses to a text file. Then use `xmllint` (found in the `libxml2` package) to test XPath expressions:

```
xmllint --shell SAMLResponse.xml

setns saml=urn:oasis:names:tc:SAML:2.0:assertion
setns samlp=urn:oasis:names:tc:SAML:2.0:protocol

xpath /samlp:Response/@Destination
```

You may also find other useful XPath testers online.

Redirect loops

If you try to access a resource and get redirected back and forth between your application and the IdP, it could be that your browser is not sending back the cookie that the SAML component just set.

- Check your `saml_cookie_secure` setting. If it is set to `1`, your browser will only use the cookie when using HTTPS, not plain HTTP.
- Check your `saml_cookie_domain` and make sure it matches the domain of your application.
- In some situations, excessively strict security settings of your browsers can also prevent the cookie from being sent to your application. Also, you might need to set the `saml_cookie_samesite` value to `None` (which may also require setting `saml_cookie_secure` to `1` on browsers).

SNMP

This page applies to:

- HAProxy Enterprise - all versions

Simple Network Management Protocol (SNMP) offers a way to collect information about network devices. Having been around for decades, you'll find it in many different types of devices such as routers, switches, servers, and printers. The HAProxy Enterprise SNMP module enables you to collect metrics from the load balancer. The module provides read access to load balancer statistics.

This guide demonstrates the setup using SNMPv3 and SNMPv2.

Install the management software

The SNMP management software is not, strictly speaking, required, but it does provide tools for creating SNMPv3 user accounts and testing that the setup works by making SNMP queries.

1. Install the SNMP management software package:

Apt:

```
sudo apt update
sudo apt install snmp
```

Yum:

```
sudo yum install net-snmp-utils
```

2. On Debian and Ubuntu, perform these additional steps:

   - Configure the management software to load Management Information Bases (MIBs) by editing the file `/etc/snmp/snmp.conf` and commenting out the `mibs` line so that it looks like this:

     snmp.conf

     ```
     # mibs :
     ```

   - Append the path to the HAProxy Enterprise MIB directory, `/opt/hapee-extras/misc`, to the end of the `mibdirs` line, and uncomment it if it is commented out:

     snmp.conf

     ```
     mibdirs /usr/share/snmp/mibs:/usr/share/snmp/mibs/iana:/usr/share/snmp/mibs/ietf:/opt/hapee-extras/misc
     ```

   - Add the `non-free` repository to the existing line in `/etc/apt/sources.list`, which will allow you to install the `snmp-mibs-downloader` package:

     sources.list

     ```
     deb https://deb.debian.org/debian bullseye main non-free
     ```

   - Add the MIB files by installing the `snmp-mibs-downloader` package:

     ```
     sudo apt update
     sudo apt install snmp-mibs-downloader
     ```

Install the agent software

An SNMP agent is software that runs on the monitored device, which in this case is the load balancer. It returns metrics when queried by the management software.

1. Install the SNMP agent software package:

   Apt:

   ```
   sudo apt update
   sudo apt install snmpd
   ```

Yum:

```
sudo yum install net-snmp
```

2. Enable the agent service:

```
sudo systemctl enable snmpd
```

3. Stop the agent service:

```
sudo systemctl stop snmpd
```

4. Configure the agent for the desired version. The agent supports both SNMPv2c and SNMPv3.

   - SNMPv3 is recommended because its security model supports encrypted passwords instead of just plain-text community strings like SNMPv2c.
   - SNMPv2c, on the other hand, is easier to configure. Configuration procedures for both are provided.

Configure SNMPv2c access

To configure agent access for SNMPv2c, which uses community strings, follow these steps:

1. Make a backup of the agent configuration file, `/etc/snmp/snmpd.conf`:

```
sudo cp /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.original
```

2. Open the agent configuration file for editing.

3. Locate the view and community definitions. To read both system information and HAProxy Enterprise information, add these definitions.

snmpd.conf

```
# System + hrSystem groups
view   haproxyview   included   .1.3.6.1.2.1.1
view   haproxyview   included   .1.3.6.1.2.1.25.1

# HAProxy Enterprise groups
view   haproxyview   included   .1.3.6.1.4.1.23263.4.3
view   haproxyview   included   .1.3.6.1.4.1.58750.4.3

rocommunity  haproxy default -V haproxyview
rocommunity6 haproxy default -V haproxyview
```

4. Optional: Add any other view and community definitions required to provide the desired access.

5. Add the following `pass_persist` lines to the end of the agent configuration file. They configure the SNMP agent service to pass requests through to the HAProxy Enterprise SNMP module:

snmpd.conf

```
# HAPEE-LEGACY.MIB
pass_persist   .1.3.6.1.4.1.23263.4.3.1.3  /opt/hapee-extras/bin/hapee-snmp-lb

# HAPEE.MIB
pass_persist   .1.3.6.1.4.1.58750.4.3.1.3  /opt/hapee-extras/bin/hapee-snmp-lb
```

6. On Debian and Ubuntu, to have the agent service listen on all interfaces instead of `127.0.0.1`, comment out the line `agentAddress udp:127.0.0.1:161` if it exists and add the line `agentAddress udp:161,udp6:[::1]:161`.

snmpd.conf

```
# Listen for connections from the local system only
# agentAddress  udp:127.0.0.1:161

# Listen for connections on all interfaces (both IPv4 *and* IPv6)
agentAddress udp:161,udp6:[::1]:161
```

7. Save and close the agent configuration file.

8. Start the agent service:

```
sudo systemctl start snmpd
```

9. Use `snmpwalk` to verify that you can make requests to the agent.

Request a metric:

```
snmpwalk -c public -v2c 127.0.0.1 sysUpTime.0
```

output

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (15234) 0:02:32.34
```

Configure SNMPv3 access

To configure agent access for SNMPv3, which uses user profiles and encrypted passwords, follow these steps:

1. Make a backup of the agent configuration file, `/etc/snmp/snmpd.conf`:

   ```
   sudo cp /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.original
   ```

2. Open the agent configuration file for editing.

3. Add the following `createUser` and `rwuser` lines to the end of the file to create an initial user account with read-write access. Note that this applies to SNMPv3 only, since older versions of the protocol do not require user accounts and instead use a community string for authentication. This initial user account will become the template from which we will create other user accounts.

   snmpd.conf

   ```
   createUser initial SHA setup_passphrase AES setup_passphrase
   rwuser initial
   ```

   Add another `rwuser` line to prepare a second user account with read-write access. Below, we define permissions for a user named `myuser`:

   snmpd.conf

   ```
   rwuser myuser
   ```

4. Locate the view definitions. To read both system information and HAProxy Enterprise information, add the following definitions.

snmpd.conf

```
# System + hrSystem groups
view   haproxyview   included   .1.3.6.1.2.1.1
view   haproxyview   included   .1.3.6.1.2.1.25.1

# HAProxy Enterprise groups
view   haproxyview   included   .1.3.6.1.4.1.23263.4.3
view   haproxyview   included   .1.3.6.1.4.1.58750.4.3
```

5. Optional: Add any other view definitions required to provide the desired access.

6. Add the following `pass_persist` lines to the end of the agent configuration file. They configure the SNMP agent service to pass requests through to the HAProxy Enterprise SNMP module:

snmpd.conf

```
# HAPEE-LEGACY.MIB
pass_persist   .1.3.6.1.4.1.23263.4.3.1.3  /opt/hapee-extras/bin/hapee-snmp-lb

# HAPEE.MIB
pass_persist   .1.3.6.1.4.1.58750.4.3.1.3  /opt/hapee-extras/bin/hapee-snmp-lb
```

7. On Debian and Ubuntu, to have the agent service listen on all interfaces instead of `127.0.0.1`, comment out the line `agentAddress udp:127.0.0.1:161` if it exists and add the line `agentAddress udp:161,udp6:[::1]:161`.

snmpd.conf

```
# Listen for connections from the local system only
# agentAddress  udp:127.0.0.1:161

# Listen for connections on all interfaces (both IPv4 *and* IPv6)
agentAddress udp:161,udp6:[::1]:161
```

8. Start the agent service:

```
sudo systemctl start snmpd
```

9. Using the `initial` user's username and passphrase, create a new user account that you'll use to make SNMP requests. This user will inherit settings from the `initial` user account. Below, we create a user named `myuser`:

```
snmpusm -v3 -n "" -u initial -a SHA -A setup_passphrase -x AES -X setup_passphrase -l authPriv 127.0.0.1 create
myuser initial
```

output

```
User successfully created.
```

10. The new user account inherited the passphrase from the `initial` user account. Use the `snmpusm` command again to change the user's passphrase. Below, we change the `myuser` user account's passphrase to `mypassword`. In a production environment, be sure to use a strong password.

```
snmpusm -v3 -n "" -u myuser -a SHA -A setup_passphrase -x AES -X setup_passphrase -l authPriv 127.0.0.1 passwd
setup_passphrase mypassword
```

output

```
SNMPv3 Key(s) successfully changed.
```

11. To verify that the user you created can make requests to the agent software, you can use the management software's `snmpget` command to fetch some metrics. Try the following:

```
snmpget -v3 -n "" -u myuser -a SHA -A mypassword -x AES -X mypassword -l authPriv 127.0.0.1 sysUpTime.0
```

output

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (34926) 0:05:49.26
```

Install the HAProxy Enterprise SNMP module

1. Install the HAProxy Enterprise SNMP module:

Apt:

```
sudo apt-get install hapee-extras-snmp-lb
```

Yum:

```
sudo yum install hapee-extras-snmp-lb
```

Zypper:

```
sudo zypper install hapee-extras-snmp-lb
```

Pkg:

```
sudo pkg install hapee-extras-snmp-lb
```

2. In the `global` section of your load balancer configuration, add a `stats socket` line that points to `/var/run/hapee-extras/hapee-lb.sock`. The SNMP module will connect to this socket to retrieve metrics data. Note that your configuration will likely come with a `stats socket` line that's different from this. You can add a second `stats socket` line to support this feature:

```
global
  stats socket /var/run/hapee-extras/hapee-lb.sock user hapee-lb group hapee mode 660 level user
```

   On Debian and Ubuntu, set the `group` parameter on this line to `Debian-snmp` instead of `hapee`. That is the group in which the SNMP service runs.

3. Optional: Add `id` directives to your `frontend` and `backend` sections. The `id` directive's value is an integer that indicates that section's unique identifier, which will appear in the SNMP output, making it easier to tell one `frontend` or `backend` from another. Otherwise, an ID will be set for you. The IDs for a frontend and backend do not need to relate in any way.

```
frontend www
  id 1

backend webservers
  id 10

backend dbservers
  id 20
```

4. Restart the HAProxy Enterprise service:

```
sudo systemctl restart hapee-3.1-lb
```

5. To verify that requests for load balancer metrics get passed to the module, use the `snmpwalk` command to return some data.

   - Verify on SNMPv2c using the community created earlier:

   ```
   snmpwalk -v2c -c haproxy 127.0.0.1 HAPROXYTECH-MIB::lbStats
   ```

   - Verify on SNMPv3 using the user created earlier:

   ```
   # On RedHat
   snmpwalk -v3 -u myuser -a SHA -A mypassword -x AES -X mypassword -l authPriv -M /usr/share/snmp/mibs:/opt/hapee-extras/misc -m HAPROXYTECH-MIB 127.0.0.1 HAPROXYTECH-MIB::lbStats
   ```

   output fragment

   ```
   HAPROXYTECH-MIB::lbProcessID.1 = INTEGER: 1
   HAPROXYTECH-MIB::lbProcessVersion.1 = STRING: "2.9.0-1.0.0-325.332"
   HAPROXYTECH-MIB::lbProcessReleaseDate.1 = STRING: "2024/05/03"
   HAPROXYTECH-MIB::lbProcessNbProc.1 = INTEGER: 1
   HAPROXYTECH-MIB::lbProcessProductName.1 = STRING: "hapee-lb"
   HAPROXYTECH-MIB::lbProcessSystemPID.1 = INTEGER: 7499
   HAPROXYTECH-MIB::lbProcessUptime.1 = STRING: "0d 0h06m38s"
   ```

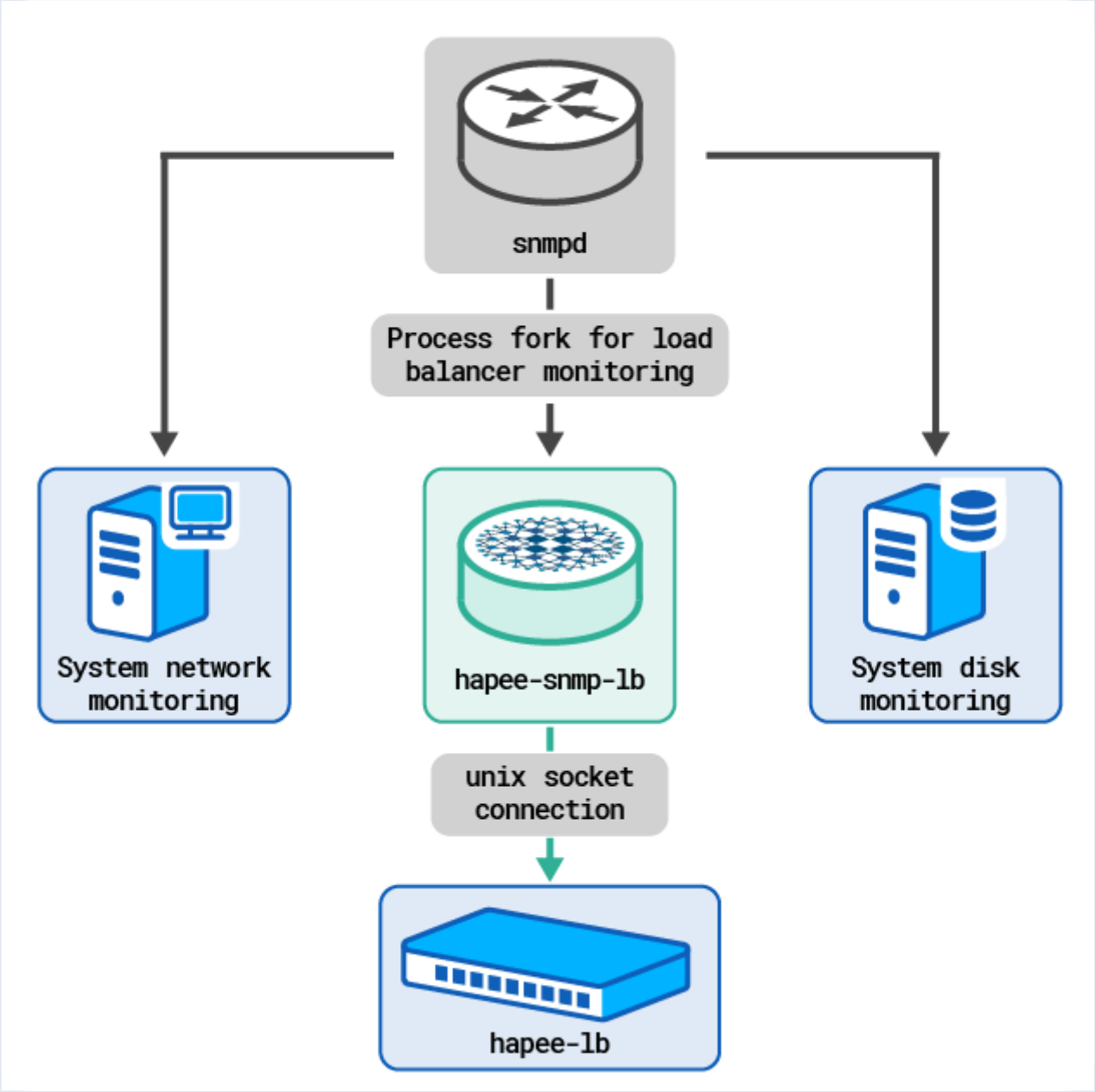
6. Configure your SNMP monitoring software to collect metrics from the HAProxy Enterprise server by using the SNMPv3 protocol with the username and passphrase you set.

How it works

The SNMP stack is split into two main components:

- The Operating system SNMP agent service, called `snmpd`. It listens on a network interface on port 161/UDP and handles SNMP requests from clients.
- The SNMP module for HAProxy Enterprise, which collects data from the load balancer for `snmpd`.

The diagram below illustrates how the SNMP agent service monitors the system's network, system disks, and HAProxy Enterprise:



Troubleshooting

You get the error 'No Such Instance currently exists at this OID'

This error indicates that although the object at the given ID (the metric) was found on this server, no value was returned for it.

- Try disabling SE Linux, which can interfere with the SNMP service.
- Check that you are using the right object ID (OID). You can either use the human readable names or the numeric names. The `snmptranslate` command shows you the OIDs that are defined in the MIB file `/opt/hapee-extras/misc/HAPEE.mib`.

  The MIBs are named `HAPROXYTECH-MIB` and `HAPROXYTECH_LEGACY-MIB`, formerly `EXCELIANCE-MIB`.

Debian:

```
snmptranslate -Pu -Tz -m HAPROXYTECH-MIB
```

Rhel:

```
snmptranslate -Pu -Tz -M /usr/share/snmp/mibs:/opt/hapee-extras/misc -m HAPROXYTECH-MIB
```

output fragment

```
"org"                   "1.3"
"dod"                   "1.3.6"
"internet"                      "1.3.6.1"
"directory"                     "1.3.6.1.1"
"mgmt"                  "1.3.6.1.2"
"mib-2"                 "1.3.6.1.2.1"
"transmission"                  "1.3.6.1.2.1.10"
"experimental"                  "1.3.6.1.3"
"private"                       "1.3.6.1.4"
"enterprises"                   "1.3.6.1.4.1"
"haproxytech"                   "1.3.6.1.4.1.58750"
```

- Check that the `pass_persist` is working. To debug the `pass_persist` line in the agent configuration file, use the following commands to run the agent in debug mode:

```
sudo systemctl stop snmpd
sudo snmpd -f -Lo -Ducd-snmp/pass
```

After invoking `snmpget` or `snmpwalk`, the debugger should show output like the following:

output fragment

```
Connection from UDP: [127.0.0.1]:56077->[127.0.0.1]:161
ucd-snmp/pass_persist: open_persist_pipe(2,'/opt/hapee-extras/bin/hapee-snmp-lb') recurse=0
ucd-snmp/pass_persist: open_persist_pipe: opened the pipes
ucd-snmp/pass_persist: persistpass-sending:
getnext
.1.3.6.1.4.1.58750.4.3.1.3
```

When finished, stop the `snmpd` command and start the `snmpd` service.

- Check that your `stats socket` line is correct in the load balancer configuration file. It should use `/var/run/hapee-extras/hapee-lb.sock` (you will probably need to add this as a new line in your configuration).

To verify that requests are passing to the stats socket, you can place a proxy in front of the socket and then monitor the traffic. In the following example, we use `socat` as a proxy. It shows that the socket is being queried and is returning data:

```
sudo systemctl restart hapee-3.1-lb
sudo apt install socat
sudo mv /var/run/hapee-extras/hapee-lb.sock /var/run/hapee-extras/hapee-lb.original
sudo socat -t100 -x -v UNIX-LISTEN:/var/run/hapee-extras/hapee-lb.sock,mode=777,reuseaddr,fork UNIX-CONNECT:/var/run/
hapee-extras/hapee-lb.original
```

In another shell, use `snmpwalk` to generate traffic.

```
snmpwalk -v2c -c haproxy 127.0.0.1 HAPROXYTECH-MIB::lbStats
```

output fragment

```
> 2024/05/21 17:21:20.326295  length=10 from=0 to=9
73 68 6f 77 20 69 6e 66 6f 0a                    show info.
--
< 2024/05/21 17:21:20.333089  length=1228 from=0 to=1227
4e 61 6d 65 3a 20 68 61 70 65 65 2d 6c 62 0a     Name: hapee-lb.
56 65 72 73 69 6f 6e 3a 20 32 2e 39 2e 30 2d 31  Version: 2.9.0-1
2e 30 2e 30 2d 33 32 35 2e 33 33 32 0a           .0.0-325.332.
52 65 6c 65 61 73 65 5f 64 61 74 65 3a 20 32 30  Release_date: 20
32 34 2f 30 35 2f 30 33 0a                       24/05/03.
4e 62 74 68 72 65 61 64 3a 20 32 0a              Nbthread: 2.
4e 62 70 72 6f 63 3a 20 31 0a                    Nbproc: 1.
50 72 6f 63 65 73 73 5f 6e 75 6d 3a 20 31 0a     Process_num: 1.
50 69 64 3a 20 38 31 38 30 0a                    Pid: 8180.
55 70 74 69 6d 65 3a 20 30 64 20 30 68 31 36 6d  Uptime: 0d 0h16m
34 37 73 0a                                      47s.
```

When finished, restart the HAProxy Enterprise service to restore the original socket.

```
sudo systemctl restart hapee-3.1-lb
```

SSL-CRL

This page applies to:

- HAProxy Enterprise 2.9r1 and newer

The SSL-CRL filter periodically checks SSL client certificates and closes any connections that rely on revoked or expired client certificates.

By default, the load balancer does not close an SSL connection between a client and backend server when an SSL certificate expires or is revoked. Consequently, the revocation of a client certificate does not prevent a client from accessing a server until the client closes their connection themselves.

A client can continue to use saved SSL session data as long as the server does not check the validity of the client certificate and the SSL session does not time out. Once the timeout period expires, new connections cannot be established, but existing connections are not closed. The timeout period is 7200 seconds or the HAProxy `tune.ssl.lifetime` configuration parameter.

To enable timely termination of connections when client certificates expire or are revoked, use the SSL-CRL module. The module implements a filter that periodically checks client certificates to see if they are valid. If a client SSL certificate expires or is revoked, the module closes the connection with the client.

For more information on certificates, see the Client certificate authentication tutorial.

Install the SSL-CRL module

1. Install the module using your package manager:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-sslcrl
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-sslcrl
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-sslcrl
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-sslcrl
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-sslcrl
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-sslcrl
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-sslcrl
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-sslcrl
```

2. In the `global` section of your configuration file, load the module:

```
global
  module-path /opt/hapee-3.1/modules
  module-load hapee-lb-sslcrl.so
```

3. In the frontend, listen, or backend sections where you want to enable the filter, add the `filter sslcrl` directive.

- The listen, frontend, or backend section must be run in TCP mode by using `mode tcp`.
- Add a `bind` directive that listens over HTTPS (port `443`).
- Verify client certificates by including `verify required` and the `ca-file` argument in the `bind` directive.
- Reference a certificate revocation list file by using the `crl-file` argument. The SSL-CRL module will watch the in-memory representation of this CRL file.

Example: Define a filter with all options set to the default values:

```
frontend fe_production
  mode tcp
  bind :443 ssl crt /certs/site.pem verify required ca-file /certs/ca.crt crl-file /certs/revoked.crl
  filter sslcrl id sslcrl-prod-1
```

Optionally, you can specify SSL-CRL configuration options in a named `sslcrl` section. Then refer to this section by name in any `filter sslcrl` directive where you wish to apply the configuration.

Example: Define a filter named `sslcrl-prod-1` with options set in a filter configuration section named `sslcrl-prod-1-config`:

```
sslcrl sslcrl-prod-1-config
  log            global
  log-format     "%ci:%cp [%t] %ft %bi:%bp"
  option         hard-errors
  check-interval 10s

frontend fe_production
  mode tcp
  bind :443 ssl crt /certs/site.pem verify required ca-file /certs/ca.crt crl-file /certs/revoked.crl
  filter sslcrl id sslcrl-prod-1 config sslcrl-prod-1-config
```

Example Configuration

hapee-lb.cfg

```
global
  # Enable the Runtime API
  stats socket /var/run/hapee-3.1/hapee-lb.sock user hapee-lb group hapee mode 660 level admin

  # Load the module
  module-path /opt/hapee-3.1/modules
  module-load sslcrl.so

sslcrl sslcrl-prod-1-config
  log            global
  log-format     "%ci:%cp [%t] %ft %bi:%bp"
  option         hard-errors
  check-interval 10s

frontend www
  mode tcp
  bind :443 ssl crt /certs/site.pem  verify required ca-file /certs/self-signed.crt crl-file /certs/revoked.crl
  filter sslcrl id sslcrl-prod-1 config sslcrl-prod-1-config
  default_backend webservers

backend webservers
  server webapp1 192.168.56.42:8080
  server webapp2 192.168.56.43:8080
```

Certificate operations

The procedures in this section are based on a simple scenario where an organization has one self-signed Certificate Authority (CA) certificate that is used both to create client certificates and to authenticate clients.

> **Info**
>
> The certificates addressed in this section are the ones your server uses to authenticate and trust clients. These are not related to the site certificate that the server provides to clients so the clients can trust your server. The site certificate, typically issued by an external third-party vendor, is specified by the `ssl crt` parameter of the `bind` directive.

Create a self-signed Certificate Authority

A self-signed Certificate Authority (CA) is the certificate that your organization uses both to create client certificates and to authenticate clients. Your organization acts as the Certificate Authority, authenticating certificates when they are presented by clients.

You create your organization's self-signed CA certificate yourself. You do not have to procure one from an external CA. In your load balancer configuration you specify the self-signed CA certificate file name using the `ca-file` parameter in the `bind` directive.

Creating a Certificate Authority involves generating a CA key pair, creating a self-signed CA certificate, and configuring the CA to sign other certificates. You can accomplish it all with a single `openssl` command.

1. Generate a CA key pair and self-signed certificate:

```
openssl req \
  -newkey rsa:2048 \
  -nodes \
  -x509 \
  -days 3650 \
  -keyout self-CA-key.pem \
  -out self-CA.crt
```

2. Learn more about this command

   The options of this command break down as follows:

   - `openssl req` creates and processes certificate signing requests (CSRs) and certificates.
   - `-newkey rsa:2048` specifies the generation of a new RSA key pair with a key size of 2048 bits.
   - `-nodes` indicates that the private key should not be encrypted with a passphrase and will be saved in an unencrypted format.
   - `-x509` tells OpenSSL to generate a self-signed certificate instead of a CSR.
   - `-days 3650` specifies the validity period of the certificate as 3650 days (approximately 10 years).
   - `-keyout` designates the file name of `self-CA-key.pem` for the private key.
   - `-out self-CA.crt` sets the certificate.

   The command also prompts you to enter information for the certificate, such as the Common Name (CN), organization details, and so on. Fill in the required fields accordingly.

3. To display the details of the CA certificate and verify its information, use the `openssl x509` command:

```
openssl x509 -in self-CA.crt -text -noout
```

After following these steps, you should have the self-signed certificate `self-CA.crt`. Remember to adjust the file names and paths according to your specific setup. Keep the CA key `self-CA-key.pem` secret.

Verify client certificates

1. In your load balancer configuration, enable verification of client certificates by setting `verify` to `required` on a `bind` line. The `ca-file` parameter specifies the CA file to use to verify:

```
frontend www
  mode tcp
  bind :443 ssl crt /certs/site.pem  verify required ca-file /certs/self-CA.crt
```

2. Reload the load balancer.

```
sudo systemctl reload hapee-3.1-lb
```

With this change, only clients with properly signed client certificates can access the server.

> **Info**
> Reloading the load balancer does not affect open connections. It only affects new connections.

Create a client certificate

A client must have a signed certificate to access your server. The client certificate is known to be authentic and reliable because you have signed it using your organization's self-signed CA certificate.

Creating a client certificate involves creating a certificate request, which can be completed by the client or by you, the administrator. Then the administrator signs the request and sends the resulting certificate to the client.

1. The client or you, the administrator, create a certificate request (CSR). Create a request for Alice:

```
openssl req \
  -newkey rsa:2048 \
  -nodes \
  -days 3650 \
  -subj "/CN=exampleUser/O=exampleOrganization" \
  -keyout alice-key.pem \
  -out alice-cert.csr
```

In addition to the certificate request, this command creates the client key, `alice-key.pem` in this example. The client needs to keep this key in a secret place.

2. You, the administrator, create an `openssl` configuration (CNF) file with these contents:

client-cert-extensions.cnf

```
basicConstraints = CA:FALSE
keyUsage = digitalSignature
extendedKeyUsage = clientAuth
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
```

3. Using your organization's self-signed CA and client request (CSR), sign the certificate:

```
openssl x509 \
  -req \
  -in alice-cert.csr \
  -out alice-cert.crt \
  -CA self-CA.crt \
  -CAkey self-CA-key.pem \
  -CAcreateserial \
  -days 7770 \
  -extfile client-cert-extensions.cnf
```

Send the certificate, `alice-cert.crt` in this example, to user Alice. Alice can now use the certificate.

If you also generated the CSR, also send the user's key, `alice-key.pem` to the user. Again, user Alice needs to keep this key in a secret place.

Certificate Revocation Lists

A Certificate Revocation List (CRL) contains a list of the certificates that have been revoked.

Create a CRL

To create a CRL:

1. Create the directory structure and necessary files. You can choose different directory paths, however ensure all created files are in the proper locations:

```
mkdir -p ./databaseCA/certs ./databaseCA/private ./databaseCA/crl
touch ./databaseCA/index.txt ./databaseCA/serial ./databaseCA/crlnumber
echo 01 > ./databaseCA/serial
echo 1000 > ./databaseCA/crlnumber
```

2. Copy your CA certificate and private key into the `databaseCA` directory.

```
cp self-CA.crt ./databaseCA
cp self-CA-key.pem ./databaseCA/private
```

3. You'll need a configuration file for OpenSSL. An example file named `openssl.cnf` can usually be found in the directories `/usr/lib/ssl/` or `/etc/ssl/`. Copy this to the current directory. Below is an example snippet from the file, modified for the certificates we created:

openssl.cnf

```
# OpenSSL example configuration file.
# See doc/man5/config.pod for more info.

##################################################################
[ ca ]
default_ca      = CA_default                    # The default ca section
##################################################################

[ CA_default ]
dir             = ./databaseCA                  # Where everything is kept
certs           = $dir/certs                    # Where the issued certs are kept
crl_dir         = $dir/crl                      # Where the issued crl are kept
database        = $dir/index.txt                # database index file.
new_certs_dir   = $dir/newcerts                 # default place for new certs.
certificate     = $dir/self-CA.crt              # The CA certificate
serial          = $dir/serial                   # The current serial number
crlnumber       = $dir/crlnumber                # the current crl number
crl             = $dir/crl.pem                  # The current CRL
private_key     = $dir/private/self-CA-key.pem  # The private key
x509_extensions = usr_cert                      # The extensions to add to the cert
crl_extensions  = crl_ext
default_days    = 3650                          # how long to certify for
default_crl_days= 30                            # how long before next CRL
default_md      = sha256                        # use SHA-256 by default
preserve        = no                            # keep passed DN ordering

[ crl_ext ]
authorityKeyIdentifier=keyid:always
```

The above is just the `CA_default` portion of a default OpenSSL configuration, not the entire `openssl.cnf` file.

In this configuration, `./databaseCA` is the directory where OpenSSL will store its database of certificates, `./databaseCA/private/self-CA-key.pem` is the CA's private key, and `./databaseCA/self-CA.crt` is the CA's certificate.

- Ensure the directory and file paths match your environment, which we created in Step 1.
- Consult the openssl ca documentation for a complete list of configuration options.

4. Generate the file `revoked.crl`, which will contain the list of revoked certificates:

```
sudo openssl ca -config openssl.cnf -gencrl -out /certs/revoked.crl
```

You can create a CRL file even before a certificate has been revoked. In this case, the revocation list will be empty inside the CRL.

Configure the load balancer

By using Certificate Revocation Lists, you can revoke client certificates in your load balancer environment. Revoked certificates will be denied access based on your configured CRL file.

Here is an example snippet in which we add the `crl-file` argument to deny access to clients who have revoked certificates:

```
frontend www
   mode tcp
   bind :443 ssl crt /certs/site.pem verify required ca-file /certs/self-CA.crt crl-file /certs/revoked.crl
   filter sslcrl id sslcrl-prod-1
```

Revoke a certificate

To revoke a client certificate:

1. Invoke the `openssl ca` command with the `-revoke` argument:

   ```
   sudo openssl ca -config openssl.cnf -revoke alice-cert.crt
   ```

   This will immediately revoke the certificate `alice-cert.crt` as confirmed by the output:

   output

   ```
   Using configuration from openssl.cnf
   Revoking Certificate 01.
   Data Base Updated
   ```

2. Update the CRL file:

   ```
   sudo openssl ca -config openssl.cnf -gencrl -out /certs/revoked.crl
   ```

3. Reload the load balancer configuration.

Update the CRL in HAProxy Enterprise

Update the in-memory representation of the CRL file in HAProxy Enterprise by using the Runtime API.

1. Replace the current CRL file with `/certs/revoked.crl`:

```
echo -e "set ssl crl-file /certs/revoked.crl <<\n$(cat /certs/revoked.crl)\n" | \
  sudo socat stdio tcp4-connect:127.0.0.1:9999
```

2. Commit the in-memory CRL change, making the change effective immediately:

```
echo -e "commit ssl crl-file /certs/revoked.crl" | \
  sudo socat stdio tcp4-connect:127.0.0.1:9999
```

Always remember to persist Runtime API changes by making analogous changes to the corresponding files on disk.

> **Info**
>
> Even after a client's certificate is revoked, web pages or other controlled resources from the server may persist in the client's cache.

Configuration directive reference

To use the SSL-CRL module, configure HAProxy Enterprise as follows.

sslcrl-add-crls

Used in the global section, the `sslcrl-add-crls` directive specifies that when new revocation lists are added to the load balancer using the Runtime API, the previous revocation lists should be retained. With this option, you can add revocation lists incrementally, in multiple Runtime API commands. By default, adding a revocation list to the load balancer deletes the previously loaded lists from memory. Optional.

```
global
  sslcrl-add-crls
```

filter directive

The `filter sslcrl` directive enables the SSL-CRL filter for the listen, frontend, or backend section. Required. The `filter sslcrl` directive is supported in the listen, frontend, and backend sections.

> **Important**
>
> The listen, frontend, or backend section must be run in TCP mode.

The directive has these parameters.

Parameter	Description
`id <name>`	This parameter specifies an ID for the filter. If you do not specify an ID, the ID is set to `<section>-<section-id>`. Use the ID when using Runtime API commands. The ID identifies the filter in log messages. Optional.
`config <section-name>`	This parameter specifies the name of the configuration section for the filter. You define a filter configuration section using the `sslcrl` directive when you need to override the default filter configuration values. The configuration section must precede the listen, frontend, or backend section that uses it. Optional.

Example: Define an SSL-CRL filter named `sslcrl-prod-1` having a configuration section named `sslcrl-prod-1-config`.

```
sslcrl sslcrl-prod-1-config
  log             global
  log-format      "%ci:%cp [%t] %ft %bi:%bp"
  option          hard-errors
  check-interval 10s

frontend fe_production
  mode tcp
  bind :443 ssl crt /certs/site.pem verify required ca-file /certs/ca.crt crl-file /certs/revoked.crl
  filter sslcrl id sslcrl-prod-1 config sslcrl-prod-1-config
```

sslcrl configuration directive

Create an `sslcrl` section to set additional options. It takes a name argument, which should match the value of the `config` argument on the `filter sslcrl` line. Multiple listen, frontend, and backend sections can refer to the same `sslcrl` configuration section.

You can specify the following directives in the `sslcrl` configuration section.

Directive	Description
`log { global | <addr> [len <len>] [format <fmt>] <facility> [<level> [<minlevel>]] }`	Configures the target log server. By default, logging from the filter is not permitted. Optional. The syntax and parameters are the same as for HAProxy Enterprise logs.
`log-format <string>`	Specifies the log format string to use for the filter's logs. For the log string, standard formats available for defining logs can be used. The default log format, if not explicitly defined, is `%ci: %cp [%t] %ft`. When logging, the prefix `[SSLCRL]: <filter-id>` and postfix `<reason>` are added to the log message. Optional.
`option disabled`	At HAProxy startup, start the filter in the disabled state. By default, the filter is on if specified in a listen, frontend, or backend section. Optional.
`option dontlog-normal`	In the case that logging in the filter is turned on, normal operation is not logged. Optional.
`option hard-errors`	In case an error occurred during the operation of the filter, the filter is temporarily (for that connection) turned off. Optional.
`check-interval <value>`	The frequency of SSL certificate validity checks. This is the maximum amount of time after which a revoked certificate causes the connection to be terminated. Default: 1 second. Optional.

Runtime API command reference

After starting the HAProxy Enterprise process, it is possible to manage the SSL-CRL filters using the Runtime API. This section describes the available commands.

> **Important**
>
> Changing filter parameters does not affect connections that are already established. Changes only apply to connections established after the change.

sslcrl add-crls

Syntax

`sslcrl add-crls [on|off]`

Description

Show or set the current `sslcrl add-crls` parameter setting, which determines whether existing CRLs are deleted when new ones are loaded. Optionally, specify `on` or `off` to set the parameter. By default, `sslcrl add-crls` is `off`, which means existing CRLs are deleted when new ones are loaded.

Example

Add new CRLs without deleting old ones:

```
echo "sslcrl add-crls on" | sudo socat stdio /var/run/hapee-lb.sock
```

sslcrl check-interval

Syntax

```
sslcrl check-interval [<value> <id>]
```

Description

Show or set the SSL certificate validation intervals for filters. The interval determines how often the validity of SSL certificates (client and server) is checked. Optionally, specify an interval and filter ID. The set value must be in milliseconds, between 1000 and 100000. Default: 1000.

Examples

Show check-interval for all SSL-CRL configurations:

```
echo "sslcrl check-interval" | sudo socat stdio /var/run/hapee-lb.sock
```

output

```
sslcrl check-interval : sslcrl-prod-1 : check-interval is currently 5s
```

Set the check-interval for SSL-CRL configuration `sslcrl-prod-1` to 10 seconds:

```
echo "sslcrl check-interval 10000 sslcrl-prod-1" | sudo socat stdio /var/run/hapee-lb.sock
```

output

```
sslcrl check-interval 10000 sslcrl-prod-1 : sslcrl-prod-1 : check-interval set to 10s
```

sslcrl disable

Syntax

```
sslcrl disable <filter-id>
```

Description

Disable the SSL-CRL filter named `<filter-id>`.

Example

Disable the SSL-CRL filter named `sslcrl-prod-1`.

```
echo "sslcrl disable sslcrl-prod-1" | sudo socat stdio /var/run/hapee-lb.sock
```

sslcrl enable

Syntax

`sslcrl enable <filter-id>`

Description

Enable the SSL-CRL filter named `<filter-id>`.

Example

Enable the SSL-CRL filter named `sslcrl-prod-1`.

```
echo "sslcrl enable sslcrl-prod-1" | sudo socat stdio /var/run/hapee-lb.sock
```

sslcrl hard-errors

Syntax

`sslcrl hard-errors <filter-id>`

Description

Enable hard-errors mode. In this mode, in the event of an error occurring in the operation of the SSL-CRL filter, the filter is turned off and does not affect the further operation of the stream.

Example

Enable hard-errors mode on the filter named `sslcrl-prod-1`.

```
echo "sslcrl hard-errors sslcrl-prod-1" | sudo socat stdio /var/run/hapee-lb.sock
```

sslcrl logging

Syntax

`sslcrl logging [<state> <filter-id>]`

Description

Show the current logging state, where state may be `off`, `on`, or `dontlog-normal` (log only errors). Optionally, set the logging state of the specified filter.

Examples

Show logging state:

```
echo "sslcrl logging" | sudo socat stdio /var/run/hapee-lb.sock
```

output

```
sslcrl logging : sslcrl-prod-1 : logging is currently enabled
```

Disable logging for SSL-CRL filter configuration `sslcrl-prod-1`:

```
echo "sslcrl logging off sslcrl-prod-1" | sudo socat stdio /var/run/hapee-lb.sock
```

output

```
sslcrl logging off sslcrl-prod-1 : sslcrl-prod-1 : logging set to disabled
```

sslcrl soft-errors

Syntax

`sslcrl soft-errors <filter-id>`

Description

Restore soft-errors mode (that is, turn off hard-errors mode) for the specified SSL-CRL filter configuration. In this mode, filter errors are logged, but the operation of the filter is not interrupted.

Example

Restore soft-errors mode on the filter named `sslcrl-prod-1`.

```
echo "sslcrl soft-errors sslcrl-prod-1" | sudo socat stdio /var/run/hapee-lb.sock
```

sslcrl status

Syntax

`sslcrl status [<filter-id>]`

Description

Show the status of all SSL-CRL filter configurations. Optionally, specify a single filter configuration.

Example

Show the status of the filter named `sslcrl-prod-1`.

```
echo "sslcrl status sslcrl-prod-1" | sudo socat stdio /var/run/hapee-lb.sock
```

See also

For more information, see the Client certificate authentication tutorial.

See also the following commands in the Runtime API:

- abort ssl cert
- add ssl crt-list
- commit ssl cert
- commit ssl crl-file
- del ssl cert
- del ssl crl-file
- del ssl crt list
- new ssl cert
- new ssl crl-file
- set ssl cert
- set ssl crl-file
- show ssl crl-file
- show ssl crt list

Traffic mirroring

This page applies to:

- HAProxy Enterprise 1.9r1 and newer

HAProxy Enterprise can mirror traffic to a different environment, even one on a different network. Traffic mirroring, also called traffic shadowing, can be useful for copying live production traffic to another environment for such purposes as:

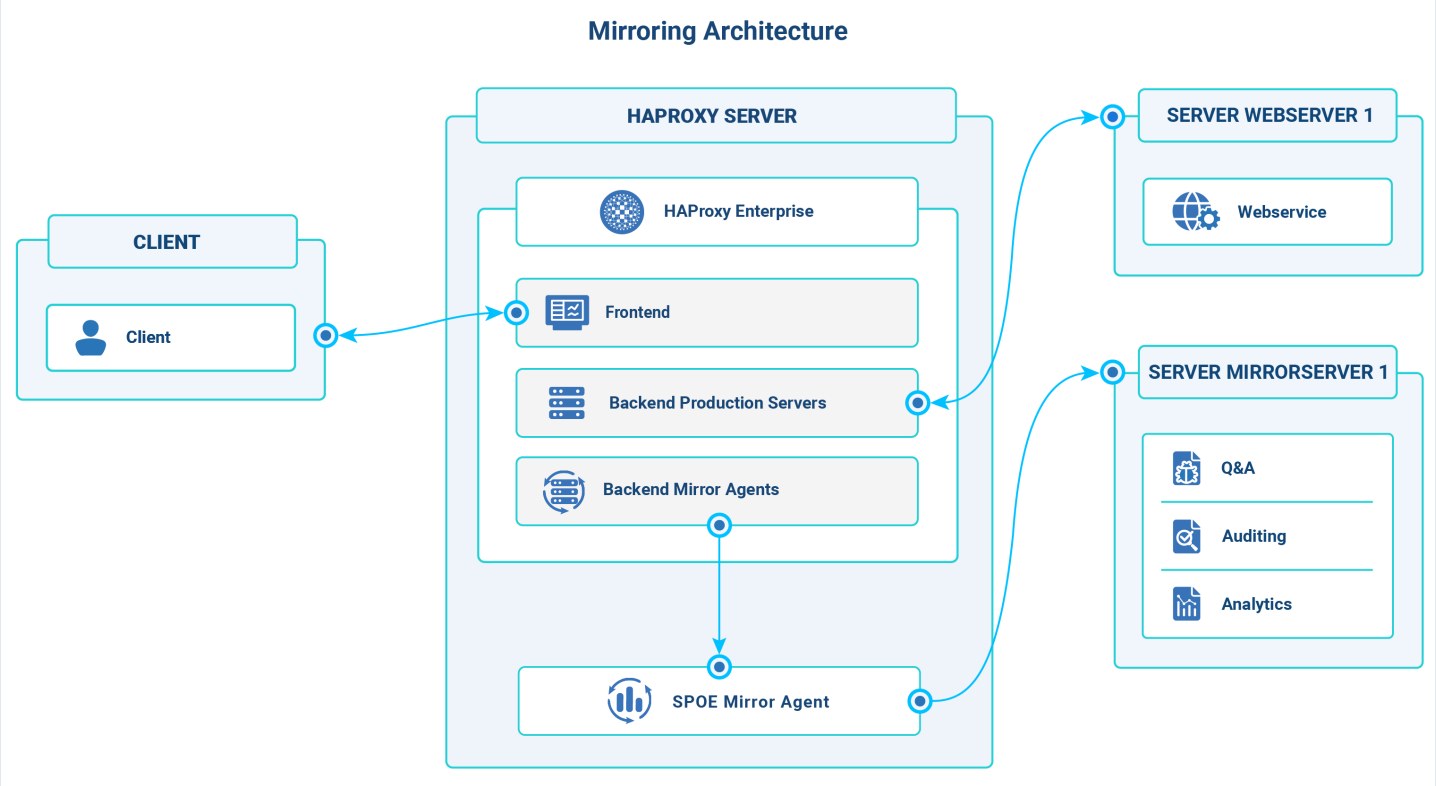
- QA
- Staging
- Auditing
- Network analytics
- Security applications (such as IDS)

Traffic mirroring has almost no impact on clients because the load balancer does not wait for a response from the mirrored environment. The mirroring process is basically “fire and forget”, where requests are copied to the mirrored environment and forgotten. Only HTTP traffic can be mirrored.

Architecture

A mirroring deployment consists of these components:

- The server where the load balancer resides.
- The Stream Processing Offload Engine (SPOE) mirroring engine, also running on the load balancer server.
- The production web servers.
- The mirror server that receives the copied traffic sent by the mirroring engine. This is the server running the mirrored services described previously, such as QA, auditing, or analytics.



The data flow occurs as follows:

1. The user client sends a request to the load balancer frontend.
2. The frontend sends the request to the regular (production) backends and mirror backends.
3. The regular backend processes the request normally, sending the request to the production web servers.
4. The mirror backend copies the request, sending it to the SPOE mirror engine.
5. The SPOE mirror engine sends the request to the mirror server in the secondary environment used for testing, auditing, or other purposes.

Configure traffic mirroring

To configure mirroring of traffic:

1. Install the required packages on the HAProxy Enterprise node:

Apt:

```
sudo apt-get install hapee-extras-spoa-mirror
```

Yum:

```
sudo yum install hapee-extras-spoa-mirror
```

Zypper:

```
sudo zypper install hapee-extras-spoa-mirror
```

Pkg:

```
sudo pkg install hapee-extras-spoa-mirror
```

2. Configure HAProxy Enterprise to send traffic to the agent. Add a `filter spoe` directive to your frontend, as shown:

```
# Production frontend
frontend fe_main
  mode http
  bind :80
  option http-buffer-request
  default_backend be_servers
  filter spoe  engine mirror  config /etc/hapee-extras/hapee-mirror-spoe.cfg
```

This directive specifies the mirror engine name and the mirror configuration file name. We cover these items in a later section.

3. In addition to the backend that specifies your production servers, add a backend that specifies the server and port of the SPOE mirror agent. Here's an example:

```
# Mirror agents
backend mirroragents
  mode tcp
  balance roundrobin
  timeout connect 5s
  timeout server 10s
  option spop-check
  server spoe-mirror-agent1 127.0.0.1:12345 check
```

This example also uses `option spop-check`, which enables health checking of SPOE agents via the SPOE protocol.

4. Edit the mirror engine configuration file, `/etc/hapee-extras/hapee-mirror-spoe.cfg`, specified by the `filter spoe` directive in the HAProxy Enterprise configuration file.

   Add the following content to the mirror engine configuration file:

   hapee-mirror-spoe.cfg

   ```
   [mirror]
   spoe-agent mirror
     log global
     messages mirror
     use-backend mirroragents
     timeout hello 500ms
     timeout idle 5s
     timeout processing 5s

   spoe-message mirror
     args arg_method=method arg_path=url arg_ver=req.ver arg_hdrs=req.hdrs_bin arg_body=req.body
     event on-frontend-http-request
   ```

   This file configures how HAProxy Enterprise communicates with the SPOE mirror agent.

5. The agent mirrors data to `http://localhost:10100/` by default. To change this URI, edit the configuration file:

   - On Debian/Ubuntu, `/etc/default/hapee-extras-spoa-mirror`
   - On Alma/Oracle/Redhat/Rocky, `/etc/sysconfig/hapee-extras-spoa-mirror`

   Modify the `-u` option in the `MIRROR_OPTIONS` environment variable to send traffic to a new URL. You can specify a domain or an IP address. The port is optional. You may define only one destination URL. Examples:

   hapee-extras-spoa-mirror

   ```
   # Set destination URL to an FQDN without port
   MIRROR_OPTIONS="-D -r0 -uhttp://mirror.mysite.com/"

   # Set destination URL to an IP address with port
   MIRROR_OPTIONS="-D -r0 -uhttp://192.168.41.10:10100"

   # Set destination URL to localhost with port
   MIRROR_OPTIONS="-D -r0 -uhttp://localhost:8100/"
   ```

6. Enable the mirror agent:

   ```
   sudo systemctl enable hapee-extras-spoa-mirror
   ```

7. Restart the mirror agent and HAProxy Enterprise:

```
sudo systemctl restart hapee-extras-spoa-mirror
sudo systemctl restart hapee-3.1-lb
```

Logging

Mirrored requests are logged to the file `/var/log/hapee-3.1/lb-access-<date>.log` by default.

An example log statement is shown below:

lb-access-20231205.log

```
Aug 25 17:48:36 node1 hapee-lb[215242]: SPOE: [mirror] <EVENT:on-frontend-http-request> sid=707 st=0 0/13/8/0/22 1/1
0/0 0/1
```

An `st` (status code) value of `0` indicates success.

Tune mirrored traffic

There are several ways to tune the traffic that is mirrored.

- Filtering
- Sampling
- Mapping key-value pairs
- Making runtime changes using the Data Plane API

Filtering

You can add an ACL that limits the requests that are captured. For instance, if you only want to mirror traffic for requests to the `/search` feature on your site, you would ignore all requests except those that have a URL path beginning with `/search`, as shown:

hapee-mirror-spoe.cfg

```
spoe-message mirror-msg
  args arg_method=method arg_path=url arg_ver=req.ver arg_hdrs=req.hdrs_bin arg_body=req.body
  event on-frontend-http-request if { path_beg /search }
```

You can also define named ACLs that do the same thing:

hapee-mirror-spoe.cfg

```
spoe-message mirror-msg
  args arg_method=method arg_path=url arg_ver=req.ver arg_hdrs=req.hdrs_bin arg_body=req.body
  acl is_search path_beg /search
  event on-frontend-http-request if is_search
```

Sampling

Suppose you don't want to capture all traffic but rather only a portion of it. You would add an ACL that collects a random sample of requests. In the next example, we generate a random number between 1 and 100 and only mirror the request if that number is less than or equal to 10:

hapee-mirror-spoe.cfg

```
spoe-message mirror-msg
  args arg_method=method arg_path=url arg_ver=req.ver arg_hdrs=req.hdrs_bin arg_body=req.body
  acl is_search path_beg /search
  event on-frontend-http-request if { rand(100) le 10 }
```

Mapping key-value pairs

Your ACL statements can also check values from map files. For example, you can switch mirroring on or off by using a map file that contains a key-value pair like `mirroring on`. Then, check the map file from your `hapee-mirror-spoe.cfg` file like this:

hapee-mirror-spoe.cfg

```
spoe-message mirror-msg
  args arg_method=method arg_path=url arg_ver=req.ver arg_hdrs=req.hdrs_bin arg_body=req.body
  acl mirroring_on str(mirroring),map(/etc/hapee-3.1/mirroring.map) -m str on
  event on-frontend-http-request if mirroring_on
```

Use the HAProxy Enterprise Runtime API to change the value in the map file to `off`.

```
# Change mirroring to off
echo "set map /etc/hapee-3.1/mirroring.map mirroring off" | nc 127.0.0.1 9999

# Show current value
echo "show map /etc/hapee-3.1/mirroring.map mirroring" | nc 127.0.0.1 9999
```

Make changes using the Data Plane API

You can also use the Data Plane API to add or remove `filter spoe` lines from the HAProxy Enterprise configuration file dynamically. In the following example, we show the existing filters, then add a new one, and then remove it:

Show existing filters:

```
curl -X GET \
  --user admin:mypassword \
  "http://localhost:5555/v1/services/haproxy/configuration/filters?parent_name=fe_main&parent_type=frontend"
```

Add a filter line:

```
curl -X POST \
  --user admin:mypassword \
  -H "Content-Type: application/json" \
  -d '{"id": 0, "spoe_config":"/etc/hapee-3.1/spoa.conf", "spoe_engine":"mirror", "type":"spoe"}' \
  "http://localhost:5555/v1/services/haproxy/configuration/filters?parent_name=fe_main&parent_type=frontend&version=1"
```

output

```
{"id":0,"spoe_config":"/etc/hapee-3.1/spoa.conf","spoe_engine":"mirror","type":"spoe"}
```

Remove a filter line:

```
curl -X DELETE \
  --user admin:mypassword \
  -H "Content-Type: application/json" \
  "http://localhost:5555/v1/services/haproxy/configuration/filters/0?parent_name=fe_main&parent_type=frontend&version=2"
```

Tip

Here are a few ways to get the most out of traffic mirroring.

- After setting up monitoring, compare the errors you get from your production servers with those you get from the new version to which you're mirroring traffic. Having a monitoring strategy in place will be key to validating a release.
- Make sure the feature you're testing has URL paths and parameters that match the existing feature so that it is forward compatible with mirrored traffic. Forward compatibility may be a valuable test in and of itself.

Troubleshooting

Traffic mirroring in HAProxy Enterprise allows you to replicate network traffic to a separate destination for monitoring and analysis purposes. Here are some suggestions to help you diagnose and resolve common problems with traffic mirroring functionality.

Confirm the HAProxy Enterprise version

Make sure you are using a version of HAProxy Enterprise that supports traffic mirroring. The mirroring feature was introduced in version 1.9.0, so if you're using an older version, consider upgrading to a compatible release.

```
/opt/hapee-3.1/sbin/hapee-lb -v
```

output

```
HAProxy version 3.1.0-1.0.0-310.374 2023/11/20 - https://haproxy.org/
Status: long-term supported branch - will stop receiving fixes around Q2 2028.
Known bugs: https://www.haproxy.com/documentation/hapee/2-9r1/onepage/changelog/#3.1.0
Running on: Linux 5.10.0-22-amd64 #1 SMP Debian 5.10.178-3 (2023-04-22) x86_64
```

Check network connectivity

Ensure that the destination IP address and port specified for mirroring are reachable from the HAProxy Enterprise server. Verify the network connectivity between the two systems using tools like `ping` or `telnet`. If there are any firewalls, security groups, or access control lists in place, ensure they allow traffic between HAProxy Enterprise and the mirroring destination.

Monitor resource utilization

Traffic mirroring can be resource-intensive, especially if the mirrored traffic volume is significant. Monitor the resource utilization of the HAProxy Enterprise, including CPU, memory, and network usage. Ensure that the server has enough capacity to handle the additional load caused by mirroring.

Check SPOE mirror agent activity status

Check if the agent is running on the system:

```
ps aux | grep spoa | grep -v grep
```

output

```
hapee-m+   63734  0.0  0.2 756464  5900 ? Sl 12:24 0:00 /opt/hapee-extras/bin/hapee-spoa-mirror
 -D -r0 -uhttp://localhost:10100/ --logfile=a:/var/log/hapee-mirror.log -F /var/run/hapee-extras/hapee-spoa-mirror.pid
```

You can also check via the system status command:

```
systemctl status hapee-extras-spoa-mirror
```

output

```
● hapee-extras-spoa-mirror.service - LSB: HAPEE HTTP requests replicator
  Loaded: loaded (/etc/init.d/hapee-extras-spoa-mirror; generated)
  Active: active (running) since Thu 2023-06-01 19:48:27 UTC; 23h ago
    Docs: man:systemd-sysv-generator(8)
Main PID: 32783 (hapee-spoa-mirr)
    Tasks: 11 (limit: 9165)
  Memory: 2.9M
```

If the process is in an error state, you can run `journalctl -xe| grep -A3 -B3 -i spoa` for more details.

Check the logs

1. Search the `/var/log/hapee-3.1/` logs for any issues. For example, `st=0` means a successful response.

   lb-access-20231205.log

   ```
   May 31 16:59:51 ip-172-31-13-131 hapee-lb[713]:: SPOE: [mirror] <EVENT:on-frontend-http-request> sid=52 st=0
   0/0/0/0/0 1/1 0/0 1/25
   ```

2. Enable logging for the mirroring agent:

   ```
   sudo touch /var/log/hapee-mirror.log
   sudo chown hapee-mirror:hapee /var/log/hapee-mirror.log
   ```

3. Use your editor to add `--logfile=a:` to the startup script:

   - On Debian/Ubuntu, `/etc/default/hapee-extras-spoa-mirror`
   - On Alma/Oracle/Redhat/Rocky, `/etc/sysconfig/hapee-extras-spoa-mirror`

This option determines the mode of logging, which allows opening and writing at end-of-file. If a capital letter is used for the mode, then line buffering is used when writing to the log file.

hapee-extras-spoa-mirror

```
MIRROR_OPTIONS="-D -r0 -uhttp://10.0.1.4:8080/ --logfile=A:/var/log/hapee-mirror.log"
```

The mirror agent will fail to start if mode is used without a defined log file.

Traffic mirroring reference

The SPOE mirror engine uses Stream Processing Offload Protocol (SPOP). The file `/etc/hapee-extras/hapee-mirror-spoe.cfg` configures how HAProxy Enterprise communicates with the SPOE mirror agent.

hapee-mirror-spoe.cfg

```
[mirror]
spoe-agent mirror
    log global
    messages mirror
    use-backend mirroragents
    timeout hello 500ms
    timeout idle 5s
    timeout processing 5s

spoe-message mirror
    args arg_method=method arg_path=url arg_ver=req.ver arg_hdrs=req.hdrs_bin arg_body=req.body
    event on-frontend-http-request
```

It supports the following directives in the `spoe-agent` section:

Directive	Description
`[*name*]`	The file begins with an engine name, `mirror`, in square brackets. As mentioned, this name must match the `engine` parameter value set on the `filter spoe` directive in the HAProxy Enterprise configuration.
`log global`	This line means that events, such as when HAProxy Enterprise sends data, will be logged to the same output defined by the `log` statement in the `global` section of the HAProxy Enterprise configuration.
`messages`	This line is a space-delimited list of labels that match up with `spoe-message` sections.
`use-backend`	This line specifies which backend in the HAProxy Enterprise configuration holds the mirror agents.
`timeout hello`	This setting limits how long HAProxy Enterprise will wait for an agent to acknowledge a connection.
`timeout idle`	This setting limits how long HAProxy Enterprise will wait for an agent to close an idle connection.
`timeout processing`	This setting limits how long an agent is allowed to process an event.

A `spoe-message` section defines which HAProxy Enterprise fetch methods will be used to capture data to send to the agents. The label here, `mirror`, is expected by this particular agent. For traffic mirroring, we capture the following:

- the HTTP method
- the URL path
- the version of HTTP
- all HTTP headers
- the request body (note that this requires `option http-buffer-request` in the HAProxy Enterprise configuration)

Data is sent every time the `on-frontend-http-request` event fires, which is before the evaluation of `http-request` rules on the frontend side.

The options supported by `hapee-spoa-mirror` can be found using `-h` or `--help`:

```
/opt/hapee-extras/bin/hapee-spoa-mirror -h
```

output

```
Usage: hapee-spoa-mirror { -h --help }
  hapee-spoa-mirror { -V --version }
  hapee-spoa-mirror { -r --runtime=TIME } [OPTION]...

  Options are:
  -a, --address=NAME              Specify the address to listen on (default: "0.0.0.0").
  -B, --libev-backend=TYPE        Specify the libev backend type (default: AUTO).
  -b, --connection-backlog=VALUE  Specify the connection backlog size (default: 10).
  -c, --capability=NAME           Enable the support of the specified capability.
  -D, --daemonize                 Run this program as a daemon.
  -F, --pidfile=FILE              Specifies a file to write the process-id to.
  -h, --help                      Show this text.
  -i, --monitor-interval=TIME     Set the monitor interval (default: 5.00s).
  -l, --logfile=[MODE:]FILE       Log all messages to logfile (default: stdout/stderr).
  -m, --max-frame-size=VALUE      Specify the maximum frame size (default: 16384 bytes).
  -n, --num-workers=VALUE         Specify the number of workers (default: 10).
  -p, --port=VALUE                Specify the port to listen on (default: 12345).
  -r, --runtime=TIME              Run this program for the specified time (0 = unlimited).
  -t, --processing-delay=TIME     Set a delay to process a message (default: 0).
  -u, --mirror-url=URL            Specify the URL for the HTTP mirroring.
  -I, --mirror-interface=NAME     Specify the interface/address for outgoing connections.
  -P, --mirror-local-port=VALUE   Specify the local port range for outgoing connections.
  -N, --noproxy=LIST              Specify a list of hosts that do not use an HTTP proxy.
  -x, --proxy-url=URL             Specify an HTTP proxy URL for reaching the mirror server. HTTPS is not supported.
Using an HTTP proxy can add latency.
  -U, --proxy-userpwd=USER:PWD    Specify the HTTP proxy's Basic authentication user and password.
  -H, --proxy-header=HDR          Pass custom header(s) to the HTTP proxy.
  -V, --version                   Show program version.
```

Supported libev backends: `select, poll, epoll, linuxaio, iouring`.

Supported capabilities: fragmentation, pipelining, async.

Allowed logging file opening modes: `a, w`. The `a` mode allows opening or creating file for writing at end-of-file. The `w` mode allows truncating the file to zero length or creating a new file. If a capital letter is used for the mode, then line buffering is used when writing to the log file.

The time delay/interval is specified in milliseconds by default, but can be in any other unit if the number is suffixed by a unit (`us, ms, s, m, h, d`).

Traffic mirroring log reference

SPOE mirror agent activity is logged using HAProxy Enterprise's logger. Mirrored requests are logged to the file `/var/log/hapee-3.1/lb-access-<date>.log` by default. A message is emitted for each mirrored request. Depending on the status code, the log level will be different. In the normal case, when no error occurred, the message is logged with the level `LOG_NOTICE`. If an error occurred, the message is logged with the level `LOG_WARNING`.

Consider the following example log message for a mirrored request:

An example log statement is shown below:

lb-access-20231205.log

```
Aug 25 17:48:36 node1 hapee-lb[215242]: SPOE: [mirror] <EVENT:on-frontend-http-request> sid=707 st=0 0/13/8/0/22 1/1 0/0 0/1
```

In this example, the `mirror` agent logged an event named `on-frontend-http-request` with a stream-id of `707`. Its status code of `0` indicates it was successful. One event was processed and had zero errors.

Mirror agent log messages follow this format:

lb-access-20231205.log

```
SPOE: [AGENT] <TYPE:NAME> sid=STREAM-ID st=STATUS-CODE reqT/qT/wT/resT/pT <idles>/<applets> <nb_sending>/<nb_waiting> <nb_error>/<nb_processed>
```

Log message item	Description
`AGENT`	The agent name. It is `mirror` for the mirror agent.
`TYPE`	For mirrored requests this is `EVENT`.
`NAME`	The event name.
`STREAM-ID`	The unique integer id of the stream.
`STATUS_CODE`	The request's status code. A status code of `0` indicates success. Other status codes include: `1`: I/O error; `2`: A timeout occurred; `3`: Frame is too big; `4`: Invalid frame received; `5`: Version value not found; `6`: `max-frame-size` value not found; `7`: Capabilities value not found; `8`: Unsupported version; `9`: `max-frame-size` too big or too small; `10`: Payload fragmentation is not supported; `11`: Invalid interlaced frames; `12`: `frame-id` not found (it does not match any referenced frame); `13`: Resource allocation error; `99`: An unknown error occurred
`reqT/qT/wT/resT/pT`	These represent the following time events: `reqT`: The encoding time. It includes ACLs processing time, if applicable. For fragmented frames, it is the sum of all fragments. `qT`: The delay before the request leaves the sending queue. For fragmented frames, it is the sum of all fragments. `wT`: The delay before the response is received. Fragmentation is not supported. `resT`: The delay in processing the response. Fragmentation is not supported. `pT`: The delay in processing the event; it is the latency added by the SPOE processing. It is more or less the sum of the other values. For all of these time events, `-1` means the process was interrupted. For example, `-1` for the queue time means the request never left the queue. Note that for fragmented frames it is harder to know when an interruption occurred.
`<idle>`	The number of idle SPOE applets.
`<applets>`	The number of SPOE applets.
`<nb_sending>`	The number of streams waiting to send data.
`<nb_waiting>`	The number of streams waiting for an `ack`.
`<nb_error>`	The number of processing errors.
`<nb_processed>`	The number of events processed.

Load balance UDP with HAProxy Enterprise

UDP load balancing on HAProxy Enterprise

This page applies to:

- HAProxy Enterprise 2.8r1 and newer

You can load balance UDP (User Datagram Protocol) by using the HAProxy Enterprise UDP module.

What is UDP?

Applications employ UDP when they need a fire-and-forget transport of messages over a network, with minimal overhead. Because UDP lacks the message delivery guarantees that TCP has, it can be ideal for some situations but not others. For example, when sending syslog messages over a network, you might prefer TCP if you require delivery of all messages. Or you might choose UDP if it's acceptable to lose some messages, but allow the sender to continue sending messages without waiting for confirmation they've been received. Under some network conditions, such as where there's significant packet loss, UDP can have better throughput. But it can perform worse in congested networks, since it lacks TCP's congestion control.

How does the UDP module work?

The UDP module adds support for a new section in your load balancer configuration called `udp-lb`. The `udp-lb` section defines the IP address and UDP port at which to accept datagrams. Also in this section, you'll define the backend servers to load balance datagrams across. The module supports health checking servers and logging traffic too.

Install the HAProxy Enterprise UDP module

This page applies to:

- HAProxy Enterprise 2.8r1 and newer

HTTP/3 over QUIC

For load balancing HTTP/3 over QUIC, instead see HTTP/3.

The HAProxy Enterprise UDP module enables you to load balance application-layer protocols that are transported over UDP such as syslog, DNS, NTP, and RADIUS.

When configuring the UDP module, you will define a `udp-lb` section that declares the address and port on which to listen and also the servers to relay UDP traffic to.

Install the UDP module

To enable UDP load balancing, first install the module:

1. Install the UDP module according to your platform:

Apt:

```
sudo apt-get install hapee-<VERSION>-lb-udp
```

Example for HAProxy Enterprise 3.1r1:

```
sudo apt-get install hapee-3.1r1-lb-udp
```

Yum:

```
sudo yum install hapee-<VERSION>-lb-udp
```

Example for HAProxy Enterprise 3.1r1:

```
sudo yum install hapee-3.1r1-lb-udp
```

Zypper:

```
sudo zypper install hapee-<VERSION>-lb-udp
```

Example for HAProxy Enterprise 3.1r1:

```
sudo zypper install hapee-3.1r1-lb-udp
```

Pkg:

```
sudo pkg install hapee-<VERSION>-lb-udp
```

Example for HAProxy Enterprise 3.1r1:

```
sudo pkg install hapee-3.1r1-lb-udp
```

2. In the `global` section of your configuration file, add a `module-load` directive to load the UDP module:

```
global
  module-path /opt/hapee-3.1/modules
  module-load hapee-lb-udp.so
```

3. Add one or more `udp-lb` sections to configure listening UDP ports and backend servers. See Examples.

Enable logging

This section applies to:

- HAProxy Enterprise 3.0r1 and newer

You can enable logging of UDP traffic through the load balancer. HAProxy Enterprise will log a message each time it receives a request datagram and forwards it to the backend server, or when the response datagram is sent back to the client. The log output format contains the source and destination addresses, bytes received or sent, the instance name, and the server if available.

1. Add the `log global` directive to your `udp-lb` section to send log messages to the syslog server declared in the `global` section:

```
global
  maxconn           10000
  log               127.0.0.1 local0
  log               127.0.0.1 local1 notice
  ...

udp-lb dns
  dgram-bind 192.168.56.25:53

  log global

  proxy-requests 1
  balance roundrobin
  option udp-check
  server dns1 10.10.10.10:53 check
  server dns2 10.10.10.20:53 check
```

   Example log messages:

log

```
Nov  7 20:40:54 hapee hapee-lb[1254]: UDP: request received (58 bytes) from 192.168.56.1:50806 to 192.168.56.25:53 (dns
Nov  7 20:40:54 hapee hapee-lb[1254]: UDP: response sent (181 bytes) from 192.168.56.25:53 to 192.168.56.1:50806 (dns/d
```

   Alternatively, you can define a `log` directive directly in the `udp-lb` section to set target syslog servers, facility code, and severity level there. For details, see log reference.

2. Optional: Set `log-tag` to indicate in the logs which load balancer server proxied the traffic. On the rsyslog side, this sets the `$programname` variable. It defaults to `hapee-lb`.

```
udp-lb dns
  dgram-bind 192.168.56.25:53

  log global
  log-tag hapee-lb-server1

  proxy-requests 1
  balance roundrobin
  option udp-check
  server dns1 10.10.10.10:53 check
  server dns2 10.10.10.20:53 check
```

Example log messages:

log

```
Nov  7 21:20:45 hapee hapee-lb-server1[1433]: UDP: request received (58 bytes) from 192.168.56.1:50768 to 192.168.56.25
Nov  7 21:20:45 hapee hapee-lb-server1[1433]: UDP: response sent (181 bytes) from 192.168.56.25:53 to 192.168.56.1:5076
```

Examples

In this section, you'll see examples of using the UDP module to load balance different types of applications. This will give you an understanding of the syntax, in case you want to load balance an application not shown here.

Load balance syslog

You can use the UDP module to load balance syslog traffic. The UDP module listens on the configured port and will load balance incoming messages to the list of configured servers. Consider the example configuration below:

```
udp-lb syslog-example
  dgram-bind 192.168.56.25:3516

  proxy-requests 1
  proxy-responses 0

  balance roundrobin

  option udp-check

  server srv1 10.10.10.10:1514 check
  server srv2 10.10.10.20:1514 check
  server srv3 10.10.10.30:1514 check
```

- We declare a UDP section using the `udp-lb` directive and name it `syslog-example`.
- We specify a `dgram-bind` on localhost port `3516`. This is where we expect to receive the UDP syslog traffic.

> **Listen port**
>
> Use caution when specifying a port for listening for syslog messages. The default rsyslog configuration for HAProxy Enterprise listens for traffic on localhost port `514`. If you try to specify the same interface and port, the load balancer will be unable to bind on that interface and will receive no messages.

- We set `proxy-requests` to `1`. This specifies that the load balancer should load balance on each datagram it receives, since each syslog message will fit into a single datagram.
- We set `proxy-responses` to `0`. This specifies that the load balancer shouldn't expect a response from the server.
- We set the load balancing algorithm to `roundrobin`.
- We enable health checks over ICMP with `option udp-check`. Be sure to enable ICMP traffic in your network to allow this behavior.
  - Note that you could also enable health checks over TCP using `option tcp-check`.
- We list three servers that will receive the load balanced syslog traffic. These servers have been configured via rsyslog to expect UDP log traffic on port `1514`.

Note that for the best performance for load balancing syslog, it's recommended that `proxy-requests` is set to `1` and `proxy-responses` is set to `0`.

Tip

For best performance, add the `shards <number> by-thread` option to your `dgram-bind` line. This will distribute incoming traffic over multiple sockets by creating this `<number>` of listeners and giving each listener its own thread. The example below is for a CPU with 48 cores so that it will use 48 threads, and the traffic will be distributed evenly among the threads since we have specified the `by-thread` option:

```
udp-lb myudp1
  dgram-bind 192.168.56.25:3516 shards 48 by-thread
```

Use caution with this option because, while it improves performance, it also increases CPU usage. For more information, see shards reference.

Load balance DNS

You can use the UDP module to load balance DNS traffic over UDP. However, in cases where the DNS response may be larger than one datagram, it's better to load balance DNS over TCP because TCP supports larger responses. This scenario may occur with DNS-based service discovery. In most cases, a DNS request fits within one datagram, and UDP is sufficient.

Consider the example configuration below:

```
udp-lb udp-dns
  dgram-bind 192.168.56.25:53

  proxy-requests 1

  balance roundrobin
  option udp-check

  server dns1 10.10.10.10:53 check
  server dns2 10.10.10.20:53 check

frontend tcp-dns
  mode tcp
  bind 192.168.56.25:53
  default_backend tcp-dns-backend

backend tcp-dns-backend
  mode tcp
  balance roundrobin
  server srv1 10.10.10.10:53 check
  server srv2 10.10.10.20:53 check
  server srv3 10.10.10.30:53 check
```

- UDP
  - We declare a UDP section using the `udp-lb` directive and name it `udp-dns`.
  - We specify a `dgram-bind` on `192.168.56.25:53`.
  - We set `proxy-requests` to `1`. This specifies that the load balancer should load balance on each datagram it receives, since each DNS request will fit into a single datagram.

    proxy-responses

    Unlike our other examples which explicitly set a value for `proxy-responses`, in the case for DNS, we leave this option unset. By leaving it unset, this specifies that the load balancer should expect an unlimited number of responses from the DNS server. It will forward all responses back to the client.

  - We set the load balancing algorithm to `roundrobin`.
  - We enable health checks over ICMP with `option udp-check`. Be sure to enable ICMP traffic in your network to allow this behavior.
    - Note that you could also enable health checks over TCP using `option tcp-check`.
  - We list two servers that will receive and provide responses for the load-balanced DNS requests.
- TCP
  - We define a frontend named `tcp-dns` and a backend named `tcp-dns-backend`. This frontend and backend will load balance DNS traffic over TCP.
  - We enable TCP healthchecks using `check`.
  - We list three servers that will receive and provide responses for the load-balanced DNS requests.

For more information, see DNS service discovery.

- Alternative configuration

  If the port on the `dgram-bind` line in the `udp-lb` section is the same as the port you specified on the `server` lines, you can omit the port from the server lines. Consider the previous example where the load balancer will listen via `dgram-bind` on 192.168.56.25 on port 53 and then forward requests to servers, also on port 53. You can configure your `udp-lb` section as follows instead, leaving off the ports on the server lines:

```
udp-lb udp-dns
  dgram-bind 192.168.56.25:53

  proxy-requests 1

  balance roundrobin
  option udp-check

  server dns1 10.10.10.10 check
  server dns2 10.10.10.20 check
```

Load balance NTP

You can use the UDP module to load balance NTP traffic. The UDP module listens on the configured port and will load balance incoming NTP requests to the list of configured NTP servers. It will then return the response to the appropriate client.

Consider the example configuration below:

```
udp-lb ntp
  dgram-bind 192.168.56.25:123

  proxy-requests 1
  proxy-responses 1

  balance roundrobin

  option udp-check

  server srv1 10.10.10.10:123 check
  server srv2 10.10.10.20:123 check
  server srv2 10.10.10.30:123 check
```

- We declare a UDP section using the `udp-lb` directive and name it `ntp`.
- We specify a `dgram-bind` on all interfaces on port `123`. This is the standard NTP port where we expect to receive NTP requests.

> **NTP servers**
>
> UDP Port `123` is the standard port for NTP and most implementations don't allow you to change it. As such, your load balancer and NTP server(s) should not be the same server. The load balancer must bind on port `123` to load balance the NTP requests, which it would be unable to do if the server it runs on is also running as an NTP server (and therefore is already using UDP port `123`).

- We set `proxy-requests` to `1`. This specifies that the load balancer should load balance on each datagram it receives, since each NTP request will fit into a single datagram.
- We set `proxy-responses` to `1`. This specifies that the load balancer should expect one response from the NTP server. It will then relay the response back to the client.
- We set the load balancing algorithm to `roundrobin`.
- We enable health checks over ICMP with `option udp-check`. Be sure to enable ICMP traffic in your network to allow this behavior.
  - Note that you could also enable health checks over TCP using `option tcp-check`.
- We list three servers that will receive the load balanced NTP traffic. These servers have been configured as NTP servers and will respond to requests on the standard UDP NTP port `123`.

- Alternative configuration

If the port on the `dgram-bind` line is the same as the port you specified on the `server` lines, you can omit the port from the server lines. Consider the previous example where the load balancer will listen via `dgram-bind` on 192.168.56.25 on port 123 and then forward requests to servers, also on port 123. You can configure your `udp-lb` section as follows instead, leaving off the ports on the server lines:

```
udp-lb ntp
  dgram-bind 192.168.56.25:123

  proxy-requests 1
  proxy-responses 1

  balance roundrobin

  option udp-check

  server srv1 10.10.10.10 check
  server srv2 10.10.10.20 check
  server srv2 10.10.10.30 check
```

Load balance RADIUS

You can use the UDP module to load balance RADIUS authentication traffic. The UDP module listens on the configured ports and will load balance incoming requests to the list of configured RADIUS servers. It will then return the responses to the appropriate client.

Consider the example configuration below where the load balancer is configured to route traffic to both RADIUS authentication (1812) and accounting (1813) ports:

```
udp-lb radius-auth
  dgram-bind 192.168.56.25:1812
  balance source

  server srv1 10.10.10.10:1812
  server srv2 10.10.10.20:1812
  server srv3 10.10.10.30:1812

udp-lb radius-accounting
  dgram-bind 192.168.56.25:1813
  balance source

  server srv1 10.10.10.10:1813
  server srv2 10.10.10.20:1813
  server srv3 10.10.10.30:1813
```

- We declare two UDP sections using the `udp-lb` directive and name them `radius-auth` and `radius-accounting`.
- We specify a `dgram-bind` on all interfaces on port `1812` for `radius-auth` and port `1813` for `radius-accounting`.
  - Ensure that the ports you specify match the ports defined in your RADIUS configuration (1812 and 1813 are the RADIUS defaults).
- We set the load balancing algorithm to `source`. This is required so that requests from the same client are routed to the same server.
- We don't set `proxy-requests`. There will be multiple requests from the client, and we want all requests from the same client to be routed to the same server. This applies regardless of any timeout value specified since we have also set `balance` to `source`.
- We don't set `proxy-responses`. There will be multiple responses from the RADIUS server.
- We list three servers that will receive the load balanced RADIUS traffic. These servers have been configured as RADIUS servers and will respond to requests on the default RADIUS ports 1812 and 1813.

- Alternative configuration

  If the ports on the `dgram-bind` lines in the `udp-lb` sections are the same as the ports you specified on the `server` lines, you can omit the ports from the server lines. Consider the previous example where the load balancer will listen via `dgram-bind` on 192.168.56.25 on ports 1812 and ports 1813 and then forward requests to servers, either on port 1812 or 1813, depending on what port the load balancer received the request. You can configure your `udp-lb` section as follows instead, leaving off the ports on the server lines and combining the two `udp-lb` sections:

```
udp-lb radius
  dgram-bind 192.168.56.25:1812
  dgram-bind 192.168.56.25:1813
  balance source

  server srv1 10.10.10.10
  server srv2 10.10.10.20
  server srv3 10.10.10.30
```

UDP load balancing reference

The UDP module uses the following directives for configuration:

The UDP module uses the following directives for configuration:

accepted-payload-size

Sets the maximum UDP datagram payload size (in bytes). The default is `1472`. The maximum allowed is `65507`.

Syntax:

```
accepted-payload-size <number>
```

balance

Sets the load balancing algorithm.

Syntax:

```
balance <algorithm>`
```

The UDP module supports the following values for `balance`:

- `static-rr`
- `roundrobin`
- `leastconn`
- `first`
- `source`
- `random`

default-server

Sets default parameters that will apply to all `server` lines within the same section. For a list of supported parameters, see default-server options.

Syntax:

```
default-server [param*]
```

dgram-bind

Configures a datagram listener to receive messages to forward. Addresses must be in IPv4 or IPv6 form, optionally followed by a port.

Syntax:

```
dgram-bind <addr> [param*]
```

The `dgram-bind` directive supports these bind parameters:

- `maxconn`
- `namespace`
- `nice`
- `shards`
- `thread`
- `transparent`

hash-balance-factor

> **This section applies to:**
>
> - HAProxy ALOHA 17.0 and newer
> - HAProxy Enterprise 3.1r1 and newer

Specifies the balancing factor for bounded-load consistent hashing. Please refer to hash-balance-factor for more details.

Syntax:

```
hash-balance-factor <factor>
```

hash-type

This section applies to:

- HAProxy ALOHA 17.0 and newer
- HAProxy Enterprise 3.1r1 and newer

Specifies a method to use for mapping hashes to servers. Please refer to the hash-type for more details.

Syntax:

```
hash-type <method> <function> <modifier>
```

log

This section applies to:

- HAProxy ALOHA 17.0 and newer
- HAProxy Enterprise 3.0r1 and newer

Enables per-instance logging of events. For requests, the source is the IP/port of the client, and the destination is the IP/port of the listener. For responses, the source is the listener, and the destination is the client.

For details, see log reference.

Syntax:

```
log <target> [len <length>] [format <format>] [sample <ranges>:<sample_size>] <facility> [<level> [<minlevel>]]
```

log global

> **This section applies to:**
>
> - HAProxy ALOHA 17.0 and newer
> - HAProxy Enterprise 3.0r1 and newer

Sets the instance's logging parameters to be the same as the global ones.

Syntax:

```
log global
```

log-tag

> **This section applies to:**
>
> - HAProxy ALOHA 17.0 and newer
> - HAProxy Enterprise 3.0r1 and newer

Sets the log tag string to use for all outgoing logs.

Syntax:

```
log-tag <string>
```

maxconn

Sets the maximum number of concurrent connections. Once the limit is reached, all datagrams received initiating new UDP connection will be dropped.

Syntax:

```
maxconn <maxconn>
```

option tcp-check

Performs health checks using TCP connection attempts.

Syntax:

```
option tcp-check
```

option udp-check

Performs health checks via ICMP.

Syntax:

```
option udp-check
```

proxy-requests

Sets the number of expected datagrams per client session. Since UDP is not a connection-oriented protocol, the UDP module must keep track of a client's session such that it can route the response datagrams from an upstream server back to the correct client. Each session is indexed by the 4-tuple consisting of source IP/port and destination IP/port corresponding to the datagram.

- If this option is not set, then the load balancer will forward all datagrams from the client to the same backend server as long as the client is considered alive. If the client becomes inactive, their session expires and the next time they send a datagram, the load balancer will again choose a server based on the load balancing algorithm.
- If this option is set to a value greater than 0, then session stickiness is disabled and the load balancer will choose the backend server on every `<number>` datagrams received. For example, if `proxy-requests 1` then a destination server will be rotated after each datagram received from the client.

Syntax:

```
proxy-requests <number>
```

proxy-responses

Sets the number of expected responses from the server. Sessions last until the timeout is reached or the expected number of responses has been received. If zero value is specified, all responses from the server will be ignored and not forwarded back to the client. If a value is not specified, the number of expected responses is set to unlimited.

Syntax:

```
proxy-responses <number>
```

server

Configures a target server.

Syntax:

```
server <name> <address>[:[port]] [param*]
```

source

Sets the source address for outgoing connections. The `<addr>` and optional `<port>` will be used for binding before connecting to the server. The `<addr2>` and `<port2>` are presented to the server when connections are forwarded in full transparent proxy mode. If `client` or `clientip` is set, the load balancer will present the client's IP address and port, or the client's IP address only.

Syntax for setting source address:

```
source <addr>[:<port>] [usesrc { <addr2>[:<port2>] | client | clientip } ]
```

Syntax for setting interface name:

```
source <addr>[:<port>] [interface <name>]
```

tcp-check

Configures TCP health checking.

Syntax:

```
tcp-check <option> [param*]
```

Supported options are:

- `comment`
- `connect`
- `send`
- `send-lf`
- `send-binary`
- `send-binary-lf`
- `expect`
- `set-var`
- `set-var-fmt`
- `unset-var`

timeout client

Sets the maximum inactivity time on the client side. If you define this value, you must define it in the `udp-lb` section. It is not inherited from the `defaults` section of the load balancer configuration.

The default is 10 seconds, but the ideal setting depends on your traffic and application. For example, if you have a large amount of traffic and a large number of client IP addresses and ports, you could lower the value in order to avoid tracking a high number of connections unnecessarily.

Syntax:

```
timeout client <timeout>
```

timeout server

Sets the maximum inactivity time on the server side. If you define this value, you must define it in the `udp-lb` section. It is not inherited from the `defaults` section of the load balancer configuration.

Syntax:

```
timeout server <timeout>
```

Integrations

- Ansible
- Consul service mesh
- Elastic Stack
- Grafana
- InfluxDB
- NS1

Ansible

> **This page applies to:**
>
> - HAProxy Enterprise - all versions

Ansible is a configuration management solution implemented primarily in the Python programming language that you can use to manage your load balancer deployment. Unlike other configuration management software, Ansible offers an ad-hoc mode in which tasks can be run manually. In many ways, this is similar to running commands via shell scripts or manually via SSH.

This guide shows how to run ad-hoc Ansible commands to control the load balancer, as well as how to organize more complex tasks into Ansible playbooks, which are YAML-formatted task definitions executed with the `ansible-playbook` command.

Install Ansible

1. Install the latest version of Ansible onto your workstation. This will be the control node from which you manage your load balancer nodes. Note that Ansible is not supported on Windows.
2. Install Ansible Lint onto your workstation, which helps to identify syntax and spacing errors and contains style recommendations and deprecation warnings.
3. Install `socat` on your load balancers so Ansible can invoke Runtime API commands:

Apt:

```
sudo apt-get install socat
```

Yum:

```
sudo yum install socat
```

Zypper:

```
sudo zypper install socat
```

Pkg:

```
sudo pkg install socat
```

4. Ansible uses SSH for communication with the remote Linux servers. Therefore, it expects that you have examined and accepted the remote server's SSH host key. Connect via SSH to the machines where the load balancer is installed and accept the host key:

```
ssh lb1.example.com
```

output

```
The authenticity of host 'lb1.example.com (100.200.1.6)' can't be established.
ECDSA key fingerprint is SHA256:dzUE7CyUTeE98A5WKUT8DyNwvNqFO3CcJtRQFvsa4xk.
Are you sure you want to continue connecting (yes/no)? yes
```

5. Update your Ansible `inventory` file, `/etc/ansible/hosts`, by adding your load balancer servers:

```
[loadbalancers]
lb1.example.com
lb2.example.com
```

6. Install Python and `pip`, the Python package manager, onto the load balancer servers. Ansible requires this on all nodes that it manages.

7. After Python is installed, run the following Ansible `ping` command to verify that everything is working:

```
ansible loadbalancers -u root -m ping
```

output

```
loadbalancers | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

The `-u` flag defines the remote user for the SSH connection and `-m` tells Ansible which module should be used. Ping is one of the pre-installed modules.

8. Many of the Ansible commands you'll use require the Runtime API. To enable it, see this guide.

Ansible ad-hoc command usage

Unlike most configuration management engines, you can use Ansible to issue on-the-fly commands to reconfigure the state on multiple machines. It shows you the state of the infrastructure and allows you to perform runtime changes to multiple machines easily.

The most basic command, which we introduced in the last section, uses the `ping` module to tell you whether the machine is alive:

```
ansible loadbalancers -u root -m ping
```

You can run shell commands on the remote machine by specifying the `-a` argument. Below, we invoke the `netstat` command on the remote load balancer servers:

```
ansible loadbalancers -u root -a "netstat -tlpn"
```

To use shell features like pipes and output redirects, you can use the `shell` module:

```
ansible loadbalancers -u root -m shell -a "netstat -tlpn | grep :80"
```

One thing to note is that if you choose to run as a non-root user, some commands will require sudo privileges to work properly. In that case it is necessary to use the `--become` flag (short form: `-b`) before executing commands. Below, we specify `--become` to ensure that the `socat` package is installed on the system:

```
ansible loadbalancers --become --ask-become-pass \
       -m apt -a "name=socat state=present update_cache=yes"
```

The `--ask-become-pass` argument prompts you to enter your sudo password. Please note that for `--ask-become-pass` to work correctly, all machines must have the same password for the sudo user.

Rather than prompting for the sudo password, you can enable passwordless sudo. To enable passwordless sudo, add the `NOPASSWD:` directive to your user or group using the `visudo` command.

```
sudo visudo
```

Then edit the file as shown below and save it:

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) NOPASSWD: ALL
```

Then you can use `--become` without `--ask-become-pass`.

Next, you can check if the load balancer service is running on a node. The `--check` argument can be used with most modules to see what changes would have been made without actually doing them. In this case, we only want to see if the load balancer is active, but not start it otherwise.

```
ansible loadbalancers -u root -m service -a "name=hapee-3.1-lb state=started" --check
```

output

```
haproxy-ams | SUCCESS => {
  "changed": false,
  "name": "hapee-3.1-lb",
  "state": "started",
  ...
```

To perform a hitless reload, use `state=reloaded` and omit the `--check` parameter:

```
ansible loadbalancers -u root -m service -a "name=hapee-3.1-lb state=reloaded"
```

We can combine the above commands with the `copy` module to sync an arbitrary configuration to multiple hosts and then reload the load balancer to apply the changes:

```
ansible loadbalancers -u root -m copy -a "src=/home/user/hapee-lb.cfg dest=/etc/hapee-3.1/hapee-lb.cfg"
```

```
ansible loadbalancers -u root -m service -a "name=hapee-3.1-lb state=reloaded"
```

Doing this for more than a few commands is quite tedious, which is why you might define an Ansible playbook instead. However, in a pinch, the ad-hoc commands are quite useful.

Ad-hoc commands can be used to interact directly with the HAProxy Runtime API.

For example, to disable a specific server from a specific backend:

```
ansible loadbalancers -u root -m shell -a "echo 'disable server bk_www/www-01-server' | socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock"
```

Or, to show different debugging statistics:

- `show stat`

```
ansible loadbalancers -u root -m shell -a "echo 'show stat' | socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock"
```

- `show info`

```
ansible loadbalancers -u root -m shell -a "echo 'show info' | socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock"
```

- `show fd`

```
ansible loadbalancers -u root -m shell -a "echo 'show fd' | socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock"
```

- `show activity`

```
ansible loadbalancers -u root -m shell -a "echo 'show activity' | socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock"
```

- `show pools`

```
ansible loadbalancers -u root -m shell -a "echo 'show pools' | socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock"
```

See the Runtime API documentation for more examples.

The ad-hoc commands are also useful for prototyping the commands which are going to become part of a larger playbook, so it's useful to be comfortable with running ad-hoc commands before beginning to write complex playbooks.

Write your first Ansible Playbook

In the last section we described a way to transfer a load balancer configuration to multiple hosts by using the `ansible` command; The commands used were the following:

```
ansible loadbalancers -u root -m copy -a "src=/home/user/hapee-lb.cfg dest=/etc/hapee-3.1/hapee-lb.cfg"
```

```
ansible loadbalancers -u root -m service -a "name=hapee-3.1-lb state=reloaded"
```

In this section, we will show how to adapt these ad-hoc commands into a playbook that can achieve the same result. The equivalent playbook would be the following:

```
- hosts: loadbalancers
  remote_user: root
  tasks:
    - name: Copy load balancer configuration
      copy:
        src: "/home/user/hapee-lb.cfg"
        dest: "/etc/hapee-3.1/hapee-lb.cfg"
        owner: root
        group: hapee
        mode: 0644
    - name: Check if there are no errors in configuration file
      command: "/opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg"
      register: hapee_check
    - name: Reload load balancer if the check passed
      service:
        name: hapee-3.1-lb
        state: reloaded
      when: hapee_check is success and not ansible_check_mode
```

Multiple configuration files

If you have multiple configuration files in your application, be sure the `hapee-lb -c` command checks them all in the correct order.

Depending on your version, the command's output indicates that the file is valid in these ways:

- In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
- In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

You might notice a few things in the above YAML snippet when moving from ad-hoc commands to a playbook:

- Under tasks, each separate command needs to be named; This name is displayed during playbook execution.
- One addition not present in the ad-hoc commands is the `register` keyword, which is used to store the success/fail result of the remote command:

```
  - name: Check if there are no errors in configuration file
    command: "/opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg"
    register: hapee_check
```

Then, the `when` line in the next block verifies that the configuration syntax check executed correctly. The block executes only upon success.

```
- name: Reload load balancer if the check passed
  service:
      name: hapee-3.1-lb
      state: reloaded
  when: hapee_check is success and not ansible_check_mode
```

If the check did not pass, we skip reloading the load balancer. It also checks that Ansible is not running in dry-run mode by verifying `ansible_check_mode`.

Depending on your version, the command's output indicates that the file is valid in these ways:

- In version 2.8 and earlier, the command indicates a valid configuration by printing `Configuration file is valid` in addition to setting the zero return status.
- In version 2.9 and later, the command sets the zero return status for a valid configuration but does not display a message. To display the message, include the `-V` option on the command line.

Ansible Lint

Save the above file as `first-haproxy-deploy.yml` and check its syntax with `ansible-lint`:

```
ansible-lint first-haproxy-deploy.yml
```

output

```
[301] Commands should not change things if nothing needs doing
first-playbook.yml:14
Task/Handler: Check if there are no errors in configuration file
```

You can ignore the `[ 301 ]` warning in this case; We are only interested in making sure there are no obvious errors, the most common being missing spaces in the YAML file.

The linter is useful, however it generally only catches Ansible syntax errors. Therefore, it is recommended to run playbooks with the `--check` flag to catch some Ansible runtime errors. Run the `ansible-playbook` command with the `--check` flag:

```
ansible-playbook first-haproxy-deploy.yml --check
```

This step only validates that there are no obvious errors in the playbook itself, and not the load balancer configuration file.

Finally, to run the playbook execute:

```
ansible-playbook first-haproxy-deploy.yml
```

Jinja templates

You can utilize Jinja templates in both the playbook YAML file itself and in configuration files it manages.

To use Jinja templates to manage an external file you can use the `template` module. In the snippet below, the `tasks` block now includes a `template` block:

```
- hosts: loadbalancers
  remote_user: root
  tasks:
    - name: Sync main HAPEE configuration
      template:
        src: ../templates/hapee-lb.cfg.j2
        dest: /etc/hapee-3.1/hapee-lb.cfg
        owner: root
        group: hapee
        mode: 0664
    - name: Check if there are no errors in configuration file
      command: "/opt/hapee-3.1/sbin/hapee-lb -c -f /etc/hapee-3.1/hapee-lb.cfg"
      register: hapee_check
    - name: Reload HAPEE if the check passed
      service:
        name: hapee-3.1-lb
        state: reloaded
      when: hapee_check is success and not ansible_check_mode
```

To start using Jinja templates, it is enough to rename a file to have a `.j2` extension. Then you can optionally introduce Jinja templating patterns into the file.

To use Jinja templates inside your playbooks directly, you can look at the following example playbook:

```
---
# updates ACLs on remote load balancer nodes via the Runtime API
- hosts: loadbalancers
  remote_user: ubuntu
  become: true
  become_user: root
  become_method: sudo
  tasks:
    - name: update ACL file
      shell: "echo '{{ acl_action }} acl {{ acl_path }} {{ acl_ip_address }}' | socat stdio unix-connect:/var/run/
hapee-3.1/hapee-lb.sock"
    - name: check ACL file for presence/absence of the IP address
      shell: "echo 'show acl {{ acl_path }}' | socat stdio unix-connect:/var/run/hapee-3.1/hapee-lb.sock | grep
{{ acl_ip_address }}"
      register: socket_show
      failed_when: "socket_show.rc != 0 and socket_show.rc != 1"
    - debug: var=socket_show.stdout_lines
```

Jinja patterns are enclosed in quotes and curly brackets `{{}}`; The variables inside of the brackets will be expanded and used as strings. Variables are set when you run the `ansible-playbook` command, as shown below:

```
ansible-playbook update-acl-playbook.yml \
       -e "acl_action=add acl_path=/etc/hapee-3.1/ip-whitelist.acl `acl_ip_address=10.10.10.10"`
```

External variables are defined via the `-e` flag. However, Jinja templates can be used with various variables like host variables, group variables, Ansible facts or custom variables set via the `register` keyword. See more examples in the Ansible Templating (Jinja2) guide.

Ansible inventory file configuration

In our previous `inventory` file, we defined two load balancer nodes that Ansible manages:

```
[loadbalancers]
lb1.example.com
lb2.example.com
```

You can create hierarchies of load balancer groups, such as to update all load balancers, only load balancers in Europe, or only load balancers in the Americas. One node can appear in multiple groups. The group names can contain any valid DNS record or IP addresses; As a suffix, the port on which SSH is listening can also be specified. A more complex inventory file could look similar to this:

```
[loadbalancers:children]
loadbalancers-europe
loadbalancer-americas

[loadbalancers-europe]
lb01.eu.example.com
lb02.eu.example.com

[loadbalancers-americas]
lb01.us.example.com
lb02.ca.example.com
lb02.mx.example.com

[loadbalancers-staging]
# Using IP address with SSH ports
10.10.12.10:2222
10.10.12.11:2223
```

The `:children` keyword creates a new group that inherits the nodes of two other groups. Set the group when calling an ad-hoc command. Below, we check whether the European load balancers are up:

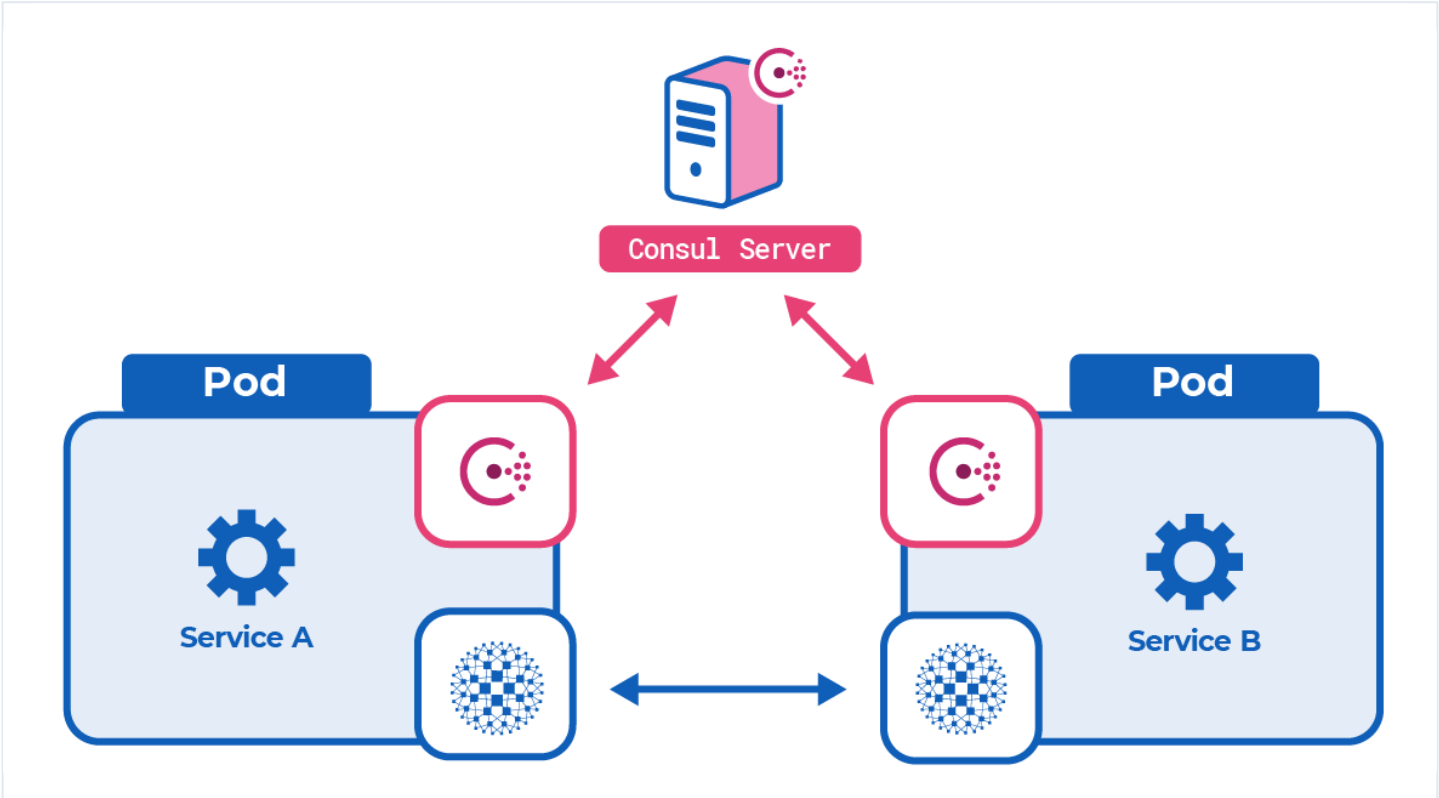
```
ansible loadbalancers-europe -u root -m ping
```

Consul service mesh

> **This page applies to:**
>
> - HAProxy Enterprise - all versions

HashiCorp Consul operates as a service mesh when you enable its Connect mode. In this mode, Consul agents integrate with HAProxy Enterprise to form an interconnected web of proxies. Whenever one of your services needs to call another, their communication is relayed through the web, or mesh, with HAProxy Enterprise nodes passing messages between all services.

An HAProxy Enterprise node exists next to each of your services on both the caller and callee end. When a caller makes a request, they direct it to `localhost` where HAProxy Enterprise is listening. HAProxy Enterprise then relays it transparently to the remote callee. From the caller's perspective, all services appear to be local, which simplifies the service's configuration.



Deploy in Kubernetes

This section describes how to deploy the Consul service mesh with HAProxy Enterprise in Kubernetes.

Deploy the Consul servers

Consul agents running in server mode watch over the cluster and send service discovery information to each Consul client in the service mesh.

1. Deploy the Consul server nodes. In Kubernetes, you can install the Consul Helm chart.

```
helm repo add hashicorp https://helm.releases.hashicorp.com
helm repo update
helm install consul hashicorp/consul \
  --set global.name=consul \
  --set connect=true
```

   If you are using a single-node Kubernetes cluster, such as minikube, then set the `server.replicas` and `server.bootstrapExpect` flags, as described in the guide Consul Service Discovery and Mesh on Minikube.

```
helm install consul hashicorp/consul \
  --set global.name=consul \
  --set connect=true \
  --set server.replicas=1 \
  --set server.bootstrapExpect=1
```

2. Create a file named `pod-reader-role.yaml` and add the following contents to it.

   This creates a Role and RoleBinding resource in your Kubernetes cluster that grant permissions to the Consul agents to read pod labels.

pod-reader-role.yaml

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]

---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: system:serviceaccount:default:default
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

3. Deploy it with `kubectl apply`:

```
kubectl apply -f pod-reader-role.yaml
```

Deploy your application

For each service that you want to include in the service mesh, you must deploy two extra containers into the same pod.

- container 1: your application
- container 2: Consul agent, `consul`
- container 3: HAProxy-Consul connector, `hapee-plus-registry.haproxy.com/hapee-consul-connect`

The three containers (application, Consul, Consul-HAProxy Enterprise connector) are defined inside a single pod.

1. Use `kubectl create secret` to store your credentials for the private HAProxy Docker registry, replacing `<KEY>` with your HAProxy Enterprise license key. You will pull the `hapee-consul-connect` container image from this registry.

```
kubectl create secret docker-registry regcred --namespace=default --docker-server=hapee-plus-
registry.haproxy.com --docker-username=<KEY> --docker-password=<KEY>
```

2. Add the `haproxy-enterprise-consul` and `consul` containers to each of your Kubernetes Deployment manifests. In the example below, we deploy these two containers inside the same pod as a service named `example-service`.

example-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: example-service
 labels:
   app: example-service
spec:
 replicas: 1
 selector:
   matchLabels:
     app: example-service
 template:
   metadata:
     labels:
       app: example-service
   spec:
     imagePullSecrets:
     - name: regcred
     containers:
     - name: example-service
       image: jmalloc/echo-server
     - name: haproxy-enterprise-consul
       image: hapee-plus-registry.haproxy.com/hapee-consul-connect
       args:
       - -sidecar-for=example-service
       - -enable-intentions
     - name: consul
       image: consul
       env:
             - name: CONSUL_LOCAL_CONFIG
               value: '{
                 "service": {
                   "name": "example-service",
                   "port": 80,
                   "connect": {
                     "sidecar_service": {}
                   }
                 }
             }'
       args: ["agent", "-bind=0.0.0.0", "-retry-join=provider=k8s label_selector=\"app=consul\""]
```

Note the following arguments for the `haproxy-enterprise-consul` container:

Argument	Description
`-sidecar-for example-service`	Indicates the name of the service for which to create an HAProxy Enterprise proxy.
`-enable-intentions`	Enables Consul intentions, which HAProxy Enterprise enforces.

Note the following arguments for the `consul` container:

Argument	Description
`agent`	Runs the Consul agent.
`-bind=0.0.0.0`	The address that should be bound to for internal cluster communications.
`-retry-join=provider=k8s label_selector="app=consul"`	Similar to `-join`, which specifies the address of another agent to join upon starting up (typically one of the Consul server agents), but allows retrying a join until it is successful. In Kubernetes, you set this to `provider=k8s` and then include a label selector for finding the Consul servers. The Consul Helm chart adds the label `app=consul` to the Consul server pods.

We've registered the `example-service` with the Consul service mesh by setting an environment variable named `CONSUL_LOCAL_CONFIG` in the Consul container. This defines the Consul configuration and registration for the service. It indicates that the service receives requests on port 80.

```
{
  "service": {
    "name": "example-service",
    "port": 80,
    "connect": {
      "sidecar_service": {}
  }
}
```

3. Deploy it with `kubectl apply`:

```
kubectl apply -f example-service.yaml
```

Deploy a second application that calls the other

The `example-service` from the previous section is published to the service mesh where other services within the mesh can call it. To define a service that calls another, add a `proxy` section to the `connect.sidecar_service` section of the Consul container's configuration.

In the example below, the service named `app-ui` adds the example-service as an upstream service, which makes it available at `localhost` at port 3000 inside the pod.

app-ui-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: app-ui
 labels:
   app: app-ui
spec:
   replicas: 1
   selector:
     matchLabels:
       app: app-ui
   template:
     metadata:
       labels:
         app: app-ui
     spec:
       containers:
       - name: app-ui
         image: jmalloc/echo-server
       - name: haproxy-enterprise-consul
         image: hapee-plus-registry.haproxy.com/hapee-consul-connect
         args:
         - -sidecar-for=app-ui
         - -enable-intentions
       - name: consul
         image: consul
         env:
         - name: CONSUL_LOCAL_CONFIG
           value: '{
               "service": {
                 "name": "app-ui",
                 "port": 80,
                 "connect": {
                   "sidecar_service": {
                       "proxy": {
                           "upstreams": [
                               {
                                 "destination_name": "example-service",
                                 "local_bind_port": 3000
                               }
                           ]
                       }
                   }
                 }
               }
             }'
```

```
        args: ["agent", "-bind=0.0.0.0", "-retry-join=provider=k8s label_selector=\"app=consul\""]
```

Note that we set the environment variable named `CONSUL_LOCAL_CONFIG` in the Consul container to register this service with the service mesh. It declares that it has an upstream dependency on the `example-service` service.

Optional: Publish the web dashboard

The Helm chart creates a Kubernetes service named `consul-server` that exposes a web dashboard on port 8500. To make it available outside of the Kubernetes cluster, you can forward the port via the HAProxy Enterprise Kubernetes Ingress Controller:

1. Deploy the HAProxy Enterprise Kubernetes Ingress Controller into your Kubernetes cluster.

2. Create a file named `consul-server-ingress.yaml` that defines an Ingress resource for the Consul service.

   In this example, we define a host-based rule that routes all requests for `consul.test.local` to the `consul-server` service at port 8500.

   consul-server-ingress.yaml

   ```
   apiVersion: networking.k8s.io/v1
   kind: Ingress
   metadata:
     name: consul-server-ingress
   spec:
     rules:
       - host: consul.test.local
         http:
           paths:
             - path: "/"
               pathType: Prefix
               backend:
                 service:
                   name: consul-server
                   port:
                     number: 8500
   ```

3. Deploy it using `kubectl apply`:

   ```
   kubectl apply -f consul-server-ingress.yaml
   ```

4. Add an entry to your system's `/etc/hosts` file that maps the `consul.test.local` hostname to the IP address of your Kubernetes cluster. If you are using minikube, you can get the IP address of the node with `minikube ip`. Below is an example `/etc/hosts` file:

```
192.168.99.125  consul.test.local
```

5. Use `kubectl get service` to check which port the ingress controller has mapped to port 80. In the example below, port 80 is mapped to port 30624.

```
minikube ip
```

output

```
192.168.99.120
```

```
kubectl get service kubernetes-ingress
```

output

```
NAME                 TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)                                 AGE
kubernetes-ingress   NodePort   10.110.104.60   <none>        80:30624/TCP,443:31147/TCP,1024:31940/TCP   7m40s
```

Open a browser window and go to the `consul.test.local` address at that port, e.g. `consul.test.local:30624`.

Optional: Enable Consul ACLs

In Consul, ACLs are a security measure that requires Consul agents to present an authentication token before they can join the cluster or call API methods.

1. When installing Consul, set the `global.acls.manageSystemACLs` flag to true to enable ACLs.

```
helm install consul hashicorp/consul \
  --set global.name=consul \
  --set connect=true \
  --set global.acls.manageSystemACLs=true
```

2. Install `jq`, a command-line utility for processing JSON data. It provides a simple and powerful way to filter, format, and transform JSON data structures.

Apt:

```
sudo apt install jq
```

Yum:

```
sudo yum install jq
```

3. Use `kubectl get secret` to get the auto-generated bootstrap token, which is base64 encoded.

```
kubectl get secret consul-bootstrap-acl-token -o json | jq -r '.data.token' | base64 -d
```

output

```
8f1c8c5e-d0fb-82ff-06f4-a4418be245dc
```

Use this token to log into the Consul web UI.

4. In the Consul web UI, go to **ACL > Policies** and select the **client-token** row. Change the policy's value so that the **service_prefix** section has a policy of `write`:

```
node_prefix "" {
  policy = "write"
}
service_prefix "" {
  policy = "write"
}
```

5. Go back to the ACL screen and select the **client-token** row. Copy this token value (e.g. `f62a3058-e139-7e27-75a0-f47df9e2e4bd`).

6. For each of your services, update your Deployment manifest so that the `haproxy-enterprise-consul` container includes the `-token` argument, set to the **client-token** value.

```
- name: haproxy-enterprise-consul
  image: hapee-plus-registry.haproxy.com/hapee-consul-connect
  args:
  args:
  - -sidecar-for=app-ui
  - -enable-intentions
  - -token=f62a3058-e139-7e27-75a0-f47df9e2e4bd
```

7. Update the `consul` container's configuration to include an `acl` section where you will specify the same **client-token** value. Also, set `primary_datacenter` to `dc1` (or to the value you've set for your primary datacenter, if you have changed it).

```
- name: consul
  image: consul
  env:
    - name: CONSUL_LOCAL_CONFIG
      value: '{
        "primary_datacenter": "dc1",
        "acl": {
          "enabled": true,
          "default_policy": "allow",
          "down_policy": "extend-cache",
          "tokens": {
            "default": "f62a3058-e139-7e27-75a0-f47df9e2e4bd"
          }
        },
        "service": {
          "name": "example-service",
          "port": 80,
          "connect": {
            "sidecar_service": {}
          }
        }
      }'
```

Elastic Stack

This page applies to:

- HAProxy Enterprise - all versions

This section illustrates how to connect the load balancer to Elastic Stack, a suite of applications for visualizing metrics along with Metricbeat and its HAProxy module, which is installed on each load balancer server to ship metrics and logs from the load balancer to the Elastic Stack.

Installation

Elastic Stack is a highly scalable and highly configurable suite of applications that can be deployed on a single machine or dozens of machines, depending upon the volume of data that must be processed, the rate at which the data is ingested, as well as the complexity of the output visualizations desired.

1. Follow the Installing Elasticsearch guide to install Elasticsearch onto a server.

   We will store metrics data in Elasticsearch.

2. Follow the Installing Kibana guide to install Kibana onto another server.

   We will visualize our data with Kibana.

3. By default, Elasticsearch is available on port 9200. You can test this using `curl`:

   ```
   curl -X GET "<elasticsearch_server_ip>:9200"
   ```

output

```
{
"name" : "es01",
"cluster_name" : "es-docker-cluster",
"cluster_uuid" : "2PSZIzo_SH-AEDK9nkPELw",
"version" : {
    "number" : "7.12.0",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "78722783c38caa25a70982b5b042074cde5d3b3a",
    "build_date" : "2021-03-18T06:17:15.410153305Z",
    "build_snapshot" : false,
    "lucene_version" : "8.8.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
},
"tagline" : "You Know, for Search"
}
```

4. Ensure that your load balancer configuration has the Runtime API enabled:

```
global
  stats socket ipv4@*:9999 user haproxy group haproxy mode 660 level admin
```

5. Follow the Installing Metricbeat guide to install Metricbeat onto your load balancer server.

6. Enable the Metricbeat HAProxy module:

```
sudo metricbeat modules enable haproxy
```

7. Edit the file `/etc/metricbeat/metricbeat.yml` so that it lists your Elasticsearch server under the output.elasticsearch section:

```
output.elasticsearch:
  hosts: ["<elasticsearch_server_ip>:9200"]
```

8. Edit the file `/etc/metricbeat/modules.d/haproxy.yml` to configure the HAProxy module so that the hosts field includes the address and port where your HAProxy Runtime API is listening:

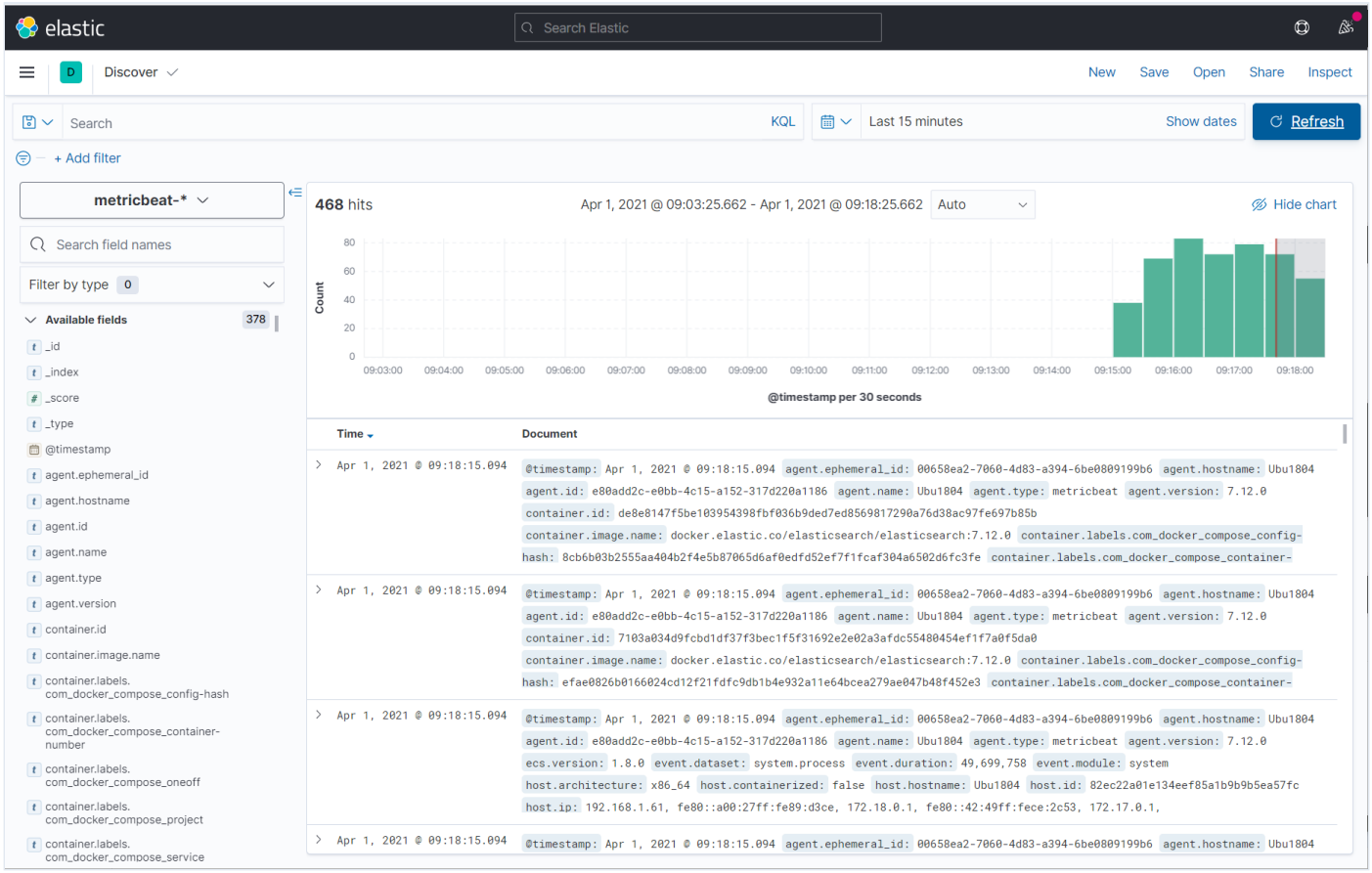
```
- module: haproxy
  metricsets: ["info", "stat"]
  period: 10s
  hosts: ["tcp://<haproxy_server_ip>:9999"]
  enabled: true
```

9. Restart the Metricbeat service to begin shipping HAProxy metrics to Elasticsearch:

```
sudo service metricbeat restart
```

10. Open Kibana's interface, `http://<kibana_server_ip>:5601/app/discover#/` using a web browser.

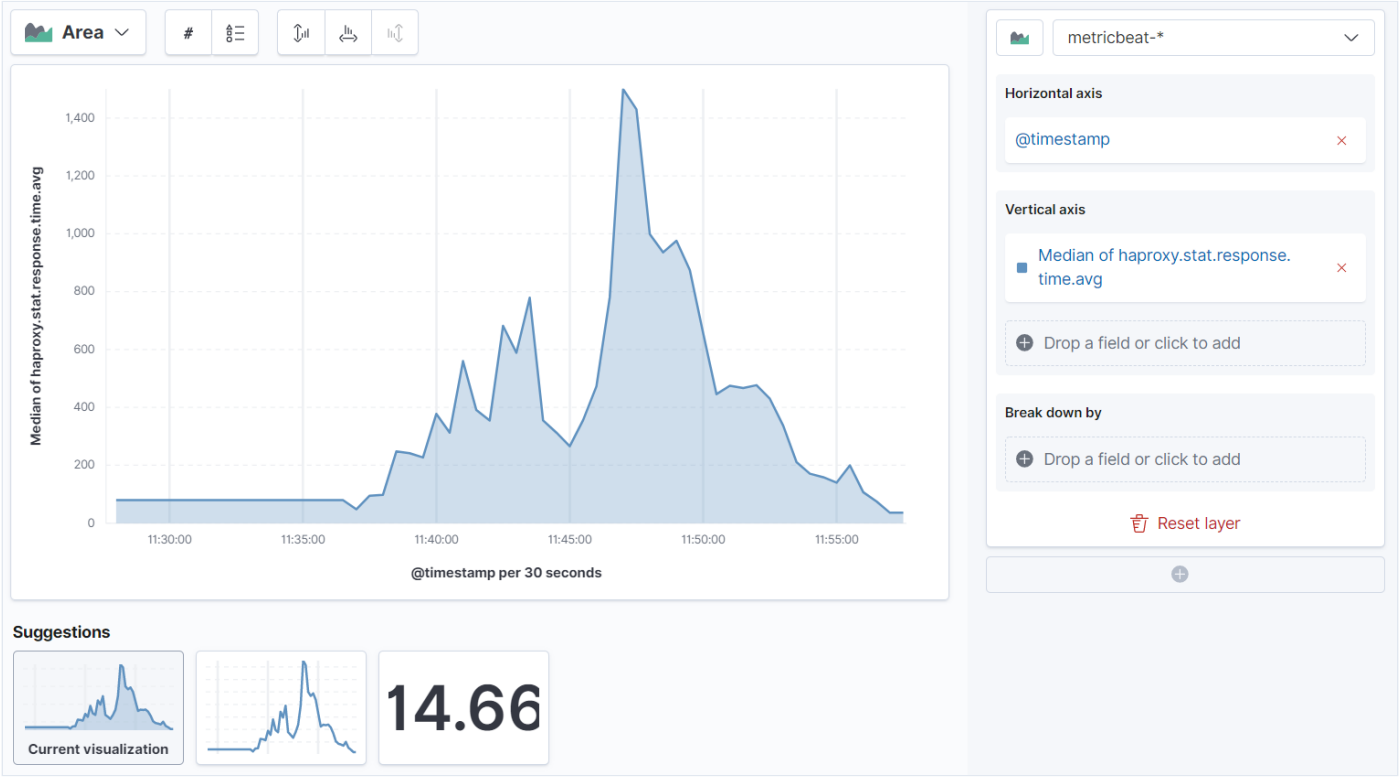
    If everything is working correctly, you will see that your metrics are being collected:



11. Begin creating visualizations using Kibana's Lens application, `http://<kibana_server_ip>:5601/app/lens/`.

    Drag and drop fields to the visualization area and adjust your timeframe.

    In this example, we added a metric for response time during a benchmarking test:


Area
metricbeat-*
Horizontal axis
@timestamp
Vertical axis
Median of haproxy.stat.response.time.avg
Drop a field or click to add
Break down by
Drop a field or click to add
Reset layer
Median of haproxy.stat.response.time.avg
1,400
1,200
1,000
800
600
400
200
0
11:30:00
11:35:00
11:40:00
11:45:00
11:50:00
11:55:00
@timestamp per 30 seconds
Suggestions
Current visualization
14.66

Grafana

This page applies to:

- HAProxy Enterprise 2.0r1 and newer

This guide shows how to export the load balancer's live traffic metrics to Grafana. We enable the built-in Prometheus metrics endpoint, which a Prometheus server scrapes at an interval. Then, Grafana displays graphs of the data stored in the Prometheus server.

Set up Prometheus and Grafana

1. Install a Prometheus server and configure it to scrape metrics from your load balancer. Follow our Prometheus guide, which shows how to:
   - enable the load balancer's built-in Prometheus metrics exporter
   - configure your Prometheus server to scrape the Prometheus endpoint for live metrics
2. Prometheus hosts a dashboard on port 9090. Open the dashboard and verify that it can scrape the endpoint successfully. Check under the menu **Status > Targets** where you should see the load balancer endpoint with a state of `UP`.
3. Follow the official Grafana documentation to install Grafana.
4. Log into the Grafana dashboard at port 3000 using the username and password `admin`. You will be prompted to immediately change the password.
5. In the left-hand navigation, go to the **Configuration** screen and add a new Prometheus data source. Set the URL to the address where your Prometheus server is listening (for example, `http://localhost:9090`), then save it.
6. Click the plus sign (+) in the left-hand navigation, then **Create > Dashboard**. Add a new panel to begin creating graphs that use metrics from the Prometheus data source.

Tip

Try importing a pre-baked dashboard, such as the Ricardo F HAProxy Dashboard. In this case, copy the JSON and paste it into the textbox on the Import screen. Then, click **Load**.

Further considerations

- The easiest way to achieve high availability for the above monitoring stack is to spin up a separate node with Prometheus and Grafana installed, and ingest the same data. For most use cases this is more than enough.
- If you require a more robust high-availability solution, you can use Thanos. The project description states:

  Thanos leverages the Prometheus 2.0 storage format to cost-efficiently store historical metric data in any object storage while retaining fast query latencies. Additionally, it provides a global query view across all Prometheus installations and can merge data from Prometheus HA pairs on the fly.
- Also consider enabling the Prometheus Node Exporter to log the CPU, memory, and other system stats on every load balancer node. This will help catch maxed out CPU, high memory, and network congestion issues.

InfluxDB

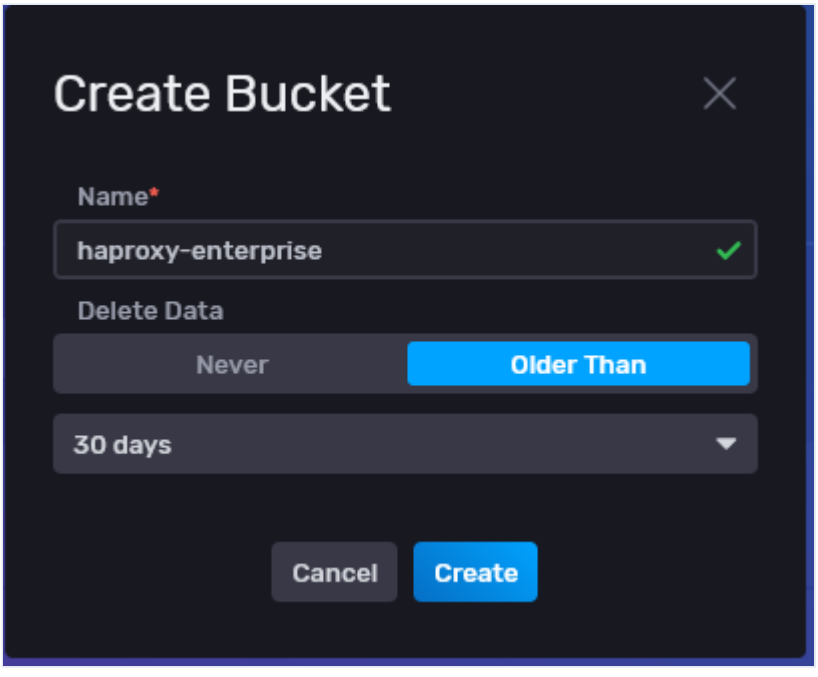
> **This page applies to:**
>
> - HAProxy Enterprise - all versions

InfluxDB is a time-series database into which you can store your load balancer metrics. You can then graph the data to see trends over time.

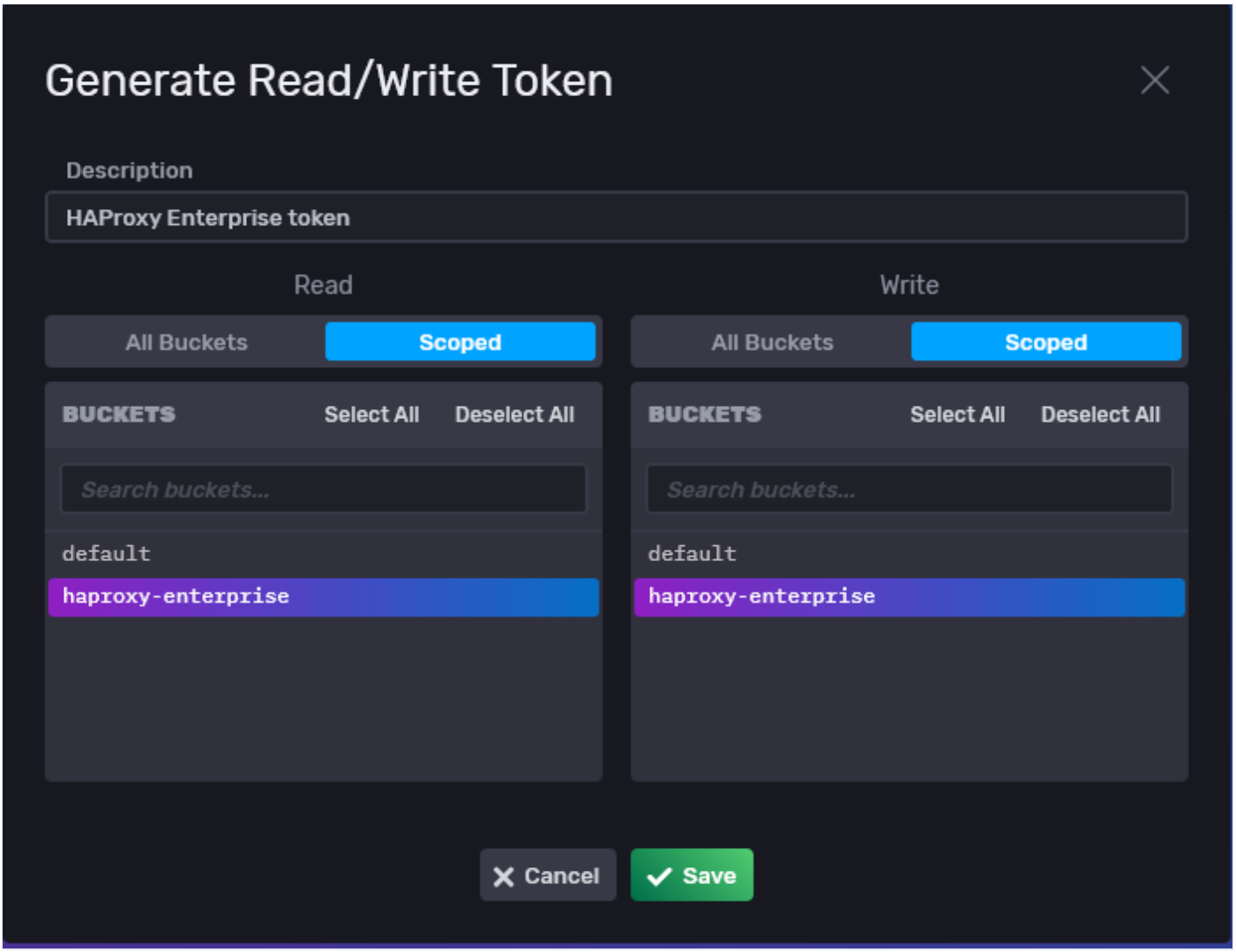
In this guide, we use Telegraf, a metrics-collections tool from InfluxData, to pull metrics from the load balancer and push them to InfluxDB.

Installation and setup

1. Follow the InfluxDB Getting Started guide to install InfluxDB onto a server.
2. Once installed, InfluxDB hosts a UI dashboard at port 8086. Go to the InfluxDB UI at `http://[INFLUXDB SERVER]:8086` and click **Get Started** to set up your initial user account. During setup, you will set the organization and default bucket for your data.
3. Go to **Data > Buckets** and create a new bucket named `haproxy-enterprise`.

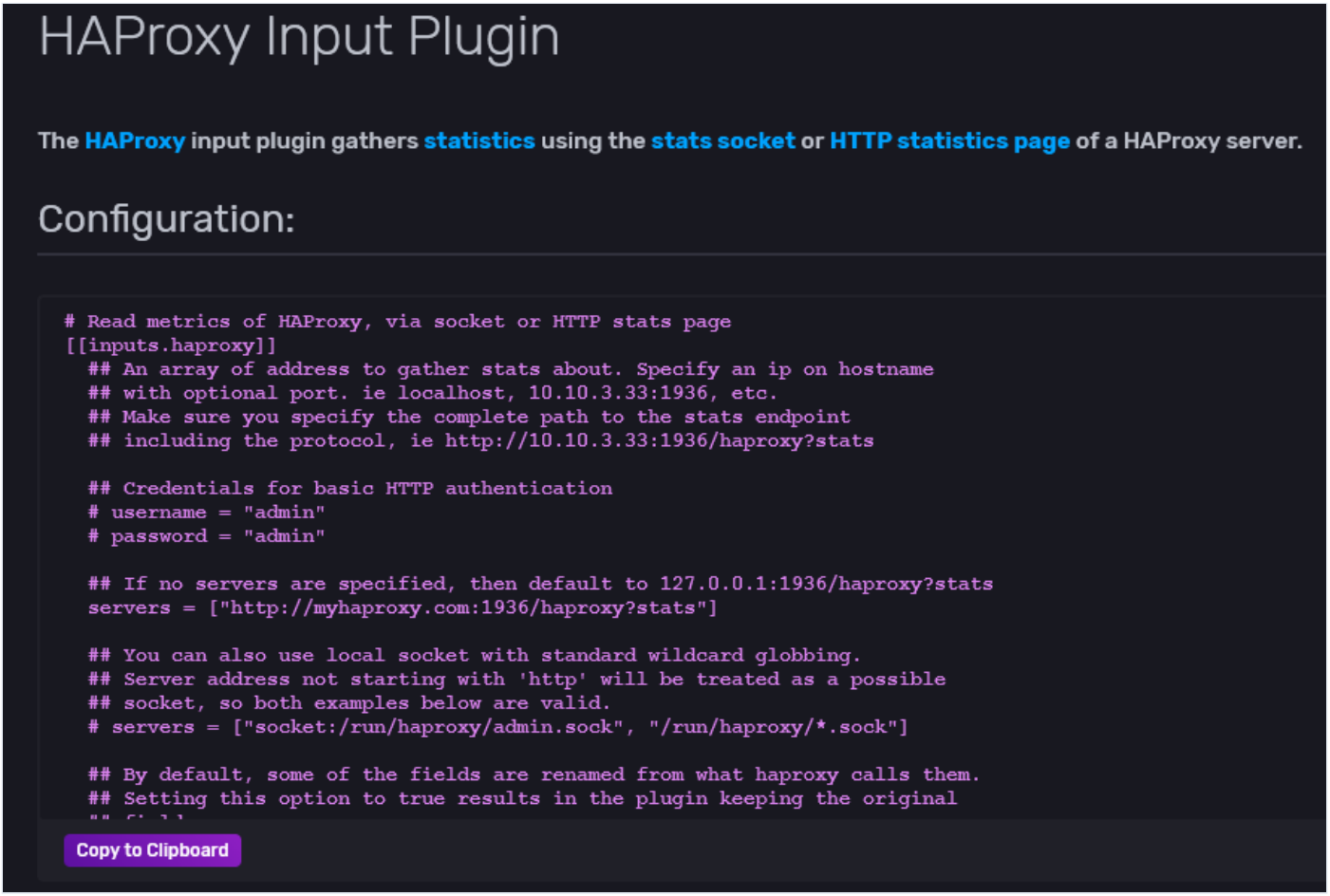


4. Go to **Data > Tokens** and click **Generate** to create a new Read/Write token. Set the Read and Write scopes to your `haproxy-enterprise` bucket. Telegraf will use this token when connecting to InfluxDB.



5. To see an example configuration for Telegraf, go to **Data > Sources**, find **HAProxy** in the list of Telegraf plugins, and click it.

Copy the sample configuration from the HAProxy Input Plugin page.



6. Follow the Introducing Telegraf guide to download and install Telegraf onto each of your load balancer servers. Telegraf is the service that exports metrics to InfluxDB.

7. Enable the Runtime API in your load balancer configuration to allow Telegraf to collect metrics. Configure it to listen on the UNIX socket `/var/run/hapee-3.1/hapee-lb.sock`:

```
global
  stats socket /var/run/hapee-3.1/hapee-lb.sock  user hapee-lb  group hapee  mode 660  level admin  expose-fd
listeners
```

8. Add the `telegraf` user to the `hapee` group so that it has access to the Runtime API socket:

```
sudo usermod -a -G hapee telegraf
```

9. Edit the file `/etc/telegraf/telegraf.conf`:

   - Uncomment the `[[outputs.influxdb_v2]]` section and set the following fields:

     - set `urls` to the address of your InfluxDB server, which listens on port 8086
     - set `token` to the value of the Read/Write token you created in InfluxDB
     - set `organization` to the organization you set when you installed InfluxDB
     - set `bucket` to haproxy-enterprise

     ```
     [[outputs.influxdb_v2]]
       urls = [http://192.168.50.26:8086]
       token = "IyjPRrnA5Fb4P_bHd8YBeea5K-y2DmSV_7D0wT5592YDc9v87vzQKKt-etS2YTHwbpo_gSrFQDhtBUfcDuIzXQ=="
       organization = "test"
       bucket = "haproxy-enterprise"
     ```

   - Uncomment the `[[inputs.haproxy]]` section. Set the `servers` field to the path of the Runtime API socket.

     ```
     # Read metrics of HAProxy, via socket or HTTP stats page
     [[inputs.haproxy]]
       servers = ["/var/run/hapee-3.1/hapee-lb.sock"]
     ```

10. Restart the Telegraf and load balancer services:

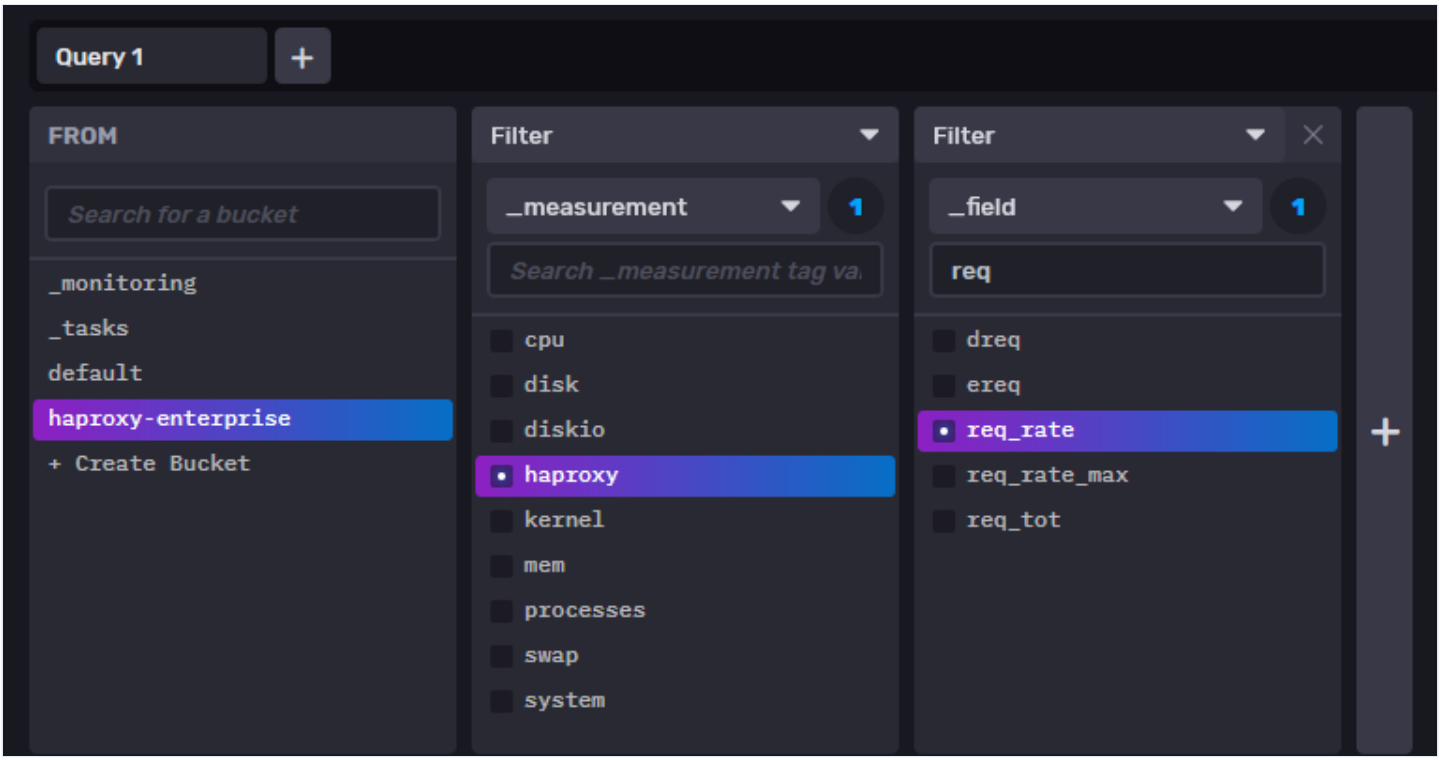
    ```
    sudo systemctl restart telegraf
    sudo systemctl restart hapee-3.1-lb
    ```

Visualize metrics

You can create graphs for your load balancer metrics in the InfluxDB UI. Go to **Explore** and create a new query.

For example, you could create the following query:

- Set **FROM** to `haproxy-enterprise`
- Set the first **Filter** to `_measurement` and select `haproxy`
- Set the second **Filter** to `_field` and select `req_rate` to see the HTTP request rate



HAProxy Monitoring Template

InfluxData also provides an HAProxy monitoring template that configures a prebaked dashboard of commonly used graphs.

1. In the InfluxDB UI, go to **Settings > Templates** and enter the URL `https://raw.githubusercontent.com/influxdata/community-templates/master/haproxy/haproxy.yml`. Then click **Install Template**.
2. The template creates a bucket named `haproxy`. Generate a new Read/Write token that has access to that bucket.
3. Update the Telegraf configuration on your load balancer servers to use the new token and `haproxy` for the bucket value.
4. In the InfluxDB UI, go to **Boards** and click the **haproxy** dashboard to see the graphs.

NS1

This page applies to:

- HAProxy Enterprise - all versions

When you run the load balancer in more than one availability zone, you may direct traffic to the geographically closest load balancer node. For example, you may use Anycast network routing to send clients to the nearest node. But where should a client be sent when the closest load balancer is experiencing high latency or is down?

Load shedding is a mechanism that allows you to send clients to a more distant load balancer if the closest one is too busy. Here you will learn how to configure load shedding at the DNS layer. You will use the NS1 DNS service to monitor your load balancers and shed load to an alternate load balancer when the closest node becomes too busy or is down.

Prerequisites

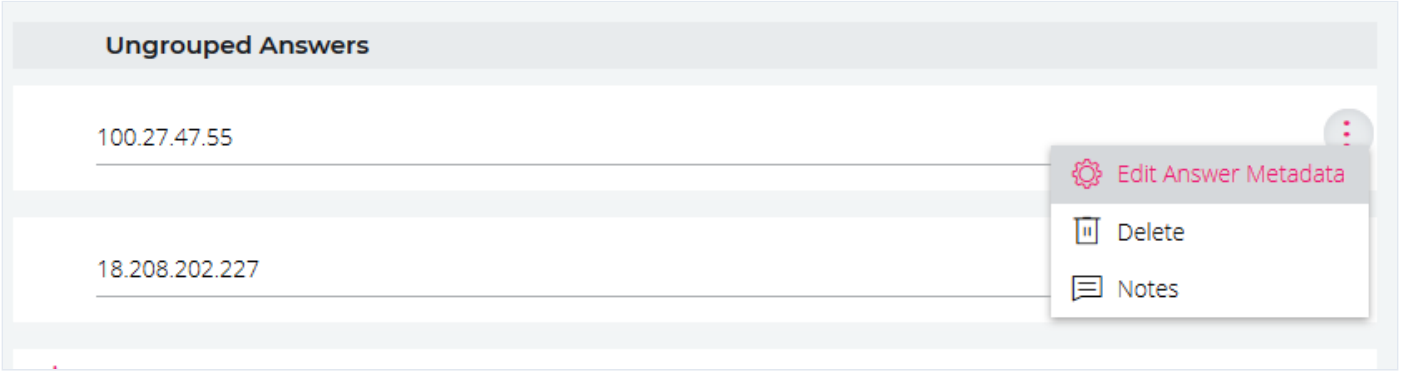
- An NS1 account at NS1.com
- A domain name to add DNS records for
- Two or more geographically dispersed load balancer nodes

Configure the NS1 records

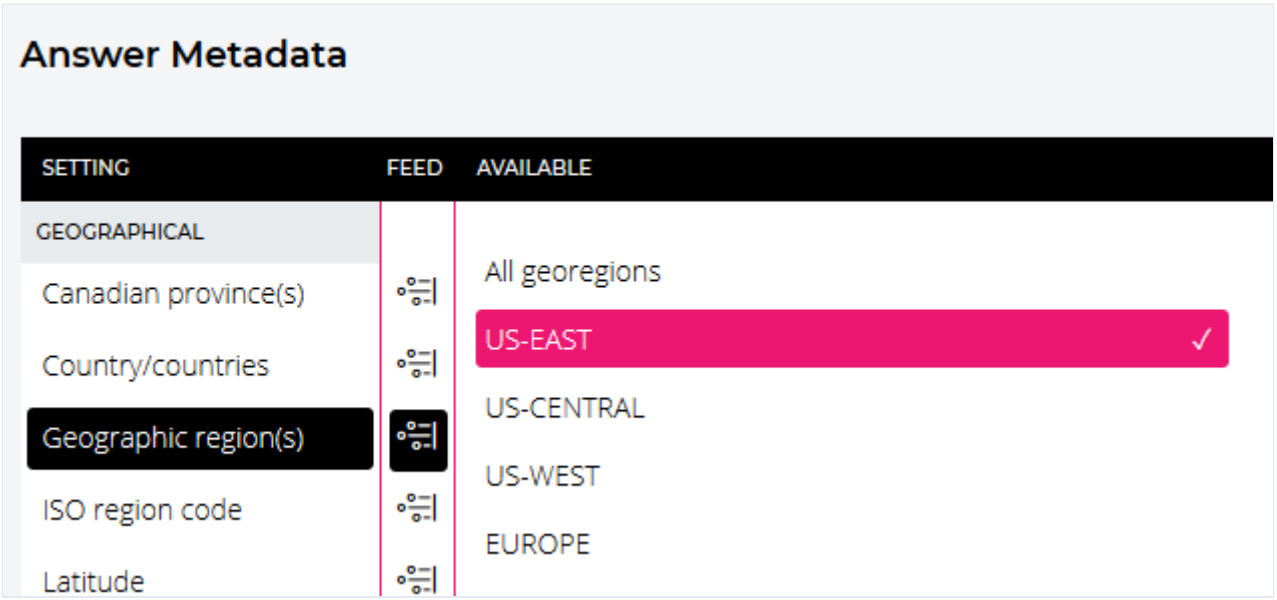
When there are several IP addresses (answer) for a given zone, NS1 must choose the best one based on metadata associated with each answer. Set up metadata to inform this decision.

1. In the NS1 Customer Portal, set up a DNS zone (for example, `www.foo.com`) and create A records within it for each of your load balancers.

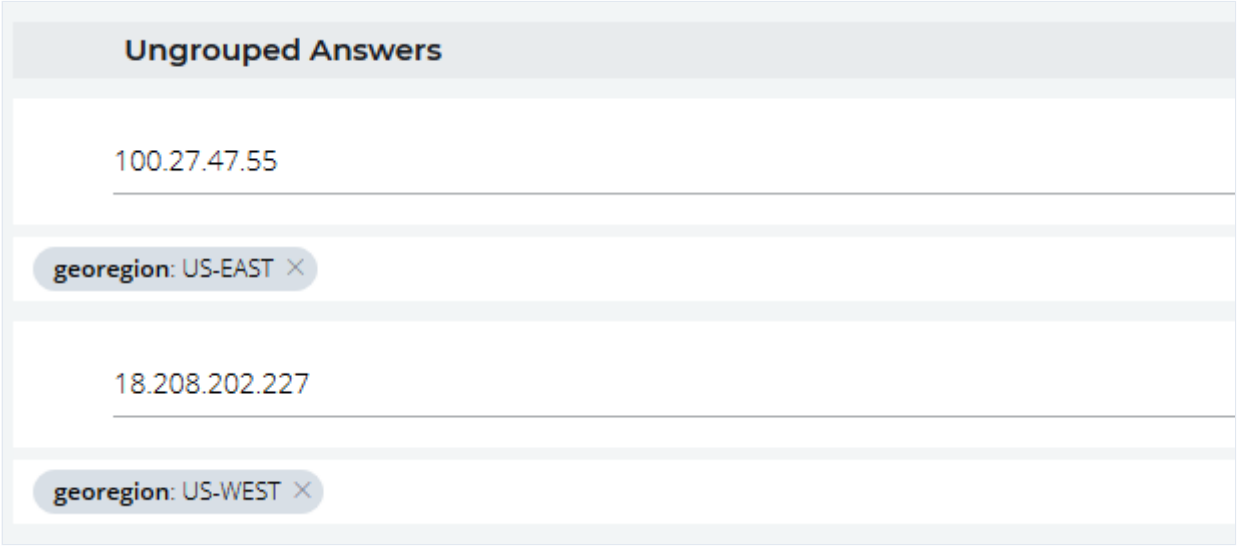
2. For each answer, click **Edit Answer Metadata** to display a settings window.



3. Set the appropriate Geographic region (for example, “US-EAST”).



4. Check that you associate each answer with a region.



5. Add metadata for the **Up/down** status and set it to `Up`. This takes into consideration the status of the load balancer in addition to its geographic location.

Configure data feeds for the load balancer

Use the NS1 API to create data feeds where the load balancer can push data.

1. Go to the NS1 portal and generate an API key. This key needs the following permissions:
   - push to datafeeds
   - manage datasources
   - manage datafeeds
2. Create a new data source with `curl` to call the `/v1/data/sources` NS1 API endpoint. Make sure you set your API key for the `X-NSONE-Key` HTTP header:

```
curl -sH 'X-NSONE-Key: <API_KEY>' \
     -X PUT 'https://api.nsone.net/v1/data/sources' \
     -d '{"sourcetype": "nsone_v1", "name": "HA_PROXY_CONNECT"}'
```

output

```
{"status": "ok", "name": "HA_PROXY_CONNECT", "feeds": [], "config": {}, "id": "760e670096f4f59dec045bed383aac5c",
"sourcetype": "nsone_v1"}
```

3. Create a new data feed for each of your load balancers:

   - Call the `/v1/data/feeds/[id]` NS1 API endpoint.
   - Set `id` in the URL to the `id` returned from the previous step.

   In the following example, we set `name` and `label` to “us-east”, but you can choose any value. NS1 uses this when it selects the data feed to monitor for the current number of connections:

   ```
   curl -sH 'X-NSONE-Key: <API_KEY>' \
        -X PUT 'https://api.nsone.net/v1/data/feeds/760e670096f4f59dec045bed383aac5c' \
        -d '{"name": "us-east", "config": {"label": "us-east"}, "destinations": []}'
   ```

4. Repeat the previous step to create a data feed for each load balancer, but change the name and label for each (for example, “us-east”, “us-west”).

5. Verify that the data feeds are listed in the NS1 Customer Portal on the **Integrations** window.

Associate DNS answers with data feeds

Add metadata to associate your A record answers with the data feeds.

1. In the NS1 Customer Portal, edit the A records for the zone.

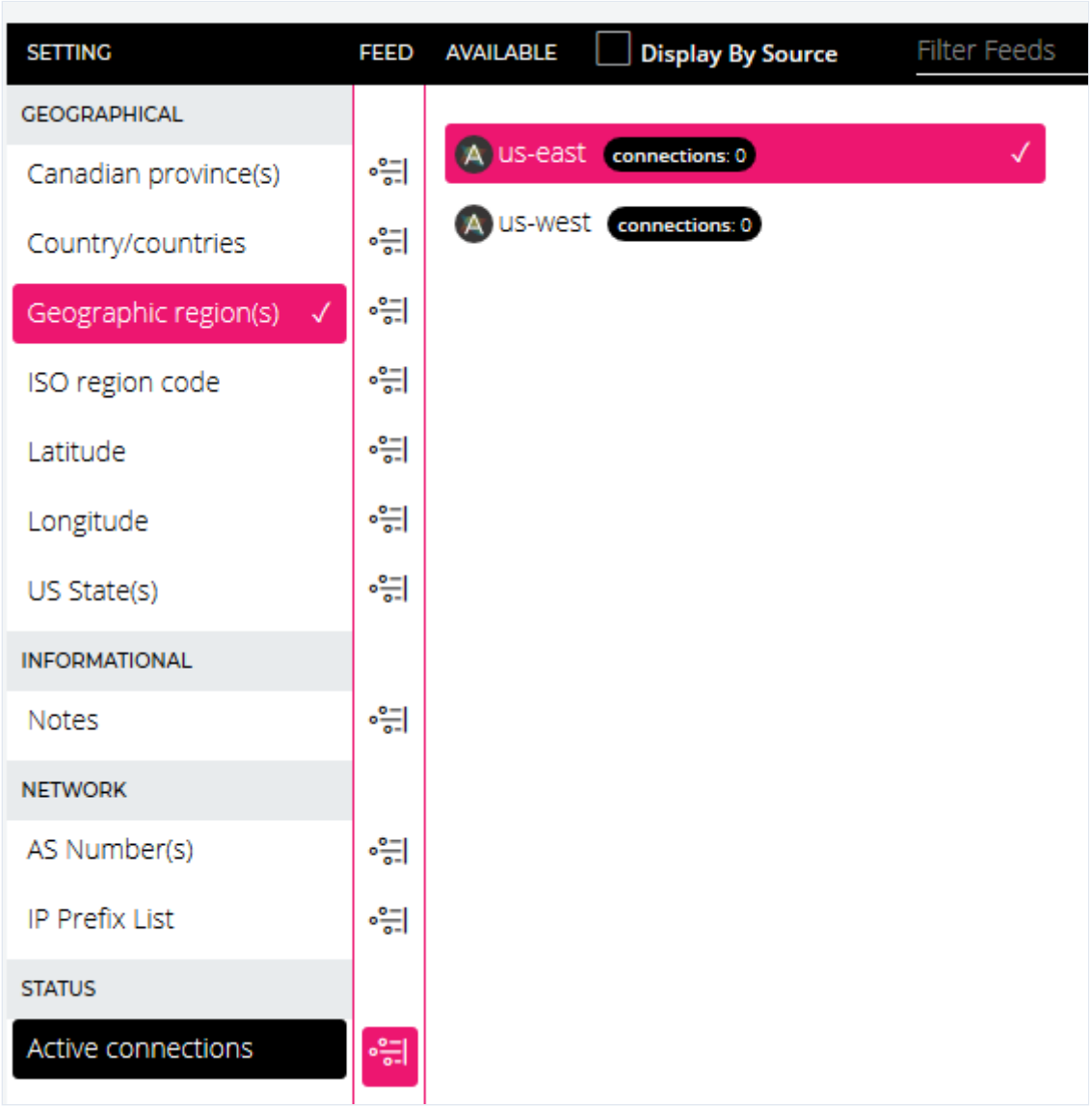
2. Click **Edit Answer Metadata** for each answer and select **Active connections**.

3. Click the **Feed**

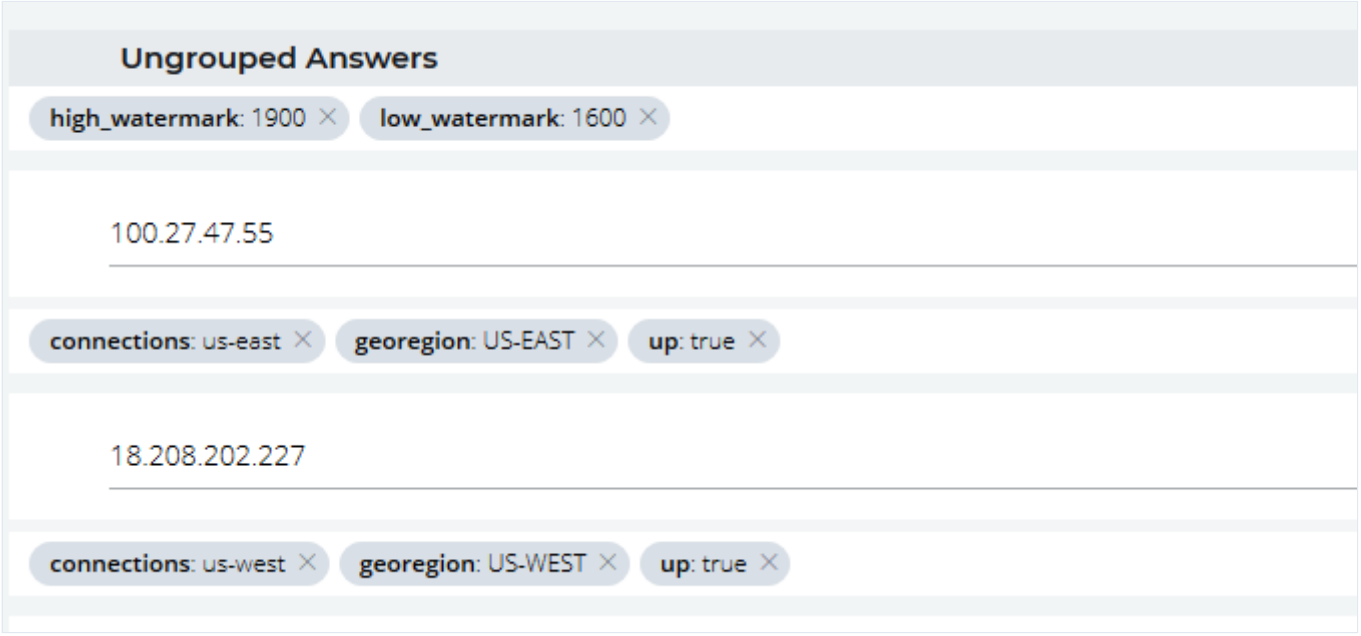
   button to toggle the source of the data to become a feed.

4. Choose the appropriate data feed for the current answer.



5. Add record metadata that applies to all answers:

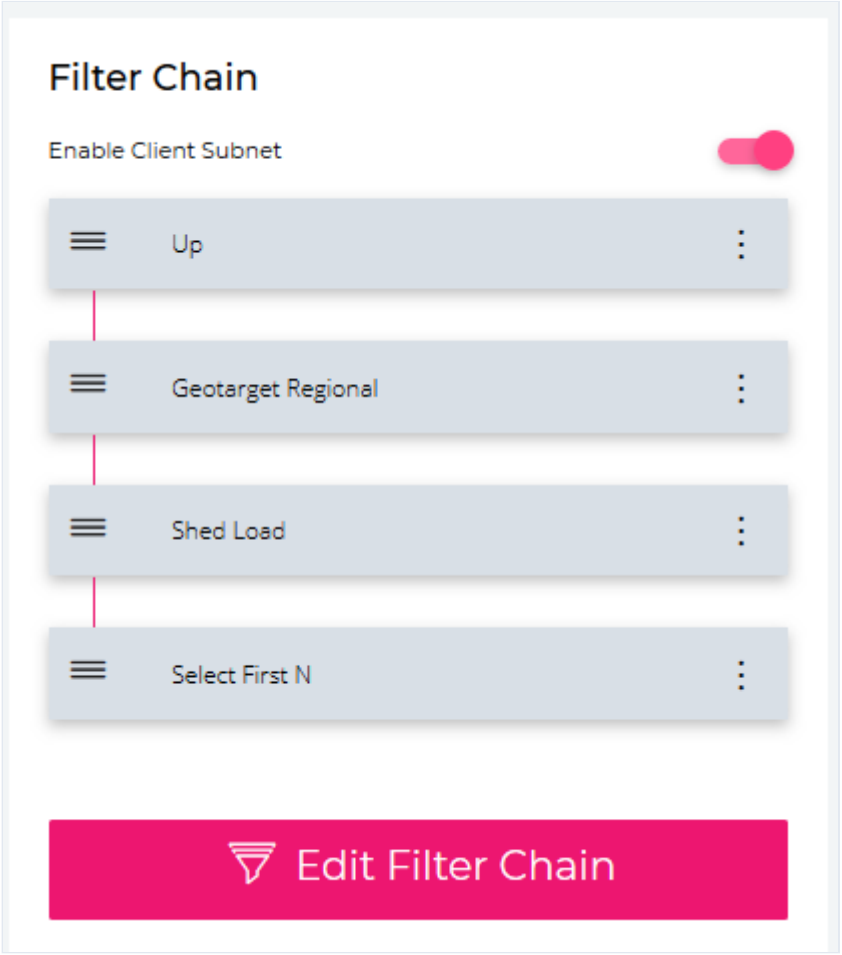
   - Specify a **Low watermark** that sets the threshold for when NS1 can begin shedding traffic away from a load balancer.
   - Specify a **High watermark** that sets the threshold for when NS1 must completely stop sending traffic to a load balancer.



Configure a filter chain

When NS1 decides which IP address to return for a DNS query, it bases its decision on a chain of filters. Each filter discards answers that fail to match some criteria. Set a filter chain to inform NS1 on how to make its decision.

1. In the NS1 Customer Portal, edit your A records and click **Edit Filter Chain**.
2. Add the following filters:
   - Up
   - Geotarget Regional
   - Shed Load
   - Select First N
3. In the **Active Filters** window, select:
   - the **Shed Load** filter and choose **connections** from the drop-down list.
   - the **Select First N** filter and set its value to `1`.
4. Save the filter chain. Verify that the filters are listed in the correct order as shown in the image below.



Configure HAProxy Enterprise to send data

Use the HAProxy Enterprise Send Metrics module to send the count of active connections for each load balancer to NS1.

1. Log into each load balancer server and install the Send Metrics module.
2. Edit the load balancer configuration file to send connection information to NS1 by adding the following to the `global` section:

```
global
  # ...
  module-path /opt/hapee-3.1/modules
  module-load hapee-lb-send-metrics.so
  send-metrics-url POST https://api.nsone.net/v1/feed/760e670096f4f59dec045bed383aac5c xdelay 1m 5s 1s 1s
timeout 100ms retries 3 log verify none
  send-metrics-header 'X-NSONE-Key: <API_KEY>'
  send-metrics-content-type application/json
  send-metrics-data '{ "us-east": { "connections": "%ac" }}'
```

3. Be sure to update the URL with your data source ID, the `X-NSONE-Key` header with your API key, and the label sent with the `send-metrics-data` line.

4. Reload the load balancer.

```
sudo systemctl reload hapee-3.1-lb
```

5. Verify that the load balancer access logs show that the Send Metrics module sent data successfully:

```
Send Metrics: metrics data successfully updated (1/0).
```

6. Repeat this procedure for each load balancer.

Licensing

HAProxy Enterprise respects licenses from each software package it embeds.

To retrieve your HAProxy Enterprise license key, see HAProxy Enterprise license key.

HAProxy Enterprise Licenses

- keepalived (hapee-extras-vrrp / hapee-extras-vrrp)

  GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or any later version

- net-snmp (hapee-extras-snmp)

  http://www.net-snmp.org/about/license.html

- socat (hapee-3.1r1-cli)

  GNU General Public License as published by the Free Software Foundation, version 2 of the License

HAProxy Enterprise (hapee-3.1r1-lb) (uses two licenses)

- All the source code is under GNU General Public License version 2
- All exportable included files are by default under GNU Lesser General Public License (LGPL) version 2.1
- For more information about HAProxy Enterprise licences, please read LICENSE file provided by haproxy.org site

HAProxy Enterprise Extensions

The code can use any license according to the LGPL mentioned above. All extensions proposed by HAProxy Technologies adhere to this proprietary license.