ALOHA Load-Balancer - Application Note

Exchange 2013 deployment guide

Document version: v1.2

Last update: 2nd June 2014

EMEA Headquarters
3, rue du petit robinson
ZAC des Metz
78350 Jouy-en-Josas
France
http://www.haproxy.com/

Purpose

ALOHA Load-Balancer deployment guide for Microsoft **Exchange 2013**.

Complexity

Versions concerned

- Aloha 5.5 and above

This guide focuses on latest **ALOHA** firmware 6.0.
To get configuration templates for older firmware version (4.2 and 5.5), please contact the support.

Microsoft Exchange 2013 versions concerned

- Microsoft Exchange 2013 SP1
- Microsoft Exchange 2013 RTM

Changelog

Version	Description
1.2	– Update followup Exchange 2013 SP1 – Update followup ALOHA 6.0 – new MAPI HTTP service – new architecture: SSL offloading
1.1	– HAProxy Tech. theme update – minor changes
1.0	Initial release

Contents

1. Introduction

1.1 About HAProxy Technologies

HAProxy Technologies is a software company, editing the **Application Delivery Controller** named **ALOHA Load-Balancer**.
Headquartered in Jouy-en-Josas (Yvelines, France), **HAProxy Tech.** teams (including R&D and technical support) are based in France. **HAProxy Tech.** has currently around 100 customers in the banking, retail groups, energy and e-commerce industries and the public sector. **HAProxy Tech.** solutions are also used by many hosting providers. The **ALOHA Load-balancer** is designed to improve performance, guarantee quality of service and ensure the availability of critical business applications, by dynamically balancing flows and queries on the company's various servers.

1.2 About ALOHA ADC

The **ALOHA Load-Balancer** is designed to load-balance at layer 4 and control application delivery at layer 7 if the OSI model. It is designed to integrate simply and quickly into any environment or architecture.
The main **ALOHA** benefits:

1. Guarantee high availability of your applications and services as well as improve web application performance
2. Makes infrastructures scalable and reliable
3. Simplifies architecture: IPv6 frontend, SSL offloading, layer 7 routing

Developed using **HAProxy** open source load balancing software, **HAProxy Tech.** solutions are known for their processing performance, reliability and wealth of features. Offered at more affordable price than other commercial solutions, they are easy to deploy and administer.

ALOHA Load-balancer is available either as a physical appliance or as a Virtual Appliance. The Virtual appliance can run on top of the most popular hypervisors available on the market.

1.3 About this guide

This guide first explains in the main lines how **Exchange 2013** is designed and what benefits an ADC can bring.
The guide also provides with configuration templates to setup the **ALOHA Load-Balancer** for Microsoft **Exchange 2013** for the two most common architectures.
The lastest version of this guide can be downloaded from **HAProxy Tech.** website: `http://www.haproxy.com/`.

1.4 Appliance supported

All **ALOHA Load-Balancers** appliances, both physical and virtual, can be used with Microsoft **Exchange 2013**.

1.5 Microsoft Exchange 2013 version supported

ALOHA load-balancer can be used with the following versions of Microsoft **Exchange 2013**:

- Microsoft exchange 2013 SP1
- Microsoft exchange 2013 RTM

1.6 Disclaimer

The **Exchange 2013** configuration tips provided in this guide are purely informational. For more information about Microsoft **Exchange 2013** tools and how to use them, please refer to Microsoft web site which is fully and properly documented.
This guide does not provide information on how to install and setup an Exchange cluster.

2. Introduction to Microsoft Exchange 2013

Exchange 2013 is Microsoft solution to provide businesses with email, calendar and contacts on the PC, phone and web browsers.

2.1 Main differences between Exchange 2010 and 2013

Exchange 2013 brings the following improvements:

1. **Client Access servers** are now session less
2. **CAS Array** has disappeared
3. **Raw TCP MAPI** protocol is not available anymore: **Outlook** clients have to use **Outlook Anywhere** (RPC over HTTPs)

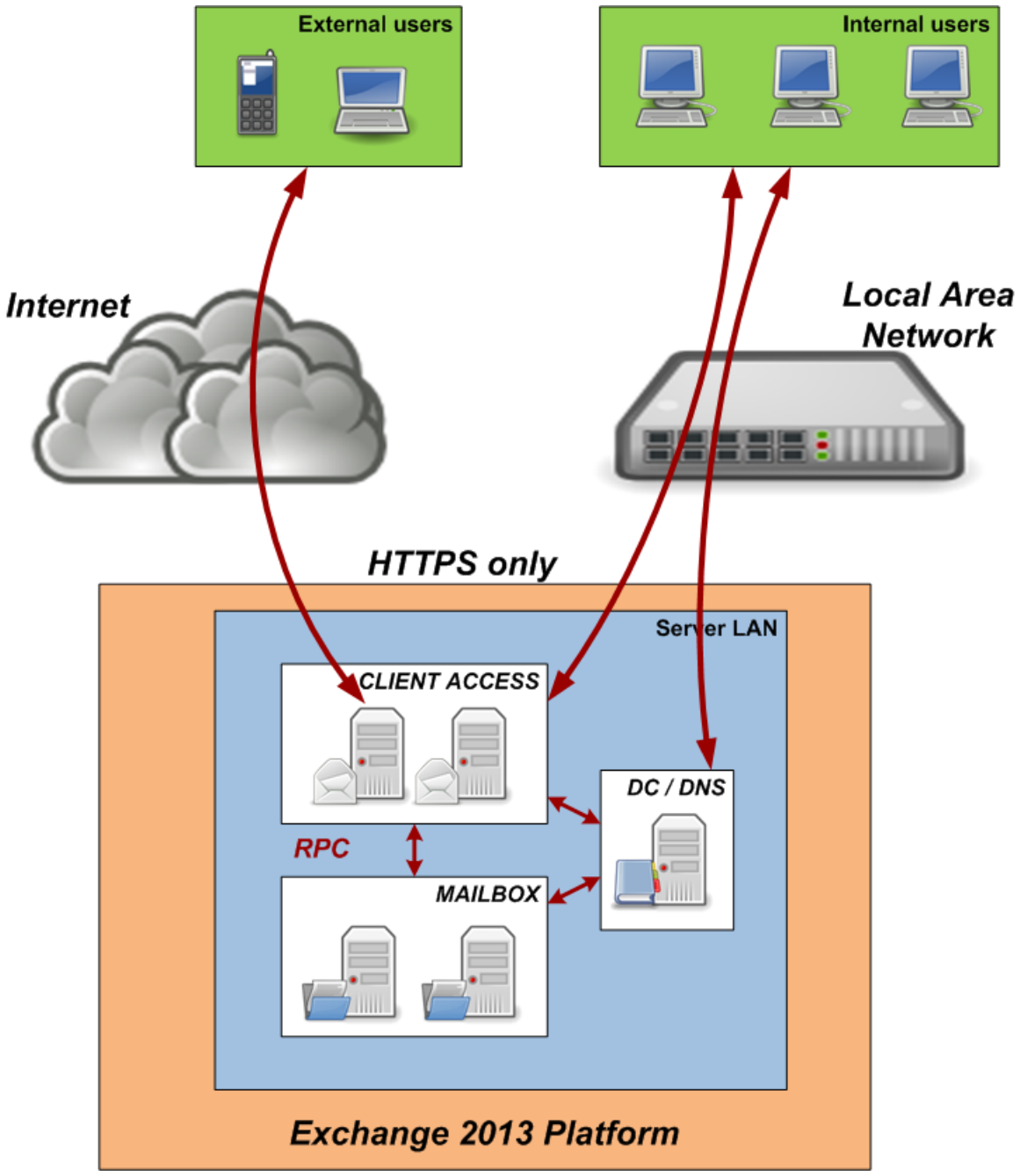
2.2 Exchange 2013 server roles

In **Exchange 2013**, servers are organised by **roles**, as described below:

Name	Description
Client Access	Frontend servers on which client gets connected to access their emails, contacts and agenda. Despite managing client connection, the **client access server** is sessionless. Often shortened as **CAS**.
Mailbox	Servers hosting mails (in mailboxes), public folders and user session information. Often shortened as **MBX**.

2.3 Exchange 2013 internal architecture

Since there are only 2 roles in **Exchange 2013**, the architecture is quite simple, as shown on the diagram below:



In Exchange 2013, **CAS** servers are pure proxies. They just forward connections from a client to the **MBX** server hosting the session. To know which **MBX** servers hosts a user's session, the **CAS** uses the **Active Directory**.

2.4 Exchange 2013 high availability

2.4.1 CAS role

In order to ensure high-availability of the **CAS** layer, you have to use a **Load-Balancer**.
As explained in introduction, there is no **CAS array** anymore in **Exchange 2013**, this is one of the positive consequence of the session-less way of working.
An other positive advantage is that Load-Balancing **CAS** servers will be very simple and straight forward.

2.4.2 Mailbox role

Mailbox servers can be configured in a **DAG** (Database Availability Group). Once in a **DAG**, the mailboxes will be highly available for the **CAS** servers, as a consequence for the end users as well.
Configuration of the **DAG** is outside the scope of this document.

DAG relies on Microsoft Clustering Services which cannot be enabled on the same server as Microsoft Network Load Balancing (NLB).

CAS servers use the **Active Directory** to know on which **Mailbox** server the user session and data are currently available.

2.5 Client access services

The table below summarizes the services hosted by **CAS** servers and the type of clients accessing them:

Service	Protocol	Clients
Autodiscover (AS)	HTTPs	Outlook, mobile devices, web browsers
Exchange ActiveSync (EAS)	HTTPs	Mobile devices
Exchange Web Services (EWS)	HTTPs	third party applications
Offline Address Book (OAB)	HTTPs	Outlook
Outlook Anywhere (OA)	HTTPs	Outlook
Messaging Application Programming Interface (MAPI)	HTTPs	Outlook (2013 SP1 and above)
Outlook Web App (OWA)	HTTPs	any web browser
Exchange Control Panel (ECP)	HTTPs	any web browser
POP	TCP	any mail user agent
IMAP	TCP	any mail user agent
SMTP	TCP	any host on the network

2.6 SMTP Load-Balancing

2.6.1 Using DNS MX entries

SMTP load-balancing can be achieved by setting up two or more DNS MX (Mail eXchanger) entries, each one pointing to an Exchange HUB server. A SMTP client would use first the MX record with the lowest preference, then try the next higher preference.

2.6.2 Using a load-balancer

A load-balancer can be used to load-balance **SMTP**. You need a single MX entry, pointing to the loadbalancer. The load-balancer would balance requests among SMTP servers configured behind it.
The current document provides different ways to load-balance **SMTP** using the **ALOHA Load-Balancer**.

Of course, you we can combine both solutions

2.7 Ports and protocols in Exchange 2013

The table below lists the services and associated ports and protocols being used in **Exchange 2013**:

TCP Port	Protocol	CAS Services
25	SMTP	SMTP
443	HTTPs	1. Autodiscover (AS) 2. Exchange ActiveSync (EAS) 3. Exchange Control Panel (ECP) 4. Exchange Web Services (EWS) 5. Offline Address Book (OAB) 6. Outlook Anywhere (OA) 7. Mapi over HTTP (MAPI) 8. Outlook Web App (OWA)
110 and 995	POP3	POP3
143 and 993	IMAP4	IMAP4

2.8 Server affinity

No **CAS** services require server affinity!

2.9 Why using a load-balancer with Exchange 2013?

First of all, even if **Exchange 2013** provides services arrays, to ensure high-availability, it does not provide any load balancing mechanism. This means a third party appliance is required to balance traffic amongst **CAS** Servers and services.

The services that can be load-balanced are the ones hosted by the **CAS** servers as well as **SMTP**.

A hardware load-balancer brings the following benefits to Microsoft **Exchange 2013** platforms:

- **Fast and Transparent failover**
  Nobody will notice a server has failed since the Load-Balancer is able to quickly and transparently redirect users to a healthy server.

- **Application aware health checking**
  A load-balancer provides application layer health check which provides the status of the service itself and are more efficient than a simple ping.
- **SSL bridging or offloading**
  A load-balancer can inspect ciphered traffic between a client and a **CAS** server to improve protection, reporting servers and URLs response time, etc...
- **No downtime during maintenance**
  The load-balancer can prevent traffic to reach a server during a maintenance window. And maintenance can now occur during business hours without generating any outage for the users.
- **Scale up**
  Building an architecture with a load-balancer allows scale up: adding new servers can be done easily with no production outage.
- **Scale out**
  Splitting services on the load-balancer side brings the ability to scale out the CAS services, dedicating servers to services.

2.10 Namespace policy: a single or multiple names?

A **Namespace** is a DNS host name which hosts one or many **Exchange 2013** services.
From a namespace point of view, **Exchange 2013** can be deployed in three modes:

1. A **single name** used to host all the services, IE: mail.domain.com.
2. **one name for each service**. IE: owa.domain.com, autodiscover.domain.com, etc...
3. A mix between the above: one name for many services, and one name for a single service

Depending on the way you configure your **Load-Balancer**, the namespace policy can have an impact.

2.10.1 A single name to host all the services

A single name means a single IP address is necessary.
If the **Load-Balancer** can't read the content of the traffic (no SSL offloading neither bridging is in use), then ALL services must follow the same configuration parameters. So only a couple of health check policy is available:

1. a basic health check is configured and the **Load-Balancer** is not able to check each service individually. So users could be forwarded to a server with a broken service
2. an advanced check is configured to ensure each service works properly. If one service fails, the server will be unavailable for all the other services as well

No problem at all if the **Load-Balancer** is configured in **SSL offloading** or **bridging**, since we can split the traffic per content and apply the right health check method for each service.

2.10.2 One name for each service

If the **Load-Balancer** can't read the content of the traffic (no SSL offloading neither bridging is in use), then it is recommended to use one IP address per service. That way, the **Load-Balancer** can probe each service individually.
No problem at all if the **Load-Balancer** is configured in **SSL offloading** or **bridging**, since we can split the traffic per content and apply the right health check method for each service.

2.10.3 Mixed

In this policy mode, then you can group services with the same type of configuration together and apply common configuration to them. You can also isolate one service on which you want to perform a specific configuration.
This mode is the most efficient since it allows mixing any type of configuration without consuming too much resources or IP addresses.

3. Load-Balancer modes

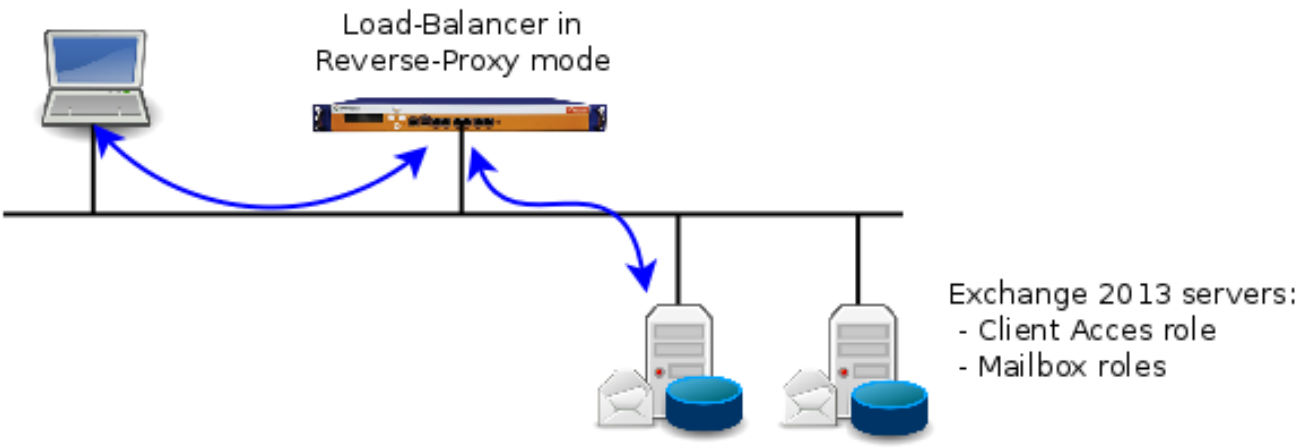
This chapter introduces the different modes a **Load-Balancer** can support, to properly spread the traffic load amongst servers.
This knowledge will be helpful when introducing **CAS** servers load-balancing, later in this document.

The **ALOHA Load-Balancer** can be used in all modes in the mean time, depending on how you want to load-balance each service individually: **HTTPs** based services, **SMTP**, **POP** and **IMAP**

3.1 Reverse proxy

The diagram below shows how things work when a **Load-balancer** is configured in **Reverse-proxy** mode:

1. The **client** establishes a connection to the **Load-Balancer**
2. The **Load-Balancer** chooses a **Client Access** server and establishes a new connection with it
3. **client** and dedicated **CAS** server discuss together through the **Load-Balancer**

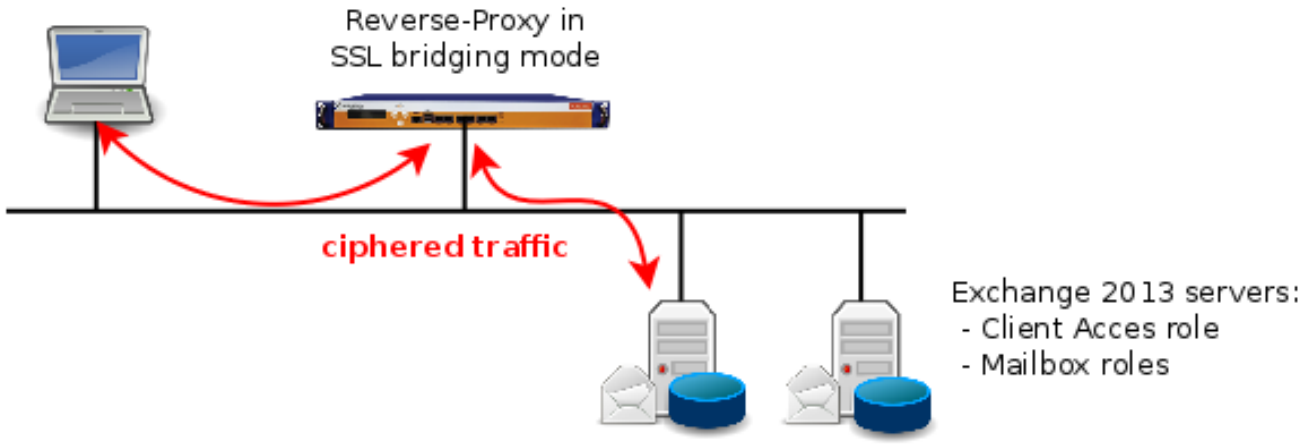


Basically, the **Load-Balancer** breaks the TCP connection between the **client** and the **CAS** server: there are 2 TCP connections established: one between the **Load-Balancer** and the **client** and an other one between the **Load-Balancer** and the **server**.
In this mode, the **Load-Balancer** uses one of its IP address to get connected on the servers. In case of the server must know the client IP address, then this mode can be turn in **transparent**: it spoof the client IP when establishing the connection on the **server**.
This mode can be deployed in a few ways:

1. raw tcp reverse proxy
2. raw tcp transparent proxy
3. HTTP reverse proxy or transparent proxy in **SSL bridging** mode
4. HTTP reverse proxy or transparent proxy in **SSL offloading** mode (Exchange 2013 SP1 and above)

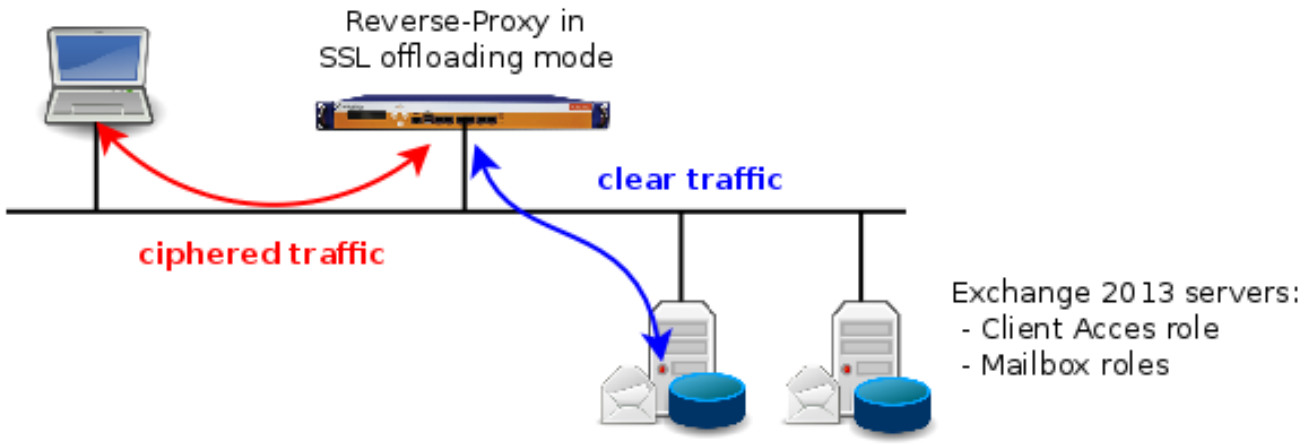
3.1.1 SSL bridging

SSL bridging means a device (here the **ALOHA Load-Balancer**) located in the middle of a ciphered stream can access the content exchanged between a **client** and a **server** by deciphering then ciphering traffic.
To achieve this, we setup the **ALOHA** to decipher the traffic from the client and to cipher the traffic to the server. That way, the **ALOHA** itself can access to the content in clear while the traffic on the network is ciphered.
The diagram below shows this type of architecture:



3.1.2 SSL offloading

SSL offloading means a device in front of application servers processes SSL traffic and gets connected in clear to the server.
The diagram below shows this type of architecture:

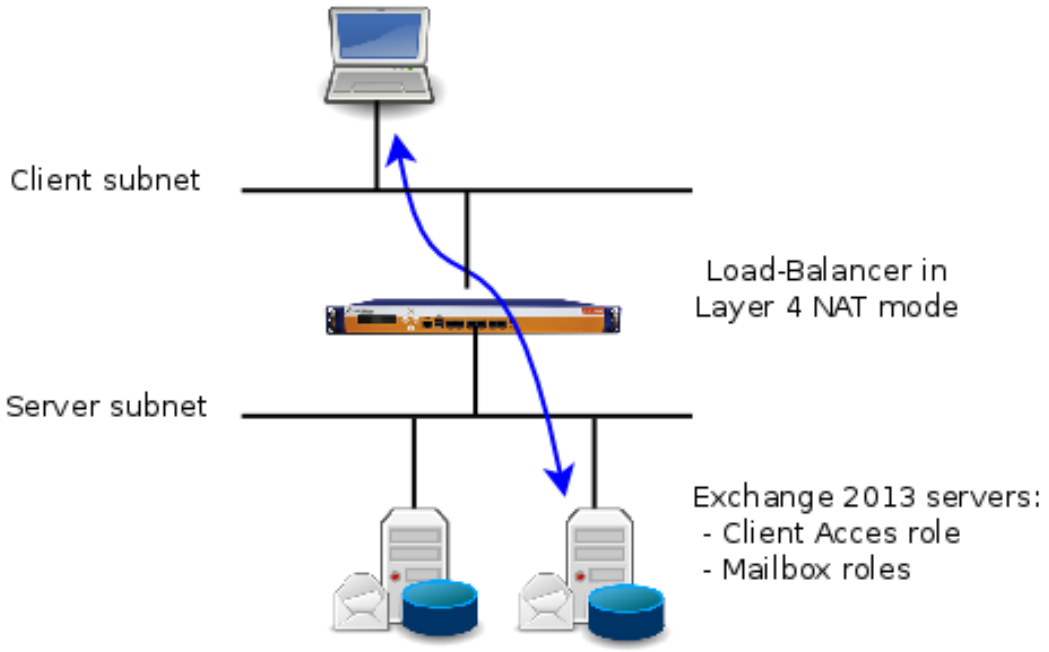


Microsoft supports SSL offloading since Exchange 2013 SP1
For this mode, some modification of the CAS server configuration must be performed

3.2 Layer 4 destination NAT

The diagram below shows how things work when a **Load-balancer** is configured in Layer 4 destination NAT mode:

1. The **client** establishes a connection to the **Client Access** server through the **Load-Balancer**
2. **Client** and **CAS** server discuss through the **Load-Balancer**



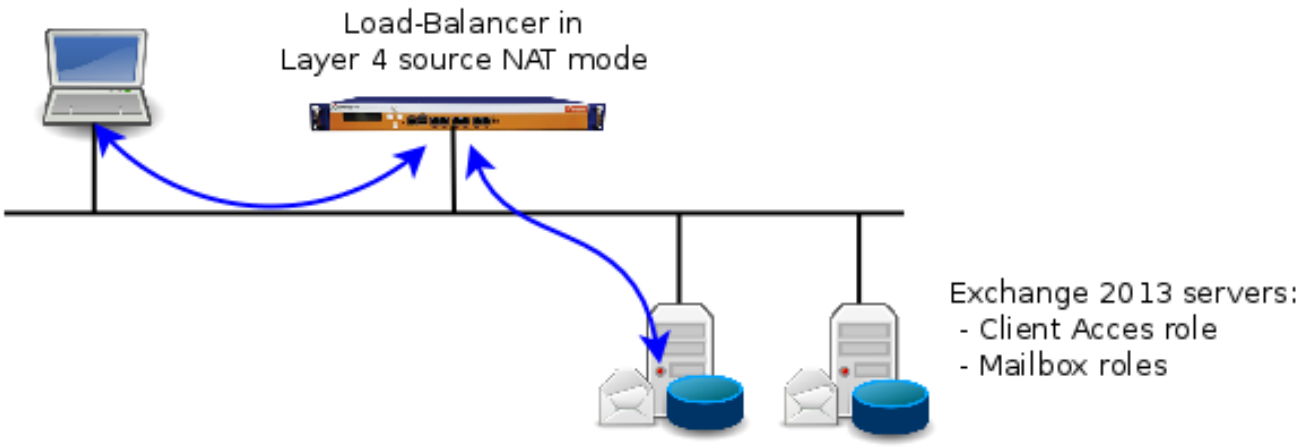
Basically, the **Load-Balancer** acts as a packet forwarder between the **client** and the **server**: a single TCP connection is established directly between the **client** and the **CAS server**, and the **Load-Balancer** just forward packets between both of them.
There a couple of drawbacks about this mode, due to the **destination** NAT:

- the client and the **server** can't be in the same subnet (otherwise the **server** will answer directly to the client, bypassing the **Load-Balancer** and the reverse NAT won't occur.
- the **server** must use the **Load-Balancer** as its default gateway

3.3 Layer 4 Source NAT

The diagram below shows how things work when a **Load-balancer** is configured in Layer 4 source NAT.

1. The **client** establishes a connection to the **Client Access server** through the **Load-Balancer**
2. When forwarding the connection to the **CAS server**, the ALOHA change the client IP address by its own IP address
3. The **Client Access server** acknowledges the connection and forward the response to the **ALOHA** IP address
4. The **ALOHA** forward the response to the client, changing IPs on the fly to match initial connection
5. **Client** and **CAS server** keep on discussing the same way

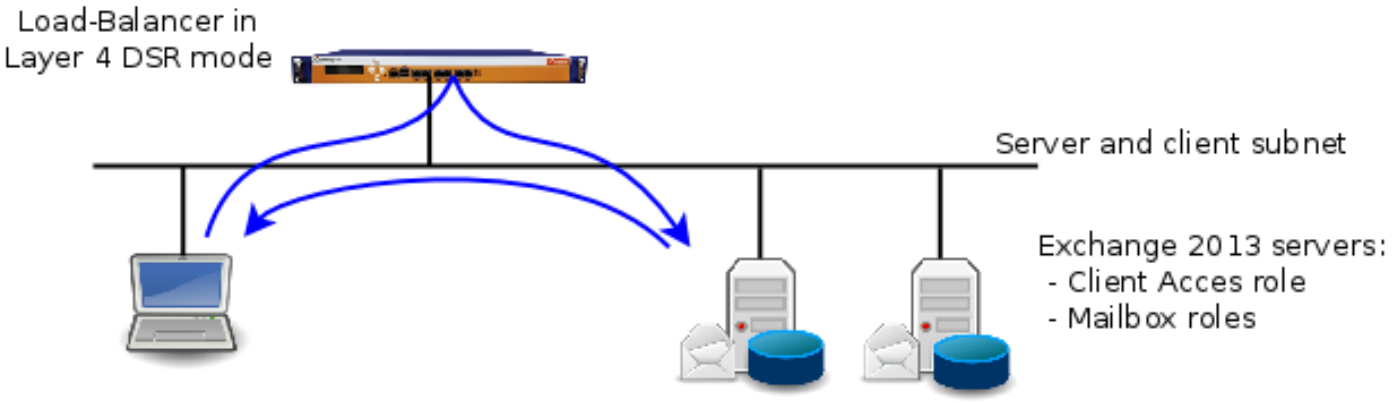


Basically, the **Load-Balancer** acts as a packet forwarder between the **client** and the **server**: a single TCP connection is established directly between the **client** and the **Client Access server**.
The **client** and the **server** can be in the same subnet.

3.4 Layer 4 Direct Server Return

The diagram below shows how things work when a **Load-balancer** is configured in Layer 4 DSR mode.

1. The **client** establishes a connection to the **Client Access server** through the **Load-Balancer**
2. The **Client Access server** acknowledges the connection directly to the **client**, bypassing the **Load-Balancer** on the way back. The **Client Access server** must have the **Load-Balancer** Virtual IP configured on a loopback interface.
3. **Client** and **CAS server** keep on discussing the same way: request through the load-balancer and answers directly from **server** to **client**.



Basically, the **Load-Balancer** acts as a packet forwarder between the **client** and the **server**: a single TCP connection is established directly between the **client** and the **Client Access server**.
The Client and the **server** can be in the same subnet, like in the diagram, or can be in two different subnet. In the second case, the **server** would use its default gateway (no need to forward the traffic back through the load-balancer).

4. Load-Balancing Exchange 2013 CAS servers

In this chapter, we'll explain the different ways of Load-Balancing the **CAS** servers, with pro and cons for each type of architecture.
When reading, keep in mind the points below:

1. **CAS** servers don't need persistence
2. All the services are delivered over HTTPS
3. **SSL Offloading** on the **CAS** servers is supported by Microsoft since Exchange 2013 SP1
4. **POP** and **IMAP** load-balancing will be described later in this document
5. **SMTP** load-balancing will be described later in this document

There is only one good architecture: the one which fits your requirements!

4.1 TCP reverse proxy

- **Load-Balancer implementation**
  - deployed in **Reverse-Proxy** mode, listening on TCP port 443.
- **Pros**
  - TCP connection is split in 2 parts, improving network protection:
    - one from **client** to **Load-Balancer**
    - one from **Load-Balancer** to **server**
  - Non intrusive implementation: the **Load-Balancer** uses its own IP address to get connected on the **server**. No route to add, no default gateway to change on servers.
  - The **Load-Balancer** can be anywhere in the infrastructure
  - **Clients** and **server**s can be in the same subnet
- **Cons**
  - the **CAS server**s don't know the**client** IP address, since it is hidden by the**Load-Balancer** one
  - the **Load-Balancer** can't access to HTTP content since its cyphered between the **client** and the **server**
  - All **CAS services** must follow the same load-balancing rules, unless you dedicate one IP address per service
  - In case of multiple namespaces, then one IP address per namespace is recommended

4.2 TCP transparent proxy

- **Load-Balancer implementation**
  - deployed in **Reverse-Proxy** mode on TCP port 443, the **Load-Balancer** spoofs the client IP address when establishing the connection on the **CAS server**
- **Pros**
  - TCP connection is split in 2 parts, improving network protection:
    - one from **client** to **Load-Balancer**
    - one from **Load-Balancer** to **server**
  - the **Load-Balancer** can be anywhere in the infrastructure
  - the **CAS servers** know the **client** IP address
- **Cons**
  - intrusive implementation: the traffic from the **server** to the **client MUST** pass through the **Load-Balancer**
  - **Client**s and **server**s can't be in the same subnet
  - the **Load-Balancer** can't access to HTTP content since it remains cyphered between the client and the server
  - All **CAS services** must follow the same load-balancing rules, unless you dedicated on IP address per service
  - In case of multiple namespaces, then one IP address per namespace is recommended

4.3 SSL bridging or offloading HTTP reverse proxy

- **Load-Balancer implementation**
  - deployed in **SSL bridging Reverse-Proxy** mode on TCP port 443, the **Load-Balancer** will act as a man-in-the-middle: traffic from **client** is decyphered, traffic to **server** is cyphered
  - deployed in **SSL offloading Reverse-Proxy** mode on TCP port 443, the **Load-Balancer** deciphers the traffic from **clients**, traffic to **servers** is sent in clear
- **Pros**
  - TCP connection is split in 2 parts, improving network protection:
    - one from **client** to **Load-Balancer**
    - one from **Load-Balancer** to **server**
  - the **Load-Balancer** can access to HTTP content and improve application protection (brute force, CSRF, etc...)
  - Each **CAS service** can have its own load-balancing rule, using a single IP address
  - non intrusive implementation: the **Load-Balancer** uses its own IP address to get connected on the **server**
  - the **Load-Balancer** can be anywhere in the infrastructure
  - **Client**s and **server**s can be in the same subnet
  - In case of multiple namespaces, then a single IP address is necessary
- **Cons**
  - the **CAS server**s don't know the **client** IP address, since its hidden by the **Load-Balancer** one
  - require **Load-Balancers** with a huge SSL capacity (furthermore in **bridging** mode)

4.4 SSL bridging or offloading HTTP transparent proxy

- **Load-Balancer implementation**
  - deployed in **SSL bridging Reverse-Proxy** mode on TCP port 443, the **Load-Balancer** will act as a man-in-the-middle: traffic from **client** is decyphered, traffic to **server** is cyphered
  - deployed in **SSL offloading Reverse-Proxy** mode on TCP port 443, the **Load-Balancer** deciphers the traffic from **clients**, traffic to **servers** is sent in clear
  - In both cases, when establishing the connection on the **CAS server**, the **Load-Balancer** spoofs the **client** IP address.
- **Pros**
  - TCP connection is split in 2 parts, improving network protection:
    - one from **client** to **Load-Balancer**
    - one from **Load-Balancer** to **server**
  - the **Load-Balancer** can access to HTTP content and improve application protection (brute force, CSRF, etc...)
  - the **CAS** servers know the **client** IP address
  - Each **CAS service** can have its own load-balancing rule, using a single IP address
  - the **Load-Balancer** can be anywhere in the infrastructure
  - In case of multiple namespaces, then a single IP address is necessary
- **Cons**
  - intrusive implementation: the traffic from the server to the client **MUST** pass through the **Load-Balancer**
  - Clients and servers can't be in the same subnet
  - require **Load-Balancers** with a huge SSL processing capacity (furthermore in **bridging** mode)

4.5 Layer4 source NAT

- **Load-Balancer implementation**
  - deployed in **packet forwarder** mode on TCP port 443, the **Load-Balancer** will just forward IP packets between **clients** and **CAS servers**.
    When forwarding the packets to the **CAS servers**, the **Load-Balancer** will spoof the **client** IP address.
- **Pros**
  - the **Load-Balancer** can be anywhere in the infrastructure
  - **Client**s and **servers** can be in the same subnet
  - Low capacity **Load-Balancers** can load-balance thousands of users
  - non-intrusive implementation
- **Cons**
  - the **Load-Balancer** can't access to HTTP content
  - the **CAS servers** don't know the **client** IP address
  - All **CAS services** must follow the same load-balancing rules unless you dedicate one IP per service
  - TCP connection is established between the **client** and the **server** directly, it means the **CAS server** IP stack is exposed to **clients**
  - In case of multiple namespaces, then one IP address per namespace is recommended

This mode is only available in **ALOHA** v6.0 and above.

4.6 Layer4 destination NAT

- **Load-Balancer implementation**
  - deployed in **Forwarder** mode on TCP port 443, the **Load-Balancer** will just forward IP packets between **clients** and **CAS servers**.
    When forwarding the packets to the **CAS servers**, the **Load-Balancer** will change the destination IP address from the Service IP to the chosen **server** IP.
- **Pros**
  - the **Load-Balancer** can be anywhere in the infrastructure
  - Low capacity **Load-Balancers** can load-balance thousands of users
  - **CAS servers** know the **client** IP address
- **Cons**
  - **Client**s and **servers** can't be in the same subnet
  - the **Load-Balancer** can't access to HTTP content
  - All **CAS services** must follow the same load-balancing rules
  - TCP connection is established between the **client** and the **server** directly, it means the **CAS server** IP stack is exposed to **clients**
  - intrusive implementation: the traffic from the **server** to the **client MUST** pass through the **Load-Balancer**
  - In case of multiple namespaces, then one IP address per namespace is recommended

4.7 Layer4 Direct Server Return

- **Load-Balancer implementation**
  - deployed in **packet forwarder** mode on TCP port 443, the **Load-Balancer** will just turn the destination MAC address of packets to the chosen **CAS servers** MAC address.

- **Pros**
  - **CAS servers** know the **client** IP address
  - **Client**s and **servers** can be in the same subnet
  - Low capacity **Load-Balancers** can load-balance thousands of users
- **Cons**
  - the **Load-Balancer** and the **CAS servers MUST** be in the same **vlan**
  - the **Load-Balancer** can't access to HTTP content
  - All **CAS services** must follow the same load-balancing rules, unless you dedicate one IP address per service
  - TCP connection is established between the **client** and the **server** directly, it means the **CAS server** IP stack is exposed to **clients**
  - intrusive implementation: the service IP must be configured on a Loopback on the **CAS server**
  - In case of multiple namespaces, then one IP address per namespace is recommended

4.8 Architecture summary table

The table below summarizes the different type of architectures with their pros and cons:

Architecture	Intrusive mode *	Source IP on CAS servers	MAX number of users **	Protection	Namespace
SSL bridging HTTP reverse proxy	no	Load-Balancer IP	3K	Network and Application	no limitation
SSL bridging HTTP transparent proxy	yes	Client IP	3K	Network and Application	no limitation
SSL offloading HTTP reverse proxy	no	Load-Balancer IP	10K	Network and Application	no limitation
SSL offloading HTTP transparent proxy	yes	Client IP	10K	Network and Application	no limitation
raw TCP reverse proxy	no	Load-Balancer IP	20K	Network	multiple namespace and IPs recommended
raw TCP transparent proxy	yes	client IP	20K	Network	multiple namespace and IPs recommended
Layer 4 source NAT	no	Load-Balancer IP	200K	no	multiple namespace and IPs recommended
Layer 4 destination NAT	yes	Client IP	500K	no	multiple namespace and IPs recommended
Layer 4 Direct Server Return	yes	Client IP	1M	no	multiple namespace and IPs recommended

* Will there by any changes to perform in an existing platform
** with the biggest **ALOHA**

5. ALOHA configuration

In this chapter, we provide **ALOHA** configuration templates for each type of architecture.
We consider you're already connected on the GUI and the service VRRP address for **Exchange 2013** is already setup.
To configure a new VRRP service IP, please refer to **HAProxy Tech.** documentation: **AN-0002-EN - Implementing High Availability via VRRP**.

5.1 HTTPS based services

5.1.1 raw TCP reverse proxy

Go on the LB Layer 7 tab, then add the configuration below:

```
frontend ft_exchange_2013
  mode tcp
  bind 192.168.13.4:443 name https
  log global
  option tcplog
  option dontlognull
  option contstats
  timeout client 300s
  maxconn 10000
  default_backend bk_exchange_2013

backend bk_exchange_2013
  mode tcp
  balance leastconn
  option tcplog
  log global
  option redispatch
  retries 3
  timeout server 300s
  timeout connect 5s
  default-server inter 3s rise 2 fall 3
  server 2013exchange1 192.168.13.21:443 check
  server 2013exchange2 192.168.13.22:443 check
```

Don't forget to update the following information:

- **bind**'s IP address
- **servers** IP addresses
- **maxconn** from the frontend section

This configuration does a basic TCP check. For an advanced TCP check read the section **"Advanced check for raw TCP modes"**

5.1.2 raw TCP transparent proxy

Go on the **LB Layer 7** tab, then add the same configuration as **raw TCP reverse proxy** and add the line below in your **backend** description:

```
source 0.0.0.0 usesrc clientip
```

Don't forget to update the following information:

- **bind**'s IP address
- **servers** IP addresses
- **maxconn** from the frontend section

This configuration does a basic TCP check. For an advanced TCP check read the section **"Advanced check for raw TCP modes"**

5.1.3 Advanced health check for raw TCP modes

As explained before in this document, the issue of TCP mode is that you can't split traffic by content, so you must forward all traffic to a single farm.
This mode has a couple of drawbacks:

- you don't know if a service is down because your health check is not application aware
- you must use one IP per service, hence one farm per service where you perform the relative application check

There is a third way: a single farm where you perform all the required checks to consider a server up and ready for production. The configuration below explain the **tcp-check** sequence for this purpose. It checks all the HTTPs based services and if any of them fails, then the server will be considered as nonoperational.
Just copy the content below in your **backend** section and remove the checks you don't need, if any:

```
 option tcp-check
 tcp-check connect port 443 ssl
# activesync
 tcp-check send GET\ /Microsoft-Server-ActiveSync/HealthCheck.htm\ HTTP/1.1\r\n
 tcp-check send Host:\ mail.2013.haproxylab.net\r\n
 tcp-check send \r\n
 tcp-check expect string 200\ OK
# autodiscover
 tcp-check send GET\ /Autodiscover/HealthCheck.htm\ HTTP/1.1\r\n
 tcp-check send Host:\ mail.2013.haproxylab.net\r\n
 tcp-check send \r\n
 tcp-check expect string 200\ OK
# exchange control panel
 tcp-check send GET\ /ECP/HealthCheck.htm\ HTTP/1.1\r\n
 tcp-check send Host:\ mail.2013.haproxylab.net\r\n
 tcp-check send \r\n
 tcp-check expect string 200\ OK
# exchange web services
 tcp-check send GET\ /EWS/HealthCheck.htm\ HTTP/1.1\r\n
 tcp-check send Host:\ mail.2013.haproxylab.net\r\n
 tcp-check send \r\n
 tcp-check expect string 200\ OK
# mapi
 tcp-check send GET\ /mapi/HealthCheck.htm\ HTTP/1.1\r\n
 tcp-check send Host:\ mail.2013.haproxylab.net\r\n
 tcp-check send \r\n
 tcp-check expect string 200\ OK
# offline address book
 tcp-check send GET\ /OAB/HealthCheck.htm\ HTTP/1.1\r\n
 tcp-check send Host:\ mail.2013.haproxylab.net\r\n
 tcp-check send \r\n
 tcp-check expect string 200\ OK
# rpc over http
 tcp-check send GET\ /RPC/HealthCheck.htm\ HTTP/1.1\r\n
 tcp-check send Host:\ mail.2013.haproxylab.net\r\n
 tcp-check send \r\n
 tcp-check expect string 200\ OK
# outlook web app
 tcp-check send GET\ /owa/HealthCheck.htm\ HTTP/1.1\r\n
 tcp-check send Host:\ mail.2013.haproxylab.net\r\n
 tcp-check send \r\n
 tcp-check expect string 200\ OK
```

5.1.4 SSL bridging HTTP reverse proxy - simple configuration

Go on the **LB Layer 7** tab, then add the configuration below:

```
frontend ft_exchange_2013
  bind 192.168.13.4:80 name http
  bind 192.168.13.4:443 name https ssl crt my_certificate_name
  mode http
  option http-keep-alive
  no option httpclose
  no option http-server-close
  no option forceclose
  option contstats
  option dontlognull
  log    global
  option httplog
  timeout client 25s
  timeout http-keep-alive 1s
  timeout http-request 15s
  maxconn 1000
  acl ssl_connection ssl_fc
  acl host_mail    hdr(Host) -i mail.mydomain.com
  acl path_slash   path        /
  http-request redirect scheme https code 302 unless ssl_connection
  http-request redirect location /owa/ code 302 if path_slash host_mail
  default_backend bk_exchange_2013

backend bk_exchange_2013
  balance roundrobin
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  log global
  option httplog
  option forwardfor
  option redispatch
  retries 3
  timeout server 25s
  timeout connect 5s
  timeout queue 30s
  default-server inter 3s rise 2 fall 3
  server 2013xchange1 192.168.13.21:443 maxconn 1000 weight 10 ssl check
  server 2013xchange2 192.168.13.22:443 maxconn 1000 weight 10 ssl check
```

Don't forget to update the following information:

- **bind**'s IP address and SSL certificate name
- **acl host_mail**'s domain name list
- **servers** IP addresses and **maxconn** parameter
- **maxconn** from the frontend section

You can use the **Advanced health check for raw TCP modes** to improve this configuration

5.1.5 SSL bridging HTTP reverse proxy - advanced configuration

Thanks to **ALOHA** powerful Layer 7 features, we can split the traffic per Exchange service and dedicate one **backend** per service.
Go on the **LB Layer 7** tab, then add the configuration below:

Frontend configuration:
This **frontend** configuration is the entry point which will dispatch traffic to different backends, one per **Exchange 2013** service, based on URL path:

```
frontend ft_XCHANGE2013_https
  bind 192.168.13.4:80 name INT_http
  bind 192.168.13.4:443 name INT_https ssl crt 2013.haproxylab.net
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  timeout client 600s
  log global
  capture request header Host len 32
  capture request header User-Agent len 64
  capture response header Content-Length len 10
  # log-format directive must br written on a single line
  # it is splitted for documentation convnience
  log-format %ci:%cp\ [%t]\ %ft\ %b/%s\ %Tq/%Tw/%Tc/%Tr/%Tt\ %ST\ %B\ %CC\ %CS\ %tsc\ %ac/%fc/%bc/%sc/%rc
\ %sq/%bq\ %hr\ %hs\ {%sslv/%sslc/%[ssl_fc_sni]/%[ssl_fc_session_id]}\
"%[capture.req.method]\ %[capture.req.hdr(0)]%[capture.req.uri]\ HTTP/1.1"
  maxconn 1000
  acl ssl_connection ssl_fc
  acl host_mail hdr(Host) -i mail.2013.haproxylab.net
  acl path_slash path /
  acl path_autodiscover path_beg -i /Autodiscover/Autodiscover.xml
  acl path_activesync path_beg -i /Microsoft-Server-ActiveSync
  acl path_ews path_beg -i /ews/
  acl path_owa path_beg -i /owa/
  acl path_oa path_beg -i /rpc/rpcproxy.dll
  acl path_ecp path_beg -i /ecp/
  acl path_oab path_beg -i /oab/
  acl path_mapi path_beg -i /mapi/
  acl path_check path_end -i HealthCheck.htm

  # HTTP deny rules
  http-request deny if path_check

  # HTTP redirect rules
  http-request redirect scheme https code 302 unless ssl_connection
  http-request redirect location /owa/ code 302 if path_slash host_mail

  # HTTP routing rules
  use_backend bk_XCHANGE2013_https_autodiscover if path_autodiscover
  use_backend bk_XCHANGE2013_https_activesync if path_activesync
  use_backend bk_XCHANGE2013_https_ews if path_ews
  use_backend bk_XCHANGE2013_https_owa if path_owa
  use_backend bk_XCHANGE2013_https_oa if path_oa
  use_backend bk_XCHANGE2013_https_ecp if path_ecp
  use_backend bk_XCHANGE2013_https_oab if path_oab
  use_backend bk_XCHANGE2013_https_mapi if path_mapi
  # other services go here
  default_backend bk_XCHANGE2013_https_default
```

Don't forget to update the following information:

- **bind**'s IP address and SSL certificate name
- **acl host_mail**'s domain name list
- **maxconn** from the frontend section

Autodiscover configuration:

```
backend bk_XCHANGE2013_https_autodiscover
  balance roundrobin
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  mode http
  log global
  option httplog
  option forwardfor
  option httpchk GET /Autodiscover/HealthCheck.htm
  http-check expect string 200\ OK
  default-server inter 3s rise 2 fall 3
  timeout server 60s
  server 2013xchange1 192.168.13.21:443 ssl maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:443 ssl maxconn 1000 weight 10 check
```

Don't forget to update the following information:

- **servers** IP addresses

Activesync configuration:

```
backend bk_XCHANGE2013_https_activesync
  balance roundrobin
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  log global
  option httplog
  option forwardfor
  option httpchk GET /Microsoft-Server-ActiveSync/HealthCheck.htm
  http-check expect string 200\ OK
  default-server inter 3s rise 2 fall 3
  timeout server 60s
  server 2013xchange1 192.168.13.21:443 ssl maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:443 ssl maxconn 1000 weight 10 check
```

Don't forget to update the following information:

- **servers** IP addresses and **maxconn** parameter

Exchange Control Panel (ECP) configuration:

```
backend bk_XCHANGE2013_https_ecp
  balance roundrobin
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  log global
  option httplog
  option forwardfor
  option httpchk GET /ECP/HealthCheck.htm
  http-check expect string 200\ OK
  default-server inter 3s rise 2 fall 3
  timeout server 60s
  server 2013xchange1 192.168.13.21:443 ssl maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:443 ssl maxconn 1000 weight 10 check
```

Don't forget to update the following information:

- **servers** IP addresses

Exchange Web Services (EWS) configuration:

```
backend bk_XCHANGE2013_https_ews
  balance roundrobin
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  log global
  option httplog
  option forwardfor
  option httpchk GET /EWS/HealthCheck.htm
  http-check expect string 200\ OK
  default-server inter 3s rise 2 fall 3
  timeout server 60s
  server 2013xchange1 192.168.13.21:443 ssl maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:443 ssl maxconn 1000 weight 10 check
```

Don't forget to update the following information:

- **servers** IP addresses

MAPI configuration:

```
backend bk_XCHANGE2013_https_mapi
  balance roundrobin
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  log global
  option httplog
  option forwardfor
  option httpchk GET /mapi/HealthCheck.htm
  http-check expect string 200\ OK
  default-server inter 3s rise 2 fall 3
  timeout server 600s
  server 2013xchange1 192.168.13.21:443 ssl maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:443 ssl maxconn 1000 weight 10 check
```

Don't forget to update the following information:

- **servers** IP addresses and **maxconn** parameter

Offline Address Book (OAB) configuration:

```
backend bk_XCHANGE2013_https_oab
  balance roundrobin
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  log global
  option httplog
  option forwardfor
  option httpchk GET /OAB/HealthCheck.htm
  http-check expect string 200\ OK
  default-server inter 3s rise 2 fall 3
  timeout server 60s
  server 2013xchange1 192.168.13.21:443 ssl maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:443 ssl maxconn 1000 weight 10 check
```

Don't forget to update the following information:

- **servers** IP addresses

Outlook Anywhere configuration:

```
backend bk_XCHANGE2013_https_oa
  balance roundrobin
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  log global
  option httplog
  option forwardfor
  option httpchk GET /RPC/HealthCheck.htm
  http-check expect string 200\ OK
  default-server inter 3s rise 2 fall 3
  timeout server 600s
  server 2013xchange1 192.168.13.21:443 ssl maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:443 ssl maxconn 1000 weight 10 check
```

Don't forget to update the following information:

- **servers** IP addresses and **maxconn** parameter

OWA (Outlook Web App) configuration:

```
backend bk_XCHANGE2013_https_owa
  balance roundrobin
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  log global
  option httplog
  option forwardfor
  option httpchk GET /owa/HealthCheck.htm
  http-check expect string 200\ OK
  default-server inter 3s rise 2 fall 3
  timeout server 60s
  server 2013xchange1 192.168.13.21:443 ssl maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:443 ssl maxconn 1000 weight 10 check
```

Don't forget to update the following information:

- **servers** IP addresses

Other services configuration:

```
backend bk_XCHANGE2013_https_default
  balance roundrobin
  mode http
  option http-keep-alive
  option prefer-last-server
  no option httpclose
  no option http-server-close
  no option forceclose
  no option http-tunnel
  log global
  option httplog
  option forwardfor
  default-server inter 3s rise 2 fall 3
  timeout server 60s
  server 2013xchange1 192.168.13.21:443 ssl maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:443 ssl maxconn 1000 weight 10 check
```

Don't forget to update the following information:

- **servers** IP addresses

5.1.6 SSL bridging HTTP reverse proxy - advanced configuration with application layer protection

Advanced application layer protection is outside the scope of this configuration.
Don't hesitate to contact **HAProxy Tech.** teams for such configuration.

5.1.7 SSL bridging HTTP transparent proxy - Simple and Advanced configuration

The configuration is exactly the same as in the previous section, just add the line below to each backend description to turn on transparent proxy mode:

```
source 0.0.0.0 usesrc clientip
```

Don't forget to update the following information:

- **bind**'s IP address and SSL certificate name
- **acl host_mail**'s domain name list
- **servers** IP addresses

5.1.8 SSL offloading HTTP reverse proxy

SSL offloading configuration is exactly the same as **SSL bridging**. The only thing which changes is the way the **ALOHA** gets connected to the servers. To configure the **ALOHA** in **SSL offloading**, just replace the server **port** and remove the **ssl** keyword from the server line description in the template provided in the **SSL bridging** sections.
In example:

```
server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

5.1.9 Layer4 destination NAT

Go on the **LB Layer 4** tab, then add the configuration below:

```
director exchange 10.0.1.9:443 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:443 weight 10 check
  server 2013exchange2 192.168.13.22:443 weight 10 check
```

Don't forget to update the following information:

- **director**'s IP address
- **servers** IP addresses

5.1.10 Layer4 source NAT

Go on the **LB Layer 4** tab, then add the configuration below:

```
director exchange 10.0.1.9:443 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:443 weight 10 check
  server 2013exchange2 192.168.13.22:443 weight 10 check
```

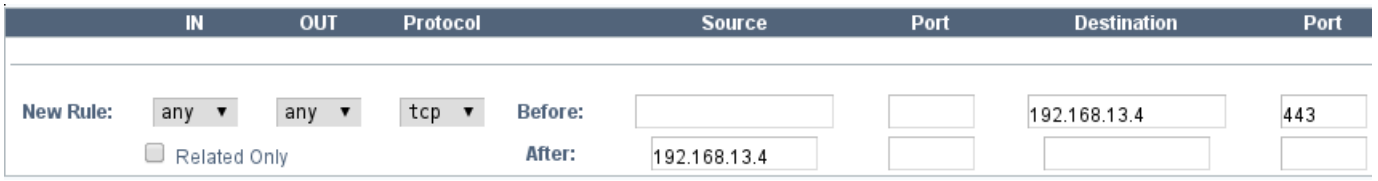
Don't forget to update the following information:

- **director**'s IP address
- **servers** IP addresses

Then, go on the **NAT** tab, and create a new rule with:

- **Protocol**: TCP
- **Source / After / IP**: the Virtual IP to NAT to (in the server LAN)
- **Destination / Before / IP**: the Virtual IP exposed to users
- **Destination / Before / port**: the port of the service exposed to users
- let the other options to default value (any)

In example, clients browse **Exchange 2013** services on 192.168.13.4:443, and we want to re-use this IP address to get connected on the server.
We have to create the following rule:



Apply this confirguration, then click on the **Apply** button.

5.1.11 Layer4 Direct Server Return

Go on the **LB Layer 4** tab, then add the configuration below:

```
director exchange 192.168.13.4:443 TCP
  balance leastconn
  mode gateway
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:443 weight 10 check
  server 2013exchange2 192.168.13.22:443 weight 10 check
```

Don't forget to update the following information:

- **director**'s IP address
- **servers** IP addresses

5.2 SMTP Load-Balancing

When Load-Balancing **SMTP** services, it is important to know the client IP address to limit access to **SMTP** relays and to improve SPAM protection.
The table below lists the different Load-Balancer deployment modes recommended for **SMTP**:

Architecture	Intrusive mode	Source IP on SMTP servers	Protection
raw TCP transparent proxy	yes	client IP	Network
Layer 4 destination NAT	yes	Client IP	no
Layer 4 Direct Server Return	yes	Client IP	no

Other deployment modes can be configured but will hide the client IP address to the **SMTP** server

5.2.1 raw TCP transparent proxy

Go on the **LB Layer 7** tab, then add the configuration below:

```
frontend ft_exchange_2013_smtp
  mode tcp
  bind 192.168.13.4:25 name smtp
  log global
  option tcplog
  option dontlognull
  option contstats
  timeout client 300s
  maxconn 10000
  default_backend bk_exchange_2013_smtp

backend bk_exchange_2013_smtp
  mode tcp
  balance leastconn
  option tcplog
  log global
  option redispatch
  retries 3
  timeout server 300s
  timeout connect 5s
  source 0.0.0.0 usesrc clientip
  option tcp-check
  tcp-check expect string 220\
  default-server inter 3s rise 2 fall 3
  server 2013exchange1 192.168.13.21:25 check
  server 2013exchange2 192.168.13.22:25 check
```

Don't forget to update the following information:

- **bind**'s IP address
- **servers** IP addresses

5.2.2 Layer 4 Destination NAT

Go on the **LB Layer 4** tab, then add the configuration below:

```
director exchange 192.168.13.4:25 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:25 weight 10 check
  server 2013exchange2 192.168.13.22:25 weight 10 check
```

Don't forget to update the following information:

- **director**'s IP address
- **servers** IP addresses

5.2.3 Layer 4 Direct Server Return

Go on the **LB Layer 4** tab, then add the configuration below:

```
director exchange 192.168.13.4:25 TCP
  balance leastconn
  mode gateway
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:25 weight 10 check
  server 2013exchange2 192.168.13.22:25 weight 10 check
```

Don't forget to update the following information:

- **director**'s IP address
- **servers** IP addresses

5.3 POP and IMAP Load-Balancing

In some deployment cases, it could be required to load-balance **POP** and **IMAP** services.
The table below summarizes the recommended load-balancing modes for **POP** and **IMAP**:

Architecture	Intrusive mode	Source IP on CAS servers	Protection
raw TCP reverse proxy	no	Load-Balancer IP	Network
raw TCP transparent proxy	yes	client IP	Network
Layer 4 full NAT	no	Load-Balancer IP	no
Layer 4 destination NAT	yes	Client IP	no
Layer 4 Direct Server Return	yes	Client IP	no

5.3.1 raw TCP reverse proxy

Go on the **LB Layer 7** tab, then add the configuration below:

```
frontend ft_exchange_2013_pop
  mode tcp
  bind 192.168.13.4:110 name pop
  bind 192.168.13.4:995 name pops
  log global
  option tcplog
  option dontlognull
  option contstats
  timeout client 300s
  maxconn 10000
  default_backend bk_exchange_2013_pop

backend bk_exchange_2013_pop
  mode tcp
  balance leastconn
  option tcplog
  log global
  option redispatch
  retries 3
  timeout server 300s
  timeout connect 5s
  option tcp-check
  tcp-check connect port 110
  tcp-check expect string +OK
  tcp-check connect port 995 ssl
  tcp-check expect string +OK
  default-server inter 3s rise 2 fall 3
  server 2013exchange1 192.168.13.21 check
  server 2013exchange2 192.168.13.22 check

frontend ft_exchange_2013_imap
  mode tcp
  bind 192.168.13.4:143 name imap
  bind 192.168.13.4:993 name imaps
  log global
  option tcplog
  option dontlognull
  option contstats
  timeout client 300s
  maxconn 10000
  default_backend bk_exchange_2013_imap

backend bk_exchange_2013_imap
  mode tcp
  balance leastconn
  option tcplog
  log global
  option redispatch
  retries 3
  timeout server 300s
  timeout connect 5s
  option tcp-check
  tcp-check connect port 143
  tcp-check expect string *\ OK
  tcp-check connect port 993 ssl
  tcp-check expect string *\ OK
  default-server inter 3s rise 2 fall 3
  server 2013exchange1 192.168.13.21 check
  server 2013exchange2 192.168.13.22 check
```

Don't forget to update the following information:

- **bind**'s IP address
- **servers** IP addresses

5.3.2 raw TCP transparent proxy

Go on the **LB Layer 7** tab, use the configuration from the section above "**raw TCP reverse proxy**" and add the following line in the **backend** section:

```
  source 0.0.0.0 usesrc clientip
```

Don't forget to update the following information:

- **bind**'s IP address
- **servers** IP addresses

5.3.3 Layer 4 Destination NAT

Go on the **LB Layer 4** tab, then add the configuration below:

```
director exchange_2013_pop 192.168.13.4:110 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:110 weight 10 check
  server 2013exchange2 192.168.13.22:110 weight 10 check

director exchange_2013_pops 192.168.13.4:995 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:995 weight 10 check
  server 2013exchange2 192.168.13.22:995 weight 10 check

director exchange_2013_imap 192.168.13.4:143 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:143 weight 10 check
  server 2013exchange2 192.168.13.22:143 weight 10 check

director exchange_2013_imaps 192.168.13.4:993 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:993 weight 10 check
  server 2013exchange2 192.168.13.22:993 weight 10 check
```

Don't forget to update the following information:

- **director**'s IP address
- **servers** IP addresses

5.3.4 Layer4 source NAT

Go on the **LB Layer 4** tab, then add the configuration below:

```
director exchange_2013_pop 192.168.13.4:110 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:110 weight 10 check
  server 2013exchange2 192.168.13.22:110 weight 10 check

director exchange_2013_pops 192.168.13.4:995 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:995 weight 10 check
  server 2013exchange2 192.168.13.22:995 weight 10 check

director exchange_2013_imap 192.168.13.4:143 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:143 weight 10 check
  server 2013exchange2 192.168.13.22:143 weight 10 check

director exchange_2013_imaps 192.168.13.4:993 TCP
  balance leastconn
  mode nat
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:993 weight 10 check
  server 2013exchange2 192.168.13.22:993 weight 10 check
```

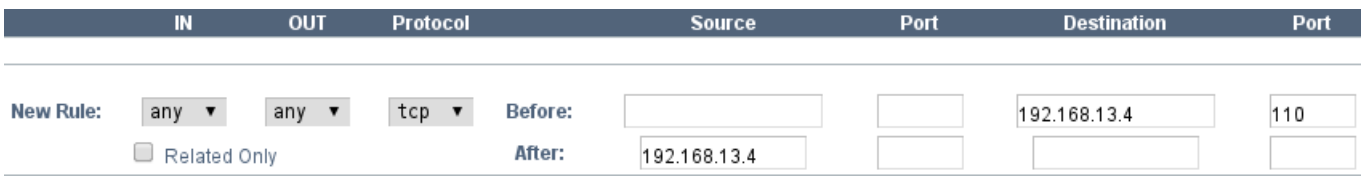
Don't forget to update the following information:

- **director**'s IP address
- **servers** IP addresses

Then, go on the **NAT** tab, and create a new rule with:

- **Protocol**: TCP
- **Source / After / IP**: the Virtual IP to NAT to (in the server LAN)
- **Destination / Before / IP**: the Virtual IP exposed to users
- **Destination / Before / port**: the port of the service exposed to users
- let the other options to default value (any)

In example, clients needs to access POP3 services on **Exchange 2013** on 192.168.13.4:110, and we want to re-use this IP address to get connected on the server.
We have to create the following rule:



Apply this confirguration, then click on the **Apply** button.
Add as many rules as you need.

5.3.5 Layer 4 Direct Server Return

Go on the **LB Layer 4** tab, then add the configuration below:

```
director exchange_2013_pop 192.168.13.4:110 TCP
  balance leastconn
  mode gateway
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:110 weight 10 check
  server 2013exchange2 192.168.13.22:110 weight 10 check

director exchange_2013_pops 192.168.13.4:995 TCP
  balance leastconn
  mode gateway
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:995 weight 10 check
  server 2013exchange2 192.168.13.22:995 weight 10 check

director exchange_2013_imap 192.168.13.4:143 TCP
  balance leastconn
  mode gateway
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:143 weight 10 check
  server 2013exchange2 192.168.13.22:143 weight 10 check

director exchange_2013_imaps 192.168.13.4:993 TCP
  balance leastconn
  mode gateway
  check interval 10 timeout 2
  option tcpcheck
  server 2013exchange1 192.168.13.21:993 weight 10 check
  server 2013exchange2 192.168.13.22:993 weight 10 check
```

Don't forget to update the following information:

- **director**'s IP address
- **servers** IP addresses

6. Recommended deployment

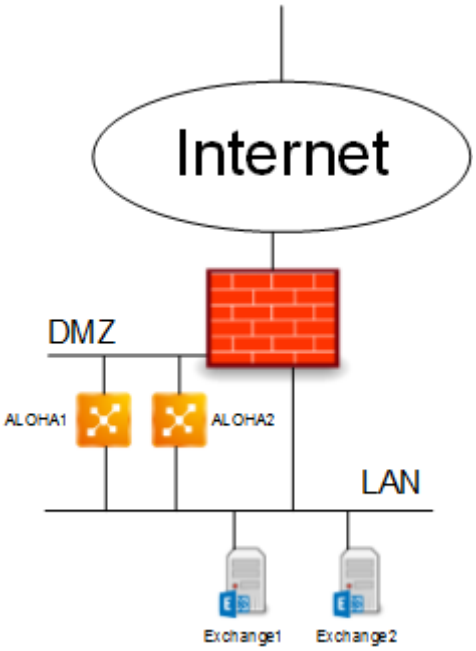
HAProxy Technologies benefits from a huge experience in **Microsoft Exchange** deployments. That's why we propose the following deployment scenario as a standard. This recommended scenario is defined with cost saving and security in mind.

6.1 Architecture

A couple (or more) of **Exchange 2013** servers in the LAN. A couple of **ALOHA Load-Balancers** are deployed with 2 interfaces at least: one in the DMZ and one in the LAN. Purpose is be able to load-balance both external and internal services using a single cluster of **ALOHA**.

In order to improve security, it is possible to deploy 2 clusters of **ALOHA**, one per zone: DMZ and LAN

The picture below shows the architecture:



6.2 Security point of view

The **ALOHA** is waterproof: no traffic won't be routed from one zone to the other one by the load-balancer. We can disable traffic routing between interface.
There won't be any asymetric routing thanks to the default gateway configured in each zone.
All the traffic from the **DMZ** remains in the **DMZ** and all the traffic from the **LAN** remains in the **LAN**. Since the ALOHA is a Reverse-Proxy, it can terminate connections in a zone and open new ones in an other zone. It is true the **ALOHA** will bypass the **Firewall** for external users. But it's not a problem at all, since your **Firewall** already allows traffic the to ALOHA **DMZ** VIP and also allows traffic from the ALOHA IPs to the Exchange servers. **Actually, you're going to save resources on your firewall, thanks to this deployment**.
That said, if security is your main focus, then it is better to setup one cluster per zone, but this is more expensive.

6.3 No DMZ and/or no external users?

Not a big deal, this setup still applies to you. Simply don't configure the binds into this zone.

6.4 Namespaces

6.4.1 External services

A single namespace, with **SSL offloading** configured on the **ALOHA** to allow traffic inspection and improve protection.

6.4.2 Internal services

A couple of namespaces:

1. one name dedicated to **outlook anywhere** or **MAPI over HTTP** service, where the **ALOHA** is configured in SSL forward mode.
2. one name for all other services where the **ALOHA** is configured in SSL offloading mode

6.4.3 Why this configuration?

1. Outlook clients (whatever the version) are not SSL friendly... They can't resume SSL connections... It is better to smartly load-balance those connections to the servers without concentrating SSL processing in a single point.
2. The **ALOHA** could process SSL offloading on Outlook Anywhere service, but you would have to order a much bigger appliance and/or licence to do it
3. We don't need traffic inspection on Outlook Anywhere on the internal network, since it's a trusted zone with trusted clients

6.5 ALOHA Configuration

6.5.1 Network configuration

You have to configure two network interfaces, with a default gateway on each of them. Required information can be found on **HAProxy Technologies** website.
In this chapter, we'll consider the **DMZ** interface is **eth0** and the **LAN** interface is **eth1**.

6.5.2 Internal and external services

First, we setup a **defaults** section in which we'll configure most defaults parameters for the frontend and backends dedicated to the **Exchange 2013** configuration.

```
defaults XCHANGE2013
 mode http
 option http-keep-alive
 option prefer-last-server
 no option httpclose
 no option http-server-close
 no option forceclose
 no option http-tunnel
 log global
 option httplog
 option forwardfor
 balance leastconn
 default-server inter 3s rise 2 fall 3
 timeout client        600s
 timeout http-request   10s
 timeout connect         4s
 timeout server         60s
```

Then the **frontend** section which listens into both DMZ and LAN zones. The **ALOHA** will automatically use the default gateway associated to the network interface associated to the incoming traffic.

```
frontend ft_XCHANGE2013_https
  bind 10.0.0.4:80 name PUB_http interface eth0
  bind 10.0.0.4:443 name PUB_https ssl crt 2013.haproxylab.net interface eth0
  bind 192.168.13.4:80 name LAN_http interface eth1
  bind 192.168.13.4:443 name LAN_https ssl crt 2013.haproxylab.net interface eth1
  capture request header Host len 32
  capture request header User-Agent len 64
  capture response header Content-Length len 10
  # log-format directive must br written on a single line
  # it is splitted for documentation convnience
  log-format %ci:%cp\ [%t]\ %ft\ %b/%s\ %Tq/%Tw/%Tc/%Tr/%Tt\ %ST\ %B\ %CC\ %CS\ %tsc\ %ac/%fc/%bc/%sc/%rc
\ %sq/%bq\ %hr\ %hs\ {%sslv/%sslc/%[ssl_fc_sni]/%[ssl_fc_session_id]}\
"%[capture.req.method]\ %[capture.req.hdr(0)]%[capture.req.uri]\ HTTP/1.1"
  maxconn 1000
  acl ssl_connection ssl_fc
  acl host_mail hdr(Host) -i mail.2013.haproxylab.net
  acl path_slash path /
  acl path_autodiscover path_beg -i /Autodiscover/Autodiscover.xml
  acl path_activesync path_beg -i /Microsoft-Server-ActiveSync
  acl path_ews path_beg -i /ews/
  acl path_owa path_beg -i /owa/
  acl path_oa path_beg -i /rpc/rpcproxy.dll
  acl path_ecp path_beg -i /ecp/
  acl path_oab path_beg -i /oab/
  acl path_mapi path_beg -i /mapi/
  acl path_check path_end -i HealthCheck.htm

  # HTTP deny rules
  http-request deny if path_check

  # HTTP redirect rules
  http-request redirect scheme https code 302 unless ssl_connection
  http-request redirect location /owa/ code 302 if path_slash host_mail

  # HTTP routing rules
  use_backend bk_XCHANGE2013_https_autodiscover if path_autodiscover
  use_backend bk_XCHANGE2013_https_activesync if path_activesync
  use_backend bk_XCHANGE2013_https_ews if path_ews
  use_backend bk_XCHANGE2013_https_owa if path_owa
  use_backend bk_XCHANGE2013_https_oa if path_oa
  use_backend bk_XCHANGE2013_https_ecp if path_ecp
  use_backend bk_XCHANGE2013_https_oab if path_oab
  use_backend bk_XCHANGE2013_https_mapi if path_mapi
  # other services go here
  default_backend bk_XCHANGE2013_https_default
```

ActiveSync configuration:

```
backend bk_XCHANGE2013_https_activesync
  option httpchk GET /Microsoft-Server-ActiveSync/HealthCheck.htm
  http-check expect string 200\ OK
  server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

Autodiscover configuration:

```
backend bk_XCHANGE2013_https_autodiscover
  option httpchk GET /Autodiscover/HealthCheck.htm
  http-check expect string 200\ OK
  server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

Exchange Control Panel (ECP) configuration:

```
backend bk_XCHANGE2013_https_ecp
  option httpchk GET /ECP/HealthCheck.htm
  http-check expect string 200\ OK
  server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

Exchange Web Services (EWS) configuration:

```
backend bk_XCHANGE2013_https_ews
  option httpchk GET /EWS/HealthCheck.htm
  http-check expect string 200\ OK
  server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

MAPI configuration:

```
backend bk_XCHANGE2013_https_mapi
  option httpchk GET /mapi/HealthCheck.htm
  http-check expect string 200\ OK
  timeout server 600s
  server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

Offline Address book (OAB) configuration:

```
backend bk_XCHANGE2013_https_oab
  option httpchk GET /OAB/HealthCheck.htm
  http-check expect string 200\ OK
  server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

Outlook Anywhere (OA) configuration:

```
backend bk_XCHANGE2013_https_oa
  option httpchk GET /RPC/HealthCheck.htm
  http-check expect string 200\ OK
  timeout server 600s
  server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

Outlook Web Application (OWA) configuration:

```
backend bk_XCHANGE2013_https_owa
  option httpchk GET /owa/HealthCheck.htm
  http-check expect string 200\ OK
  server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

Other requests configuration:

```
backend bk_XCHANGE2013_https_default
  timeout server 60s
  server 2013xchange1 192.168.13.21:80 maxconn 1000 weight 10 check
  server 2013xchange2 192.168.13.22:80 maxconn 1000 weight 10 check
```

6.5.3 Outlook Anywhere for LAN users

And finally, the Outlook Anywhere configuration for internal users, in raw TCP mode:

```
defaults XCHANGE2013_internal_oa
 mode tcp
 log global
 option tcplog
 option dontlognull
 option contstats
 timeout client 600s
 timeout server 600s
 timeout connect 5s

frontend ft_XCHANGE2013_internal_oa
  bind 192.168.13.5:443 name LAN_https interface eth1
  maxconn 10000
  default_backend bk_XCHANGE2013_internal_oa

backend bk_XCHANGE2013_internal_oa
  balance leastconn
  option redispatch
  retries 3
  option httpchk GET /RPC/HealthCheck.htm
  http-check expect string 200\ OK
  default-server inter 3s rise 2 fall 3
  server 2013xchange1 192.168.13.21:443 weight 10 check check-ssl
  server 2013xchange2 192.168.13.22:443 weight 10 check check-ssl
```

7. Apendix 1: Exchange 2013 configuration summary

Print and fill this table. It can be used as a basis for your configuration.

7.1 Publications for internal users

Service	number of users	Configuration type *
Exchange ActiveSync (EAS)		
Outlook Anywhere (OA) or Mapi over HTTP (MAPI)		
Outlook Web App (OWA)		
POP3		
IMAP4		

* configuration type: Layer 4, Reverse-Proxy raw TCP, SSL offloadin, SSL bridging

7.2 Publications for external users

Service	number of users	Configuration type *
Exchange ActiveSync (EAS)		
Outlook Anywhere (OA) or Mapi over HTTP (MAPI)		
Outlook Web App (OWA)		
POP3		
IMAP4		

* configuration type: Layer 4, Reverse-Proxy raw TCP, SSL offloadin, SSL bridging