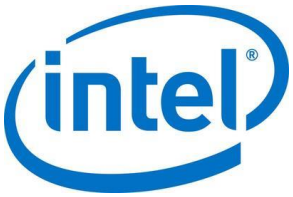


HAProxy* with Intel® QuickAssist Technology

Application Note

April 2018

Revision 001

Document Number: 337430-001US

Contents

1 Introduction 5
1.1 Network Topology 5
1.2 Resources and Prerequisites 5
1.3 Terminology 5
1.4 Reference Documents 6
2 Operating System and Virtual Machine Setup 7
2.1 Install the Host Operating System 7
2.2 Install and Configure the Virtual Machines 7
2.3 Test the Virtual Machines 7
3 HAProxy* Setup and Testing for HTTP Connections 9
4 HAProxy* Setup and Testing for HTTPS Connections 12
4.1 Generate a Self-Signed Certificate 12
4.2 Update the HAProxy Configuration File 12
5 Intel® QuickAssist Technology Setup and Testing 13
5.1 OpenSSL and QAT Engine Setup and Testing 13
5.2 HAProxy*+Intel® QAT Setup and Testing 13
6 HAProxy*+QAT Performance Testing 15

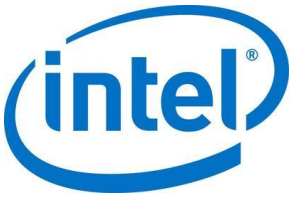
Tables

Table 1. Terminology 5
Table 2. Reference Documents 6

Revision History

Document Number	Revision Number	Description	Revision Date
337430	001	Initial release.	April 2018

§



1 Introduction

This document details the steps necessary to configure HAProxy* to work with Intel® QuickAssist (Intel® QAT) Technology.

1.1 Network Topology

While other configurations are possible, this document focuses on a simple "Secure Sockets Layer (SSL) Termination" topology in which a frontend proxy server with Intel® QuickAssist Technology handles traffic between clients and backend servers.

In this case, the connections between the proxy server and clients use secure protocols, but connections between the proxy and backend servers do not use secure protocols. This configuration essentially offloads the security workload to the proxy server so the backend servers don't have to carry the overhead of the secure protocols.

In practice, this topology uses multiple systems: for easier configuration, this application note has been written such that the setup may be tested with just one system. The backend servers will be Virtual Machines (VMs) on the one system, and the client traffic can also be generated on the same system.

1.2 Resources and Prerequisites

Before working through this document, the following fundamentals are required:

- General familiarity with Intel® QAT.

  Technical collateral, including links to tutorial videos, are available at https://01.org/intel-quickassist-technology.

- Familiarity with the OpenSSL* QAT engine:

  Details are available via the "*Intel® QuickAssist Technology - libcrypto/openssl resources*", Table 2, which includes the link to the Intel® QAT Engine GitHub page: https://github.com/intel/QAT_Engine/.

- A system with Intel® QAT installed.

1.3 Terminology

Table 1. Terminology

Term	Description
Intel® QAT	Intel® QuickAssist Technology
SSL	Secure Sockets Layer

Term	Description
VMs	Virtual Machines

1.4 Reference Documents

Table 2. Reference Documents

Document	Document No./Location
Intel® QuickAssist Technology - libcrypto/openssl resources	https://01.org/intel-quickassist-technology
Intel® QuickAssist Technology Software for Linux - Getting Started Guide*	336212/ https://01.org/intel-quickassist-technology
Intel® QuickAssist Technology Performance Sample Code	https://software.intel.com/en-us/videos/intel-quickassist-technology-performance-sample-code
Intel® QuickAssist Technology: Performance Sample Code Debug	https://software.intel.com/en-us/videos/intel-quickassist-technology-performance-sample-code-debug
Intel® QuickAssist Technology (Intel® QAT): OPENSSL 1.1.x+ Intel® QAT Engine	https://software.intel.com/en-us/videos/intel-quickassist-technology-openssl-1-1-x-qat-engine

§

2 Operating System and Virtual Machine Setup

This section provides instructions on how to install the Linux* operating system (OS) on the host system. Instructions are provided for the setup of two virtual machines (VMs), which are used as backend web servers for testing purposes.

2.1 Install the Host Operating System

From https://01.org/intel-quickassist-technology, find the applicable "*Intel® QuickAssist Technology Software for Linux* - Getting Started Guide*." Follow the "Installing the Operating System" chapter to install Linux* on your system. It isn't a requirement to follow the steps exactly, but following the steps should ensure that you do not encounter build errors or other errors.

2.2 Install and Configure the Virtual Machines

For functional testing, there are no specific requirements for the VMs and, in fact, they do not have to be VMs at all. These will be acting as backend web servers; for testing purposes we'll set up two of these. For ease of setup and configuration, the VM Manager GUI can be used to install the latest Ubuntu* Server distribution on each of these virtual machines. Name the virtual machines intuitively: for instance, "**MyWebServer1**" and "**MyWebServer2**". Select the option to enable ssh access to make remote configuration and debug easier.

Once the operating systems for the backend web servers have been installed and configured, you may optionally shut down the VMs and then use virsh and ssh to access these, for easier remote access.

2.3 Test the Virtual Machines

With the virtual machines shut down and the Virtual Machine Manager GUI closed, run "`sudo virsh list --all`" to see the available virtual machines: for instance, "**MyWebServer1**" and "**MyWebServer2**" should show these are "**off**".

From this point forward, assume the names of the virtual machines are "**MyWebServer1**" and "**MyWebServer2**".

1. Start **MyWebServer1** using "`sudo virsh start MyWebServer1`".
2. Obtain the IP address associated with **MyWebServer1** using "`sudo virsh domifaddr MyWebServer1`".
3. Connect to **MyWebServer1** using "`ssh 192.168.122.xxx`".

   Insert the correct IP address obtained in Step two.

4. If necessary, update the `apt-get` proxy for the host environment.

   This may be enabled by adding the following to a new file located at `/etc/apt/apt.conf` using the following script, substituting your specific details for the placeholders:

   ```
   Acquire::http::Proxy "http://<yourproxyIP>:<yourproxyport>";
   ```

5. After a "`sudo apt-get update`" (or equivalent), use "`sudo apt-get install nginx`" to install nginx*.
6. From the host operating system, enter "`wget <IPWebServer1>`".

   This should download an `index.html` file to the current working directory. If so, **MyWebServer1** VM web server has been configured correctly.

Note: Successive requests of `wget` will not overwrite the `index.html` by default; instead, it will save the file with a slightly different filename.

Look at the nginx config file located in `/etc/nginx/nginx.conf` to determine where the main html page is located. It may be located at `/var/www/html/index.nginx-debian.html`. Copy or move the config file as necessary and/or edit `/etc/nginx/nginx.conf` to point to your main html page.

Make the index.html (or other main html page file) unique to distinguish it from the other backend web server. For instance, change the text in the <title> tag to "**MyWebServer1**" and the text in the <body> section to display a unique string. For instance, you can have this paragraph in index.html:

```
<p>MyWebServer1</p>
```

7. Repeat Steps 1 through 6 of this section to setup **MyWebServer2**, substituting "**MyWebServer1**" with "**MyWebServer2**" and using the **MyWebServer2** IP address.

§

3 HAProxy* Setup and Testing for HTTP Connections

HAProxy added support for asynchronous crypto engines beginning with v1.8.0.

Generally speaking, for best results, start with the latest stable HAProxy package located here: http://www.haproxy.org/.

For more information, refer to release announcement located here: https://www.mail-archive.com/haproxy@formilux.org/msg28004.html.

As noted in the announcement, support for asynchronous engines requires OpenSSL 1.1.x or later.

In many, if not most cases building HAProxy from the source may be required for the foreseeable future if support for asynchronous engines is required. If you are installing HAProxy from a package manager (such as `dnf`, `yum`, or `apt-get`), check for the **OpenSSL 1.1.x** dependency, using the following command:

```
# haproxy -vv
```

This command will show information about the HAProxy version (e.g. v1.8 or greater) and also the **OpenSSL** version (e.g. v1.1.0 or greater). Running "`ldd haproxy`" also gives insight into the HAProxy assumptions and environment.

Note: It's strongly recommend to remove old HAProxy versions when installing a newer version.

From here, assume HAProxy will be built from the source. Download the latest stable branch from http://www.haproxy.org/. Untar the source file and enter the HAProxy root directory.

Use the following commands to ensure that **OpenSSL 1.1.0** or later is being used for the HAProxy build, set `SSL_INC` and `SSL_LIB` to **OpenSSL 1.1.0+** and include library directories, respectively. For instance:

```
# export SSL_INC=/usr/local/ssl/include
# export SSL_LIB=/usr/local/ssl/lib
```

Note: If you did not do a "`make install`" of the **OpenSSL 1.1.0+** or if you installed it in different directories, adjust the environment variables above to point to the correct directories.

Use the following command to build HAProxy:

```
# make TARGET=linux2628 USE_OPENSSL=1
```

Assuming that this compiles correctly, verify immediately that "`./haproxy -vv`" shows it has been built and is running against the 1.1.0+. You can also run "`ldd haproxy`". Verify that it does not show `libssl.so.10`.

Note: With a typical OpenSSL 1.1.0+ installation, the following command may appear when trying to run HAProxy:

```
# ./haproxy -vv
```

`./haproxy`: error while loading shared libraries: `libssl.so.1.1`: cannot open shared object file: No such file or directory

Run the following command to avoid this error:

```
# export LD_LIBRARY_PATH=/usr/local/ssl/lib
```

The output of "`haproxy -vv`" should be similar to the following:

```
# ./haproxy -vv
...
  OPTIONS = USE_OPENSSL=1
...

Built with OpenSSL version : OpenSSL 1.1.0g  2 Nov 2017

Running on OpenSSL version : OpenSSL 1.1.0g  2 Nov 2017
...
```

The output of "`ldd haproxy`" should be similar to the following:

```
# ldd ./haproxy
        linux-vdso.so.1 =>  (0x00007fff72bb6000)
        libcrypt.so.1 => /lib64/libcrypt.so.1 (0x00007f26c49b5000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f26c47b0000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f26c4594000)
        libssl.so.1.1 => /usr/local/ssl/lib/libssl.so.1.1
(0x00007f26c4325000)
        libcrypto.so.1.1 => /usr/local/ssl/lib/libcrypto.so.1.1
(0x00007f26c3e9f000)
        libc.so.6 => /lib64/libc.so.6 (0x00007f26c3adc000)
        libfreebl3.so => /lib64/libfreebl3.so (0x00007f26c38d9000)
        /lib64/ld-linux-x86-64.so.2 (0x0000558b75ebd000)
```

Optionally, do a "`make install`" of HAProxy.

Note: To start HAProxy on boot: because of the differences in distributions, the instructions to do so are outside of the scope of this document.

There are many HAProxy configuration options. Consult the "**examples**" directory located in the HAProxy directory to understand which options are available.

To test a simple HAProxy configuration, use the following HAProxy configuration file:

```
frontend myfrontend
     bind *:80
     default_backend mybackend

backend mybackend
     balance roundrobin
     mode http
     server myvm1 <ipaddress1>:80 check # e.g. 192.168.1.101:80
     server myvm2 <ipaddress2>:80 check # e.g. 192.168.1.101:80
```

Note: Change the `<ipaddress#>` placeholders so they point to your **MyWebServer1** and **MyWebServer2** VM IP addresses.

Save the configuration file to any accessible directory. For testing purposes, invoke HAProxy with an explicit path to the configuration file. Optionally, you may need to save this as `/etc/haproxy/haproxy.cfg`. For our purposes we assume the HAProxy configuration file will reside at `/etc/haproxy/haproxy.cfg`.

Invoke HAProxy as follows:

```
# haproxy -f /etc/haproxy/haproxy.cfg
```

If any errors or warnings are reported, be sure to understand these and deal with them as necessary.

Test that HAProxy is working correctly on the host operating system by using the following command:

```
# wget 127.0.0.1
```

Alternatively, run `wget` or access the service IP address from a client system using `wget` or a Web Browser. If set up correctly, the `index.html*` file will include the default web page of the virtual machine, along with any modifications that were made (e.g. changing the <title> tag to "**MyWebServer1**"). Each successive invocation should show the `index.html` file of the next web server virtual machine, since we told HAProxy to use the roundrobin algorithm.

§

4 HAProxy* Setup and Testing for HTTPS Connections

To test HAProxy with HTTPS connections, create or obtain a certificate, update the HAProxy configuration file to redirect the HTTPS requests (via port 443) to the backend Servers (on port 80).

4.1 Generate a Self-Signed Certificate

Follow the steps below to create a self-signed certificate for HTTPS testing:

```
# sudo mkdir /etc/ssl/myhaproxy
# ./openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout\
server.key -out server.crt
# sudo cat /etc/ssl/myhaproxy/myhaproxy.crt\
/etc/ssl/myhaproxy/myhaproxy.key > \
/etc/ssl/myhaproxy/myhaproxy.pem
```

4.2 Update the HAProxy Configuration File

Just one additional line is required in the `haproxy.cfg`, to redirect the port 443 traffic to port 80 on the backend servers:

```
frontend myfrontend
     bind *:80
     bind *:443 ssl crt /etc/ssl/myhaproxy/myhaproxy.pem
     default_backend mybackend

backend mybackend
     balance roundrobin
     mode http
     server myvm1 <ipaddress1>:80 check # e.g. 192.168.1.101:80
     server myvm2 <ipaddress2>:80 check # e.g. 192.168.1.102:80
```

Now invoke HAProxy as follows:

```
# haproxy -f /etc/haproxy/haproxy.cfg
```

If any errors or warnings are reported, be sure to understand these and deal with them as necessary.

To test that HAProxy is working correctly, run the following command on the host operating system:

```
# wget --no-check-certificate https://127.0.0.1
```

Alternatively, run `wget` or access the service IP address from a client system using `wget` or a web browser with "**https://**" explicitly specified before the IP address. When set up correctly, you should see the `index.html*` file has been downloaded successfully.

§

5 Intel® QuickAssist Technology Setup and Testing

Obtain a copy of the *Intel® QuickAssist Technology Software for Linux* - Getting Started Guide* (see Table 2). Follow these instructions to install and test the Intel® QAT package. Ensure that some Intel® QAT sample code can be run successfully before continuing.

5.1 OpenSSL and QAT Engine Setup and Testing

Refer to OpenSSL and Intel® QAT Engine materials for setup and testing. Refer to Table 2, "*Intel® QuickAssist Technology - libcrypto/openssl resources*" which includes the link to the Intel® QAT engine GitHub page: https://github.com/intel/QAT_Engine/.

Note: Versions of OpenSSL earlier than v1.1.0 do not support Intel® QAT engine.

5.2 HAProxy*+Intel® QAT Setup and Testing

Enable Intel® QAT in HAProxy by adding the following to the bottom of the global section in the `haproxy.cfg` file:

```
ssl-engine qat algo RSA
```

As desired, experiment with other variants of the ssl-engine line.

For asynchronous operations, which should generally give better performance, include this at the bottom of the global section in the `haproxy.cfg` file:

```
ssl-mode-async
```

Consult the HAProxy documentation for additional information on these parameters.

You may want to consider other HAProxy options, including "`tune.ssl.default-dh-param 2048`".

Now invoke HAProxy as follows:

```
# haproxy -f /etc/haproxy/haproxy.cfg
```

If any errors or warnings are reported, be sure to understand these and deal with them as necessary.

Now test that HAProxy is working correctly using the following command:

```
# wget --no-check-certificate https://127.0.0.1
```

Alternatively, run `wget` or access the service IP address from a client system using `wget` or a web browser with "**https://**" explicitly specified before the IP address. When set up correctly, you should see that the **index.html*** file is downloaded successfully.

To verify Intel® QAT is being used successfully, note that the latest Intel® QAT driver has a `/sys/kernel/debug/qat_*/fw_counters` which can be "**cat**"ed out to show the firmware requests. If this number increases when the web request is made, then Intel® QAT is being used. If this number does not increase, Intel® QAT is not being used.

If this test is not successful, double-check the steps of each previous section, paying careful attention to the fact that the minimum required version of HAProxy is v1.8, and it must be explicitly built with OpenSSL 1.1.0 or greater.

§

6 HAProxy*+QAT Performance Testing

Note: Performance testing is outside of the scope of this document at this time.

Before concluding that Intel® QAT is a bottleneck in any configuration, first rule out other possible bottlenecks. These could be related to the following, on the frontend or the backend Servers:

- System memory
- CPU utilization
- Network bandwidth
- PCIe* bandwidth
- Other system settings or limitations.

As a general rule, to be sure that the right performance conclusions are made, ensure that you can get the performance expected in each of the following configurations:

- HAProxy without HTTPS
- HAProxy with HTTPS, but without Intel® QAT being used
- HAProxy with HTTPS and with Intel® QAT being used.

If these tests lead you to believe that Intel® QAT is the bottleneck, first check for the performance of Intel® QAT using the performance sample code and also via OpenSSL speed, as discussed in these videos:

- Intel® QuickAssist Technology Performance Sample Code: https://software.intel.com/en-us/videos/intel-quickassist-technology-performance-sample-code
- Intel® QuickAssist Technology: Performance Sample Code Debug: https://software.intel.com/en-us/videos/intel-quickassist-technology-performance-sample-code-debug
- Intel® QuickAssist Technology (Intel® QAT): OPENSSL 1.1.x+ Intel® QAT Engine: https://software.intel.com/en-us/videos/intel-quickassist-technology-openssl-1-1-x-qat-engine

Note: You may have to change the value of `LimitDevAccess` in the Intel® QAT configuration files (and then restart the `qat_service`) to use more than one Intel® QAT endpoint.

§