Integration Guide

CryptoServer Simulator in a Docker Container

utimaco®

Imprint

Copyright 2022	Utimaco IS GmbH Germanusstr. 4 D-52080 Aachen Germany
Phone	AMERICAS: +1-844-UTIMACO (+1 844-884-6226) EMEA: +49 800-627-3081 APAC: +81 800-919-1301
Internet	https://support.utimaco.com
Email	support@utimaco.com
Document version	1.0
Date	2022-05-18
Document No.	2022-0076
Author	Utimaco IS GmbH

Contents

1 Introduction

This guide is part of the information and support provided by Utimaco. Additional documentation produced to support your Utimaco SecurityServer product can be found in the document directory of the Utimaco SecurityServer product bundle. All Utimaco SecurityServer product documentation is available from Utimaco's web site at **https://utimaco.com/**

1.1 About This Guide

This guide describes how to enable HSM integration with Docker. The instructions in this document have been thoroughly tested and provide a straightforward integration process. There may be other untested ways to achieve interoperability.

1.1.1 Target Audience for This Guide

This guide is intended for using Containerized HSM Simulators HSM administrators.

1.1.2 Contents of This Guide

After the introduction this manual is divided up as follows:

Chapter 2 Overview

Chapter 3 Integration Requirements and Prerequisites

Chapter 4 Install Docker Engine

Chapter 5 Creating Docker Image

Chapter 6 Using the CryptoServer Simulator Image

Chapter 7 Troubleshooting

Chapter 8 Further Information

1.1.3 Document Conventions

The following conventions are used in this guide:

Convention	*Use*	*Example*
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press the **OK** button.

Convention	*Use*	*Example*
`Monospaced`	File names, folder and directory names, commands, file outputs, programming code samples	You will find the file `example.conf` in the `/exmp/demo/` directory.
Italic	References and important terms	See Chapter 3, "Sample Chapter", in the *CryptoServer - csadm Manual* or [CSADMIN].

Table 1: Document conventions

Special icons to highlight the most important notes and information.

Here you find important safety information that should be followed.

Here you find additional notes or supplementary information.

1.1.4 Abbreviations

The following abbreviations are used in this guide:

Abbreviation	*Meaning*
API	Application Programming Interface
CD	Compact Disc
CSADM	CryptoServer Command-line Administration Tool
CSAR	Cloud Service Architecture
CNG	Cryptography API Next Generation
CSP	Cryptographic Service Provider
GUI	Graphical User Interface

Abbreviation	*Meaning*
HSM	Hardware Security Module
ID	Identity
IP	Internet Protocol
JCE	Java Cryptography Extension
LAN	Local area network
PCIe	PCI Express Interface
PKCS#11	Public-Key Cryptography Standard #11
Pseudo TTY	Pseudo Terminals
SDK	Software Development Kit
STDERR	Standard Error
STDOUT	Standard output
TCP	Transmission Control Protocol
URL	Uniform Resource Locator

Table 2: List of Abbreviations

2 Overview

2.1 Docker Minimal Client

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so the developer can assure that the application will run on any other Linux machine regardless of any customized settings that machine might have.

In a similar way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker does allow applications to use the same Linux kernel as the system that they are running on and only requires applications be shipped with things not already running on the host computer. This significantly reduces the footprint of size of the application.

Utimaco offers a fully functional HSM Software Simulator (CryptoServer Simulator). The CryptoServer Simulator package comes with 100% functional runtime, including all administration and configuration toolsets. The Utimaco CryptoServer Simulator facilitates evaluation, development and integration testing without purchase, delivery, or installation of hardware. For a development teams but even more so for partners' and customers' development teams, the CryptoServer Simulator serves in the process of (application) development. Customers that use the CryptoServer Software Development Kit (SDK) to develop their own firmware module can use the CryptoServer Simulator for testing and validation. Where hardware security modules are integrated into existing IT infrastructure, multiple users can test their developments and corresponding interfaces on the CryptoServer Simulator without affecting production. The CryptoServer Simulator can be used to integrate the HSM with third party applications that provide standardized cryptographic API. These include PKCS#11, CSP/CNG and JCE. The CryptoServer Simulator can be used in plug & play deployments for evaluation of different configuration options, application settings, as well as load-balancing or high-availability scenarios. This guide shows how to build an image for the CryptoServer Simulator. With this Docker image it is simple to start and stop a single instance of a CryptoServer Simulator or run several CryptoServer Simulator simultaneously as a CryptoServer cluster. Development teams can easily deploy custom firmware modules to the CryptoServer Simulator container without the need to revert the current firmware module configuration manually right after integration or unit testing. The Docker image will not store changes to CryptoServer Simulator permanently, so that after stopping the Docker container changes are lost. This allows also to use these Docker images for automated testing of custom firmware module development.

2.2 Utimaco CryptoServer HSM

CryptoServer is a hardware security module developed by Utimaco IS GmbH. CryptoServer is a physically protected, specialized computer unit designed to perform sensitive cryptographic tasks and to securely manage as well as store cryptographic keys and data. It can be used as a universal, independent security component for heterogeneous computer systems.

3 Integration Requirements and Prerequisites

Ensure that the system environment you will be using, meets the following hardware and software requirements.

This guide assumes that the user has already installed and configured required Software.

3.1 Tested Versions

The integrations that have been successfully tested with the Utimaco HSM with Docker Container.

Operating System	*Docker Version*	*Utimaco Security Server Version*	*Utimaco HSM*
CentOS 8.2 Debian Ubuntu	20.10.7 5.19 20.04	SecurityServer V4.45.3.0	CryptoServer CSe-Series/Se-Series u.trust Anchor Se*k and u.trust Anchor CSAR

Table 3: List of Software Requirements

3.2 Software Requirements

Software	*Version*
HSM Interfaces	SecurityServer PKCS#11 Provider
Utimaco Crypto Server Software	V4.45.3.0
Docker Image Version	20.10.7

Table 4: List of Software Requirements

3.3 Hardware Requirements

Hardware	*Hardware Requirements*
Utimaco LAN HSM	CryptoServer CSe-Series/Se-Series LAN with firmware SecurityServer 4.45.3 or higher u.trust Anchor Se*k and u.trust Anchor CSAR with firmware 4.50 or higher
Utimaco PCI-e HSM	CryptoServer CSe-Series/Se-Series PCI-e with firmware SecurityServer 4.45.3 or higher

Table 5: List of Hardware Requirements

Setup an account on the Utimaco support portal and request download access at the following URL.

https://support.hsm.utimaco.com/

3.4 Prerequisites

Before you begin, please ensure that you have installed/setup:

- Operating system listed in Tested Versions
- SecurityServer listed in Tested Version
- Familiarize yourself with the docker minimal client documentation and setup process and have the Utimaco documentation available.

4 Install Docker Engine

To install docker engine on your Linux operating system we do refer to the docker documentation website. (Refer Link, **Install Docker Engine on CentOS | Docker Documentation**).

1. To get started with Docker Engine on CentOS, make sure you meet the prerequisites
2. Install the latest version of Docker Engine, containerd using below mentioned command
3. After installation of the docker engine, you can check whether the docker engine is running and setup correctly by starting a hello-world container

›_ Console

```
# yum install docker-ce docker-ce-cli containerd.io

# docker run --rm hello-world && docker image rm hello-world
Hello from Docker!
```

If you are running the above and it results in an error, please verify that either you are running the commands as root or your account is a member of the docker group.

5 Creating Docker Image

Docker containers are based on a Docker image. Those images consist of multiple immutable layers of data. The immuteable layers are generated on build time. One image can be based off of another image. By starting a container, a slim read/write-layer is generated on top of the static image which holds all runtime data. Changing files in the underlying layers will start a copy-on-write action with the new file in the r/w-layer overlaying the older one. Due to this, multiple containers can be based on a single image with minimal footprint. A Docker image is built by using a so called Dockerfile.

5.1 Creating the Dockerfile

A Dockerfile contains plain text instructions on how to build a docker image.

- Create a directory `cssim` to the appropriate location and change the directory to `cssim`
- Create a new file Dockerfile to build our simulator image

›_ Console

```
# mkdir ~/cssim
# cd ~/cssim
# vi Dockerfile
```

- Copy the following lines into the file (Dockerfile).

›_ Console

```
FROM debian:stretch-slim
LABEL version="4.45.3.0"
 # description= "Utimaco CryptoServer Simulator"
  maintainer = "support@utimaco.com"

ARG SimulatorPort=3001

RUN dpkg --add-architecture i386; \

apt-get update; \
apt-get install -y \

libc6-i386 \
lib32gcc1; \

rm -rf /var/lib/apt/lists/*

WORKDIR /simulator/bin
```

```
RUN mkdir -p /etc/utimaco/
RUN mkdir -p /opt/utimaco/bin
RUN mkdir -p /opt/utimaco/lib
RUN chmod -R 777 /opt/utimaco
RUN chmod -R 777 /etc/utimaco

COPY sim5_linux /simulator

RUN chmod u+x ./bl_sim5

ENV SDK_PORT=$SimulatorPort

EXPOSE $SimulatorPort

STOPSIGNAL SIGINT
ENTRYPOINT ["./bl_sim5"]
CMD ["-o","-h"]
```

Change the CryptoServer Simulator version="4.45.3.0" appropriately.

As you can see from FROM debian:stretch-slim that we have derived our image from an offical slim Debian Stretch image. Those images are specifically build for containers as they do not contain unnecessary files. Due to the CryptoServer Simulator being an x86 application and Debian only releasing x86-64 operating systems we need to enable x86 support with dpkg --add-architecture i386. Furthermore you need to install some x86 libraries. To connect to the CryptoServer Simulator we will expose a TCP/IP port. The default port 3001 can be changed by supplying a paramter during the build process of our image. The Dockerfile moreover requires the CryptoServer Simulator files to be present in the project directory.

- Copy the CryptoServer Simulator files from your product CD. They are located in <cd>/Software/Linux/Simulator. Copy the sim5_linux folder to the project folder ~/cssim

›_ Console

```
# cp -r /media/cdrom/Software/Linux/Simulator/sim5_linux ~/cssim
```

- After this initial setup your project folder should look like this below

›_ Console

```
# tree
```

```
.
└── sim5_linux
    ├── ReadmeMBK.txt
    ├── bin
    │   ├── bl_sim5
    │   ├── cs_multi.sh
    │   ├── cs_sim.ini
    │   └── cs_sim.sh
    └── devices
        ├── ALARM.curr
        ├── FLASHFILE
        ├── MK
        ├── NVRAMFILE
        ├── SDRAM
        │   └── SDRAMFILE
        └── swap.com
```

5.2 Building the Docker Image

To start containers, the user can now build a Docker image that contains the CryptoServer Simulator. Use the build command inside of the project directory, created in the previous paragraph. Tag the image with the appropriate version of the CryptoServer Simulator. In this document we use version 4.45.3 as it represents the current product CD version.

›_ Console

```
# cd ~/cssim

# docker build -t simulator:4.45.3 .
Sending build context to Docker daemon
Successfully built e4ba909b69ca
Successfully tagged simulator:4.45.3
```

If you receive an error message on build you may check that you have setup the project folder as described in the previous paragraph. Also, you may check the connection to the apt repository as it is required to install the necessary additional packages.

Verify that the Docker image is created successfully using below command.

›_ Console

```
# docker image ls
REPOSITORY   TAG            IMAGE ID       CREATED        SIZE
```

```
simulator    4.45.3          e4ba909b69ca    31 hours ago    138MB
simulator    latest          e4ba909b69ca    31 hours ago    138MB
```

6 Using the CryptoServer Simulator Image

Now that you have created succesfully a docker image for the CryptoServer Simulator we can start a CryptoServer Simulator container. After you have started the container, you can then verify the functionality of the CryptoServer Simulator.

6.1 Launching a CryptoServer Simulator Container

Start an instance of CryptoServer Simulator docker image as a named container.

›_ Console

```
# docker run -dt --rm --name cssim453 simulator:4.45.3
0dd6753a0dc7021d4b9121553df429b892d078b6298c02bdfc04458ba1a8e1cd
```

Let us break down the command.

`docker run`

will create and start a container in the same step.

`-dt`

will run the container in the background (detached) and allocate a pseudo TTY for the process.

`--rm`

will remove the container when it is stopped.

`--name cssim453`

will make the container accessible by the name in addition to the container ID.

`simulator:4.45.3`

tells docker to use the image tagged with 4.45.3

The resulting alpha numerical string (here 0dd6753a0dc7021…) is the unique container ID. With this ID you can always change settings of the container.

6.2 Checking the Running Status of the Container

›_ Console

```
# docker container list
CONTAINER ID   IMAGE   COMMAND    CREATED    STATUS    PORTS     NAMES
e4ba909b69ca   simulator:4.45.3   "./bl_sim5 -o -h"   3 minutes ago   Up
3 minutes   288/tcp   cssim453
```

6.3 Viewing the Container Output

When the container was started with a pseudo TTY allocated we can view the output of the CryptoServer Simulator. Therefore, we use the docker logs command. It shows all the output

of the container process made to STDOUT and STDERR. You could also directly attach to the container although this is not recommended.

A newly created simulator container without modifications will have an output similar to the next console output. If the build was successful, you now have a docker image of the CryptoServer Simulator available to your local docker installation. You can view the image with the docker image command.

›_ Console

```
# docker logs cssim453

Utimaco CryptoServer Simulator HSD process started


22.04.26 10:00:22 SMOS SDK Ver. 5.6.4.0 (Oct 13 2021) started [0]
22.04.26 10:00:22 Compiler Ver. 4.8.5
22.04.26 10:00:22 CPU clock frequency: 1000000000
22.04.26 10:00:22 Devices directory: '/simulator/devices'
22.04.26 10:00:22 Sensory Controller Ver. 2.0.0.42 [0/0]
22.04.26 10:00:22 Real Random Number Generator initialized with:
  RESEED_INTERVAL = 1000
  PREDICTION_RESISTANCE = 0
  REALRANDOM_SHARE = 3
22.04.26 10:00:22 Pseudo Random Number Generator initialized with:
  RESEED_INTERVAL = 1000
  PREDICTION_RESISTANCE = 0
  REALRANDOM_SHARE = 0
22.04.26 10:00:22 Load module 'adm.msc' from FLASHFILE
22.04.26 10:00:22 Load module 'cmds.msc' from FLASHFILE
22.04.26 10:00:22 Load module 'crypt.msc' from FLASHFILE
22.04.26 10:00:22 Load module 'cxi.msc' from FLASHFILE
22.04.26 10:00:22 Load module 'db.msc' from FLASHFILE
22.04.26 10:00:22 Load module 'mbk.msc' from FLASHFILE
22.04.26 10:00:22 Load module 'ntp.msc' from FLASHFILE
22.04.26 10:00:22 Load module 'pp.msc' from FLASHFILE
22.04.26 10:00:22 Load module 'sc.msc' from FLASHFILE
22.04.26 10:00:22 Load module 'util.msc' from FLASHFILE
22.04.26 10:00:22 module 0x83 (CMDS) initialized successfully
22.04.26 10:00:22 module 0x86 (UTIL) initialized successfully
22.04.26 10:00:22 module 0x9f (CRYPT) initialized successfully
22.04.26 10:00:22 module 0x88 (DB) initialized successfully
22.04.26 10:00:22 PP: Setting PIN pad type to AUTO1
22.04.26 10:00:22 module 0x82 (PP) initialized successfully
22.04.26 10:00:22 module 0x85 (SC) initialized successfully
22.04.26 10:00:22 module 0x96 (MBK) initialized successfully
22.04.26 10:00:22 module 0x69 (MBK_EI) initialized successfully
```

```
22.04.26 10:00:22 module 0x68 (CXI) initialized successfully
22.04.26 10:00:22 module 0x87 (ADM) initialized successfully
22.04.26 10:00:22 module 0x9a (NTP) initialized successfully
```

6.4 Get the IP address of CryptoServer Simulator

When the container is running properly you can connect to this simulator using the csadm console application. To do this you need to get the current IP address of the running container. To determine the IP address, use the docker inspect command.

The next command requests the IP address of our cssim453 container.

›_ Console

```
# docker inspect cssim453 | grep -m 1 \"IPAddress\"
"IPAddress": "172.17.0.2",
```

Now you can connect to the CryptoServer Simulator via this IP address using csadm. You can run a csadm GetState command to ensure the basic connection is successful and retrieve some status information from the CryptoServer Simulator.

›_ Console

```
# ./csadm Dev=172.17.0.2 GetState

mode = Operational Mode
state = INITIALIZED (0x00100004)
temp = 30.0 [C]
alarm = OFF
bl_ver = 5.01.6.0 (Model: Se-Series Gen2)
hw_ver = 0.00.8.15
uid = 68534d32 35317f53 |hSM251 S |
adm1 = 5554494d 41434f20 53493030 30323838 |UTIMACO SI000288|
adm2 = 53696d75 6c61746f 72000000 00000000 |Simulator |
adm3 = 496e6974 2d446576 2d312d4b 65790000 |Init-Dev-1-Key |
```

6.5 Connecting to the Container

When the container was created in above steps, we can connect to the container using the docker command along with the ID of that container.

›_ Console

```
# docker exec -i -t e4ba909b69ca sh
# pwd
/simulator/bin
# ls -ltr
total 128
-rwxr-xr-x 1 root root     70 Apr 22 05:27 cs_sim.sh
-rwxr-xr-x 1 root root      0 Apr 22 05:27 cs_sim.ini
```

```
-rwxr-xr-x 1 root root   1275 Apr 22 05:27 cs_multi.sh
-rwxr-xr-x 1 root root 120760 Apr 22 05:27 bl_sim5
```

Here the value e4ba909b69ca is the container ID derived in the ***section 6.2***

6.6 Stop the CryptoServer Simulator Container

When the container is no longer needed, it can be removed via the docker container rm command. For docker container you have started earlier that requires the next command to stop the container from running.

›_ Console

```
# docker container stop cssim453
cssim453
```

7 Troubleshooting

Error	*Diagnosis*
LoginUser= failed: 05.12.2021 23:45:45 src/p11adm_R2.c[429] p11_login: C_Login [type=1] returned Error 0x00000102 (CKR_USER_PIN_NOT_INITIALIZED)	PKCS#11 Slot is not initialized.

Table 6: List of Error and its Diagnosis

8 Further Information

This document forms a part of the information and support which is provided by the Utimaco IS GmbH. Additional documentation can be found on the product CD in the Documentation directory.

All CryptoServer product documentation is also available at the Utimaco IS GmbH website:
https://utimaco.com/

References

Reference	*Title/Company*	*Document No.*
[CSADMIN]	CryptoServer – csadm Manual/Utimaco IS GmbH	2009-0003
[CSTrSh]	CryptoServer Troubleshooting/Utimaco IS GmbH	M011-0008-en

Contact

Utimaco IS GmbH
Germanusstr. 4
D-52080 Aachen
Germany

Phone: AMERICAS: +1-844-UTIMACO (+1 844-884-6226)
EMEA: +49 800-627-3081
APAC: +81 800-919-1301

Web: https://support.utimaco.com

Email: support@utimaco.com