VeeaHub ToolKit R0.5: Container Flow

Contents

Preface

Information in this document is provided solely in connection with Veea Inc. and its affiliates (collectively "Veea") products. Veeareserves the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

Use of Veea products and services is subject to the terms of use and/or separate agreements and warranties applicable to those products and services. Please visit www.veea.com/legal for these terms.

Evaluators (as defined in the Evaluation Agreement) are solely responsible for the choice, selection and use of the Veea products and services described herein, and Veea assumes no liability whatsoever relating to the choice, selection or use of the Veea products and services described herein.

Trademark Credits

Copyright Information and Restrictions

Document Feedback

Veea welcomes your suggestions for improving our documentation. If you have comments, send your feedback to: support@veeahub.com

Document History

Issue	Issue Date	Author	Description
1	2019-10-30	RB	First published version for VeeaHub Toolkit Release 0.5

Reference Documents

Document Ref.	Issue	Document Title
[RD/1]		VeeaHub Toolkit Guide (Veea Systems Ltd)
[RD/2]		VeeaHub ToolKit Container Design (Veea Systems Ltd)

Reference Software

Software Ref.	Release	Description/Title
[RS/1]		Installation file: *https://c3templates.veeaplatform.net/0.5.0/setup-veeahub-dev-env_0.5.0.sh*

1. Introduction

1.1. Purpose of this Document

This document describes the processes of developing custom applications for the VeeaHub platform.

It should be read in conjunction with the document on installing and using the VeeaHub Toolkit [RD/1] and the Container Design document [RD/2].

This document covers release 0.5 of the VeeaHub Toolkit software.

1.2. Terminology

VHT	VeeaHub Toolkit
VHC	VeeaHub Client or “the client” included as part of VHT
VH	VeeaHub
Template	A zip archive containing a fully functional application that can be modified.

1.3. VeeaHub Client: VHC

VeeaHub Client is a command-line utility designed to help developers create, build, and deploy applications to the VeeaHub platform.

The client is built to run on Ubuntu Linux 18.04.3 or later.

1.4. Organisation of this Document

- Section 2 covers requirements and the process for getting set up to produce secure and certified apps for the VeeaHub.
- Section 3 covers how to build and certify containers. This should be read in conjunction with [RD/1].
- Section 4 lists the commands in VeeaHub Toolkit for working with secure containers.
- Section 6 gives the API for sideloading containers on to the VeeaHub.

1.5. Signed vs Unsigned Applications

Unsigned applications can be developed by anyone and allow the wider Linux community to benefit from developing new functionality to work and operate on a mesh-based platform.

Signed applications allow Veea commercial partners to develop applications that will benefit from the additional security features provided on the VH platform.

The signing process for a container, and its associated app, enables the following:

- It allows for the application to be hosted on Veea’s servers and downloaded by the user through Control Center or (by enterprise users) from Enterprise Center.
- It enables the container to run on the VeeaHub with higher privileges, enabling various kinds of behavior that are not available to unsigned containers.

The VeeaHub can be configured to run unsigned containers to run. This is currently the default configuration when the hub is bootstrapped. This is normally used for container development but should be disabled for deployed VeeaHubs.

By default, a container will not run on a VeeaHub that allows unsigned containers; this can be changed by setting the **allowOnUnauthenticatedHost** label in the VHC (see section 4.2.3).

Application developers can:

- Access their containers
- Access services that they choose to install within their containers, for example, a database or a web server.
- Can run, monitor and diagnose the container.
- Obtain debug and logging information for the app from the MEN.

Developers of signed containers can additionally:

- Build signed containers using a Hardware Signing Module (HSM)

1.6. Documentation

This is one of three documents describing VeeaHub Toolkit release 0.5.

[RD/1] describes the installation and use of VeeaHub Toolkit.

[RD/2] describes guidelines for creating containers for the VeeaHub.

2. Requirements for Secure Containers on the VeeaHub

2.1. The VeeaHub Security Model

VeeaHubs run applications that are based on the open **containerd** format. Veea's implementation of the **containerd** standard enforces a secure environment. The VeeaHub uses enterprise-level cryptography to protect developers' host integrity as well as the integrity of the authenticated applications.

These security restrictions will:

- use hardware security modules (HSM) to enforce the root of trust
- ensure that the integrity of the container can always be validated
- ensure that containers are only run in the ways intended by the container developer
- ensure that containers share only the data that is intended to be shared (including running processes, environment, data stored on disk, and networked data)
- block unauthorized access to the containers

The security restrictions will not:

- prevent reasonable operation of the containers
- prevent authorized access to devices
- prevent a developer from debugging processes running within their container or the start-up and shutdown of their container

Part of the security model on the VeeaHub is a system of licenses. These licenses are provided by Veea for specific purposes, and relate to:

- Access permissions for devices
- Access permissions for elevated container execution privileges
- Additional privileges for the container when performing Linux system calls

2.2. Veea Customer Support

Contact Veea Customer Support at the address in section 7. You may also wish to sign up for the developer support community at https://developers.veea.com/.

You will require from Veea Support:

- **A partner number**. This may already have been assigned to your company.
- **A secure USB token.** A Hardware Signing Module (HSM) will be supplied with specific instructions for using it. This is required for certification of your secure containers.

In return, you should provide the following to Veea Support by email:

- The file *partner_certificates.tgz* created in section 2.3.
- A list of devices you intend to develop applications for.

Veea Customer Support will then email you the relevant licenses in accordance with the SLA.

2.3. Generating Credentials

The Toolkit must first be installed as described in [RD/1]. The Secure Container Tools are installed as part of the Toolkit installation process. They include custom versions of both OpenSC and PKCS11 as well as a script used to generate credentials and set up the HSM required for secure development.

Insert the HSM into a USB drive that can be accessed by the Ubuntu virtual machine that you are using for development.

Run the script *generate-veeahub-credentials_0.5.0.sh* that was downloaded as part of the Toolkit and is located in */opt/veea/vht/bin.*

```
$ generate-veeahub-credentials_0.5.0.sh

************************************************************
VeeaHub Development Key and Certificate Utility
************************************************************


Empty secure token... initializing.
Erasing and setting defaults for the secure token...
Please enter a six (6) digit PIN (for example, 123456): <PIN>
Please enter a sixteen (16) hex-digit SO-PIN (for example, 0123456789012345): <SO-PIN>
Initializing the secure token...
Enter your Contact Name: <contact-name>
Enter your Organization Name: <organization-name>
Enter your Organizational Unit: <organizational-unit>
Enter your Email address: <email-address>
Enter your Domain Component: <domain-component>
Enter your City or Town: <city>
Enter your State or Province: <state>
Enter your Country: <country>
Generating the Root Authority Key...
Generating the VHC Networking Certificate (vhc-nw-cert.pem)...
Generating the VHC TLS Key...
Generating the VHC TLS CSR (vhc-tls-csr.pem)...
Generating the VHC TLS Certificate (vhc-tls-cert.pem)...
Generating the VHC Signing Key...
Generating the VHC Signing Certificate (vhc-signing-cert.pem)...
Updating developer configuration...

*************************************************************
Key and certificate generation was successful.
Please email partner_certificates.tgz to support@veeahub.com.
*************************************************************
```

Record your PINs in a secure location.

This script creates the following certificates:

- VHC networking certificate (*vhc-nw-cert.pem*)
- VHC TLS certificate (*vhc-tls-cert.pem*)
- VHC signing certificate (*vhc-sign-cert.pem*)

It also creates the following private cryptographic keys within the HSM:

- Root authority key
- VHC TLS key
- VHC signing key

It adds the following items to the VHC global user configuration *~/.vhc/user-config.yaml*:

- TLS certificate and URL
- Signing certificate and URL

As indicated by the script, you should email the output tarball *partner_certificates.tgz* to Veea support, support@veea.com.

You should indicate the devices for which you require feature licenses. The list of available devices is available as part of the main VHC README document.

Veea Support will in return supply the following additional credentials:

- Partner license with partner ID, used for creating secure applications
- Transport license, used for secure hub communications
- One or more partner feature licenses, used for securely accessing hub resources

2.4. Installing the Licenses

The credentials need to be added to your VHC configuration and hub once they are received.

To add the partner license to the VHC global user configuration, run:

```
$ vhc secure credential --set-partner-lic /path/to/partner-license.txt
User configuration modified. Please rebuild image(s).
```

To add the transport license to the hub, run:

```
$ vhc hub licenses --add /path/to/transport-license.cms
Adding license file [partner-license.cms] to C05BCB00C0A000001127:9000
(https://10.20.0.90:9000/licenses)...
 3.26 KB / 3.26 KB
[======================================================================] 100.00%
Success
```

Once the transport license is added to the hub, reboot the hub for the changes to take effect:

```
$ vhc hub --reboot
Rebooting C05BCB00C0A000001127:9000 (https://10.20.0.90:9000/reboot)...
Success
```

When the transport license is in place and the hub rebooted, you should test that secure communication between VHC and the VeeaHub is working correctly by running the following command:

```
$ vhc hub --get-partner-id --tls-pin <PIN>
Retrieving Partner ID from C05BCB00C0A000001127:9000
(https://10.20.0.90:9000/partner/id)...
{
  "partner_id": "XXXXXXXX"
}
```

If everything is working correctly, VHC should display a value for the returned partner ID other than the insecure value of `FFFFFFFF`.

To add partner feature licenses to the VHC application configuration, run the following command for each partner feature file:

```
$ vhc secure credential --add /path/to/partner-featureX.txt
Project configuration modified. Please rebuild image(s).
```

The secure credential commands are described in greater detail in the next section.

2.5. Validating the Secure Configuration

Before building the containers, you should validate the secure configuration using the **--validate** option of the **secure** command.

```
$ vhc secure --validate
=> Checking system tools...
  OK   - openssl: installed
  OK   - sign image: /opt/veea/vht/bin/sign-image.py
  OK   - verify image: /opt/veea/vht/bin/verify-image.py
  OK   - Veea Container: /opt/veea/vht/bin/VeeaContainer.py
  OK   - Veea Crypto: /opt/veea/vht/bin/VeeaCrypto.py
=> Checking common credentials...
  OK   - license authority: /opt/veea/vht/cert/veeahub-license-authority.pem
  OK   - license server: /opt/veea/vht/cert/veeahub-license-server.pem
  OK   - client certificate: /path/to/client-cert.pem
  OK   - client key: /path/to/client-priv.pem
  OK   - partner certificate: /path/to/partner-cert.pem
  OK   - partner license: /path/to/partner-license.txt
  OK   - partner ID: 00000001
  OK   - token URL: set
  OK   - token PIN: set
  OK   - token key: set
=> Checking application credentials...
  OK   - no partner features configured
=> Checking application options...
  OK   - allowOnUnathenticatedHost: true
  OK   - UUID: a1654e88-f683-43b5-8ce5-e09711cb6314
=> Checking application devices...
  OK   - no devices configured
=> Checking application features...
  OK   - DEVELOPER
=> Checking application persistant volumes...
  OK   - no persistant volumes configured
DONE!
```

This identifies any devices for which you do not have licenses.

3. Process for Secure Containers on the VeeaHub

3.1. Workflow

This is the general workflow when creating secure containers:

1. Work with Veea customer support to obtain the necessary credentials for the devices you need (section 2.2).
2. Set the credentials paths etc. in the secure configuration (section 2.3).
3. Set up base container options as described in the VeeaHub Toolkit document [RD/1], with reference to the Container Design guidelines.
4. Set the UUID in the secure configuration (section 4.2.2).
5. Add devices to the secure configuration (section 4.4).
6. Add volume persists, offers, and requests to the secure configuration (sections 4.7, 4.8 and 4.9).
7. Build the container(s) for your target(s) (section 5.2).
8. Deploy the container(s) on your target(s) (section 3.2).

Note: In release 0.5 of the software, the previously separate steps **generate/build/sign/verify** are a single step.

3.2. Deploying

For test deployments, see Sideloading, section 6.

For production deployment to Control Center, you must provide Veea Support with the container files and the Application Metadata (JSON) file: see section 3.3.

3.3. Compatibility Information

The partner must release a *application.metadata.cms* file, which is required for the deployment on the Veea Cloud. This is a signed JSON file in Cryptographic Message Syntax (DER) format that must include the following information about the application release:

- The application name
- The application description
- The application version which should adhere to the Semantic Version standard (semver.org)
- Contact details for your company
-
- As an option, you can also specify:
- Which VeeaHub hardware platform your software supports, for example, VHC05, VHE09, etc
- If there are any supporting apps, for example, pre-requisites
- If there are any licensing requirements
- If there are any hardware requirements, for example, Zigbee support

An example of the JSON file:

```
{
  "components": [
    {
      "sha256": "4f86a4519a639aaf41a31603548c3252b66825aa5e3ddf981075e77259c19558",
      "size": 68950088,
      "source": "database-iesv1.0.tar.gz",
      "uuid": "8695c938-0cb1-4429-8eca-d8442dca8168"
    }
  ],
  "conflicts": {},
  "hardware_list": [
    {
      "platform": "iesv0.9"
    },
    {
      "platform": "iesv1.0"
    }
  ],
  "files": [],
  "metadata": {
    "tags": [
      "DEVELOPER"
    ],
    "timestamp": 1557827496
  },
  "provides": {
    "description": "Provides the Veea Database",
    "name": "com.veea.database",
    "platform": "iesv1.0",
    "uuid": "00000000-BA5E-5EED-AA64-000000000002"
  },
  "requires": [
    {
      "type": "system",
      "min_version": "2.2.1"
    },
    {
      "type": "com.veea.system",
      "min_version": "2.2.1"
    },
    {
      "type": "com.veea.network",
      "min_version": "2.2.1"
    }
  },
  "runspec": {
    "docker": {
      "extract": {
        "${BUILD_NAME}": {
          "args": "--init --detach --net=host --restart=always --oom-score-adj=-10",
          "cmd": "database",
          "name": "${BUILD_NAME}${BUILD_TAG}",
          "ports": "5984:5984",
          "repo": "${BUILD_REPO}",
          "tag": "${BUILD_TAG}",
          "volumes": "/var/run:/var/run /vmrun/db:/vmrun/db /vmrun/node:/vmrun/node
/srv/node:/srv/node"
        }
      },
      "initial-install": {
        "${BUILD_NAME}": {
          "args": "--tty --net=host",
          "cmd": "db-install",
          "name": "${BUILD_NAME}${BUILD_TAG}_initial_install",
          "repo": "${BUILD_REPO}",
          "tag": "${BUILD_TAG}",
          "volumes": "/var/run:/var/run /vmrun/db:/vmrun/db /vmrun/node:/vmrun/node
/srv/node:/srv/node"
        }
      },
      "service": {
        "${BUILD_NAME}": {
          "mode": "global",
          "name": "${BUILD_NAME}${BUILD_TAG}",
          "repo": "${BUILD_REPO}",
          "tag": "${BUILD_TAG}",
          "volumes": "/var/run:/var/run /srv/node:/srv/node"
        }
      }
    }
  },
  "seccomp": {},
  "type": "container",
  "uuid": "7d2bdbf1-b16a-475f-985a-2ff25e712421",
  "vendor": {
    "address": [
      "Veea Systems Ltd.",
      "Cambridge House",
      "Henry Street",
      "Bath",
      "BA1 1JS",
      "UK"
    ],
    "email": "support@veea.com",
    "name": "Veea Systems Limited",
    "uuid": "9549ad36-6687-4e72-8ec2-6f454e33d476",
    "web": "https://veea.com"
  },
  "version": "2.2.0-35"
}
{
  "components": [
    {
      "sha256": "4f86a4519a639aaf41a31603548c3252b66825aa5e3ddf981075e77259c19558",
      "size": 68950088,
      "source": "database-iesv1.0.tar.gz",
      "uuid": "8695c938-0cb1-4429-8eca-d8442dca8168"
    }
  ],
  "conflicts": {},
  "hardware_list": [
    {
      "platform": "iesv0.9"
    },
    {
      "platform": "iesv1.0"
    }
  ],
  "files": [],
  "metadata": {
    "tags": [
      "DEVELOPER"
    ],
    "timestamp": 1557827496
  },
  "provides": {
    "description": "Provides the Veea Database",
    "name": "com.veea.database",
    "platform": "iesv1.0",
    "uuid": "00000000-BA5E-5EED-AA64-000000000002"
  },
  "requires": [
    {
      "type": "system",
      "min_version": "2.2.1"
    },
    {
      "type": "com.veea.system",
      "min_version": "2.2.1"
    },
    {
      "type": "com.veea.network",
      "min_version": "2.2.1"
    }
  },
  "runspec": {
    "docker": {
      "extract": {
        "${BUILD_NAME}": {
          "args": "--init --detach --net=host --restart=always --oom-score-adj=-10",
          "cmd": "database",
          "name": "${BUILD_NAME}${BUILD_TAG}",
          "ports": "5984:5984",
          "repo": "${BUILD_REPO}",
          "tag": "${BUILD_TAG}",
          "volumes": "/var/run:/var/run /vmrun/db:/vmrun/db /vmrun/node:/vmrun/node
/srv/node:/srv/node"
        }
      },
      "initial-install": {
        "${BUILD_NAME}": {
          "args": "--tty --net=host",
          "cmd": "db-install",
          "name": "${BUILD_NAME}${BUILD_TAG}_initial_install",
          "repo": "${BUILD_REPO}",
          "tag": "${BUILD_TAG}",
          "volumes": "/var/run:/var/run /vmrun/db:/vmrun/db /vmrun/node:/vmrun/node
/srv/node:/srv/node"
        }
      },
      "service": {
        "${BUILD_NAME}": {
          "mode": "global",
          "name": "${BUILD_NAME}${BUILD_TAG}",
          "repo": "${BUILD_REPO}",
          "tag": "${BUILD_TAG}",
          "volumes": "/var/run:/var/run /srv/node:/srv/node"
        }
      }
    }
  },
  "seccomp": {},
  "type": "container",
  "uuid": "7d2bdbf1-b16a-475f-985a-2ff25e712421",
  "vendor": {
    "address": [
      "Veea Systems Ltd.",
      "Cambridge House",
      "Henry Street",
      "Bath",
      "BA1 1JS",
      "UK"
    ],
    "email": "support@veea.com",
    "name": "Veea Systems Limited",
    "uuid": "9549ad36-6687-4e72-8ec2-6f454e33d476",
    "web": "https://veea.com"
  },
  "version": "2.2.0-35"
}
```

4. VHC Secure Container Commands

4.1. Commands

These commands are available to set secure configuration options:

- secure
  - credential
  - device
  - feature
  - volumes
    - offer
    - persist
    - request

4.2. *secure* Command

4.2.1. Usage

The **secure** command is the topmost command for setting secure configuration options.

To display general usage for the secure container functionality within **vhc**, run:

```
$ vhc secure --help
Adds and removes entries to the secure portion of the configuration.
Also used to generate/set the UUID and enable/disable allowOnUnauthorisedHosts.
Use the 'show' option to display configured arguments.
Please use the 'validate' option before generating the Dockerfile.

Usage:
  vhc secure [flags]
  vhc secure [command]

Examples:
  vhc secure --auth-host
  vhc secure --unauth-host
  vhc secure --gen-uuid
  vhc secure --set-uuid <uuid>
  vhc secure --clear-uuid (USE WITH CAUTION!)
  vhc secure --show
  vhc secure --validate

Available Commands:
  credential  Adds/removes partner credentials to/from the secure container
configuration.
  device      Adds/removes devices to/from the secure container configuration.
  feature     Adds/removes features to/from the secure container configuration.
  volumes     Adds/removes volume entries to/from the secure container configuration.

Flags:
  -a, --auth-host         Allows the container to only run on an authorized host.
  -c, --clear-uuid        Clears the UUID in the secure container configuration.
  -g, --gen-uuid          Generates a UUID for the secure container configuration.
  -h, --help              help for secure
  -S, --set-uuid string   Sets the UUID for the secure container configuration.
  -s, --show            Shows configured options.
  -u, --unauth-host     Allows the container to run on an unauthorized host.
  -v, --validate        Validates the secure container configuration.

Use "vhc secure [command] --help" for more information about a command.
```

Note: Please note the spelling of **allowOnUnauthorisedHosts**.

Each command is presented in more detail in the following sections.

4.2.2. UUID Options

To have VHC auto-generate the UUID run:

```
$ vhc secure --gen-uuid
Generated UUID: EBB1160E-E686-4C20-982D-5028EDF97B5A
Project configuration modified. Please rebuild image(s).
```

To set your own UUID run:

```
$ vhc secure --set-uuid EBB1160E-E686-4C20-982D-5028EDF97B5A
Set UUID: EBB1160E-E686-4C20-982D-5028EDF97B5A
Project configuration modified. Please rebuild image(s).
```

Once a UUID is set via either method, to set a new value you must explicitly clear the existing value. This is a safety mechanism to prevent downstream problems with licenses and volumes being changed unnecessarily, because internally they depend on the UUID.

To clear the UUID run:

```
$ vhc secure --clear-uuid
UUID cleared. Please generate or set the UUID before building.
Project configuration modified. Please rebuild image(s).
```

4.2.3. Unauthorized Hosts Options

To enable the secure container to run on unauthorized hosts, run:

```
$ vhc secure --unauth-host
Project configuration modified. Please rebuild image(s).
```

To disable the secure container from running on unauthorized hosts (required for real deployments), run:

```
$ vhc secure --auth-host
Project configuration modified. Please rebuild image(s).
```

4.2.4. Showing Options

To display options configured with the **secure** command, run:

```
$ vhc secure --show
Secure Options:
```

```
+-------------+--------------------------------------+
|    NAME     |                VALUE                 |
+-------------+--------------------------------------+
| unauth_host | false                                |
| uuid        | EBB1160E-E686-4C20-982D-5028EDF97B5A |
+-------------+--------------------------------------+
```

4.3. *secure credential* Command

4.3.1. Usage

The **credential** command is a sub-command of **secure**. It adds and removes credentials to and from the secure configuration. Depending on what devices are needed and when they are licensed, more than one feature license may need to be added. In this case the path and name of each license file should be unique.

To display help/usage for the **credential** command run:

```
$ vhc secure credential --help
Adds/removes partner credentials to/from the secure container configuration.
Use the 'show' option to display configured credentials.

When signing and verifying application images, vhc expects that the
following tools and credentials are present:

    /opt/veea/vht/bin/sign-image.py
    /opt/veea/vht/bin/verify-image.py
    /opt/veea/vht/cert/veeahub-license-authority.pem
    /opt/veea/vht/cert/veeahub-license-server.pem

The location of various partner credentials must be specified using this
command. Note that partner features are stored in the current project
configuration (i.e. ./config.yaml), while all other credentials are stored
in the user configuration (i.e. ~/.vhc/user-config.yaml).

URLs should be enclosed in double-quotes, i.e. "<url>".

Usage:
  vhc secure credential [flags]

Examples:
  vhc secure credential --clear-partner-lic
  vhc secure credential --set-partner-lic /path/to/partner/license
  vhc secure credential --clear-sign-cert
  vhc secure credential --set-sign-cert /path/to/signing/certificate
  vhc secure credential --clear-sign-url
  vhc secure credential --set-sign-url "<url>"
  vhc secure credential --clear-tls-cert
  vhc secure credential --set-tls-cert /path/to/tls/certificate
  vhc secure credential --clear-tls-url
  vhc secure credential --set-tls-url "<url>"
  vhc secure credential --add-partner-feature /path/to/partner/feature
  vhc secure credential --remove-partner-feature /path/to/partner/feature
  vhc secure credential --show

Flags:
  -f, --add-partner-feature string      Adds the Partner Feature to the configuration.
  -L, --clear-partner-lic                    Clears the Partner License for the
configuration.
  -C, --clear-sign-cert                      Clears the Signing Certificate for the
configuration.
  -U, --clear-sign-url                       Clears the Signing URL for the configuration.
  -D, --clear-tls-cert                       Clears the TLS Certificate for the
configuration.
  -V, --clear-tls-url                        Clears the TLS URL for the configuration.
  -h, --help                                 help for credential
  -F, --remove-partner-feature string        Removes the Partner Feature from the
configuration.
  -l, --set-partner-lic string               Sets the Partner License for the configuration.
  -c, --set-sign-cert string                 Sets the Signing Certificate for the
configuration.
  -u, --set-sign-url string                  Sets the Signing URL for the configuration.
  -d, --set-tls-cert string                  Sets the TLS Certificate for the configuration.
  -v, --set-tls-url string                   Sets the TLS URL for the configuration.
  -s, --show                                 Shows configured credentials.
```

4.3.2. Partner License Options

To set the partner license for the secure configuration run:

```
$ vhc secure credential --set-partner-lic /path/to/partner-license.txt
User configuration modified. Please rebuild image(s).
```

To clear the partner license for the secure configuration run:

```
$ vhc secure credential --clear-partner-lic
User configuration modified. Please rebuild image(s).
```

4.3.3. Signing Certificate Options

To set the signing certificate for the secure configuration run:

```
$ vhc secure credential --set-sign-cert /path/to/vhc-signing-cert.pem
User configuration modified. Please rebuild image(s).
```

To clear the signing certificate for the secure configuration run:

```
$ vhc secure credential --clear-sign-cert
User configuration modified. Please rebuild image(s).
```

4.3.4. Setting Signing URL

For secure containers the signing and TLS URLs must be set. These URLs were created when the token was initially set up. To get the URL after the initial setup, run the following command (**note:** serial and ID have been removed, yours will be the actual values) and use the **second** URL for TLS and the **third** for signing:

```
$ p11tool --login --provider=/opt/veea/tools/lib/x86_64-linux-gnu/opensc-pkcs11.so --
list-privkeys
Token 'UserPIN (SmartCard-HSM)' with URL
'pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;toke
n=UserPIN%20%28SmartCard-HSM%29' requires user PIN
```

```
Enter PIN:
Object 0:
   URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=root_auth_key;type=private
   Type: Private key
   Label: root_auth_key
   Flags: CKA_PRIVATE; CKA_NEVER_EXTRACTABLE; CKA_SENSITIVE;
   ID: <id>

Object 1:
   URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=tls_key;type=private
   Type: Private key
   Label: tls_key
   Flags: CKA_PRIVATE; CKA_NEVER_EXTRACTABLE; CKA_SENSITIVE;
   ID: <id>

Object 2:
   URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=signing_key;type=private
   Type: Private key
   Label: signing_key
   Flags: CKA_PRIVATE; CKA_NEVER_EXTRACTABLE; CKA_SENSITIVE;
   ID: <id>
```

To set the signing URL for the secure configuration, run:

```
$ vhc secure credential --set-sign-url
"pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;toke
n=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=signing_key;type=private"
User configuration modified. Please rebuild image(s).
```

Note that the URL must be enclosed in double quotes: "<url>".

4.3.5. Clearing Signing URL_**

To clear the signing URL for the secure configuration, run:

```
$ vhc secure credential --clear-sign-url
User configuration modified. Please rebuild image(s).
```

4.3.6. TLS Certificate Options

To set the TLS certificate for the secure configuration, run:

```
$ vhc secure credential --set-sign-cert /path/to/vhc-tls-cert.pem
User configuration modified. Please rebuild image(s).
```

To clear the TLS certificate for the secure configuration run:

```
$ vhc secure credential --clear-sign-cert
User configuration modified. Please rebuild image(s).
```

4.3.7. TLS URL Options

To set the TLS URL for the secure configuration, run:

```
$ vhc secure credential --set-tls-url
"pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;toke
n=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=tls_key;type=private"
User configuration modified. Please rebuild image(s).
```

To clear the TLS URL for the secure configuration, run:

```
$ vhc secure credential --clear-tls-url
User configuration modified. Please rebuild image(s).
```

4.3.8. Partner Feature Options

To add a partner feature to the secure configuration run:

```
$ vhc secure credential --add /path/to/partner-feature1.txt
Project configuration modified. Please rebuild image(s).
```

To remove a partner feature from the secure configuration run:

```
$ vhc secure credential --remove /path/to/partner-feature1.txt
Project configuration modified. Please rebuild image(s).
```

4.3.9. Showing Configured Credentials

To display the configured credentials with the **credential** command run:

```
$ vhc secure credential --show
Secure Credentials:
+---------------------+-------------------------------+
|     CREDENTIAL      |             PATH              |
+---------------------+-------------------------------+
| Partner License     | /path/to/partner-license.txt  |
| Signing Certificate | /path/to/vhc-signing-cert.pem |
| TLS Certificate     | /path/to/vhc-tls-cert.pem     |
| Partner Feature 0   | /path/to/partner-feature1.txt |
+---------------------+-------------------------------+

Signing URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=signing_key;type=private
TLS URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=tls_key;type=private
```

4.4. *secure device* Command

4.4.1. Usage

The **device** command is a sub-command of **secure** and enables users to add and remove devices to and from the secure configuration.

To display help/usage for the **device** command, run:

```
$ vhc secure device --help
Adds/removes devices to/from the secure container configuration.
Use the 'list' option to display available device options.
Use the 'show' option to display configured devices.

Usage:
  vhc secure device [flags] [device]

Examples:
  vhc secure device --add exclusive-zwave
  vhc secure device --remove exclusive-zwave
  vhc secure device --list
  vhc secure device --show

Flags:
  -a, --add      Adds a device to the secure configuration.
  -h, --help     help for device
  -l, --list     Lists available device options.
  -r, --remove   Removes a device from the secure configuration.
  -s, --show     Shows configured devices.
```

4.4.2. Listing Available Devices

To display the list of available devices, run:

```
$ vhc secure device --list
Available Secure Devices:
+----------------------------+-----------+
|           NAME             | TARGET(S) |
+----------------------------+-----------+
| DEV:EXCLUSIVE:audio        |   vhc05   |
| DEV:EXCLUSIVE:bluetooth    |    all    |
| DEV:EXCLUSIVE:ext-serial0  | vhx09-10  |
| DEV:EXCLUSIVE:movidius     | vhx09-10  |
| DEV:EXCLUSIVE:video        |   vhc05   |
| DEV:EXCLUSIVE:webcam       |    all    |
| DEV:EXCLUSIVE:zigbee       |    all    |
| DEV:EXCLUSIVE:zwave        |    all    |
| DEV:SHARED:bluetooth       |    all    |
| DEV:SHARED:movidius        | vhx09-10  |
| DEV:SHARED:webcam          |    all    |
| DEV:SHARED:zigbee          |    all    |
| DEV:SHARED:zwave           |    all    |
+----------------------------+-----------+
```

4.4.3. Adding and Removing a Device

To add a device to the secure configuration, run (as example):

```
$ vhc secure device --add DEV:EXCLUSIVE:zwave
Project configuration modified. Please rebuild image(s).
```

To remove a device from the secure configuration, run (as example):

```
$ vhc secure device --remove DEV:EXCLUSIVE:zwave
Project configuration modified. Please rebuild image(s).
```

4.4.4. Showing Configured Devices

To display the devices configured with the **device** command, run:

```
$ vhc secure device --show
Secure Devices:
+----------------------+
|         NAME         |
+----------------------+
| DEV:EXCLUSIVE:zwave |
+----------------------+
```

4.5. *secure feature* Command

4.5.1. Usage

The **feature** command is a sub-command of **secure** and enables users to add and remove features to and from the secure configuration.

To display help/usage for the **feature** command, run:

```
$ vhc secure feature --help
Adds/removes features to/from the secure container configuration.
Use the 'list' option to display available features.
Use the 'show' option to display configured features.

Usage:
  vhc secure feature [flags] [feature]

Examples:
  vhc secure feature --add developer
  vhc secure feature --remove developer
  vhc secure feature --list
  vhc secure feature --show

Flags:
  -a, --add      Adds a feature to the secure configuration.
  -h, --help     help for feature
  -l, --list     Lists available features.
  -r, --remove   Removes a feature from the secure configuration.
  -s, --show     Shows configured features.
```

4.5.2. Listing Available Features

To display the list of available features, run:

```
$ vhc secure feature --list
Available Secure Features:
+-----------+
|   NAME    |
+-----------+
| DEVELOPER |
+-----------+
```

4.5.3. Adding or Removing a Feature

To add a feature to the secure configuration, run:

```
$ vhc secure feature --add DEVELOPER
Project configuration modified. Please rebuild image(s).
```

To remove a feature from the secure configuration, run:

```
$ vhc secure feature --remove DEVELOPER
Project configuration modified. Please rebuild image(s).
```

4.5.4. Showing Configured Features

To display the features configured with the **feature** command, run:

```
$ vhc secure feature --show
Secure Features:
+-----------+
|   NAME    |
+-----------+
| DEVELOPER |
+-----------+
```

4.6. *secure volumes* Command

The **volumes** command is a sub-command of **secure** and allows users to add and remove volume entries (offers, persists, and requests) to and from the secure configuration.

To display help/usage for the **volumes** command, run:

```
$ vhc secure volumes --help
Adds/removes volume entries to/from the secure container configuration.

Usage:
  vhc secure volumes [command]

Available Commands:
  offer       Adds/removes offers to/from the secure container configuration.
  persist     Adds/removes persistent volumes to/from the secure container
configuration.
  request     Adds/removes requests to/from the secure configuration configuration.

Flags:
  -h, --help   help for volumes

Use "vhc secure volumes [command] --help" for more information about a command.
```

4.7. *secure volumes offer* Command

4.7.1. Usage

The offer command is a sub-command of secure volumes and allows users to add and remove offers to/from the secure configuration.

```
$ vhc secure volumes offer --help
Adds/removes offers to/from the secure container configuration.
Use the show option to display configured offers.

Usage:
  vhc secure volumes offer [flags] [offer]

Examples:
  vhc secure volumes offer --add foo
  vhc secure volumes offer --remove foo
  vhc secure volumes offer --show

Flags:
  -a, --add      Adds a offer to the secure configuration.
  -h, --help     help for offer
  -r, --remove   Removes a offer from the secure configuration.
  -s, --show     Shows configured offers.
```

4.7.2. Adding and Removing Volume Offers

To add an offer to the secure configuration, run:

```
$ vhc secure volumes offer --add <…>
```

To remove an offer from the secure configuration, run:

```
$ vhc secure volumes offer --remove <…>
```

4.7.3. Showing Configured Volume Offers

To display the offer configured with the **offer** command, run:

```
$ vhc secure volumes offer --show
```

4.8. *secure volumes persist* Command

4.8.1. Usage

The **persist** command is a sub-command of **secure volumes** and allows users to add and remove persists to/from the secure configuration.

```
$ vhc secure volumes persist --help
Adds/removes persistent volumes to/from the secure container configuration.
Use the 'show' option to display configured persistent volumes.

Usage:
  vhc secure volumes persist [flags] [persist]

Examples:
  vhc secure volumes persist --add <volume-name>
  vhc secure volumes persist --remove <volume-name>
  vhc secure volumes persist --show

Flags:
  -a, --add      Adds a persistent volume to the secure configuration.
  -h, --help     help for persist
  -r, --remove   Removes a persistent volume from the secure configuration.
  -s, --show     Shows configured persistent volumes.
```

4.8.2. Adding or Removing Volume Persists

To add a volume persist to the secure configuration, run:

```
$ vhc secure volumes persist --add my-directory
Project configuration modified. Please rebuild image(s).
```

To remove a volume persist from the secure configuration, run:

```
$ vhc secure volumes persist --remove my-directory
Project configuration modified. Please rebuild image(s).
```

4.8.3. Showing Configured Volume Persists

To display the volume persists configured with the **persist** command, run:

```
$ vhc secure volumes persist --show
Secure Persists:
+--------------+
|     NAME     |
+--------------+
| my-directory |
+--------------+
```

4.9. *secure volumes request* Command

4.9.1. Usage

The **request** command is a sub-command of **secure volumes** and allows users to add and remove requests to/from the secure configuration.

```
$ vhc secure volumes request --help
Adds/removes requests to/from the secure configuration configuration.
Use the 'show' option to display configured requests.

Usage:
  vhc secure volumes request [flags] [request]

Examples:
  vhc secure volumes request --add foo
  vhc secure volumes request --remove foo
  vhc secure volumes request --show

Flags:
  -a, --add      Adds a request to the secure configuration.
  -h, --help     help for request
  -r, --remove   Removes a request from the secure configuration.
  -s, --show     Shows configured requests.
```

4.9.2. Adding or Removing Volume Requests

To add a volume request to the secure configuration, run:

```
$ vhc secure volumes request --add <…>
```

To remove a volume from to the secure configuration, run:

```
$ vhc secure volumes request --remove <…>
```

4.9.3. Showing Configured Volume Requests

```
$ vhc secure volumes request --show
```

5. Validating, Building and Deploying Secure Containers

5.1. Validating

Before building an application, you should validate the secure configuration using the **--validate** option to the **secure** command:

```
$ vhc secure --validate
=> Listing targets...
  OK    - vhx09-10, vhc05
=> Checking system tools...
  OK    - openssl: installed
  OK    - sign image: /opt/veea/vht/bin/sign-image.py
  OK    - verify image: /opt/veea/vht/bin/verify-image.py
  OK    - Veea Container: /opt/veea/vht/bin/VeeaContainer.py
  OK    - Veea Crypto: /opt/veea/vht/bin/VeeaCrypto.py
=> Checking common credentials...
  OK    - license authority: /opt/veea/vht/cert/veeahub-license-authority.pem
  OK    - license server: /opt/veea/vht/cert/veeahub-license-server.pem
  OK    - partner license: /path/to/partner-license.test.txt
  OK    - partner ID: XXXXXXXX
  OK    - signing certificate: /path/to/vhc-signing-cert.pem
  OK    - signing URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id<id>;object=signing_key;type=private
  OK    - TLS certificate: /path/to/vhc-tls-cert.pem
  OK    - TLS URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=tls_key;type=private
=> Checking application credentials...
  OK    - partner features 1: /path/to/partner-feature1.txt
=> Checking application options...
  OK    - allowOnUnathenticatedHost: true
  OK    - UUID: EBB1160E-E686-4C20-982D-5028EDF97B5A
=> Checking application capabilities...
  OK    - no capabilities configured
=> Checking application devices...
  OK    - no devices configured
=> Checking application privileges...
  OK    - no privileges configured
=> Checking application features...
  OK    - DEVELOPER
=> Checking application persistant volumes...
  OK    - no persistant volumes configured
DONE!
```

5.2. Building

To build the secure image, including signing and verifying, run:

```
$ vhc build --target vhc05 --sign-pin <pin>
Making vh_io_attach image for target vhc05...
Dockerfile.vhc05 does not exist.
Generating Dockerfile for Dockerfile.vhc05...
bin/vh_io_attach:vhc05.tar does not exist.
Building vh_io_attach image for vhc05...
Command Started
```

```
Sending build context to Docker daemon  19.97kB
Step 1/25 : FROM arm32v7/busybox:latest
 ---> 723ce4685637
Step 2/25 : RUN mkdir /app
 ---> Running in 1f21372141ea
Removing intermediate container 1f21372141ea
 ---> 7b296e46f49e
Step 3/25 : COPY src/ /app/
 ---> 4f33ee077fd0
Step 4/25 : WORKDIR /app
 ---> Running in ad7c7f54d609
Removing intermediate container ad7c7f54d609
 ---> 5321b169f47b
Step 5/25 : LABEL com.veea.vhc.target="vhc05"
 ---> Running in 5f213e8b7e65
Removing intermediate container 5f213e8b7e65
 ---> c10a2afaa25d
Step 6/25 : LABEL com.veea.vhc.version="0.5.0"
 ---> Running in 554cea0f1ac9
Removing intermediate container 554cea0f1ac9
 ---> 3e3f3869c6e3
Step 7/25 : LABEL com.veea.vhc.app.name="vh io attach"
 ---> Running in 83202f0aa8b7
Removing intermediate container 83202f0aa8b7
 ---> fd2d2189c27a
Step 8/25 : LABEL com.veea.vhc.app.version="1"
 ---> Running in 9983903a3296
Removing intermediate container 9983903a3296
 ---> 7829cad92f94
Step 9/25 : LABEL com.veea.vhc.config.proj.version="1"
 ---> Running in 71dd2bc9a0bf
Removing intermediate container 71dd2bc9a0bf
 ---> 1ad073676cbd
Step 10/25 : LABEL com.veea.vhc.config.user.version="1"
 ---> Running in 4ce471437939
Removing intermediate container 4ce471437939
 ---> e2badafa01d6
Step 11/25 : ARG PARTNER_LICENSE
 ---> Running in 154417e93b22
Removing intermediate container 154417e93b22
 ---> 2feb2dd60778
Step 12/25 : LABEL com.veea.authentication.identifier="$PARTNER_LICENSE"
 ---> Running in e6630eee004d
Removing intermediate container e6630eee004d
 ---> 54497345eac3
Step 13/25 : ARG PARTNER_ID
 ---> Running in 2037412a149b
Removing intermediate container 2037412a149b
 ---> c945d282bfe5
Step 14/25 : LABEL com.veea.image.persistent_uuid="$PARTNER_ID-E686-4C20-982D-
5028EDF97B5A"
 ---> Running in 9f58955a8b89
Removing intermediate container 9f58955a8b89
 ---> 237e207ddbb4
Step 15/25 : LABEL com.veea.authorisation.allowOnUnauthenticatedHost="true"
 ---> Running in 30b9dbfe7897
Removing intermediate container 30b9dbfe7897
 ---> 47d84beb2f21
Step 16/25 : LABEL com.veea.authorisation.feature1="DEVELOPER"
```

```
 ---> Running in 2ad9f50d7645
Removing intermediate container 2ad9f50d7645
 ---> 14ba764aae37
Step 17/25 : ARG PARTNER_FEATURE1
 ---> Running in 3e317e5c1a09
Removing intermediate container 3e317e5c1a09
 ---> dbd3d916a10c
Step 18/25 : LABEL com.veea.authorisation.license1="$PARTNER_FEATURE1"
 ---> Running in a3bbee809da9
Removing intermediate container a3bbee809da9
 ---> 6840cbfb52c2
Step 19/25 : ARG SIGNING_CERTIFICATE
 ---> Running in f4c9ba952f12
Removing intermediate container f4c9ba952f12
 ---> 1a70e01e8b92
Step 20/25 : LABEL com.veea.authentication.certificates.partner="$SIGNING_CERTIFICATE"
 ---> Running in 021b55fb5b08
Removing intermediate container 021b55fb5b08
 ---> cf29b5e5a0e1
Step 21/25 : ARG LICENSE_SERVER
 ---> Running in 3897c624365e
Removing intermediate container 3897c624365e
 ---> 8529058b4d77
Step 22/25 : LABEL
com.veea.authentication.certificates.veeahub_license_server="$LICENSE_SERVER"
 ---> Running in 1887335a0790
Removing intermediate container 1887335a0790
 ---> 8e4abdcf9aff
Step 23/25 : ARG LICENSE_AUTHORITY
 ---> Running in 294c1f080a80
Removing intermediate container 294c1f080a80
 ---> dbeefb1a7330
Step 24/25 : LABEL
com.veea.authentication.certificates.veeahub_license_authority="$LICENSE_AUTHORITY"
 ---> Running in 8c13080a7d4c
Removing intermediate container 8c13080a7d4c
 ---> 2ae7a75cbdf2
Step 25/25 : CMD ["./start.sh"]
 ---> Running in a756db18f7b5
Removing intermediate container a756db18f7b5
 ---> d3d7e82085e7
Successfully built d3d7e82085e7
Successfully tagged vh_io_attach:vhc05


Saving vh_io_attach image for vhc05.
bin/vh_io_attach:vhc05.signed.tar does not exist.
Signing vh_io_attach image for target vhc05...
**********************************************************
** Processing: vh_io_attach:vhc05
**********************************************************

  Docker TAG:      vh_io_attach:vhc05
  Feature Set:     /opt/veea/vht/bin/features.json
  Rule Set:        /opt/veea/vht/bin/verify.json
  Working folder:  bin
  Signing Server   None
  Signing Key      <id>
  Signing Cert     /path/to/vhc-signing-cert.pem
  OpenSSL Arguments:
    -engine=pkcs11
    -keyform=engine

Verifying container
Info: **************************************
Info: ** Starting verification
Info: **************************************
Info: Checking path: config
Info: *************************************
Info: ** Checking entry point rules
Info: *************************************
Info: *************************************
Info: ** Completed entry point rules
Info: *************************************
Info: Checking path: config.ArgsEscaped
Info: Checking path: config.AttachStderr
Info: Checking path: config.AttachStdin
Info: Checking path: config.AttachStdout
Info: Checking path: config.Cmd
Info: Checking path: config.Entrypoint
Info: Checking path: config.Env
Info: Checking path: config.ExposedPorts
Info: Checking path: config.Healthcheck
Info: Checking path: config.Healthcheck.Interval
Info: Checking path: config.Healthcheck.Retries
Info: Checking path: config.Healthcheck.StartPeriod
Info: Checking path: config.Healthcheck.Test
Info: Checking path: config.Healthcheck.Timeout
Info: Checking path: config.Image
Info: Checking path: config.Labels
Info: *************************************
Info: ** Checking label syntax
Info: *************************************
Info: Found 2 licensed features: ['DEV:EXCLUSIVE:audio', 'DEV:EXCLUSIVE:video']
Info: Using 0 license feature: []
Info: Making 0 persistent volume: []
Info: Found 0 volume offer: []
Info: Found 0 volume request: []
Info: Container allowed on unauthenticated host: true
Info: *************************************
Info: *************************************
Info: ** Completed label syntax checks
Info: *************************************
Info: Checking path: config.Labels.com.veea.authentication.certificates.partner
Info: Checking path:
config.Labels.com.veea.authentication.certificates.veeahub_license_authority
Info: Checking path:
config.Labels.com.veea.authentication.certificates.veeahub_license_server
Info: Checking path: config.Labels.com.veea.authentication.identifier
Info: Checking path: config.Labels.com.veea.authorisation.allowOnUnauthenticatedHost
Info: Checking path: config.Labels.com.veea.authorisation.feature1
Info: Checking path: config.Labels.com.veea.authorisation.license1
Info: Checking path: config.Labels.com.veea.image.persistent_uuid
Info: Checking path: config.Labels.com.veea.vhc.app.name
Info: Checking path: config.Labels.com.veea.vhc.app.version
Info: Checking path: config.Labels.com.veea.vhc.config.proj.version
Info: Checking path: config.Labels.com.veea.vhc.config.user.version
Info: Checking path: config.Labels.com.veea.vhc.target
```

```
Info: Checking path: config.Labels.com.veea.vhc.version
Info: Checking path: config.NetworkDisabled
Info: Checking path: config.OnBuild
Info: Checking path: config.OpenStdin
Info: Checking path: config.Shell
Info: Checking path: config.StdinOnce
Info: Checking path: config.StopSignal
Info: Checking path: config.StopTimeout
Info: Checking path: config.Tty
Info: Checking path: config.User
Info: Checking path: config.Volumes
Info: Checking path: config.WorkingDir
Info: **************************************
Info: ** Image verification complete
Info: **************************************
Info: ** Warnings:   0
Info: ** Errors:     0
Info: **************************************
Signing container
Writing: bin/vh_io_attach:vhc05.signed.tar
**********************************************************
** Success!
**********************************************************

Verifying vh_io_attach image for target vhc05...
Info: **************************************
Info: ** Verifying Container: bin/vh_io_attach:vhc05.signed.tar
Info: openssl: Verified OK
Info: Found valid partner Signature!
Info: Found matching Partner certificate.
Info: Partner Identity XXXXXXXX signature verified!
Info: Partner License signature verified: com.veea.authorisation.license1
Info: **************************************
Info: ** Cerification complete
Info: **************************************
Info: ** Warnings:   0
Info: ** Errors:     0
Info: **************************************
```

As with insecure applications, a target-specific Dockerfile now exists in the project directory:

```
$ ls
bin  config.yaml  Dockerfile  Dockerfile.vhc05  README.md  src
```

Examine its contents:

```
$ more Dockerfile.vhc05

#BEGIN vhc05
FROM arm32v7/busybox:latest
#END

RUN mkdir /app
COPY src/ /app/
WORKDIR /app
```

```
#BEGIN AUTO-GENERATED - DO NOT EDIT!!!
LABEL com.veea.vhc.target="vhc05"
LABEL com.veea.vhc.version="0.5.0"
LABEL com.veea.vhc.app.name="vh io attach"
LABEL com.veea.vhc.app.version="1"
LABEL com.veea.vhc.config.proj.version="1"
LABEL com.veea.vhc.config.user.version="1"
ARG PARTNER_LICENSE
LABEL com.veea.authentication.identifier="$PARTNER_LICENSE"
ARG PARTNER_ID
LABEL com.veea.image.persistent_uuid="$PARTNER_ID-E686-4C20-982D-5028EDF97B5A"
LABEL com.veea.authorisation.allowOnUnauthenticatedHost="true"
LABEL com.veea.authorisation.feature1="DEVELOPER"
ARG PARTNER_FEATURE1
LABEL com.veea.authorisation.license1="$PARTNER_FEATURE1"
ARG SIGNING_CERTIFICATE
LABEL com.veea.authentication.certificates.partner="$SIGNING_CERTIFICATE"
ARG LICENSE_SERVER
LABEL com.veea.authentication.certificates.veeahub_license_server="$LICENSE_SERVER"
ARG LICENSE_AUTHORITY
LABEL
com.veea.authentication.certificates.veeahub_license_authority="$LICENSE_AUTHORITY"
#END AUTO-GENERATED - DO NOT EDIT!!!

CMD ["./start.sh"]
```

The signed containers are in tar archives located in the *bin* directory:

```
$ ls bin
vh_io_attach:vhc05.signed.tar  vh_io_attach:vhc05.tar
```

5.3. Notes

Now that the VHC configuration is set for secure/trusted operation, in order to build an untrusted application you must build with the **--untrusted** option, for example:

```
$ vhc build --untrusted --target vhc05.
```

There are three ways to specify the signing PIN when building secure/trusted applications:

- Via the **--sign-pin <pin>** command-line option, for example:

```
$ vhc build --sign-pin <pin>
```

- Via the environment variable **VHC_SIGNING_PIN**,for example:

```
export VHC_SIGNING_PIN=<pin>; vhc build
```

- Via prompt on the command line after running the **build** command.

5.4. Deploying a Container

While the deployment mechanism for secure containers is covered under Sideload (section 6), please note that once your transport license is installed on the hub, communication with the hub from that point forward will be secure and therefore require providing the TLS PIN for *every* **hub** sideload command. There are three ways to do this, none of which allow for the PIN to be stored on disk. In internal VHC priority order they are:

- Via the **--tls-pin <pin>** command-line option, for example:

```
$ vhc hub --get-partner-id --tls-pin <pin>
```

- Via the environment variable **VHC_TLS_PIN**,for example:

```
export VHC_TLS_PIN=<pin>; vhc hub --get-partner_id
```

- Via prompt on the command line after running a **hub** command.

5.5. Troubleshooting

When troubleshooting problems related to secure applications the following environment variables should be set:

```
export OPENSSL_ENGINES=/opt/veea/tools/lib/ssl/engines
export LD_LIBRARY_PATH=/opt/veea/tools/lib/x86_64-linux-gnu
```

5.5.1. VHC-to-Hub Communication

If secure communication between VHC and the hub fails to return the correct partner ID please check the following:

1. Ensure that the HSM is inserted into a valid and working USB slot on your development machine. If you are developing under a Virtual Machine (VM) such as Parallels be sure that the HSM is connected to the VM (appears with **uTrust** in the name). The HSM should be identifiable using **lsusb** as follows:

```
$ lsusb
...
   Bus 002 Device 024: ID 04e6:5816 SCM Microsystems, Inc.
...
```

2. Verify that the three private cryptographic keys are present in the HSM. Note each key's name, for example, **Private EC Key [tls_key]**:

```
$ pkcs15-tool -D 2
    Using reader with a card: Identiv uTrust 3512 SAM slot Token [CCID Interface]
(55511747600241) 00 00
    PKCS#15 Card [SmartCard-HSM]:
        Version        : 0
        Serial number  : <serial>
        Manufacturer ID: www.CardContact.de
        Flags          :
```

```
PIN [UserPIN]
    Object Flags   : [0x3], private, modifiable
    Auth ID        : 02
    ID             : 01
    Flags          : [0x812], local, initialized, exchangeRefData
    Length         : min_len:6, max_len:15, stored_len:0
    Pad char       : 0x00
    Reference      : 129 (0x81)
    Type           : ascii-numeric
    Path           : e82b0601040181c31f0201::
    Tries left     : 3

PIN [SOPIN]
    Object Flags   : [0x1], private
    ID             : 02
    Flags          : [0x9A], local, unblock-disabled, initialized, soPin
    Length         : min_len:16, max_len:16, stored_len:0
    Pad char       : 0x00
    Reference      : 136 (0x88)
    Type           : bcd
    Path           : e82b0601040181c31f0201::
    Tries left     : 15

Private EC Key [root_auth_key]
    Object Flags   : [0x3], private, modifiable
    Usage          : [0x10C], sign, signRecover, derive
    Access Flags   : [0x1D], sensitive, alwaysSensitive, neverExtract, local
    FieldLength    : 256
    Key ref        : 1 (0x1)
    Native         : yes
    Auth ID        : 01
    ID             : <id>
    MD:guid        : 5e85e122-146f-c8fe-f7db-f45c53d777de

Private EC Key [tls_key]
    Object Flags   : [0x3], private, modifiable
    Usage          : [0x10C], sign, signRecover, derive
    Access Flags   : [0x1D], sensitive, alwaysSensitive, neverExtract, local
    FieldLength    : 256
    Key ref        : 2 (0x2)
    Native         : yes
    Auth ID        : 01
    ID             : <id>
    MD:guid        : 788db1b3-dee8-74a9-f4d7-aab1dca10dd4

Private EC Key [signing_key]
    Object Flags   : [0x3], private, modifiable
    Usage          : [0x10C], sign, signRecover, derive
    Access Flags   : [0x1D], sensitive, alwaysSensitive, neverExtract, local
    FieldLength    : 256
    Key ref        : 3 (0x3)
    Native         : yes
    Auth ID        : 01
    ID             : <id>
    MD:guid        : 3e899d36-1c20-029d-7697-4f9eca6a5544

Public EC Key [root_auth_key]
    Object Flags   : [0x0]
```

```
        Usage          : [0x40], verify
        Access Flags   : [0x2], extract
        FieldLength    : 256
        Key ref        : 0 (0x0)
        Native         : no
        ID             : <id>
        DirectValue    : <present>

    Public EC Key [tls_key]
        Object Flags   : [0x0]
        Usage          : [0x40], verify
        Access Flags   : [0x2], extract
        FieldLength    : 256
        Key ref        : 0 (0x0)
        Native         : no
        ID             : <id>
        DirectValue    : <present>

    Public EC Key [signing_key]
        Object Flags   : [0x0]
        Usage          : [0x40], verify
        Access Flags   : [0x2], extract
        FieldLength    : 256
        Key ref        : 0 (0x0)
        Native         : no
        ID             : <id>
        DirectValue    : <present>
```

3. Dump the URLs for the private keys and make sure they match what is configured in VHC (the Root key is unused by VHC):

```
$ p11tool --login --provider=/opt/veea/tools/lib/x86_64-linux-gnu/opensc-pkcs11.so --
list-privkeys
    Token 'UserPIN (SmartCard-HSM)' with URL
'pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;toke
n=UserPIN%20%28SmartCard-HSM%29' requires user PIN
    Enter PIN:
    Object 0:
        URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=root_auth_key;type=private
        Type: Private key
        Label: root_auth_key
        Flags: CKA_PRIVATE; CKA_NEVER_EXTRACTABLE; CKA_SENSITIVE;
        ID: <id>

    Object 1:
        URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=tls_key;type=private
        Type: Private key
        Label: tls_key
        Flags: CKA_PRIVATE; CKA_NEVER_EXTRACTABLE; CKA_SENSITIVE;
        ID: <id>

    Object 2:
        URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=signing_key;type=private
        Type: Private key
        Label: signing_key
        Flags: CKA_PRIVATE; CKA_NEVER_EXTRACTABLE; CKA_SENSITIVE;
        ID: <id>
```

And now VHC:

```
$ vhc secure credential --show
    Secure Credentials:
    +---------------------+-------------------------------+
    |     CREDENTIAL      |              PATH             |
    +---------------------+-------------------------------+
    | Partner License     |                               |
    | Signing Certificate | /path/to/vhc-signing-cert.pem |
    | TLS Certificate     | /path/to/vhc-tls-cert.pem     |
    | Partner Features    | <No Configuration Available>  |
    +---------------------+-------------------------------+

    Signing URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=signing_key;type=private
    TLS URL:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=tls_key;type=private
```

Update any incorrect URLs using **vhc secure credential --set-sign-url** and/or **vhc secure credential --set-tls-url**.

4. Verify that the signing and TLS PINs match what was entered when running the key generation script (note: VHC intentionally avoids displaying these PINs via the **secure credentials --show** command):

```
$ cat ~/.vhc/user-config.yaml
    credentials:
      signing:
        certificate: /path/to/vhc-signing-cert.pem
        pin: <pin>
        url:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=signing_key;type=private
      tls:
        certificate: /path/to/vhc-tls-cert.pem
        pin: <pin>
        url:
pkcs11:model=PKCS%2315%20emulated;manufacturer=www.CardContact.de;serial=<serial>;token
=UserPIN%20%28SmartCard-HSM%29;id=<id>;object=tls_key;type=private
    version: 1
```

Update any incorrect URLs using **vhc secure credential --set-sign-pin** and/or **vhc secure credential --set-tls-pin**.

5. Check that the transport license contains the correct certificate data. First, query the hub for the licenses:

```
$ vhc hub licenses --get
    ...
          "partner_ca": "<certificate-contents>",
          "partner_id": "XXXXXXXX"
        }
    ...
```

Next, dump the contents of `vhc-nw-cert.pem`:

```
$ more vhc-nw-cert.pem
    -----BEGIN CERTIFICATE-----
    <certificate contents>
    -----END CERTIFICATE-----
```

6. Verify that the public key for each private cryptographic key in the HSM matches the public key in the associated certificate. First, using the TLS key ID from Step 2 above, query the public key from the HSM:

```
$ openssl ec -pubin -noout -text -inform engine -engine pkcs11 -in <id>
    engine "pkcs11" set.
    read EC key
    Public-Key: (256 bit)
    pub:
        04:c3:fe:7d:e9:45:6d:b6:c2:86:ae:29:13:78:4e:
        1a:06:f3:76:81:67:8d:74:98:2f:c0:ab:93:5d:7b:
        7b:87:a0:3c:2c:7c:97:c7:c2:f9:69:fd:e4:5a:24:
        d1:0c:fd:15:6d:7b:43:6b:de:80:40:c0:99:c2:e8:
        aa:99:e5:99:d4
    ASN1 OID: prime256v1
    NIST CURVE: P-256
```

7. Dump the public key from the TLS certificate:

```
$ openssl x509 -noout -text -in vhc-tls-cert.pem
    Certificate:
        Data:
            Version: 3 (0x2)
            Serial Number: 1 (0x1)
            Signature Algorithm: ecdsa-with-SHA256
            Issuer: <issuer-information>
            Validity
                Not Before: Sep 27 15:22:30 2019 GMT
                Not After : Sep 24 15:22:30 2029 GMT
            Subject: <subject-information>
            Subject Public Key Info:
                Public Key Algorithm: id-ecPublicKey
                    Public-Key: (256 bit)
                    pub:
                        04:c3:fe:7d:e9:45:6d:b6:c2:86:ae:29:13:78:4e:
                        1a:06:f3:76:81:67:8d:74:98:2f:c0:ab:93:5d:7b:
                        7b:87:a0:3c:2c:7c:97:c7:c2:f9:69:fd:e4:5a:24:
                        d1:0c:fd:15:6d:7b:43:6b:de:80:40:c0:99:c2:e8:
                        aa:99:e5:99:d4
                    ASN1 OID: prime256v1
                    NIST CURVE: P-256
            X509v3 extensions:
                X509v3 Basic Constraints: critical
                    CA:FALSE
                X509v3 Key Usage:
                    Digital Signature, Certificate Sign
        Signature Algorithm: ecdsa-with-SHA256
             30:45:02:20:48:23:0b:11:10:46:7d:50:7e:51:fa:6b:26:17:
             cf:60:f8:08:40:3e:9a:63:97:63:08:2b:ad:60:02:ce:ec:b9:
             02:21:00:d2:f5:c8:82:ef:b2:c0:c6:8a:81:0a:f4:ef:68:f0:
             0b:49:7b:81:91:d1:ee:bc:e4:53:22:3a:15:2d:6f:ff:6d
```

8. Repeat the process for the signing key.
9. Test that Transport Layer Security (TLS) is working correctly outside of VHC by using **openssl**. To do this, plug the TLS key ID from the previous step and path to your VHC TLS certificate into the following command:

```
$ openssl s_client -host 10.20.0.90 -port 9000 -key <id> -engine pkcs11 -keyform engine
-cert /path/to/vhc-tls-cert.pem -CAfile /opt/veea/vht/cert/veea_root_ca.pem
    engine "pkcs11" set.
    Enter PKCS#11 token PIN for UserPIN (SmartCard-HSM):
    CONNECTED(00000006)
    depth=2 generationQualifier = 0, O = Veea Inc, OU = PKI, CN = Veea Root Authority
    verify return:1
    depth=1 O = Veea Inc, CN = Veea Device Authority, generationQualifier = 0
    verify return:1
    depth=0 O = Veea Inc, CN = C05BCB00C0A000001127@veea.com, generationQualifier = 0
    verify return:1
    ---
    Certificate chain
     0 s:O = Veea Inc, CN = C05BCB00C0A000001127@veea.com, generationQualifier = 0
       i:O = Veea Inc, CN = Veea Device Authority, generationQualifier = 0
     1 s:O = Veea Inc, CN = Veea Device Authority, generationQualifier = 0
       i:generationQualifier = 0, O = Veea Inc, OU = PKI, CN = Veea Root Authority
     2 s:generationQualifier = 0, O = Veea Inc, OU = PKI, CN = Veea Root Authority
       i:generationQualifier = 0, O = Veea Inc, OU = PKI, CN = Veea Root Authority
    ---
    Server certificate
    -----BEGIN CERTIFICATE-----
    <certificate contents>
    -----END CERTIFICATE-----
    subject=O = Veea Inc, CN = C05BCB00C0A000001127@veea.com, generationQualifier = 0

    issuer=O = Veea Inc, CN = Veea Device Authority, generationQualifier = 0

    ---
    Acceptable client certificate CA names
    <subject-information>
    Requested Signature Algorithms:
ECDSA+SHA256:ECDSA+SHA384:ECDSA+SHA512:Ed25519:Ed448:RSA-PSS+SHA256:RSA-PSS+SHA384:RSA-
PSS+SHA512:RSA-PSS+SHA256:RSA-PSS+SHA384:RSA-
PSS+SHA512:RSA+SHA256:RSA+SHA384:RSA+SHA512:ECDSA+SHA224:ECDSA+SHA1:RSA+SHA224:RSA+SHA1
    Shared Requested Signature Algorithms:
ECDSA+SHA256:ECDSA+SHA384:ECDSA+SHA512:Ed25519:Ed448:RSA-PSS+SHA256:RSA-PSS+SHA384:RSA-
PSS+SHA512:RSA-PSS+SHA256:RSA-PSS+SHA384:RSA-
PSS+SHA512:RSA+SHA256:RSA+SHA384:RSA+SHA512
    Peer signing digest: SHA256
    Peer signature type: ECDSA
    Server Temp Key: X25519, 253 bits
    ---
    SSL handshake has read 2397 bytes and written 999 bytes
    Verification: OK
    ---
    New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
    Server public key is 256 bit
    Secure Renegotiation IS NOT supported
    Compression: NONE
    Expansion: NONE
    No ALPN negotiated
    Early data was not sent
    Verify return code: 0 (ok)
    ---
    ---
    Post-Handshake New Session Ticket arrived:
    SSL-Session:
        Protocol  : TLSv1.3
        Cipher    : TLS_AES_256_GCM_SHA384
        Session-ID: 276599C4986D8B71EB215713C2CB68CB9D5F0BA020999FC5E19BA88AD5030410
        Session-ID-ctx:
        Resumption PSK:
45724494837B78304F1C08D5E9E161B286F49C99E487DDE1C7B486793CCBFB2E2E469169E77280758FE6507
F66CD06D4
        PSK identity: None
        PSK identity hint: None
        SRP username: None
        TLS session ticket lifetime hint: 7200 (seconds)
        TLS session ticket:
        0000 - <data>
        ...
        02a0 - <data>

        Start Time: 1569611280
        Timeout   : 7200 (sec)
        Verify return code: 0 (ok)
        Extended master secret: no
        Max Early Data: 0
    ---
    read R BLOCK
    ---
    Post-Handshake New Session Ticket arrived:
    SSL-Session:
        Protocol  : TLSv1.3
        Cipher    : TLS_AES_256_GCM_SHA384
        Session-ID: 87DA2314FABB3151F18360A609BD7B0759C564DFE7FD2B8693273F58866D8D38
        Session-ID-ctx:
        Resumption PSK:
5035448F973391A2E9B21B27D464FBC1305D4D08B198FC9FDAF8558255EF3DAF2B577779880524B65C6A878
3A6E77812
        PSK identity: None
        PSK identity hint: None
        SRP username: None
        TLS session ticket lifetime hint: 7200 (seconds)
        TLS session ticket:
        0000 - <data>
        ...
        02a0 - <data>

        Start Time: 1569611280
        Timeout   : 7200 (sec)
        Verify return code: 0 (ok)
        Extended master secret: no
        Max Early Data: 0
    ---
    read R BLOCK
    ^C
```

If none of the previous steps are able to help resolve the problem, please contact Veea Support.

5.5.2. Resetting your HSM

If all else fails it is worth considering resetting or initializing the HSM and repeating the process. To reset everything in order to re-run the key generation script, run the following **destructive** commands from the directory in which the script was originally run:

```
rm certs/01.pem
rmdir certs
rm db.txt*
rm *.cnf
rm *.pem
rm serial.*
rm partner_certificates.tgz


pkcs11-tool --login --login-type so --so-pin <so-pin> --change-pin --new-pin 3537363231383830
pkcs11-tool --login --pin <pin> --change-pin --new-pin 648219
```

6. Sideloading

6.1. Sideload API

This describes how VeeaHub Toolkit supports sideloading of secure containers.

Sideloading is the process of loading containers directly on to VeeaHubs, usually for testing purposes. For release of signed containers, Veea will be providing a process for submitting the containers for users to install via Control Center.

6.2. Usage

To display general usage for the sideload container functionality within VHC, run:

```
$ vhc hub -h
Used to load, start, and stop containers on and communicate with a VeeaHub.

Notes:
  1. Only one local option should be used at a time.
  2. The format for setting the hub configuration is a colon-separated string
     2- or 3-tuple.
  3. When communicating securely with the hub, which is required for secure/
     trusted application development, the TLS PIN should be specified via
     either the --tls-pin option or the environment varible VHC_TLS_PIN,
     otherwise VHC will prompt the user for the PIN.

Usage:
  vhc hub [flags]
  vhc hub [command]

Examples:
  vhc hub --add-hub <hub-id>:<serial-number>
  vhc hub --add-hub <hub-id>:<serial-number>:<ipv4-addr>
  vhc hub --remove-hub <hub-id>
  vhc hub --update-hub <hub-id>:<new-ip4v-addr>
  vhc hub --update-hub <hub-id>:
  vhc hub --active-hub <hub-id>
  vhc hub --ping --hub-id <hub-id>
  vhc hub --ping --hub-id <hub-id> --tls-pin <pin>
  vhc hub --get-partner-id --hub-id <hub-id>
  vhc hub --get-partner-id --hub-id <hub-id> --tls-pin <pin>
  vhc hub --reboot --hub-id <hub-id>
  vhc hub --reboot --hub-id <hub-id> --tls-pin <pin>
  vhc hub --show

Available Commands:
  config      Used to export/unexport config on a VeeaHub.
  container   Used to create, delete, start, stop, and get information about containers
on a VeeaHub.
  image       Used to create, delete, and get information about images on a VeeaHub.
  licenses    Used to add and get information about licenses on a VeeaHub.
  software    Used to upgrade the software image on a VeeaHub.
  volume      Used to create, delete, and get information about volumes on a VeeaHub.

Flags:
  -A, --active-hub string   Used to specify the hub for all subsequent operations.
  -a, --add-hub string      Adds a hub to the configuration. See Examples above for
format.
```

```
  -i, --get-partner-id      Returns the Partner ID from the hub.
  -h, --help                help for hub
  -H, --hub-id string       Specifies the hub ID. Required for most commands.
  -p, --ping                Pings the hub specified by serial number or Ipv4 address.
  -R, --reboot              Reboots the hub specified by serial number or Ipv4 address.
  -r, --remove-hub string   Removes a hub from the configuration.
  -s, --show                Shows hub information.
  -T, --tls-pin string      Specifies the TLS PIN.
  -u, --update-hub string   Used to update the IPv4 address for a configured hub.
Format is <id>:<new-ipv4-addr>.

Use "vhc hub [command] --help" for more information about a command.
```

6.3. Commands

The following commands are available for sideloading:

- hub

  - config

  - container

  - image

  - licenses

  - software

  - volume

6.4. Exporting/Unexporting Configuration

The **config** command is a sub-command of **hub**. It is used to export/unexport configuration on a VeeaHub.

```
$ vhc hub config -h
Used to export/unexport config on a VeeaHub.

Note: When communicating securely with the hub, which is required for secure/
      trusted application development, the TLS PIN should be specified via
      either the --tls-pin option or the environment variable VHC_TLS_PIN,
      otherwise VHC will prompt the user for the PIN.

Usage:
  vhc hub config [flags]

Examples:
  vhc hub config --export --hub-id <hub-id>
  vhc hub config --export --hub-id <hub-id> --tls-pin <pin>
  vhc hub config --unexport --hub-id <hub-id>
  vhc hub config --unexport --hub-id <hub-id> --tls-pin <pin>

Flags:
  -e, --export            Exports a config on the hub via WebDAV.
  -h, --help              help for config
  -H, --hub-id string     Specifies the hub ID. Required for all commands.
  -T, --tls-pin string    Specifies the TLS PIN.
  -u, --unexport          Unexports a config on the hub via WebDAV.
```

6.5. *hub container* Command

The **container** command is a sub-command of **hub**. It is used to create, delete, start, stop, and get information about containers on a VeeaHub.

```
$ vhc hub container -h
Used to create, delete, start, stop, and get information about containers on a VeeaHub.

Notes:
  1. Only one local option should be used at a time.
  2. The format for getting a specific container label or parameter is a colon-
separated string 2-tuple.
  3. The format for setting a specific container parameter is a colon-     separated
string 3-tuple.
  4. When communicating securely with the hub, which is required for secure/
     trusted application development, the TLS PIN should be specified via
     either the --tls-pin option or the environment varible VHC_TLS_PIN,
     otherwise VHC will prompt the user for the PIN.

Usage:
  vhc hub container [flags]

Examples:
  vhc hub container --get-containers --hub-id <hub-id>
  vhc hub container --delete <cont-id> --hub-id <hub-id>
  vhc hub container --delete-all --hub-id <hub-id>
  vhc hub container --delete-all --force --hub-id <hub-id>
  vhc hub container --get-info <cont-id> --hub-id <hub-id>
  vhc hub container --get-labels <cont-id> --hub-id <hub-id>
  vhc hub container --get-label <cont-id>:<label> --hub-id <hub-id>
  vhc hub container --start <cont-id> --hub-id <hub-id>
  vhc hub container --stop <cont-id> --hub-id <hub-id>
  vhc hub container --export <cont-id> --hub-id <hub-id>
  vhc hub container --unexport <cont-id> --hub-id <hub-id>
  vhc hub container --attach <cont-id> --hub-id <hub-id>
  vhc hub container --attach <cont-id> --hub-id <hub-id> --no-ws
  vhc hub container --attach <cont-id> --hub-id <hub-id> <attach-command> (e.g.
"/bin/sh -il")
  vhc hub container --detach <cont-id> --hub-id <hub-id>
  vhc hub container --get-syslog <cont-id> --hub-id <hub-id>
  vhc hub container --put-syslog <cont-id> --hub-id <hub-id> <config-file>
  vhc hub container --delete-syslog <cont-id> --hub-id <hub-id>
  vhc hub container --get-status <cont-id> --hub-id <hub-id>

Flags:
  -a, --attach string          Attaches to a container on the hub. Can specify an
attach command such as "/bin/sh -il". Use --no-ws to skip Websockets connection.
  -d, --delete string          Deletes a container from the hub.
  -R, --delete-all             Deletes all containers from the hub. Use --force to skip
confirmation.
  -D, --delete-syslog string   Removes the syslog configuration and turns off streaming
to a local syslog server.
  -y, --detach string          Detaches from a container on the hub.
  -x, --export string          Exports a container on the hub.
  -f, --force                  Skip confirmation when deleting all containers.
  -c, --get-containers         Returns a list of containers on the hub.
  -I, --get-info string        Gets a container's info.
  -L, --get-label string       Gets a container's label. Format: <cont-id>:<label-
name>.
  -l, --get-labels string      Gets a container's labels.
  -A, --get-status string      Get the status for a container on the hub.
  -g, --get-syslog string      Returns the syslog configuration for the container.
  -h, --help                   help for container
  -H, --hub-id string          Specifies the hub ID. Required for all commands.
  -n, --no-ws                  Specifies no websockets to attach option.
  -e, --put-syslog string      Enables syslog events for the container to be streamed
to a local syslog server.
  -s, --start string           Starts a container.
  -t, --stop string            Stops a container.
  -T, --tls-pin string         Specifies the TLS PIN.
  -u, --unexport string        Unexports a container on the hub.
```

6.6. *hub image* Command

The image command is a sub-command of **hub**. It is used to create, delete, and get information about images on a VeeaHub.

```
$ vhc hub image -h
Used to create, delete, and get information about images on a VeeaHub.

Notes:
  1. Only one local option should be used at a time.
  2. The format for getting a specific image label or parameter is a colon-separated
string 2-tuple.
3. When communicating securely with the hub, which is required for secure/
     trusted application development, the TLS PIN should be specified via
     either the --tls-pin option or the environment varible VHC_TLS_PIN,
     otherwise VHC will prompt the user for the PIN.

Usage:
  vhc hub image [flags]

Examples:
  vhc hub image --get-images --hub-id <hub-id>
  vhc hub image --create <file> --hub-id <hub-id>
  vhc hub image --delete <image-id> --hub-id <hub-id>
  vhc hub image --delete-all --hub-id <hub-id>
  vhc hub image --delete-all --force --hub-id <hub-id>
  vhc hub image --create-container <image-id> --hub-id <hub-id>
  vhc hub image --create-container <image-id> --hub-id <hub-id> --detach
  vhc hub image --create-container <image-id> --hub-id <hub-id> --restart-policy
<policy>
  vhc hub image --create-container <image-id> --hub-id <hub-id> --detach --restart-
policy <policy>
  vhc hub image --get-info <image-id> --hub-id <hub-id>
  vhc hub image --get-labels <image-id> --hub-id <hub-id>
  vhc hub image --get-label <image-id>:<label-name> --hub-id <hub-id>

Flags:
  -c, --create string              Creates an image on the hub via push.
  -C, --create-container string    Creates a container on the hub.
  -d, --delete string              Deletes an image from the hub.
  -R, --delete-all                 Deletes all images from the hub. Use --force to skip
confirmation.
  -D, --detach                     Specifies to start the container in the background.
  -f, --force                   Skip confirmation when deleting all images.
  -i, --get-images              Returns a list of images on the hub.
  -I, --get-info string         Gets an image's info.
  -L, --get-label string        Gets an image's label. Format: <image-id>:<label-
name>.
  -l, --get-labels string       Gets an image's labels.
  -h, --help                    help for image
  -H, --hub-id string           Specifies the hub ID. Required for all commands.
  -r, --restart-policy string   Specifies the restart policy when creating a
container. Valid values are: no, unless-stopped, and always.
  -T, --tls-pin string          Specifies the TLS PIN.
```

6.7. *hub licenses* Command

The **licenses** command is a sub-command of **hub**. It is used to add and get information about licenses on a VeeaHub.

```
$ vhc hub licenses -h
Used to add and get information about licenses on a VeeaHub.

Note: When communicating securely with the hub, which is required for secure/
      trusted application development, the TLS PIN should be specified via
      either the --tls-pin option or the environment variable VHC_TLS_PIN,
      otherwise VHC will prompt the user for the PIN.

Usage:
  vhc hub licenses [flags]

Examples:
  vhc hub licenses --add <license> --hub-id <hub-id>
  vhc hub licenses --add <license> --hub-id <hub-id> --tls-pin <pin>
  vhc hub licenses --get --hub-id <hub-id>
  vhc hub licenses --get --hub-id <hub-id> --tls-pin <pin>

Flags:
  -a, --add string       Adds a license to the hub.
  -g, --get              Returns a list of licenses on the hub.
  -h, --help             help for licenses
  -H, --hub-id string    Specifies the hub ID. Required for all commands.
  -T, --tls-pin string   Specifies the TLS PIN.
```

6.8. *hub software* Command

The **software** command is a sub-command of **hub**. It is used to upgrade the software image on a VeeaHub.

```
$ vhc hub software -h
Used to upgrade the software image on a VeeaHub.

Note: When communicating securely with the hub, which is required for secure/
      trusted application development, the TLS PIN should be specified via
      either the --tls-pin option or the environment variable VHC_TLS_PIN,
      otherwise VHC will prompt the user for the PIN.

Usage:
  vhc hub software [flags]
```

```
Examples:
  vhc hub software --get-info
  vhc hub software --get-info --tls-pin <pin>
  vhc hub software --upgrade <software-image>
  vhc hub software --upgrade <software-image> --tls-pin <pin>

Flags:
  -i, --get-info          Returns hub software information.
  -h, --help              help for software
  -H, --hub-id string     Specifies the hub ID. Required for all commands.
  -T, --tls-pin string    Specifies the TLS PIN.
  -u, --upgrade string    Upgrades the software on a hub.
```

6.9. *hub volume* Command

The **volume** command is a sub-command of **hub**. It is used to create, delete, and get information about volumes on a VeeaHub.

```
$ vhc hub volume -h
Used to create, delete, and get information about volumes on a VeeaHub.

Notes:
  1. Only one local option should be used at a time.
  2. The format for most volume commands is a colon-separated string 2-tuple.
3. When communicating securely with the hub, which is required for secure/
     trusted application development, the TLS PIN should be specified via
     either the --tls-pin option or the environment varible VHC_TLS_PIN,
     otherwise VHC will prompt the user for the PIN.

Usage:
  vhc hub volume [flags]

Examples:
  vhc hub volume --get-all-volumes --hub-id <hub-id>
  vhc hub volume --get-uuid-volumes --hub-id <hub-id>
  vhc hub volume --delete <volume-name> --hub-id <hub-id>
  vhc hub volume --get-info <volume-name> --hub-id <hub-id>
  vhc hub volume --create <volume-name> --hub-id <hub-id>
  vhc hub volume --export <volume-name> --hub-id <hub-id>
  vhc hub volume --unexport <volume-name> --hub-id <hub-id>

Flags:
  -c, --create string      Creates a volume on the hub.
  -d, --delete string      Deletes a volume from the hub.
  -x, --export string      Exports a volume on the hub via WebDAV.
  -v, --get-all-volumes    Returns a list of all volumes on the hub.
  -I, --get-info string    Gets a volume's info.
  -V, --get-uuid-volumes   Returns a list of the UUID's volumes on the hub.
  -h, --help               help for volume
  -H, --hub-id string      Specifies the hub ID. Required for all commands.
  -T, --tls-pin string     Specifies the TLS PIN.
  -u, --unexport string    Unexports a volume on the hub via WebDAV.
```

7. Technical Support and Developer Community

Before contacting Technical Support, please consult the documentation, tutorials, and community topics available on the support web site www.veea.com/support/. Please sign up, if you don't already have an account, and sign in.

For unresolved queries, click on the Submit a request link located near the top of the screen.

Please complete the form with an appropriate subject, and as much detail as possible in the description field. Please include any relevant information such as Veea hardware serial numbers, logs, screenshots, etc.

An email will automatically be sent to your registered email address to confirm the request has been received.

There is a community for developers at developers.veea.com, where you may register to discuss all aspects of development of apps for the VeeaHub.

You can provide feedback on the development process through the support web site.