System Requirements Specification

for

Intel Cloud Integrity Technology Docker Proxy

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel products are not intended for use in medical, life-saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined."

Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Intel® Cloud Integrity Technology may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

\*Other names and brands may be claimed as the property of others.

Copyright © 2015 Intel Corporation. All rights reserved.

**Table of Contents**

[Introduction 4](#_Toc455563113)

[Purpose 4](#_Toc455563114)

[Scope 4](#_Toc455563115)

[Terminology 4](#_Toc455563116)

[Overview 4](#_Toc455563117)

[Description 5](#_Toc455563118)

[Background and Perspective 5](#_Toc455563119)

[Demonstration 6](#_Toc455563120)

[Docker Proxy Workflow 6](#_Toc455563121)

[Logistics 8](#_Toc455563123)

[Distribution 8](#_Toc455563124)

[Installation 9](#_Toc455563125)

[Uninstallation 9](#_Toc455563126)

[External Interfaces 9](#_Toc455563127)

[Software 9](#_Toc455563128)

[Hardware 9](#_Toc455563129)

[Communications 9](#_Toc455563130)

[Security 9](#_Toc455563131)

# Introduction

This document contains the system requirements specification for the Cloud Integrity Technology Docker Proxy.

## Purpose

The purpose of this document is to clearly articulate the features required to be in the software and document considerations for improvements or additional features that may addressed by future versions of the software.

The intended audience is developers, system engineers, product marketing team, and managers.

## Scope

The Docker binary in the current CIT setup is patched to invoke the CIT components. This needs to be done for all the versions of the Docker installation.

To get around this issue, the proposed solution is to implement a plugin:   
[https://docs.docker.com/engine/extend/](https://docs.docker.com/engine/extend/#_blank)

<https://docs.docker.com/engine/extend/plugins_authorization/>

According to this solution, the plugin will receive the requests that are received by the Docker engine, and make calls to the CIT components as necessary and allow or disallow the launch.

## Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

## Overview

The requirements in this specification are organized into packaging, external interfaces, features, and other requirements.

Packaging covers distribution, installation, and uninstallation.

External interfaces include human, software, hardware, and communication interfaces.

Features include all functional requirements, which are things the system “does”.

Quality requirements include all other requirements, commonly known as non-functional requirements, which are qualities or attributes of the system or how it “behaves” either normally or under special or stressful conditions.

There are aspects of the system, such as security, that include both feature requirements and quality requirements.

A list of requirement categories is provided in the appendix because not all categories are covered by this specification.

Possibilities that have been considered but are not currently requirements are listed in the backlog appendix.

# Description

## Background and Perspective

Currently the Docker binary is patched so that before a launch of a Docker image, the required CIT components are invoked and necessary validation checks have been performed.

The drawback of this approach is that it needs to be repeated for all the supported versions of Docker.

**Pros:**

1. We don't have to modify docker engine, maintain code base of different versions and do the patching.
2. If there is change/addition/deletion in api calls, required changes can be handled easily.
3. Proxy would be able to intercept command line calls as well. So the solution should work in standalone mode as well.

**Cons:**

1. Mounting would take more time as we do it manually.
2. Currently only supports aufs and devicemapper. We would need to provide mount scripts for each different union file systems supported by docker engine.

**List API calls on which docker proxy will invoke CIT component.**

This list will be included in docker proxy but it is not limited to, list of API calls may increase while proxy’s actual implementation.

1. Start
2. Rm
3. Stop
4. Kill
5. Restart

### Demonstration

A plugin is a process running on the same or a different host as the Docker daemon, which registers itself by placing a file on the same Docker host in one of the plugin directories described in [Plugin discovery](https://docs.docker.com/engine/extend/plugin_api/#plugin-discovery).

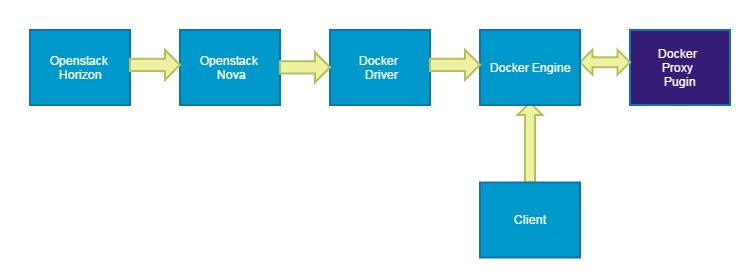
Plugins have human-readable names, which are short, lowercase strings. For example docker\_proxy in our case.

Plugins can run inside or outside containers. Currently running them outside containers is recommended.

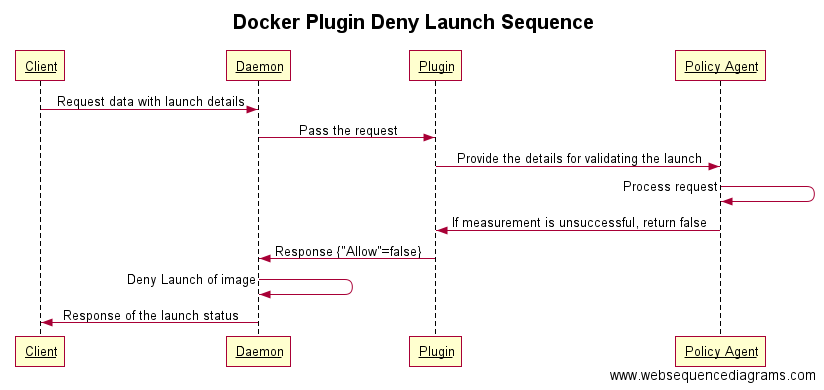
**Control Flow:**

1. Nova docker driver invokes docker engine
2. Docker engine forwards the request to the plugin
3. If it is a start container request, plugin mounts the image and invokes the policy agent.
4. Policy agent verifies the trust policy signature using tagent, creates manifest and invokes vRTM.
5. vRTM invokes MA. MA does the measurement of the files. Calculate cumulative hash and checks the integrity of the image.  Based on the result vRTM maintains a memory map and generates trust report. Also takes care of the cleanup.
6. vRTM return response to Policy agent whether verification is successful or not.
7. On success/failure policy agent returns call to docker proxy plugin. The image is now unmounted.
8. In case of successful measurement, docker proxy returns the response to the docker engine to allow the launch

### Docker Proxy Workflow



# 



**Docker Proxy behavior on these API calls:**

1. **Start:** 
   1. On getting this request Docker Proxy mounts the docker image.
   2. Invoke Policy Agent.
   3. Then PA will call the vRTM API to measure the instance with its name as its UUID, and waits for its response.
   4. After getting request to measure a docker instance, vRTM read trustpolicy.xml, extract out Golden Hash (Precalculated Hash) and Measurement Policy (ME or MO) along with some other values for its internal data structure, then it invokes MA to do measurement.
   5. After MA is done with measuring, vRTM verifies its measurement by comparing measured hash with golden hash, allows or denies the launch based on combination of verification and Measurement Policy, and then send its response back to Policy Agent.
   6. Policy Agent forward the response to Docker Proxy. Docker proxy unmounts the image.
   7. Depending on the response from PA, docker proxy sends response back to docker engine to allow/disallow the launch.
2. **Rm:**
   1. On getting this request Docker Proxy will call vRTM API by its container name to update the status of instance to DELETED.
   2. vRTM will update status and eventually remove this instance entry from its internal table. It will also remove this instance trustreports directory, then it will send the response back to docker proxy.
   3. After receiving the response docker proxy will forward the request to docker engine.
3. **Stop and Kill:**
   1. On getting this request Docker Proxy will call vRTM API to update the status of instance with its name.
   2. vRTM update the status of instance by its name and send the response back to docker proxy.
   3. After receiving the response docker proxy will forward the request to docker engine.
4. **Restart:**

# Logistics

## Distribution

The Docker proxy will be packaged as a Linux self-extracting installer.

## Installation

The Docker proxy installation will update the “plugins” directory of the Docker installation to add the plugin details for this proxy. The setup will also update the DOCKER\_OPTS in the /etc/default/docker to add flag to enable the plugin.

It will specify the host and port where the API requests would be sent and restart the docker engine.

The installation will start a jetty server at a configured port and listen to requests from the Docker engine.

By default the proxy will be configured to run on port 22080.

## Uninstallation

The Docker Proxy will have a “one touch” uninstallation procedure that leaves related data and configuration intact (uninstall).

## Purge

The uninstallation with the “--purge” option will remove the reference of the docker proxy plugin from the DOCKER\_OPTS and also remove the spec file in /etc/docker/plugins folder.

# External Interfaces

## Software

The Docker proxy will execute as a Linux process. The proxy would be invoked via Openstack Horizon and Docker client.

## Hardware

The Docker proxy will operate inside a physical machine.

## Communications

The Docker proxy will provide an HTTP interface for Docker engine to interact via a representational state transfer (REST) message style.

## Security

Note for Sysadmin : Please use firewall to block incoming access to the port number configured for the plugin.