System Requirements Specification

for

Intel Cloud Integrity Technology Attestation Report Hub

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 5](#_Toc452534782)

[Purpose 5](#_Toc452534783)

[Scope 5](#_Toc452534784)

[Terminology 6](#_Toc452534785)

[References 6](#_Toc452534786)

[Overview 6](#_Toc452534787)

[Description 7](#_Toc452534788)

[Background and Perspective 7](#_Toc452534789)

[User Stories 8](#_Toc452534790)

[Demonstration 8](#_Toc452534791)

[Sample Attestation Hub Workflow 8](#_Toc452534792)

[Logistics 8](#_Toc452534793)

[Distribution 8](#_Toc452534794)

[Installation 9](#_Toc452534795)

[Uninstallation 9](#_Toc452534796)

[External Interfaces 10](#_Toc452534797)

[Human 10](#_Toc452534798)

[Software 10](#_Toc452534799)

[Hardware 10](#_Toc452534800)

[Communications 10](#_Toc452534801)

[Features 10](#_Toc452534802)

[Quality Requirements 11](#_Toc452534803)

[User Experience 11](#_Toc452534804)

[Performance 12](#_Toc452534805)

[Scalability 12](#_Toc452534806)

[Capacity 12](#_Toc452534807)

[Availability 13](#_Toc452534808)

[Reliability 13](#_Toc452534809)

[Recoverability 13](#_Toc452534810)

[Maintainability 13](#_Toc452534811)

[Serviceability 14](#_Toc452534812)

[Security 14](#_Toc452534813)

[Regulatory 14](#_Toc452534814)

[Manageability 14](#_Toc452534815)

[Appendix 1 Requirement categories 15](#_Toc452534816)

[Appendix 2 Data model 16](#_Toc452534817)

[Database 16](#_Toc452534818)

[Appendix 4 APIs 17](#_Toc452534819)

[General 17](#_Toc452534820)

[URL 17](#_Toc452534821)

[Headers 17](#_Toc452534822)

[Responses 17](#_Toc452534823)

[Tenants 18](#_Toc452534824)

[Create Tenant 18](#_Toc452534825)

[Delete Tenant 19](#_Toc452534826)

[Retrieve Tenant 19](#_Toc452534827)

[Search Tenant 19](#_Toc452534828)

[Host Assignments 20](#_Toc452534829)

[Create Host-Assignment 20](#_Toc452534830)

[Appendix 5 Configuration 21](#_Toc452534831)

[Application Settings 21](#_Toc452534832)

[Tenant Settings 21](#_Toc452534833)

# Introduction

This document contains the system requirements specification for the Cloud Integrity Technology Attestation Hub (The Cache).

## 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 Attestation Hub regularly pushes attestation information to a remote scheduler service. The intent is that the Attestation Hub shall utilize a plugin architecture allowing easily extensible endpoints for pushing attestation information to different schedulers; for example, one plugin would push attestation data to OpenStack Nova, while another would push attestation data to Kubernetes or Mesos.

The Attestation Hub serves to disassociate the process of retrieving attestations from actual scheduler queries, so that scheduler services can adhere to best practices and retain better performance. The Attestation Hub will regularly query the CIT Attestation Service for SAML attestations for each host. The period for this repeated query will be tied to the validity period of the SAML attestation. The Attestation Hub will maintain only the most recent currently valid attestation for each host, and will refresh attestations when they would expire. The Attestation Hub will verify the signature of the SAML attestation for each host assigned to a tenant, then parse the attestation status and asset tag information, and then will push the parsed key/value pairs to the plugin endpoints enabled

Initially the Attestation Hub will include the necessary plugin to push attestation information OpenStack Nova. Future versions of the Attestation Hub may include plugins for other orchestration services.

The initial development for the Attestation Hub will also include development of the Nova API extension needed to receive the attestation information. Development of modifications to Nova shall follow OpenStack best practices and must be done with the goal of being upstreamed into the community Nova project.

The Attestation Hub will maintain its own Access Control List mapping tenants to hosts and plugin endpoints. The database used for this ACL will be Postgres, so that the Attestation Hub and the CIT Attestation Service may use the same database server.

## 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.

CSP - cloud service provider

CSC - cloud service consumer (the enterprise customer that is using the CSP), this term is used only to refer to an enterprise in situations when it is acting specifically as a customer of a CSP

Tenant – A single customer of the CSP. For the purpose of the Attestation Hub, a single OpenStack Controller environment, or a single Docker environment, etc, are defined as single Tenants. Services like OpenStack may further subdivide their environments with their own multi-tenancy support, but this is outside the scope of the Attestation Hub.

## References

ISO/IEC/IEEE 12207-2008

IEEE 830

RFC 2119 [http://tools.ietf.org/html/rfc2119]

<http://requirements.intel.com>

## 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 Cloud Integrity Technology provides attestation information (for physical and virtual servers, and future support for containers) including “trust status” and “asset tag” attributes. One of the intended use cases for these attributes is workload orchestration. An example of this use case is the current implementation of the CIT integration with OpenStack, which adds a scheduler filter which calls the Attestation Service to retrieve attestation and tag attributes for use in making workload placement decisions.

However, this current implementation has been rejected by the OpenStack community, as the OpenStack Nova scheduler should pull information only from Nova and HostState information. External calls to the CIT Attestation Service result in extreme performance degradation that scales multiplicatively with the number of hosts managed by a given Nova instance and the number of instances being launched. In addition to severely degrading the performance of the scheduler in OpenStack, the large number of simultaneous calls to the Attestation Service repeatedly for the same information results in an eventual crash of the Attestation Service.

The current implementation also has no support for multi-tenancy. Customers have indicated a desire to restrict users to only be able to attest specific hosts. The current implementation grants blanket access to all attestations of all hosts registered to the Attestation Service using the same credentials. For example, if Tenant 1 were to have an account with attestation permissions, they could use the same credentials to discover hosts assigned to Tenant 2, since currently the Attestation Service permissions model does not support granularity beyond allowing specific actions (like retrieving/creating attestations).

To resolve this issue, a new Attestation Hub has been proposed to decouple the retrieval of attestation and asset tag attributes in the CIT Attestation Service from direct scheduler activities. This allows schedulers and orchestrators to continue to use their standard practices for determining host information.

Since the Attestation Cache pushes attestation information to different orchestrator databases, it can be used to provide multi-tenancy by maintaining its own mapping of tenants to hosts, and pushing attestation to a tenant’s orchestrator database for those hosts the tenant is granted access to.

## Solution Summary

* Bullet list of main features
* The Hub maintains its own database of users with relevant permissions.
  + First release will include only Administrator permissions for the Hub.
  + Tenants will not access the Hub at all.
  + Hub users must be distinct and separate from Attestation Service users
* The Hub maintains a database mapping Tenants to hosts
  + Tenants for the Hub are defined as an entity like a CSP customer; one Tenant may have more than one Endpoint (OpenStack controllers; Kubernetes controllers; etc), but each Plugin Endpoint may be associated with only one Tenant. This may be different from the definition of a “tenant” in other solutions (OpenStack for example).
* On a regular basis, the Hub retrieves all SAML attestations performed by the Attestation Service since the last time the Hub made this query
  + The time between refresh will be user configurable and have a default of 10 minutes
  + Attestation and Asset Tag attributes will be updated for all hosts by correlating the host hardware UUID (included in SAML)
  + If the Hub cannot reach the Attestation Service for a certain length of time, the Hub will count all hosts as Untrusted and push that status to all endpoints.
* Only the CSP or similar entity has access to the Hub. At no point shall any Tenant have any credentials with which to access the Hub.
* The Hub pushes Attestation and Asset Tag attributes to all hosts assigned to each Tenant.
  + The endpoints the information will be pushed to are defined in a configuration file for each Tenant. Each configuration file will contain endpoint configurations for each Plugin type used by that Tenant. For example, the config file for Tenant A might have a section for configuration of an OpenStack Nova endpoint.
* The Hub uses a plugin architecture to add support for different endpoints (OpenStack Nova, Kubernetes, etc)
* First release of the Hub will include necessary changes to the OpenStack Nova Trust Filter; Nova Conductor;

## User Stories

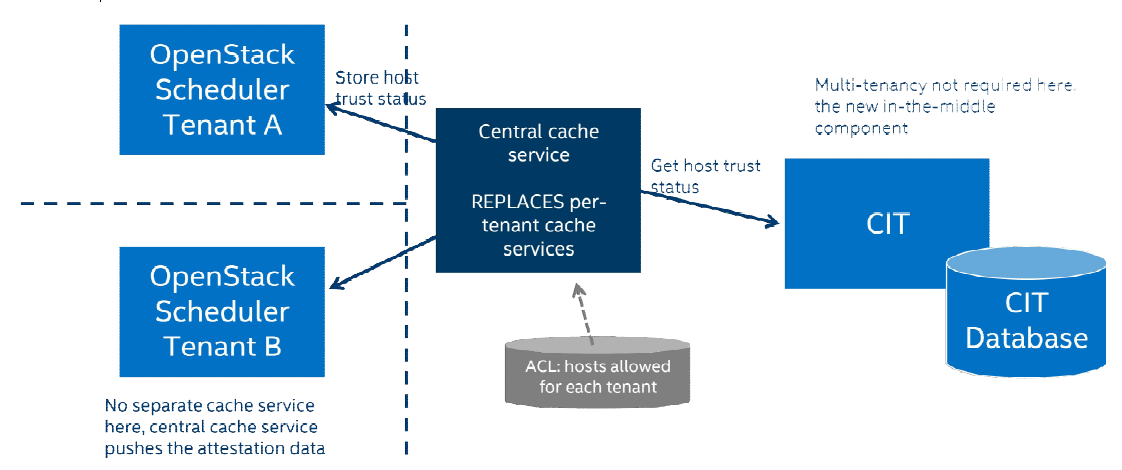
### Demonstration

The Attestation Hub will be used to periodically retrieve all new, valid SAML attestations for all hosts registered with a given Attestation Service, and then push the parsed Attestation and Asset Tag attributes to endpoints (typically some form of orchestration or scheduler service, such as OpenStack Nova) via an endpoint-specific plugin (for example, a plugin that tells the Attestation Hub how to use the OpenStack Nova Conductor) on a per-tenant basis.

The Attestation Hub should support configurations with multiple tenants each with one or more plugins, each with one or more endpoints, where each host is assigned to one or more tenants.

In cases where multiple Tenants use the same Plugin, only a single Plugin is used, but each Tenant may have a unique endpoint configuration for each installed Plugin.

### Sample Attestation Hub Workflow



A Cloud Service Provider uses the Attestation Hub ACL to map hosts registered with the Attestation Service to one or more tenants (one host may be assigned to one tenant, or one host may be assigned to multiple tenants, or a host may remain unassigned).

The CSP administrator creates Plugin configurations for each tenant defining the Attestation Hub plugins, and endpoints for each of those Plugins to use for that tenant.

The Attestation Hub periodically automatically retrieves all SAML attestations that have been created since the last time the Hub retrieved attestations. Only the “new” attestations are retrieved, by searching for all attestations created after the last retrieval timestamp.

The Attestation Hub parses the SAML attestations for their trust and asset tag attributes.

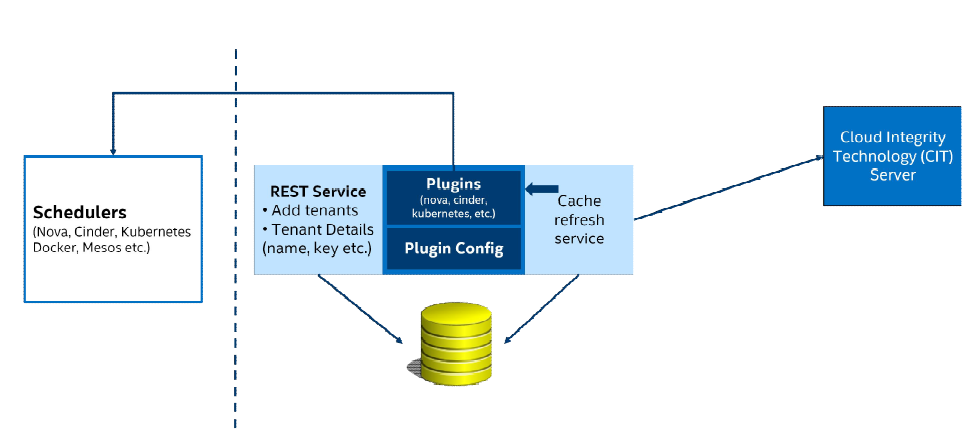
If the Attestation Hub bas not received an updated SAML attestation for a given host for a configurable amount of time, the host will be considered Untrusted. The default timeout time shall be 90 minutes, which is the same as the default SAML validity period in the Attestation Service.

Whenever a host’s SAML is retrieved, the Attestation Hub finds any/all tenants that host is assigned to in the ACL. This should be a single job that runs after the set of new SAML attestations is retrieved; only those hosts whose SAML has been refreshed or timed out should be selected.

Attestation and Asset Tag attributes are optionally signed using a Hub private key. This signing feature MUST be a toggle configured for each Plugin configuration for each Tenant, and should default to “on.”

The Hub pushes the Attestation and Asset Tag attributes to the endpoints defined for each of those Tenants’ plugin configurations.

### Sample Plugin Workflow



The core Attestation Hub manages communication with the Attestation Service; retrieval, verification, and parsing of SAML attestations; signing of attribute data; and maintains the ACL mapping hosts to Tenants.

Plugins are then used to add specific endpoint types, allowing the Attestation Hub to be easily extensible to support new scheduler/orchestrator services.

When a new Plugin is installed, the Attestation Hub will use the Tenant-specific configuration files to find endpoints defined for the new Plugin. A Plugin Endpoint may be an OpenStack Controller for an OpenStack Nova Plugin, for example.

Plugin Endpoints are Tenant-specific. Tenant A may have an OpenStack Controller defined as an Endpoint for an OpenStack Nova Plugin, while Tenant B may have a separate OpenStack Controller defined as a different Endpoint for the same OpenStack Nova Plugin. Tenant C may not use the OpenStack Nova Plugin, but may have an Endpoint defined for a Kubernetes Plugin.

### Sample OpenStack Nova Scheduler Workflow

This section MAY change based on OpenStack community feedback.

The OpenStack Nova Conductor will be modified to include metadata attributes (TBD – need to determine whether we will create a new table or use an existing one). This will include the ability to store signed Attestation and Asset Tag attributes for a given hypervisor, but must also be sufficiently flexible to allow other attributes as well.

In a similar fashion to how the OpenStack Compute Node pushes its own RAM, Disk, CPU and other attributes through the Nova Conductor, the Metadata attributes will also be pushed from a 3rd party source (in this case, the Attestation Hub OpenStack Nova Plugin).

The OpenStack Trust Filter will be modified to access the Nova HostState database to find Attestation and Asset Tag attributes, and will be configured to decrypt the encrypted attributes using a Tenant-specific symmetric key. The Trust Filter will no longer make external calls.

Trust and Asset Tag policy controls will be defined in the Image metadata, and will be retrieved from Glance during Instance launch. This SHOULD be a part of the filter process, pending community approval.

When an Instance is launched, any Trust and Asset Tag policy metadata will be required by the filter to match the Attestation and Asset Tag attributes of the hypervisor during scheduling.

# Logistics

## Distribution

The Attestation Hub is REQUIRED to be packaged as a Linux self-extracting installer.

Plugins for the Attestation Hub are REQUIRED to be packaged as Linux self-extracting installers.

## Installation

The Attestation Hub MUST install a base package containing the Attestation Service integration, ACL tenant-host mapping database, plugin architecture, and all required interfaces.

The Attestation Hub installer MUST include creation of a default administrative user with rights over the Attestation Hub tenant ACL and any API interfaces.

The Attestation Hub installation MUST include configuration of the Attestation Service base URL.

The Attestation Hub installation MUST include the automatic creation of a new Attestation Service API Client request with the “Attestations” role, to be approved by the Attestation Service administrator.

Tenant creation MUST NOT be a part of the installation process. Tenant creation (including the configuration of all plugin endpoints) MUST be supported dynamically and MUST NOT require a service restart.

Plugins to add support for endpoint types (for example – OpenStack Nova; Kubernetes; OpenStack Cinder; Docker; etc) for the Attestation Hub MUST be installed separately as Linux self-extracting installers.

Plugin installers MUST NOT require any configuration at installation time. Configuration of plugin endpoints (URLs, credentials, etc) MUST be part of the standard tenant configuration process.

The Attestation Hub SHOULD support the installation of new Plugins or the uninstallation of existing Plugins without requiring a service restart.

## Uninstallation

The Attestation Hub MUST have a “one touch” uninstallation procedure that leaves related data and configuration intact (uninstall).

The Attestation Hub MUST have a “one touch” uninstallation procedure that erases related data and configuration (purge).

The Attestation Hub Plugins MUST each support a “one touch” uninstallation procedure that removes the plugin while leaving the Attestation Hub and all tenant configurations intact (including any configurations for the removed plugin).

# External Interfaces

## Human

The Attestation Hub SHALL provide a human user command line interface supporting standard functions from other CIT components.

## Software

The Attestation Hub SHALL execute as a Linux process.

The Attestation Hub MUST provide a non-interactive method for providing input and initiating the installation of services.

## Hardware

The Attestation Hub SHALL operate inside a virtual machine or physical machine.

## Communications

The Attestation Hub SHALL provide an HTTP interface for local or remote clients to interact via a representational state transfer (REST) message style.

The Attestation Hub MUST use SSL for all external communication, including the Attestation Service and any Plugin endpoints.

# Features

The Attestation Hub is REQUIRED to support a Plugin architecture for adding additional endpoints to which Attestation and Asset Tag attributes will be pushed. The installation of a new Plugin to support a new endpoint type SHALL NOT require a service restart.

The Attestation Hub is REQUIRED to maintain an Access Control List (ACL) mapping “Tenants” to one or more hosts that are registered in the Attestation Service. This mapping is REQUIRED to support mapping one host to zero or more “Tenants.”

The Attestation Hub is REQUIRED to identify hosts by their hardware UUIDs, which can be parsed from the SAML attestation.

The Attestation Hub is REQUIRED to retrieve all SAML attestation of all hosts registered in the Attestation Service that are newer than the last time the Hub has retrieved attestations, on a regular repeating task. The repetition period for this task MUST be configurable, and default to 10 minutes.

The Attestation Hub is REQUIRED to verify the SAML attestation validity (including time and signature validation) when the attestation is retrieved from the Attestation Service. At this time the attestation and tag attributes as well as the SAML expiration time MUST be stored in the Attestation Hub database.

The Attestation Hub is REQUIRED to track hosts that have not received a new attestation beyond a configurable timeout time. This timeout MUST use either the SAML expiry time OR the time that the SAML was retrieved from the Attestation Service as the start time. The timeout SHOULD default to 90 minutes. Hosts that have not received a new SAML attestation for longer than the timeout period MUST be flagged as “untrusted” by the Hub, and the “untrusted” Attestation attribute MUST be pushed to all Plugin endpoints for any Tenants to whom the “untrusted” host is assigned.

The Attestation Hub is REQUIRED to push attestation and asset tag attributes to each Plugin-supported endpoint defined in each tenant configuration.

The Attestation Hub is REQUIRED to support pushing attestation attributes as “TRUSTED” “UNTRUSTED” or “UNKNOWN”

The Attestation Hub is REQUIRED to support pushing Asset Tag attributes as zero or more text key/value pairs.

The Attestation Hub Plugins are REQUIRED to push host attributes according to the host hardware UUID. Plugins MUST NOT identify hosts by hostname or IP address.

The Attestation Hub is REQUIRED to support signing the attestation and asset tag attributes. Plugin Endpoints (such as the OpenStack Nova Trust Filter) will verify the signature on the signed attributes using the Hub’s public key. The OpenStack Nova Trust Filter modifications to be included in the initial Hub release are REQUIRED to support this signature validation. This feature MUST be a configurable property, set for each Plugin endpoint in each Tenant configuration file.

The first release of the Attestation Hub is REQUIRED to include a Plugin that supports pushing attestation and asset tag data to OpenStack Nova via Nova Conductor. This will include changes to OpenStack Nova and the Trust Filter to support the Hub. This Plugin SHALL NOT be included in the base Attestation Hub Installer, but will be delivered in its own separate installation package.

The OpenStack Nova Trust Filter is REQUIRED to support signature verification of Attestation and Asset Tag attributes that have been pushed by the Hub. Signature validation MUST occur each time the filter accesses host state information to check the Attestation and Asset Tag attributes. This feature is REQUIRED to default to “on,” but also MUST support being disabled by a configuration property.

The OpenStack Nova Trust Filter is REQUIRED to retrieve Attestation and Asset Tag policy information from the Image metadata stored in Glance. The Filter SHALL require that all Attestation and Asset Tag attributes in the Image metadata correspond to the Attestation and Asset Tag attributes on a hypervisor before that host shall be passed by the filter. The host may have additional tags beyond what the Image policy requires, but MUST NOT allow a hypervisor to pass if it does not have all of the attributes required by policy.

The OpenStack Nova Trust Filter is REQUIRED to identify hypervisors by their hardware UUID.

The Attestation Hub is REQUIRED to be able to install using the same host as the Attestation Service, including using the same database server.

The Attestation Hub MUST securely configure TLS between installed services that communicate with each other using HTTPS during operation.

# Quality Requirements

## User Experience

The Attestation Hub MUST operate invisibly to the user, with no user interaction required from either the Administrator or any Tenants, to trigger attestation refreshes either from the Attestation Service or the Plugin endpoints.

The Attestation Hub MUST NOT require service restarts to support the addition of new Tenants, new Plugins, or configuration changes for Tenants or Plugin endpoints.

The Attestation Hub MUST use default configuration options that match any corresponding default configuration options in the Attestation Service such that, for default installations of both the Attestation Service and the Attestation Hub, minimal configuration is required before and after installation.

## Performance

## Scalability

The Attestation Hub MUST support High Availability (HA) and Load Balancing (LB) configurations for both the Attestation Hub itself and the Attestation Service.

The Attestation Hub MUST limit its requests to the Attestation Service such that it does not create a Denial of Service (DOS) for the Attestation Service. This SHOULD be accomplished by limiting the frequency of attestation retrievals, and by retrieving all (and only) new attestations in a single request each time.

## Capacity

The Attestation Hub SHALL operate on a host with at least 2GB memory total and at least 2GB memory free prior to starting the Attestation Hub, at least 20GB disk total and at least 5GB disk available prior to installation, running on an Intel Xeon CPU or 1 vCPU.

Table Capacity Requirements for Quick Start

|  |  |  |
| --- | --- | --- |
| **Resource** | **Minimum** | **Notes** |
| CPU | Intel Xeon or 1 vCPU (Standalone); 2vCPU (if running on the same host as the Attestation Service) | Separate requirements based on whether the Attestation Hub is running on its own dedicated host/VM, or if the Attestation Hub is running on the same host/VM as the Attestation Service. |
| Memory | 2GB (standalone); 4GB (if running on same host as Attestation Service) | Separate requirements based on whether the Attestation Hub is running on its own dedicated host/VM, or if the Attestation Hub is running on the same host/VM as the Attestation Service. |
| Disk | 5GB | Application should not consume much disk space, as old attestations are not retained. |
| Network | 1GBit |  |

## Availability

The Attestation Service is REQUIRED to ensure that attestation and asset tag data are refreshed on the Plugin endpoints before the expiration of the SAML attestation for that host in 99% of cases.

## Reliability

The Attestation Hub MUST verify Tenant-specific Plugin endpoint credentials prior to pushing attestation and asset tag data. In the event of an error, the log MUST clearly indicate the HTTP error code, the Tenant, the Plugin, and the Plugin endpoint related to the error.

The Attestation Hub MUST verify credentials for the Attestation Service during installation.

The Attestation Hub installation MAY fail if the provided Attestation Service credentials are invalid.

The Attestation Hub MAY fail if network errors occur during operation.

The Attestation Hub MAY fail to update attestation and asset tag data if the Plugin endpoints lack prerequisite support for the relevant Plugin.

## Recoverability

The Attestation Hub MUST restart automatically after a reboot.

The Attestation Hub MAY restart automatically after a crash, if sufficient disk space is available.

## Maintainability

The Attestation Hub MUST allow setting the log level to DEBUG, INFO, WARN, or ERROR.

The Attestation Hub MUST log internal details at the DEBUG level.

The Attestation Hub MUST log user actions and results at the INFO level.

The Attestation Hub MUST log alerts and notices to the administrator at the WARN level.

The Attestation Hub MUST log error conditions at the ERROR level.

The Attestation Hub MAY log an incident identification tag and provide the same tag to the client in its error response.

The Attestation Hub MUST allow an administrator to update or patch the code and restart the server for the patches to take effect.

The Attestation Hub MUST allow a new Plugin to be installed without requiring a server restart.

## Serviceability

The Attestation Hub MUST allow an administrator to update any Plugins by installing the new version of the Plugin without needing to uninstall the older version.

The Attestation Hub MUST allow an administrator to view and edit its configuration file and restart the server for the changes to take effect.

The Attestation Hub MUST NOT require a server restart when changing Tenant configurations.

The Attestation Hub MUST NOT require any routine service by the administrator beyond ensuring sufficient memory and disk space is available in accordance with the capacity specification.

## Security

The Attestation Hub MUST NOT log any Tenant or Attestation Service credentials at any log level.

The Attestation Hub SHALL use HTTP Basic authentication for REST interfaces.

The Attestation Hub SHALL include its own user implementation. The initial release MAY include exclusively Administrative users without granularity of permissions.

## Regulatory

The Attestation Hub MUST qualify for export outside the United States.

## Manageability

Configuring Tenant Plugin Endpoints SHOULD be managed via a REST API. In the initial release, this configuration MAY be performed by adding/editing configuration files for each tenant in a designated folder.

Table Sample Tenant Plugin configuration

[DEFAULT]

tenant\_id=<tenant\_name>

enabled\_plugins=NovaCachePlugin, CinderCachePlugin, DockerCachePlugin

[nova\_plugin]

# Plugin specific configuration

nova\_api\_endpoint=<url>

nova\_api\_user=<username>

nova\_api\_password=<password>

[cinder\_plugin]

# Plugin specific configuration

cinder\_api\_endpoint=<url>

cinder\_api\_user=<username>

cinder\_api\_password=<password>

# Can have more configurations based on the plugins need

[docker\_plugin]

# Plugin specific configuration. Plugin can either talk to an API server or directly talk to the DB

docker\_db\_endpoint=<url>

docker\_db\_user=<username>

docker\_db\_password=<password>

# Appendix 1 Requirement categories

The following list shows typical non-functional requirement categories. If the specification includes at least one requirement from a category, it is marked as being included in this specification.

It’s possible to organize the requirements into groups such as reliability (functionality, availability, recoverability, maintainability, and manageability), performance (response time, memory usage, hardware requirements, scalability, and capacity), usability, external interfaces (human, software, hardware, and communication), and security (integrity, confidentiality, availability). Some of these are mentioned below but no effort is made at this time to fully organize the list.

Table Requirement categories

|  |  |
| --- | --- |
| **Category** | **Included** |
| User Experience | Yes |
| Packaging (Distribution, Installation, Uninstallation) | Yes |
| External interfaces (Human, Software, Hardware, Communication, Interoperability) | Yes |
| Performance (Response Time, Memory Usage) | Yes |
| Scalability | Yes |
| Capacity | Yes |
| Availability | Yes |
| Reliability | Yes |
| Recoverability | Yes |
| Maintainability | Yes |
| Serviceability | Yes |
| Security (Privacy, Data Integrity, Confidentiality, Detect, Protect, Recovery) | Yes |
| Regulatory | Yes |
| Manageability | Yes |
| Environmental | No |

# Appendix 2 Data model

The Attestation Hub accepts user input for the following data:

## Database

The Attestation Hub maintains a list of Tenants, a list of all Hosts registered in the Attestation Service, and an ACL mapping the hosts registered in the Attestation Service to zero or more Tenants, where Tenants represent end customers who are allowed to use some subset of available hosts. The tenant\_key column contains the symmetric encryption key used to encrypt attestation and asset tag attributes pushed to the Plugin endpoints assigned to that Tenant.

|  |  |  |
| --- | --- | --- |
| Tenants |  |  |
| UUID | Name | tenant\_key |

The ACL is represented by the tenant\_uuid column in the Hosts table, which defines a mapping of the specified Host to the specified Tenant. The same host MAY be listed multiple times with different tenant\_uuid entries to represent a single host used by multiple tenants. A given Host record MAY contain no tenant\_uuid entry to represent a host that has not been assigned to any Tenant. The Host UUID MUST the Host Hardware UUID. The host\_assignment\_id SHALL be a unique identifier in UUID format for each record in the Hosts table, and is used for the purpose of identifying records to be deleted if a host will no longer be assigned to a given tenant.

|  |  |  |  |
| --- | --- | --- | --- |
| Hosts |  |  |  |
| Host\_uuid | HostName | tenant\_uuid | Host\_assignment\_id |

The Attestation Hub maintains in its database the most recent SAML for each host registered in the Attestation Service. The saml\_expiry column SHOULD be the expiration timestamp parsed from the SAML, but alternatively MAY be the timestamp from when the SAML was retrieved. The host\_uuid MUST be the host hardware UUID, which MUST match the host hardware UUID parsed from the SAML attestation.

|  |  |  |
| --- | --- | --- |
| Host\_SAML\_Report |  |  |
| Host\_uuid | SAML\_report | SAML\_expiry |

# Appendix 3 APIs

## General

Unless specified otherwise, clients can incorporate the following specifications for each API:

### URL

The Attestation Hub server base URL is assumed to be prepended to all the API paths. If the Attestation Hub server is installed at cacheservice.cit.com, the base URL would typically be https://cacheservice.cit.com:9443. For an API with a specified path of /v1/rpc/example the full URL would be https://cacheservice.cit.com:9443/v1/rpc/example.

### Headers

The Attestation Hub application allows HTTP BASIC authentication.

In the global request headers below, the HTTP BASIC authorization parameter is the base-64 encoding of username anonymous with an empty password.

Clients should send the following HTTP headers with all API requests unless specified otherwise:

Authorization: Basic YW5vbnltb3VzOg==

Content-Type: application/json

Accept: application/json

### Responses

#### Bad Request

HTTP code will be 400. This indicates the server could not understand (parse) the request. The response may look like this:

Incident-Tag: 7053316b

Content-Type: text/plain

Content-Length: 12

Bad argument

#### Unauthorized

HTTP code will be 401.

#### Not Found

HTTP code will be 404.

#### Success

HTTP code will be 200.

A success response MAY also include data, an object containing primary data the server is sharing with the client.

A success response MAY also include extra, an object describing related information such as the linked data. For example if data contains blog posts, extra may contain comments for those posts or author information.

#### Failure

HTTP code will be 500. This indicates the server parsed the request successfully, but the request may be invalid (asking for something that isn’t possible) or there may have been an error during processing.

A failure response MUST include faults, a non-empty array with elements describing what went wrong.

A failure response MAY also include extra, an object describing related information such as the request input.

A failure response MUST NOT include data, an object that contains successful response data.

## Tenants

Tenants should correspond to an end customer of a Cloud Service Provider, who has access to some number of hosts, and who has an orchestrator, scheduler, or similar application that can receive CIT attestation and asset tag information via a Attestation Hub Plugin (for example, OpenStack Nova). NOTE – This section may be updated to include tenant configurations as well. The assumption is that, for the initial release, tenant configuration will be done via text configuration files rather than APIs.

### Create Tenant

Creates a new Tenant record in the database. The name is required to be a numeric or text identifier corresponding to a “tenant” or “customer” of the CSP.

#### Request

POST /v1/tenant

#### Request body

{

"name":"Sample Tenant Name",

}

#### Response on success

{

"data": {

"name": "Sample Tenant Name",

"id": “5e494131-3df8-4a84-94e8-a37a1d5a0b8a”

}

}

#### Response on failure

### Delete Tenant

Deletes the Tenant with the provided TenantId. The TenantId is required to be the UUID of the Tenant. SHOULD also delete all records in the Hosts table where the tenant\_id matches that of the deleted Tenant.

#### Request

DELETE /v1/tenant/TenantId

#### Request body

None

#### Response on success

No Content

#### Response on failure

### Retrieve Tenant

Returns Name and ID for a Tenant specified by its TenantId. The TenantId is required to be the UUID of the Tenant.

#### Request

GET /v1/tenant/TenantId

#### Request body

None

#### Response on success

{

"data": {

"name": "Sample Tenant Name",

"id": “5e494131-3df8-4a84-94e8-a37a1d5a0b8a”

}

}

#### Response on failure

### Search Tenant

Returns an array of Name, and Id for all Tenants that match the specified parameters. Valid search parameters include: nameEqualTo=, nameContains=. The parameter “filter=false” will return all tenants.

#### Request

GET /v1/tenant?nameContains=Tenant

#### Request body

None

#### Response on success

{

"data": [{

"name": "Sample Tenant Name1",

"id": “5e494131-3df8-4a84-94e8-a37a1d5a0b8a”

}

{

"name": "Sample Tenant Name2",

"id": “12494131-31f8-4a84-94e8-a37a1d512b82”

}

]}

#### Response on failure

## Host Assignments

Host objects are REQUIRED to be created automatically by a periodically recurring background process that queries the Attestation Service for all registered hosts. Host records in the Attestation Hub SHALL use the same host\_name and host\_id to represent a Host that is used in the Attestation Service (returned via the query /mtwilson/v2/hosts?filter=false sent to the Attestation Service). The host-assignments resource will allow a Attestation Hub administrator to assign a Tenant to a Host. Host assignments are performed via matching a host\_name against a tenant\_id, such that attestation and asset tag attribute data will be pushed to all Plugin endpoints configured for the specified Tenant. When the recurring process pulls the list of Hosts from the Attestation Service, the host\_id field will be updated to match the ID from the Attestation Service that matches the host\_name in the Attestation Hub.

### Create Host-Assignment

Creates a new Host record in the database where the host\_uuid is the hardware UUID of the Host and the provided tenant\_id matches an existing Tenant in the Attestation Hub database. If the host\_uuid does not match any Host currently registered in the Attestation Service, the record will still be created and an error will not be returned. In this way phishing for actual hosts cannot be performed using the Attestation Hub, and a Host matching the host\_uuid may be added to the Attestation Service later. If the host\_name does not match any Host currently known to the Attestation Hub, the Attestation Hub will create the database host record, and then automatically attempt to retrieve a SAML attestation for that host from the Attestation Service. In this way, creating a new host-assignment will retrieve an attestation outside of the automated repeating task, which otherwise may cause a significant delay in host attestation availability.

#### Request

POST /v1/host-assignment

#### Request body

{

"tenant\_id":"<TenantUuid>",

"host\_uuid":"<HostHardwareUuid>",

}

#### Response on success

{

"data": {

"tenant\_name": "Sample Tenant Name",

"host\_uuid": "22952a72e24194f208200e76fd3900da",

"host\_assignment\_id": “5e494131-3df8-4a84-94e8-a37a1d5a0b8a”

}

}

#### Response on failure

### Create Host-Assignment

Deletes the record in the Hosts table where the specified host\_uuid is mapped to the specified tenant\_id.

#### Request

DELETE /v1/host-assignment

#### Request body

{

"tenant\_id":"<TenantUuid>",

"host\_uuid":"<HostHardwareUuid>"

}

#### Response on success

204 No Content

***Response on failure***

# Appendix 4 Configuration

The configuration settings described in this section are used to affect the behavior of the Attestation Hub server. The Application Settings section describes settings that are specific to the Attestation Hub server. The Tenant Settings section describes settings for individual Tenants, including configurations for each Plugin the Tenant will use.

## Application Settings

[DEFAULT]

debug=False

logdir=/var/log/cit-cache-svc

refresh\_interval=10 # in Minutes

tenant\_confi\_dir=tenant.conf/

[database]

connection = psql://user:password@xxx.xxx.xxx.xxx/cit-cache-svc-db

[attestation\_server\_details]

attestation\_server\_ip=xxx.xxx.xxx.xxx

attestation\_server\_port=8443

attestation\_server\_auth\_blob=admin:password

attestation\_service\_url=<url to get SAML report>

## Tenant Settings

Each tenant would have a configuration file like below. Config lists the

enabled plugins and also contains the required configurations for each of

those plugins

[DEFAULT]

tenant\_id=<tenant\_name>

enabled\_plugins=NovaCachePlugin, CinderCachePlugin, DockerCachePlugin

[nova\_plugin]

# Plugin specific configuration

nova\_api\_endpoint=<url>

nova\_api\_user=<username>

nova\_api\_password=<password>

[cinder\_plugin]

# Plugin specific configuration

cinder\_api\_endpoint=<url>

cinder\_api\_user=<username>

cinder\_api\_password=<password>

# Can have more configurations based on the plugins need

[docker\_plugin]

# Plugin specific configuration. Plugin can either talk to an API server or directly talk to the DB

docker\_db\_endpoint=<url>

docker\_db\_user=<username>

docker\_db\_password=<password>

# Appendix 5 Plugin Architecture

The Attestation Hub is REQUIRED to utilize a “plugin” architecture. The core Attestation Hub manages communication with the Attestation Service; retrieval, verification, and parsing of SAML attestations; encryption of attribute data; and maintains the ACL mapping hosts to Tenants.

Plugins are then used to add specific endpoint types, allowing the Attestation Hub to be easily extensible to support new scheduler/orchestrator services.