OpenAttestation SDK Overview

Version: 1.5
October 2012

OpenAttestation SDK (OAT)

A SDK for Remote Attestation

Table of Contents

1 Introduction

The Trusted Computing Group has defined a series of specifications that define how a commercial computing platform can support code measurement in a trusted manner. Intel has developed an OpenAttestation SDK built on NSA's National Information Assurance Research Laboratory (NIARL) developed Host Integrity at Startup to measure and report status for host platforms which contain a Trusted Platform Module (TPM). The implementation takes advantage of Infrastructure Work Group Integrity Report Schema Specification
http://www.trustedcomputinggroup.org/resources/infrastructure_work_group_integrity_report_schema_specification_version_10/

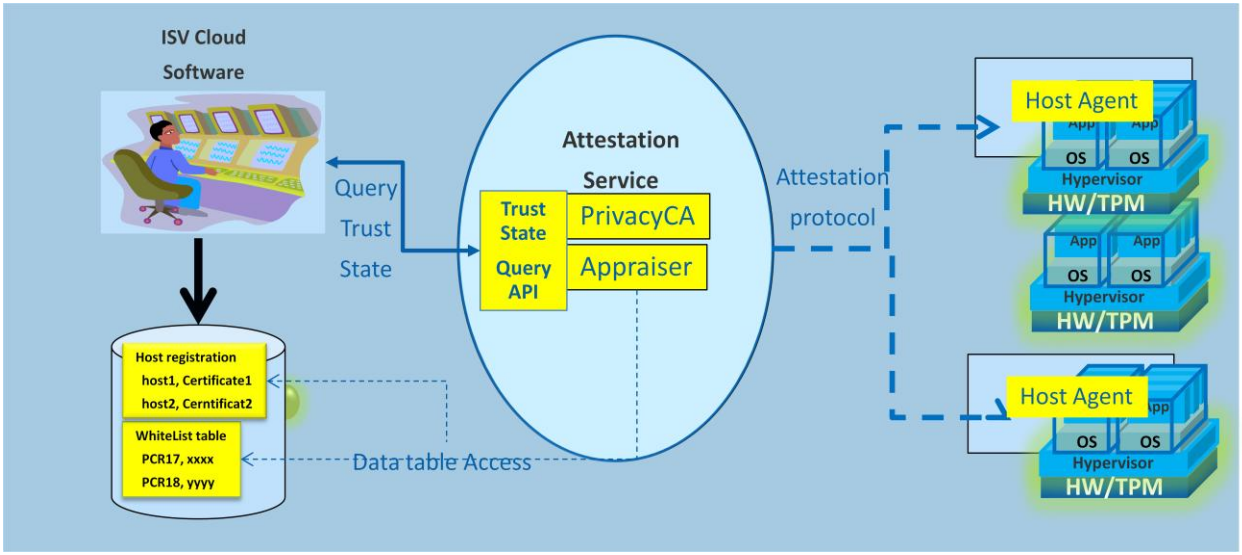
The OpenAttestation SDK supports web API for 3rd party software to integrate and access web-based attestation appraisals in order to support cloud usage model.

The OpenAttestation SDK is intended to be merged, modified and distributed as 3rd party ISV's cloud management stacks. ISV should enhance OAT for its distribution or enhance security implementation into their products

1.1 Scope

This guide describes OpenAttestation architecture and exported APIs. The intended audients can be ISV developers or administers of cloud providers.

1.2 Architecture



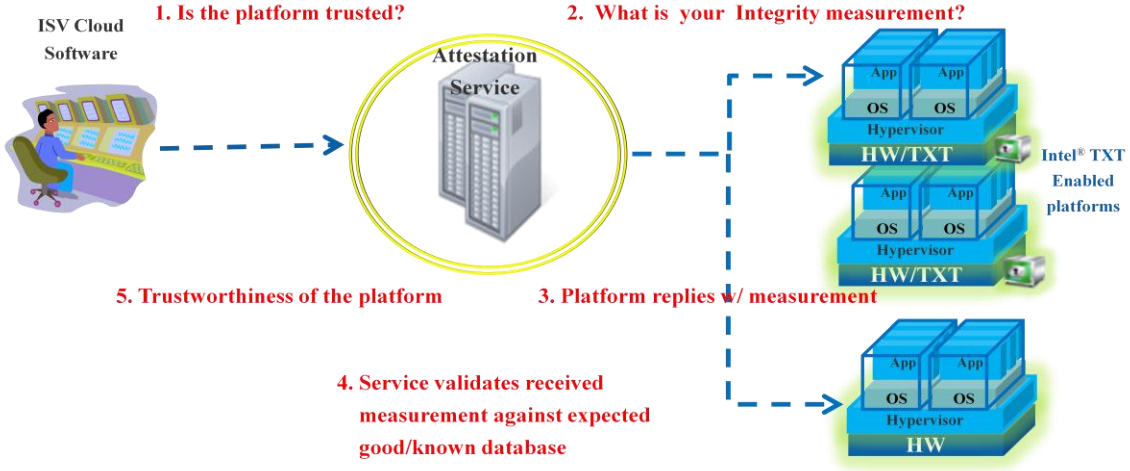
This above diagram highlights OpenAttestation architecture. OpenAttestation, as part of 3rd party ISV's software stack, enables ISV software remotely to retrieve and verify target

hosts' TPM PCRs and verify hosts' integrity through exported Query API. Accordingly, administer of cloud providers can pre-setup WhilteList tables, e.g., good/known PCR values, a.k.a, measurements, through SDK exported data table access APIs.

OpenAttestation service runs with Privacy Certificate Authority (PrivacyCA) and Appraiser to communicate with HostAgent runs on target hosts. PrivacyCA provisions and certifies hosts' Attestation Identify key, where Appraiser communicates with HostAgent, access through TrouserS TSS stack, to retrieve and verify a host's PCRs.

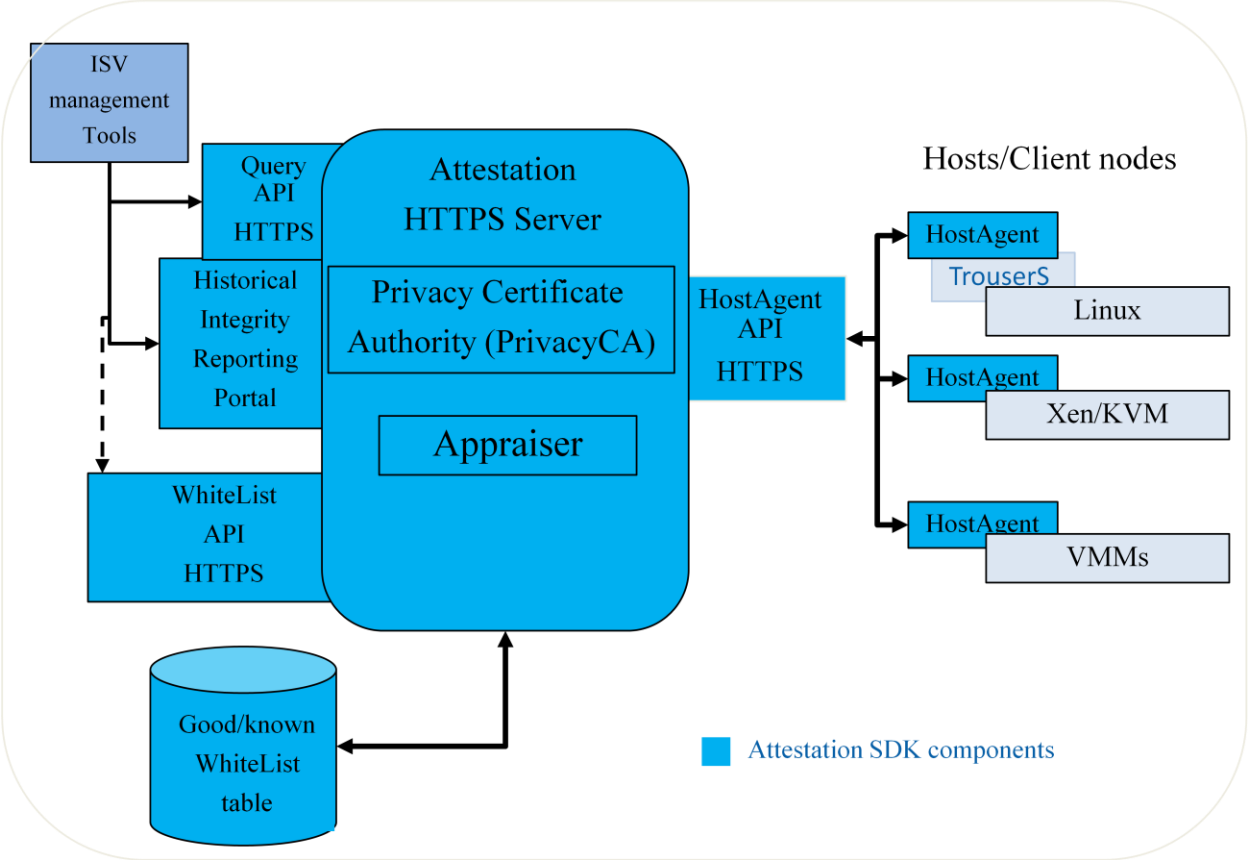
2 A Sample Usage model

Following diagram exemplifies possible usage model in cloud computing environment. Where in a cloud computing environment, a subscriber may require a service must be run on a trusted platform. That is, hosts running with good/known software. Attestation service provides ISV software with capability to verify target hosts' integrity before dispatch task(s) onto the host.



3 OpenAttestation Components

Following diagram describes exported components of Attestation service



Attestation Service API HTTPS service exporting Restful API to maintain Hosts and enable ISV software to query hosts' integrity status

WhiteList Service API HTTPS service providing the APIs and storage to define the various MLEs in the environment, their attributes, policy-driven Trust definition, the 'Golden Measurements' for the modules, and the PCR Manifests.

Historical Integrity Reporting Portal Appraiser, as a reference implementation, internally tracks each host's historical integrity reports. The Portal provides an interface to view historical data as well as details of integrity data reported by hosts

HostAgent API HTTPS Connection point for HostAgent to send in integrity report with TCG Integrity Report Schema

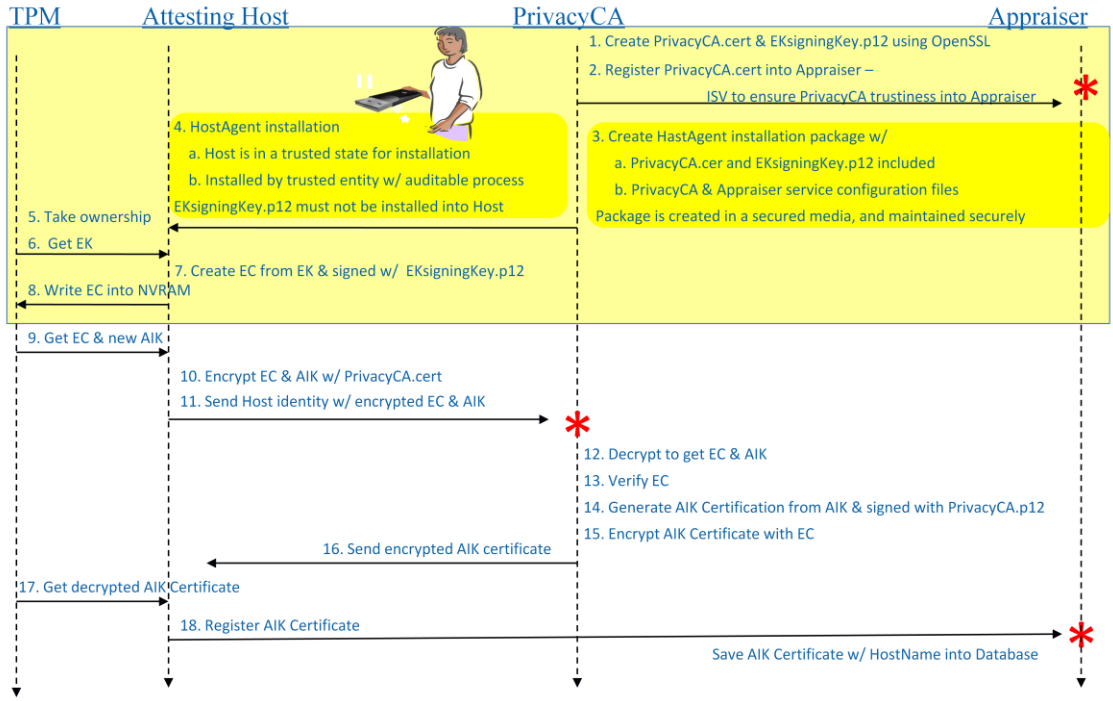
ProvacyCA verifies Hosts' TPM Endorsement Key (EK) & issue EK Certificate (EKC). EKC used by Host to request Attestation Identification Key (AIK) Certificate (AIC)

Appraiser verifies Host measurement which signed by Host AIC

4 OpenAttestation control flow

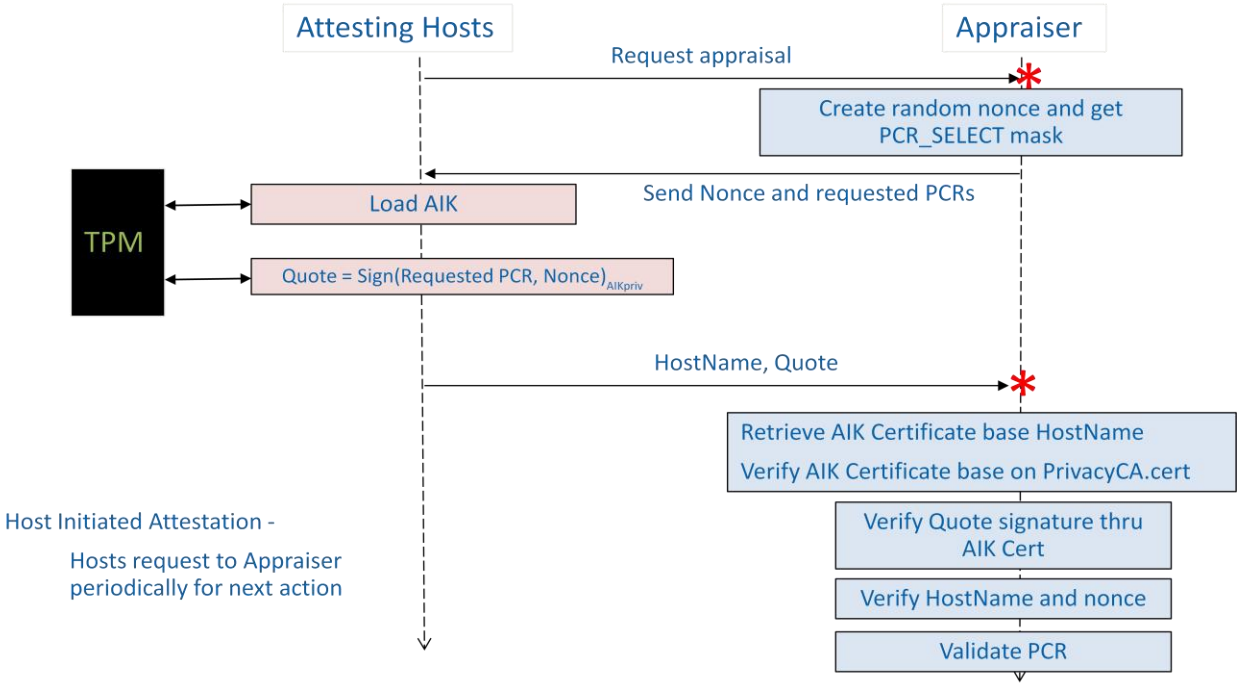
4.1 HostAgent Installation/Provisioning – EK and AIK certificates

The above flow describes HostAgent provisioning to get Endorsement Key (EK) certificate and how it gets Attestation Identity Key (AIK) certificate from ProvicyCA, and then register its Hostname associated with AIC into Appraiser



5 Attestation Control flow

5.1 Host initiated attestation



Attestation flow is run as *Host initiated Attestation*, that is, Host system will poll Appraiser periodically to query if any active task it should perform, Appraiser can then set active task to request integrity report.

6 OpenAttestation API

We will describe OpenAttestation exported Restful APIs, which is Integrity Attestation Service API enables querying a platform's integrity state, and WhiteList API which enables administrator to setup good/known measurement PCRs for appraiser.

6.1 Security Notes

For access control and security purpose, all the APIs exported from Attestation service will be HTTPS based. However the SDK internal implementation doesn't impose any authentication or access control verification, rather it solely depends following 2 mechanisms for authentication –

- ISV specific authentication verification – An *Auth-blob* is included in all the headers as request to attestation service. Attestation service will pass this blob to ISV specific function for authentication verification.
- Tomcat Truststore client and server certificate authentication for platform accesses – Tomcat provided SSL authentication through Keystore/Truststore authentication for client to access server as well as client verifying server before connection

It is strongly recommended that ISV must implement its own software specific authentication logic, and Attestation service provider should also implement Tomcat 2-way authentication to securely control platforms that are allowed to access to API services

6.2 API Request/Response Types

Current Attestation APIs support JSON data serialization format, where

- The request format is specified by using *Context-Type* header and is required for operations that havea request body
- The response format is specified in request header using *Accept*
- *Accept* type is default to JSON if not specified in request header

6.3 Authentication header

All requests to attestation server is required to include an authentication blob, Auth-blob, within request header. Where Auth-blob is ISV authentication specific and is opaque to OpenAttestation API.

Note: ISV must implement its own ***ISV_Authentication_verification()*** function to complete the authentication logic. The default ***ISV_Authentication_Verification()*** logic is true.

Context-Type: application/json

Accept: application/json

Auth-Blob: ISV_specific_string

6.4 Example of request with headers: JSON

Following example should a request to attestation server and its response in JSON format

Request	Response
POST AttestationService/resources/PollHosts Host: Attestation.ras.com:8443 Context-Type: application/json Accept: application/json Auth_blob: authenticationBlob	HTTP/1.1 200 OK Server: BaseHTTP/0.3 Python/2.7.1+ Date: Mon, 15 Oct 2012 22:36:58 CST Context-Type: application/json

{
“hosts”: [
“host1”,
“host2”
]
}
	{
	"hosts":[
	{"host_name":" host1","trust_lvl":"trusted",
	"vtime":"2012-10-15T22:36:58.836+08:00"},
	{"host_name":" host2","trust_lvl":"trusted",
	"vtime":"2012-10-15T22:36:58.836+08:00"}
	]
	}

6.5 Attestation Service API

Following lists Attestation Service APIs

API Type	Method	Method Name	Description
Provisioning	POST	/hosts	Adds/Registers a new host
	PUT	/hosts	Updates the configuration of an existing host
	DELETE	/hosts?Hostname	Deletes the specified configured host
	GET	/hosts?searchCriteria	Retrieves the list of all the hosts matching the search criteria. If the search criteria is empty, all the hosts registered are retrieved.
Query	POST	/PollHosts	Gets the current trust status of all the hosts requested.

6.5.1 Add Host

This API registers a new host with the system for attestation.
Method type: POST
Sample Call: https://Server_Name:8443/AttestationService/resources/hosts
Sample Input:
{"HostName":"TestHost1","IPAddress":"192.168.1.1","Port":"9999","BIOS_Name":"EPSD","BIOS_Version":"55","BIOS_Oem":"EPSD","VMM_Name":"Xen","VMM_Version":"4.1.1","VMM_OSName":"RHEL","VMM_OSVersion":"6.1","Email":"","AddOn_Connection_String":"","Description":""}
Sample Output: True (HTTP Status code: 200)
If proper BIOS or VMM or OEM or OS is not chosen, accordingly the error message would be different.
Sample Input:
{"HostName":"TestHost1","IPAddress":"192.168.192.168","Port":"9999","BIOS_Name":"EPSD","BIOS_Version":"55","BIOS_Oem":"EPSD","VMM_Name":"Xen","VMM_Version":"4.1.1","VMM_OSName":"RHEL","VMM_OSVersion":"6.1","Email":"","AddOn_Connection_String":"","Description":""}
Sample Output: HTTP Status code: 400
{
"error_code": 2001,
"error_message": "Network error - Error while connecting to TrustAgent on host [192.168.192.168]"
}

6.5.2 Edit Host

This API allows the user to update all the host details except for the host name. Note that the IP address and Port number are required for OpenSource hosts.
Method Type: PUT
Sample Call: https://Server_Name:8443/AttestationService/resources/hosts
Sample Input:
{"HostName":"TestHost1","IPAddress":"192.168.1.1","Port":"9999","BIOS_Name":"EPSD","BIOS_Version":"55","BIOS_Oem":"EPSD","VMM_Name":"Xen","VMM_Version":"4.1.1","VMM_OSName":"RHEL","VMM_OSVersion":"6.1","Email":"testemail@intel.com","AddOn_Connection_String":"","Description":"Updated Description"}
Sample Output: True (HTTP Status code: 200)

6.5.3 Delete Host

This API deletes a host that is already configured in the system.
Method type: DELETE
Sample Call:
https://Server_Name:8443/AttestationService/resources/hosts?hostName=TestHost1
Sample Output: true (HTTP Status code: 200)

If in case the host that the user is trying to delete does not exist, an error would be returned back to caller.
Sample Call:
https://Server_Name:8443/AttestationService/resources/hosts?hostName=TestHost25
Sample Output: HTTP Status code: 400


```
{
   "error_code": 2000,
   "error_message": "Host not found - Host - 'TestHost25' that is being deleted does not exist."
}
```


6.5.4 Query for Hosts

This API retrieves the list of the hosts registered with OAT based on the search criteria specified. If the search criteria is empty, then all the hosts are retrieved or else only the specific hosts whose name matched the criteria is retrieved.
Method Type: GET
Sample Call: https://Server_Name:8443/AttestationService/resources/hosts?searchCriteria=192
Sample Output:


```
"[{"HostName":"Server_Name","IPAddress":"127.0.0.1","Port":9999,"BIOS_Name":"EPSD","BIOS_Version":"v60","BIOS_Oem":"EPSD","VMM_Name":"Xen","VMM_Version":"4.1.0","VMM_OSName":"SUSE","VMM_OSVersion":"11 P2","AddOn_Connection_String":"","Description":"10.1.71.149 SUSE","Email":"","Location":null},
{"HostName":"192.168.1.101","IPAddress":"192.168.1.101","Port":9999,"BIOS_Name":"Intel_Ubuntu","BIOS_Version":"T060","BIOS_Oem":"Intel Corp.","VMM_Name":"QEMU","VMM_Version":"11.10-0.14.1","VMM_OSName":"UBUNTU","VMM_OSVersion":"11.10","AddOn_Connection_String":null,"Description":null,"Email":null,"Location":null}]"
```


6.5.5 Poll Hosts for Trust Status

This API was specifically added for the OpenStack integration, which also provides the current trust status of all the hosts passed in.
Method Type: POST
Sample Call: https://Server_Name:8443/AttestationService/resources/PollHosts
Sample Input: {"hosts":["ubuntu-txtnode","Phase2Host1"]}
Sample Output:


```
{"hosts":[
{"host_name":"localhost","trust_lvl":"trusted",
"vtime":"2012-10-15T22:36:58.836+08:00"},
{"host_name":"ubuntu1104","trust_lvl":"trusted",
```

"vtime":"2012-10-15T22:36:58.836+08:00"}
]}

6.6 WhiteList Service API

Following lists WhiteList Service APIs

API Type	Method	Method Name	Description
OEM Provisioning	POST	/OEM	Adds a new OEM
	PUT	/OEM	Updates the existing OEM
	DELETE	/OEM?Name	Deletes the specified OEM
	GET	/OEM	Retrieves the list of all OEMs
OS Provisioning	POST	/OS	Adds a new OS information into the DB
	PUT	/OS	Updates an existing OS details
	DELETE	/OS?Name&Version	Deletes the specified OS details
	GET	/OS	Retrieves the list of all the OS configured in DB

API Type	Method	Method Name	Description
MLE Provisioning	POST	/mles	Adds/Registers a new MLE
	PUT	/mles	Updates the MLE & its manifest list
	DELETE	/mles?mleName&mleVersion&osName&osVersion&oemName	Deletes an existing MLE
	GET	/mles?searchCriteria	Gets a list of MLEs based on the search criteria. If criteria is empty all the MLE are returned back.
	GET	/mles/manifest?mleName&mleVersion&osName&osVersion&oemName	Retrieves the details of the MLE including the PCR manifests (Note: Module manifests are not included)
WhiteList Mgt	POST	/mles/whitelist/pcr	Adds a new PCR white list entry for the specified MLE
	PUT	/mles/whitelist/pcr	Updates an existing PCR white list entry
	DELETE	/mles/whitelist/pcr?pcrName&mleName&mleVersion&osName&osVersion&oemName	Deletes an existing PCR white list entry

6.6.1 Add Operating System (OS)

This API creates new OS information in the system.
Method type: POST
Sample Call: https://Server_Name:8443/WLMService/resources/os
Sample Input: {"Name":"OS Name 1","Version":"v1234","Description":"Test OS"}
Sample Output: True (HTTP Status code: 200)
If the OS Name/Version combination already exists in the database an appropriate error would be returned back.
Sample Input: {"Name":"OS Name 1","Version":"v1234","Description":"New description"}
Sample Output: (Http Status Code: 400)


```
{
    "error_code": 1006,
    "error_message": "Data Error - OS OS Name 1 Version v1234 already exists in the database"
}
```


6.6.2 Edit OS

This API updates the details of an existing OS that is configured in the system. Only the description field is editable.
Method type: PUT
Sample Call: https://Server_Name:8443/WLMService/resources/os
Sample Input: {"Name":"OS Name 1","Version":"v1234","Description":"Test OS description updated"}
Sample Output: True (HTTP Status code: 200)
If in case the user tries to update a non-existent OS Name/Version combination, the follow error would be sent back to the caller.
Sample Input: {"Name":"OS Name 1","Version":"v12345","Description":"Test OS description updated"}
Sample Output: (Http Status Code: 400)


```
{
    "error_code": 1006,
    "error_message": "Data Error - OS OS Name 1 Version v12345 does not exist in the database"
}
```


6.6.3 Delete OS

This API deletes an existing OS that is configured in the system.
Method type: DELETE

Sample Call: https://Server_Name:8443/WLMService/resources/os?Name=OS Name 1&Version=v1234
Sample Output: True (HTTP Status code: 200)
If the OS that the user is trying to delete does not exist, an appropriate error message would be returned back to the caller.
Sample Call: https://Server_Name:8443/WLMService/resources/os?Name=OS Name 1&Version=v12
Sample Output: (Http Status Code: 400)


```
{
  "error_code": 1006,
  "error_message": "Data Error - OS OS Name 1 Version v12 does not exist in the database"
}
```


6.6.4 View OS

Retrieves the list of all the OS that is configured in the system.
Method type: GET
Sample Call: https://Server_Name:8443/WLMService/resources/os
Sample Output: (HTTP Status code: 200)


```
[(3)
  {
    "Name": "RHEL",
    "Version": "6.1",
    "Description": null
  },
  {
    "Name": "UBUNTU",
    "Version": "11.10",
    "Description": null
  },
]
```

6.6.5 Add OEM

This API creates new OEM information in the system.
Method type: POST
Sample Call: https://Server_Name:8443/WLMService/resources/oem
Sample Input: {"Name":"OEM1","Description":"New OEM description"}
Sample Output: True (HTTP Status code: 200)
If the OEM Name already exists in the database an appropriate error would be returned back.
Sample Input: {"Name":"OEM1","Description":"New description"}
Sample Output: (Http Status Code: 400)


```
{
   "error_code": 1006,
   "error_message": "Data Error - OEM OEM1 already exists in the database"
}
```


6.6.6 Edit OEM

Updates the description of the OEM already configured in the system.
Method type: PUT
Sample Call: https://Server_Name:8443/WLMService/resources/oem
Sample Input: {"Name":"OEM1","Description":"Updated description"}
Sample Ouput: True (HTTP Status code: 200)
If in case the OEM does not exist in the database, an appropriate error would be returned back.
Sample Input: {"Name":"OEM2","Description":"OEM2 Description"}
Sample Output: (Http Status Code: 400)


```
{
   "error_code": 1006,
   "error_message": "Data Error - OEM OEM2 does not exist in the database"
}
```

6.6.7 View OEM

This API retrieves the list of all the OEMs that are currently configured in the system.
Method type: GET
Sample Call: https://Server_Name:8443/WLMService/resources/oem
Sample Output: (HTTP Status code: 200)


```
[(3)
  {
    "Name": "GENERIC",
    "Description": "Default Oem for testing"
  },
  {
    "Name": "EPSD",
    "Description": "Intel white boxes"
  },
  {
    "Name": "HP",
    "Description": "HP Systems"
  }
]
```


6.6.8 Delete OEM

This API deletes an existing OEM configured in the system.
Method type: DELETE
Sample Call:
https://Server_Name:8443/WLMService/resources/oem?Name=OEM1
Sample Output: True (HTTP Status code: 200)
If in case the OEM that the user is trying to delete does not exist, an appropriate error would be returned back.
Sample Call:
https://Server_Name:8443/WLMService/resources/oem?Name=OEM2
Sample Output: (Http Status Code: 400)


```
{
  "error_code": 1006,
  "error_message": "Data Error - OEM OEM2 does not exist in the database"
}
```

6.6.9 Add MLE (Measured Launch Environment)

This API creates a new MLE in the system. There will be 2 types of MLEs: BIOS and VMM. BIOS MLEs are tied to OEMs that are already configured. VMM type MLEs are tied to the OS that is pre-configured in the system. Each host created in the system, will be tied to both a BIOS MLE and a VMM MLE.
Note: The manifest list can be empty and can be added later on using the white list management REST APIs.
Method type: POST
Sample Call: https://Server_Name:8443/WLMService/resources/mles
Sample Input: {"Name":"New RHEL MLE","Version":"123","OsName":"RHEL","OsVersion":"6.1","Attestation_Type": "PCR","MLE_Type":"VMM","Description":"Test","MLE_Manifests": [{"Name": "18", "Value": "87654321"}]}
Sample Output: True (HTTP Status code: 200)
If the OS information does not exist in the system, an appropriate error would be returned back.
Sample Input: {"Name":"New RHEL MLE","Version":"123","OsName":"RHEL","OsVersion":"6.9","Attestation_Type": "PCR","MLE_Type":"VMM","Description":"Test","MLE_Manifests": [{"Name": "18", "Value": "87654321"}]}
Sample Output: (Http Status Code: 400)


```
{
   "error_code": 1006,
   "error_message": "Data Error - OS [RHEL] Version [6.9] does not exist."
}
```


6.6.10 Edit MLE

This API allows the user to edit an already configured MLE in the system.
Method Type: PUT
Sample Call: https://Server_Name:8443/WLMService/resources/mles
Sample Input: {"Name":"New RHEL MLE","Version":"123","OsName":"RHEL","OsVersion":"6.1","Attestation_Type":"PCR","MLE_Type":"VMM","Description":"Updated Description","MLE_Manifests": [{"Name": "18", "Value": "87654321"},{"Name": "19", "Value": "12345678"}]}
Sample Output: True (HTTP Status code: 200)

6.6.11 View MLE

This API retrieves the details of the MLE including the associated Good Known Values (GKVs). For MLEs of BIOS type, the OEM information needs to be passed. For MLEs of VMM type, the OS Name/OS Version has to be passed in.
Method Type: GET
Sample Call:
https://Server_Name:8443/WLMService/resources/mles/manifest?mleName=EPSD&mleVersion=55&oemName=EPSD
Sample Output:


```
{
   "Name": "EPSD",
   "Version": "55",
   "Description": "",
   "OsName": null,
   "OsVersion": null,
   "Attestation_Type": "PCR",
   "OemName": "EPSD",
   "MLE_Manifests": [(1)
      {
         "Name": "0",
         "Value": "E3A29BD603BF9982113B696CD37AF8AFC58E2877"
      }],
   "MLE_Type": "BIOS"
}
```


For the MLEs of type VMM:
Sample Call:
https://Server_Name:8443/WLMService/resources/mles/manifest?mleName=Xen&mleVersion=4.1.1&osName=RHEL&osVersion=6.1
Sample Output:


```
{
   "Name": "Xen",
   "Version": "4.1.1",
   "Description": "",
   "OsName": "RHEL",
   "OsVersion": "6.1",
   "Attestation_Type": "PCR",
   "OemName": null,
   "MLE_Manifests": [(3)
      {
         "Name": "17",
         "Value": "EC2CF197E639AFA7D30EE800723FDEBF609A7B6A"
      },
      {
         "Name": "18",
         "Value": "709057CABCFCFCB8C12C7759369ADBC180B568FA"
```

```
      },
      {
        "Name": "19",
        "Value": "6E8042086A6A383CCAD60C064DD521335FDC9BCE"
      }
    ],
    "MLE_Type": "VMM"
}
```


6.6.12 Search MLE

This API provides the user capability to retrieve the list of MLEs matching the search criteria. If the search criterion is empty, it retrieves the list of all the MLEs.
Method Type: GET
Sample Call:
https://Server_Name:8443/WLMService/resources/mles?searchCriteria=HP
Sample output:


```
[(2)
  {
    "Name": "HP",
    "Version": "P67",
    "Description": "",
    "OsName": null,
    "OsVersion": null,
    "OemName": "HP",
    "MLE_Manifests": null,
    "Attestation_Type": "PCR",
    "MLE_Type": "BIOS"
  },
  {
    "Name": "HP",
    "Version": "P67-2",
    "Description": "",
    "OsName": null,
    "OsVersion": null,
    "OemName": "HP",
    "MLE_Manifests": null,
    "Attestation_Type": "PCR",
    "MLE_Type": "BIOS"
  }
]
```

6.6.13 Delete MLE

This API deletes an existing MLE if it is not associated with any host. If the MLE type is VMM, then the OS name & version information have to be provided. If the MLE is of BIOS type, then the OEM information has to be passed into the API call.
Method Type: DELETE
Sample Call:
https://Server_Name:8443/WLMService/resources/mles?mleName=New RHEL MLE&mleVersion=123&osName=RHEL&osVersion=6.1
Sample Output: True (HTTP Status code: 200)
If any of the information does not match with the configuration in the database, an appropriate error would be returned back.
Sample Call:
https://Server_Name:8443/WLMService/resources/mles?mleName=New RHEL MLE&mleVersion=123&osName=RHEL&osVersion=6.9
Sample Output:

```
{
   "error_code": 1007,
   "error_message": "WLM Service Error - MLE not found in attestion data to update"
}
```

6.6.14 Add PCR White List

This API adds a white list value (good known value) for the specified MLE. If the MLE type is VMM, then the OS name & version information have to be sent. If the MLE type is BIOS, then the OEM information has to be passed into the API call.
Note: This API is only for adding white lists for MLE's that support PCR attestation. These include BIOS MLEs and OpenSource VMM MLEs.
Method type: POST
Sample Call:
https://Server_Name:8443/WLMService/resources/mles/whitelist/pcr
Sample Input:

```
{"pcrName":"0","pcrDigest":"ACE7AB9D3582097C9BC739C9311D60B5B5F5603A", "mleName":"Intel_BIOS","mleVersion":"s60", "oemName": "Intel Corporation"}
```

Sample Output: True (HTTP Status code: 200)
If the MLE information does not exist in the system, an appropriate error would be returned back.

6.6.15 Update PCR White List

This API updates the specified white list value (good known value) for the specified MLE.
Note: This API is only for updating white lists for MLE's that support PCR attestation. These include BIOS and OpenSource hypervisors.
Method type: PUT
Sample Call:
https://Server_Name:8443/WLMService/resources/mles/whitelist/pcr
Sample Input:
{"pcrName":"0","pcrDigest":"DE1343582097C9BC739C9311D60B5B5F5603A", "mleName":"Intel_BIOS","mleVersion":"s60", "oemName": "Intel Corporation"}
Sample Output: True (HTTP Status code: 200)
If in case either the PCR details or the MLE details do not match, an appropriate error message would be returned back to the caller.

6.6.16 Delete PCR White List

This API deletes the specified white list value (good known value) for the specified MLE.
Note: This API is only for deleting white lists for MLE's that support PCR attestation. These include BIOS & OpenSource hypervisors.
Method type: DELETE
Sample Call:
https://Server_Name:8443/WLMService/resources/mles/whitelist/pcr?pcrName=0&mleName=Intel_BIOS&mleVersion=s60&oemName=Intel Corporation
Sample Output: True (HTTP Status code: 200)
If in case either the PCR details or the MLE details do not match, an appropriate error message would be returned back to the caller.