Design Guide for Intel® SGX Provisioning Certificate Caching Service (Intel® SGX PCCS)

Rev 0.72
Jan, 2023

Table of Contents

1. Introduction..3

1.1. Terminology..3

2. Overview..5

2.1. Architecture Overview – ECDSA-Based Data Center Attestation..5

2.2. Intel® SGX ECDSA Public Key and Data Structure Hierarchy..6

3. API Specification for PCCS..7

3.1. Get PCK Certificate..7

3.2. Get PCK Cert CRL..9

3.3. Get TCB Info..11

3.4. Get Intel's QE Identity..13

3.5. Get Intel's QvE Identity..14

3.6. Get Intel's TD QE Identity..16

3.7. Get Root CA CRL..18

3.8. Post Platforms IDs..19

3.9. Get Platform IDs..20

3.10. Put Platform Collateral to Cache..22

3.11. Cache Data Refresh..24

3.11.1. Refresh through HTTP Request..24

3.11.2. Scheduled Cache Data Refresh..26

3.12. Get CRL by endpoint..26

3.13. Get default platform policy..27

3.14. Put appraisal policy..28

4. Database..30

4.1. Schema Definition..30

4.2. Data Access Objects..31

4.2.1. platformsDao..31

4.2.2. pckcertDao..32

4.2.3. fmspcTcbDao..32

4.2.4. pckCertchainDao..33

4.2.5. pckcrlDao..33

4.2.6. pcsCertificatesDao..33

4.2.7. platformsRegDao..34

4.2.8. platformTcbsDao..34

4.2.9. enclaveIdentityDao 35

4.2.10. crlCacheDao 35

4.2.11. appraisalPolicyDao 35

5. Cache Fill Mode 37

6. Configuration File 38

7. Administration Tool (Admin Tool) 40

8. Cache Management Flows 43

8.1. Platform Registration 43

8.2. Handling TCB Recoveries 44

8.2.1. Special handling of Multi-Package TCB Recovery 45

8.3. Refreshing Expiring Collateral 46

8.4. Database migration 47

8.4.1. Migrating from v2 caching database (DCAP v1.8 and before) to v3 caching database (DCAP v1.9 and after) 47

8.4.2. Migrating database from DCAP v1.9 to newer versions of DCAP 47

1. Introduction

This document describes software architecture for the Provisioning Certification Caching Service (PCCS) delivered as part of Intel® Software Guard Extensions Data Center Attestation Primitives (Intel® SGX DCAP) in order to support third party attestation model in data center environment.

1.1.Terminology

PCCS	Provisioning Certification Caching Service.
PCS	Intel® SGX Provisioning Certification Service.
Intel® SGX Quote	Data structure used to provide proof to an off-platform entity that the application's enclave is running with SGX protections on a trusted SGX-enabled platform.
Quoting Enclave (QE)	Enclave signed trusted by the attestation infrastructure owner to sign and issue Quotes/attestations about other enclaves.
Elliptic Curve Digital Signature Algorithm (ECDSA)	Signing cryptographic algorithm as described in FIPS 186-4.
Provisioning Certification Enclave (PCE)	Architectural enclave that uses a Provisioning Certification Key (PCK) to sign QE REPORT structures for Provisioning or Quoting Enclaves. These signed REPORTS contain ReportData that indicates that attestation keys or provisioning protocol messages were created on genuine hardware.
Provisioning Certification Key (PCK)	Signing key available to the Provisioning Certification Enclave for signing certificate-like QE REPORT structures. The key is unique to the processor package or the platform instance, the HW TCB, and the PCE version (PSVN).
Provisioning Certification Key Certificate (PCK Cert)	The x.509 Certificate chain signed and distributed by Intel for every SGX-enabled platform. Quote verifiers use this cert to verify that the QE-generating quotes are valid and running on a trusted SGX platform at the particular PSVN. It matches the private key generated by the PCE.
Platform Provisioning ID (PPID)	Provisioning ID for the processor package or the platform instance. PPID is not TCB-dependent.
Security Version Number (SVN)	Version number that indicates when the relevant security updates have occurred. New versions can have increased functional versions without incrementing the SVN.
Intel® SGX Provisioning TCB	Trusted Computing Base of Intel® SGX provisioning that includes the platform HW TCB and the PCE SVN.

PCEID	Identifies the version of the PCE used to generate the PPID and the PCK signing key. The length of PCEID is 2 bytes.
QEID	The unique ID of an SGX platform which is generated by ID Enlave. The size of QEID is 16 bytes.
CPUSVN	CPU security version number. The size of CPUSVN is 16 bytes.
PCESVN	The security version number of PCE enclave. The size of PCESVN is 2 bytes.
Encrypted_PPID	PPID encrypted with PPIDEK. The size of Encrypted_PPID is 384 bytes.
FMSPC	Description of the processor package or platform instance including its. Family, Model, Stepping, Platform Type, and Customized SKU. The size of FMSPC is 6 bytes.

Table 1-1: Terminology

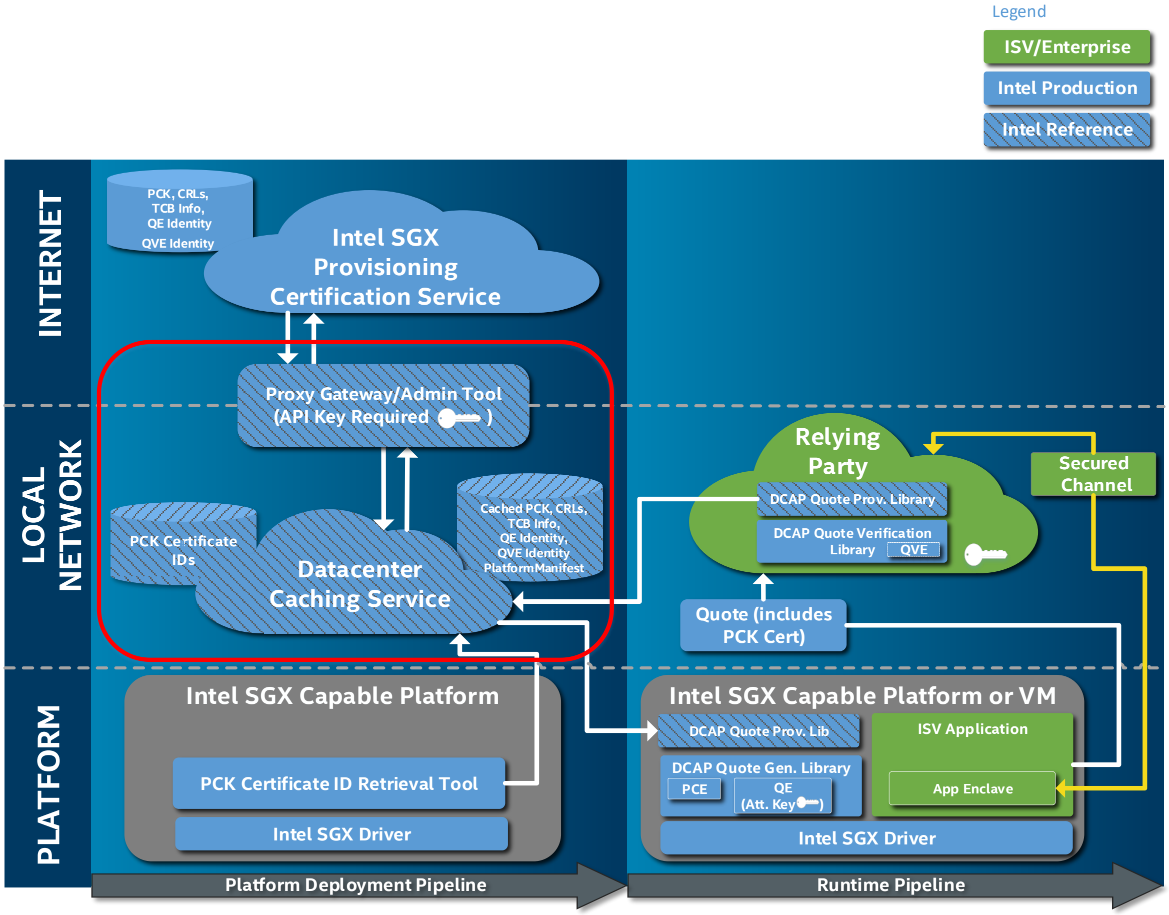
2.Overview

Intel provides a reference Quote Provider Library (QPL) and a reference Provisioning Certification Caching Service (PCCS) to enable SGX attestation run-time workloads without a dependence on the Intel® services. Intel SGX DCAP also provides the network interface layer called PCK Certificate Collateral Network Library (QCNL). The libraries and the PCCS interaction with the Intel Provisioning Certification Service (PCS) can be configured in a number of ways to fit the customer's attestation infrastructure.

This document covers the high-level design details for the PCCS.

2.1. Architecture Overview – ECDSA-Based Data Center Attestation

The proposed architecture centers around a caching service that maintains the Intel® SGX attestation collateral for all servers in the CPP or datacenter, and provides that collateral to quote-generating servers and verifiers that verify the quotes during runtime workloads. This document describes the PCCS RESTful APIs and the tools needed to get the attestation collateral into the PCCS for Intel® SGX-enabled server platforms. It also describes the RESTful APIs exposed to platforms to retrieve that collateral during runtime.



Legend
ISV/Enterprise
Intel Production
Intel Reference
INTERNET
LOCAL NETWORK
PLATFORM
PCK, CRLs, TCB Info, QE Identity QVE Identity
Intel SGX Provisioning Certification Service
Proxy Gateway/Admin Tool (API Key Required)
PCK Certificate IDs
Datacenter Caching Service
Cached PCK, CRLs, TCB Info, QE Identity, QVE Identity PlatformManifest
Relying Party
DCAP Quote Prov. Library
DCAP Quote Verification Library
QVE
Secured Channel
Quote (includes PCK Cert)
Intel SGX Capable Platform
PCK Certificate ID Retrieval Tool
Intel SGX Driver
Intel SGX Capable Platform or VM
DCAP Quote Prov. Lib
ISV Application
DCAP Quote Gen. Library
PCE
QE (Att. Key)
App Enclave
Intel SGX Driver
Platform Deployment Pipeline
Runtime Pipeline

Figure 1: Intel® SGX DCAP Architecture Diagram

2.2. Intel® SGX ECDSA Public Key and Data Structure Hierarchy

The figure below shows the relationship between the SGX Attestation collateral (keys and data structures) used for Intel® SGX ECDSA Quote Generation and Quote Verification. This collateral is generated and signed by Intel and stored in the Intel SGX PCS.

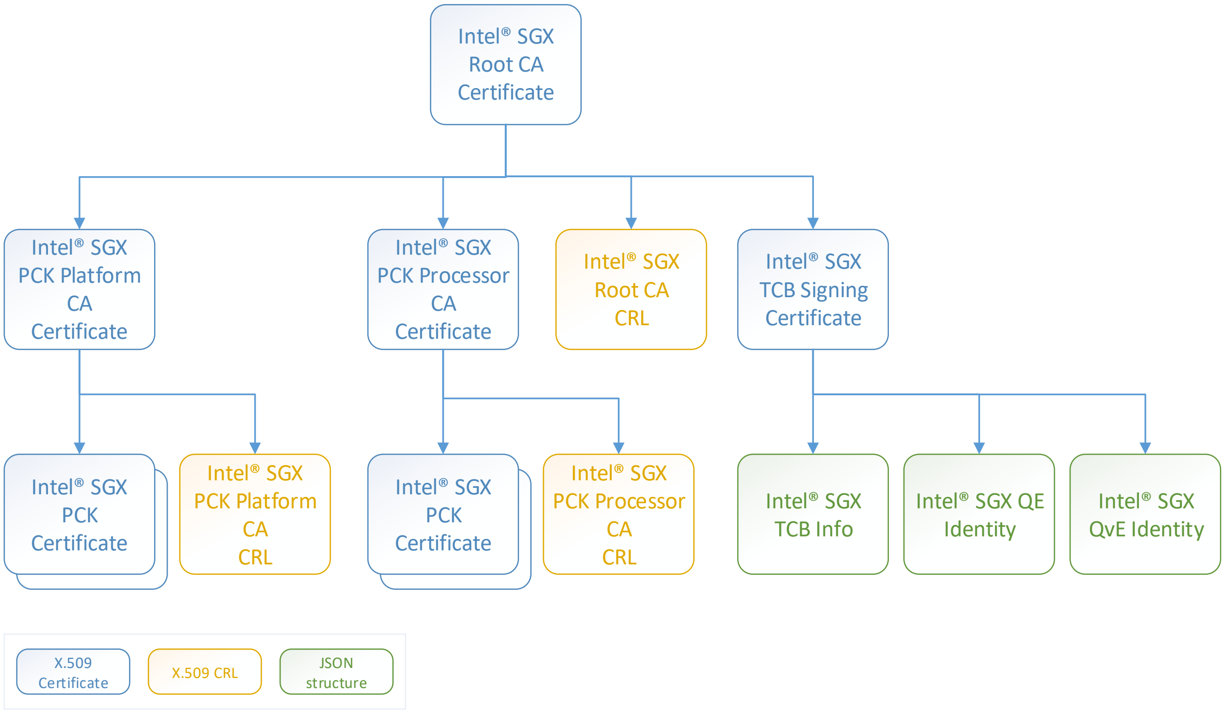

Intel® SGX Root CA Certificate
Intel® SGX PCK Platform CA Certificate
Intel® SGX PCK Processor CA Certificate
Intel® SGX Root CA CRL
Intel® SGX TCB Signing Certificate
Intel® SGX PCK Certificate
Intel® SGX PCK Platform CA CRL
Intel® SGX PCK Certificate
Intel® SGX PCK Processor CA CRL
Intel® SGX TCB Info
Intel® SGX QE Identity
Intel® SGX QvE Identity
X.509 Certificate
X.509 CRL
JSON structure

Figure 2: Intel® SGX ECDSA Public Key and Data Structure Hierarchy

3. API Specification for PCCS

3.1. Get PCK Certificate

Retrieve the X.509 SGX Provisioning Certification Key (PCK) certificate for an SGX-enabled platform at a specified TCB level.

GET https://pccs-server-url:8081/sgx/certification/v3/pckcert

GET https://pccs-server-url:8081/sgx/certification/v4/pckcert

Request

Name	Type	Request Type	Required	Pattern	Description
encrypted_ppid	String	Query	False	^[0-9a-fA-F]{512}$	Base16-encoded Encrypted_PPID
cpusvn	String	Query	True	^[0-9a-fA-F]{32}$	Base16-encoded CPUSVN value
pcesvn	String	Query	True	^[0-9a-fA-F]{4}$	Base16-encoded PCESVN value
pceid	String	Query	True	^[0-9a-fA-F]{4}$	Base16-encoded PCEID value
qeid	String	Query	True	^[0-9a-fA-F]{32}$	Base16-encoded QEID value

Response

PckCert (x-pem-file) - PEM-encoded representation of Intel® SGX PCK Certificate in case of success (200 HTTP status code)

Status Codes

Code	Model	Headers	Description
200	PckCert	**SGX-PCK-Certificate-Issuer-Chain (String)** - URL-encoded Issuer Certificate chain for SGX PCK Certificate in PEM format. It consists of SGX Intermediate CA Certificate (Processor CA) followed by SGX Root CA Certificate **SGX-TCBm (String)** - Hex-encoded string representation of concatenation of	Successfully completed

	CPUSVN (16 bytes) and PCESVN (2 bytes) as returned in corresponding SGX PCK Certificate	
400		Invalid request parameters
404		No cache data for this platform
461		The platform was not found in the cache.
462		Certificates are not available for certain TCBs.
500		Internal server error occurred
502		Unable to retrieve the collateral from the Intel SGX PCS.

Process

1. Checks request parameters upon client request and returns the 400 error if any parameter is invalid.
2. Gets the platform object from **platforms** table with the input {qeid, pceid} as key (see platformsDao.getPlatform).
3. If the platform was found, which means the platform was already cached:
    a. Queries PCK cert for this platform and PCK certificate issuer chain from cache db with the input {qeid, cpusvn, pcesvn, pceid } as key (see pckcertDao.getCert).
    b. Goes to step 4.
4. If collateral was not retrieved in step 3:
    a. If platform is not cached
        i. If caching fill mode is 'LAZY':
            i) Gets all PCK certs for this platform from Intel PCS v4 API with {encrypted_ppid, pceid} for single-package platform, or {platform_manifest,pceid} for multi-package platform.
            ii) Parses the first cert (X.509) in the array to get FMSPC and ca type ('processor' or 'platform').
            iii) Contacts Intel PCS again to get SGX TCB info as well as TDX TCB info (if available) with the above FMSPC value.
            iv) Gets the best cert with PCKCertSelectionTool using {cpusvn, pcesvn, pceid, SGX TCB info, PCK certs}.
            v) Updates the cache tables:
            "platforms" table: calls platformsDao.upsertPlatform to update the platforms table;
            "pck_cert" table: first calls pckcertDao.deleteCerts to delete old records associated with the {qeid, pceid}, then for each certificate fetched in i., calls pckcertDao.upsertPckCert to insert the certificate;
            "platform_tcbs" table: for the new raw TCB in the request and all old cached raw TCBs, inserts/updates the new TCB mapping by calling platformTcbsDao.upsertPlatformTcbs;

“fmspc_tcbs” table: calls fmspcTcbDao.upsertFmspcTcb to update fmspc_tcbs table for both SGX TCB info and TDX TCB info (if available);
“pck_certchain” table: calls pckCertchainDao.upsertPckCertchain to update pck_certchain table with the ca type in step ii.;
“pcs_certificates” table: calls pcsCertificatesDao.upsertPckCertificateIssuerChain to update PCK certificate issuer chain with the ca type in step ii.

vi) Returns the PCK cert in the response body and PCK

certificate issuer chain in the response header.

vii) Responds with the 200 status code.

ii. Else return 461 (not found) error

b. Else

i. Gets PCK certificates from cache DB with {qeid, pceid} (see pckcertDao.getCerts).

ii. Gets SGX TCBInfo from cache DB with the fmpsc of the platform (see fmspcTcbDao.getTcbInfo).

iii. Runs PCk Cert selection tool with the raw TCB, PCK certificates, and SGX TCB Info.

iv. Gets PCK certificate issuer chain from cache DB (see pckCertchainDao.getPckCertChain).

v. If success, returns the “best” certificate and certificate chain, else returns the 404 error.

vi. Updates platform_tcbs table for this raw TCB (see platformTcbsDao.upsertPlatformTcbs).

5. Else

a. Returns the PCK cert in the response body and PCK certificate issuer chain in the response header.

3.2.Get PCK Cert CRL

Retrieve the X.509 Certificate Revocation List with the revoked Intel® SGX PCK Certificates. The CRL is issued either by Intel® SGX Platform CA or by Intel® SGX Processor CA.

GET https://pccs-server-url:8081/sgx/certification/v3/pckcrl

GET https://pccs-server-url:8081/sgx/certification/v4/pckcrl

Request

Name	Type	Request Type	Required	Pattern	Description
ca	String	Query	True	Enum: “processor”, “platform”	Identifier of the CA that issued the requested CRL

encoding	String	Query	True	Enum: “der”	Optional identifier of the encoding for the requested CRL. If the parameter is not provided, HEX-encoded DER is assumed. If “der” is provided, raw DER format CRL will be returned.

Response

PckCrl (PKIX-CRL) – DER or HEX-encoded DER representation of SGX Platform CA CRL or SGX Processor CA CRL in case of success.

Status codes

Code	Model	Headers	Description
200	PckCrl	**SGX-PCK-CRL-Issuer-Chain** Issuer Certificate chain for SGX PCK CRL. It consists of SGX Intermediate CA Certificate (Processor CA) followed by SGX Root CA Certificate	Successfully completed
400			Invalid request parameters
404			PCK CRL cannot be found
500			Internal server error occurred
502			Unable to retrieve the collateral from the Intel SGX PCS.

Process

1. Checks request parameters and returns the 400 error if any input parameter is invalid
2. Queries PCK CRL along with CRL issuer chain with the key {ca} (see pckcrlDao.GetPckCrl).
    a. If record exists:
        i. Returns pck_crl in the response body and the certchain in the response header.
        ii. Responds with the 200 status code.
    b. Else:
        i. If caching fill mode is not ‘LAZY’, returns the 404 (No cache data) error to client.
        ii. If caching fill mode is ‘LAZY’
            1. Gets PCK CRL from Intel PCS v4 API with {ca}. If failed, returns the 404 error.
            2. Updates “**pck_crl**” and “**pcs_certificates**” table:
                a. Calls pckcrlDao.upsertPckCrl(ca, crl), crl is a response body.
                b. Calls pcsCertificatesDao.upsertPckCrlCertchain with the ca and certchain.

3. Returns PCK CRL in the response body and CRL certchain in the response header.
4. Responds with the 200 status code.

3.3. Get TCB Info

Retrieve Intel® SGX or TDX TCB information for the given FMSPC

GET https://pccs-server-url:8081/sgx/certification/v3/tcb

GET https://pccs-server-url:8081/sgx/certification/v4/tcb

GET https://pccs-server-url:8081/tdx/certification/v4/tcb

Request

Name	Type	Request Type	Required	Pattern	Description
fmspc	String	Query	True	^[0-9a-fA-F]{12}	Base16-encoded FMSPC value
update	String	Query	False		Type of update to TCB Info. Enum: early, standard If not provided standard is assumed. early indicates an early access to updated TCB Info provided as part of a TCB recovery event (commonly the day of public disclosure of the items in scope**) standard indicates standard access to updated TCB Info provided as part of a TCB recovery event (commonly approximately 6 weeks after public disclosure of the items in scope**)

Response
TcbInfo (JSON) - Intel® SGX TCB Info encoded as JSON string in case of success

Status codes

Code	Model	Headers	Description
200	TcbInfo	**SGX-TCB-Info-Issuer-Chain (v3)** **TCB-Info-Issuer-Chain (v4)**- Issuer Certificate chain for Intel® SGX TCB Info. It consists of Intel® TCB Signing Certificate followed by Root CA Certificate	Successfully completed
400			Invalid request parameters
404			TCB information for provided {fmspc} cannot be found
500			Internal server error occurred
502			Unable to retrieve the collateral from the Intel SGX PCS

Process

1. Checks request parameters and returns the 400 error if any parameter is invalid.
2. Queries TCB Info along with Intel® TCB Info issuer chain with key {type, fmspc, version} (see fmspcTcbDao. getTcbInfo). Type can be 0:SGX or 1:TDX.
    a. If record exists:
        i. Returns tcb_info in the response body and the issuer chain in the response header.
        ii. Responds with the 200 status code.
    b. Else:
        i. If caching fill mode is not 'LAZY', returns the 404 (No cache data) error to client.
        ii. If caching fill mode is 'LAZY'
            1. Gets TCB info from Intel PCS with {type, fmspc, version}. If failed, return the 404 error.
               If type is 0:SGX, send request to /sgx/ url, else if type is 1:TDX, send request to /tdx/ url. Version will be translated to either /v3/ or /v4/.
            2. Updates "**fmspc_tcbs**" and "**pcs_certificates**":
               fmspc_tcbs table: calls fmspcTcbDao.upsertFmspcTcb;
               pcs_certificates table: calls pcsCertificatesDao.upsertTcbInfoIssuerChain.
            3. Returns TCB info in the response body and TCB info certchain in the response header.
            4. Responds with the 200 status code.

3.4. Get Intel’s QE Identity

Retrieve the Quote Identity information for the Quoting Enclave issued by Intel.

REST API

GET https://pccs-server-url:8081/sgx/certification/v3/qe/identity

GET https://pccs-server-url:8081/sgx/certification/v4/qe/identity

Name	Type	Request Type	Required	Pattern	Description
update	String	Query	False		Type of update to QE Identity. Enum: early, standard If not provided standard is assumed. early indicates an early access to updated QE Identity provided as part of a TCB recovery event (commonly the day of public disclosure of the items in scope**) standard indicates standard access to updated QE Identity provided as part of a TCB recovery event (commonly approximately 6 weeks after public disclosure of the items in scope**)

Response

QEIdentity (JSON) - QE Identity data structure encoded as JSON string in case of success

Status codes

Code	Model	Headers	Description

200	QEIdentity	**SGX-Enclave-Identity-Issuer-Chain** - Issuer Certificate chain for Intel® SGX QE Identity. It consists of Intel® SGX TCB Signing Certificate followed by Intel® SGX Root CA Certificate	Successfully completed
404			QE identity information cannot be found
500			Internal server error occurred
502			Unable to retrieve the collateral from the Intel SGX PCS

Process

1. Queries QE identity along with SGX Enclave identity issuer chain (see enclaveIdentityDao. getEnclaveIdentity).
   a. If record exists:
      i. Returns qe_identity in the response body and the issuer certchain in the response header.
      ii. Returns the 200 status code.
   b. Else:
      i. If caching fill mode is not 'LAZY', returns the 404 (No cache data) error to client.
      ii. If caching fill mode is 'LAZY':
         1. Gets QE identity from the correspoding API version of Intel PCS. If failed, returns the 404 error.
         2. Updates "**enclave_identities**" and "**pcs_certificates**" table:
            enclave_identities table: calls enclaveIdentityDao.upsertEnclaveIdentity.
            pcs_certificates table: calls pcsCertificatesDao.upsertEnclaveIdentityIssuerChain.
         3. Returns qe_identity in the response body and the certchain in the response header.
         4. Returns the 200 status code.

3.5. Get Intel's QvE Identity

Retrieve Identity information for Quote Verification Enclave issued by Intel.

GET https://pccs-server-url:8081/sgx/certification/v3/qve/identity

GET https://pccs-server-url:8081/sgx/certification/v4/qve/identity

Name	Type	Request Type	Required	Pattern	Description

update	String	Query	False		Type of update to QVE Identity. Enum: early, standard If not provided standard is assumed. early indicates an early access to updated QVE Identity provided as part of a TCB recovery event (commonly the day of public disclosure of the items in scope**) standard indicates standard access to updated QVE Identity provided as part of a TCB recovery event (commonly approximately 6 weeks after public disclosure of the items in scope**)

Response

QvEIdentity (JSON) - QvE Identity data structure encoded as JSON string in case of success

Status codes

Code	Model	Headers	Description
200	QvEIdentity	**SGX-Enclave-Identity-Issuer-Chain** - Issuer Certificate chain for Intel® SGX QvE Identity. It consists of Intel® SGX TCB Signing Certificate followed by Intel® SGX Root CA Certificate	Successfully completed
404			QvE identity information cannot be found
500			Internal server error occurred
502			Unable to retrieve the collateral from the Intel SGX PCS

Process

1. Queries QvE identity along with Intel® SGX Enclave identity issuer chain (see enclaveIdentityDao.getEnclaveIdentity).
    a. If record exists:
        i. Returns qve_identity in the response body and certchain in the response header.
        ii. Returns the 200 status code.
    b. Else:
        i. If caching fill mode is not 'LAZY', returns the 404 (No cache data) error to client.
        ii. If caching fill mode is 'LAZY':
            1. Gets QvE identity from the correspoding API version of Intel PCS . If failed, returns the 404 error.
            2. Updates "**enclave_identities**" and "**pcs_certificates**" table:
               enclave_identities table: calls enclaveIdentityDao.upsertEnclaveIdentity;
               pcs_certificates table: calls pcsCertificatesDao.upsertEnclaveIdentityIssuerChain.
            3. Returns qve_identity in the response body and certchain in the response header.
            4. Returns the 200 status code.

3.6.Get Intel's TD QE Identity

Retrieve Identity information for TD Quote Enclave issued by Intel.

GET https://pccs-server-url:8081/tdx/certification/v4/qe/identity

Name	Type	Request Type	Required	Pattern	Description
update	String	Query	False		Type of update to QE Identity. Enum: early, standard If not provided standard is assumed. early indicates an early access to updated QE Identity provided as part of a TCB recovery event (commonly the

					day of public disclosure of the items in scope**) standard indicates standard access to updated QE Identity provided as part of a TCB recovery event (commonly approximately 6 weeks after public disclosure of the items in scope**)

Response

TDQEIdentity (JSON) - TDQE Identity data structure encoded as JSON string in case of success

Status codes

Code	Model	Headers	Description
200	TDQEIdentity	**SGX-Enclave-Identity-Issuer-Chain** - Issuer Certificate chain for Intel® SGX TDQE Identity. It consists of Intel® SGX TCB Signing Certificate followed by Intel® SGX Root CA Certificate	Successfully completed
404			TDQE identity information cannot be found
500			Internal server error occurred
502			Unable to retrieve the collateral from the Intel SGX PCS

Process

1. Queries TDQE identity along with Intel® SGX Enclave identity issuer chain (see enclaveIdentityDao.getEnclaveIdentity).
    a. If record exists:
        i. Returns tdqe_identity in the response body and certchain in the response header.
        ii. Returns the 200 status code.
    b. Else:
        i. If caching fill mode is not 'LAZY', returns the 404 (No cache data) error to client.
        ii. If caching fill mode is 'LAZY':
            1. Gets TDQE identity from Intel PCS v4 API. If failed, returns the 404 error.
            2. Updates "**enclave_identities**" and "**pcs_certificates**" table:

enclave_identities table: calls enclaveIdentityDao.upsertEnclaveIdentity;
pcs_certificates table: calls pcsCertificatesDao.upsertEnclaveIdentityIssuerChain.

3. Returns tdqe_identity in the response body and certchain in the response header.
4. Returns the 200 status code.

3.7. Get Root CA CRL

Retrieve Root CA CRL.

GET https://pccs-server-url:8081/sgx/certification/v3/rootcacrl

GET https://pccs-server-url:8081/sgx/certification/v4/rootcacrl

Request

No parameters

Response

Root CA CRL – The HEX-encoded DER representation of Root CA CRL in case of success

Status codes

Code	Model	Headers	Description
200			Successfully completed
404			Root CA CRL cannot be found
500			Internal server error occurred
502			Unable to retrieve the collateral from the Intel SGX PCS

Process

1. Queries root CA record(id=1) from **pcs_ceritificates** table (see pcsCertificatesDao.getCertificateById)
   a. If the root CA record exists and the CRL field is not empty:
      i. Returns the CRL in the response body.
      ii. Returns the 200 status code.
   b. Else:
      i. If caching fill mode is not 'LAZY', return the 404 error to client
      ii. If caching fill mode is 'LAZY':
         1. Calls Intel PCS v4 API to get QE identity and extracts the root CA from the certificate chain in response header. If failed, returns empty body.
         2. Parses the root CA to get cdp uri

3. Contacts the cdp uri to get root CA CRL. If failed, returns the 500 error.
4. Updates **pcs_certificates** table with root CA and CRL(see pcsCertificatesDao.upsertPcsCertificates)
5. Returns the root CA CRL in the response body (hex-encoded)
6. Returns the 200 status code.

3.8. Post Platforms IDs

This API stores platform identity information provided in the request. This API is restricted to users with the access to the user-token.

POST https://pccs-server-url:8081/sgx/certification/v3/platforms

POST https://pccs-server-url:8081/sgx/certification/v4/platforms

Request Parameters

Name	Type	Request Type	Required	Pattern	Description
user-token	String	Header	True	String	PCCS user token which provides access to this API.

Request Body (JSON)

```
{
        “qe_id”: “Base16-encoded QE-ID value”,
        “pce_id”: “Base16-encoded PCE-ID value”,
        “cpu_svn”: “Base16-encoded CPUSVN value”,
        “pce_svn”: “Base16-encoded PCESVN value”,
        “enc_ppid”: “Base16-encoded PPID encrypted with PPIDEK”,
        “platform_manifest”: “Base16-encoded platform manifest value”
}
```

Response

Status codes

Code	Model	Headers	Description
200			Successfully completed (entry updated)
201			Successfully completed (entry created)
400			Invalid request parameters
401			Authentication failed
500			Internal server error occurred
502			Unable to retrieve the collateral from the Intel SGX PCS

Process

1. Validates the user token(caclulate sha-512 hash of the token and compare the hash value with UserTokenHash in the configuration file). If validation fails, returns the 401 error.
2. Validates the request body with pre-defined JSON schema. If the validation fails, returns the 400 error to client.
3. Checks cache status for this platform.
    a. Gets the platform object from **platforms** table based on the provided {qeid, pceid} (see platformsDao.getPlatform) .
    b. If the platform_manifest in the request does not match the one in the cache (**Note:** Treat the absence of the platform_manifest in the request while there is a PLATFORM_MANIFEST in the cache as a match):
        i. Updates **platforms** table of the cache with the new manifest.
        ii. Sets cache status to FALSE.
    c. Else if platform_manifest matches:
        i. Queries PCK certificate from cache db with {qeid, cpusvn, pcesvn, pceid} to check whether PCK certificate for this platform is cached (see pckcertDao.getCert).
        ii. If found, sets cache status to TRUE.
        iii. Else: sets cache status to FALSE.
4. If caching fill mode is OFFLINE:
    a. If cache status is FALSE:
        i. Adds the platform registration data to **platforms_registered** table and returns SUCCESS (call platformsRegDao.registerPlatform with state=NEW).
5. Else:
    a. If cache status is FALSE:
        i. If caching fill mode is REQ, adds the platform registration data to platforms_registered table (calls platformsRegDao.registerPlatform with state=NEW)
        ii. Uses the same logic in chapter 3.1 Get PCK Certificate to get PCK certificate from Intel PCS.
        iii. If caching fill mode is REQ, deletes the platform registration data from **platforms_registered** table (calls platformsRegDao.registerPlatform with state=DELETED).
    b. Checks quote verification collateral (PCK CRL, QE identity, QvE identity, Root CA CRL). If not cached, retrieves them from Intel PCS and fills the cache
        pckcrlDao.getPckCrl
        qeidentityDao.getQEIdentity
        qveidentityDao.getQvEIdentity
        pcsCertificatesDao.getCertificateById(root_cert_id=1)

3.9. Get Platform IDs

Administrators use this API to retrieve the platform ID information for registered platforms or cached platforms. This API is restricted to users with the access to the admin-token.

GET https://pccs-server-url:8081/sgx/certification/v3/platforms

GET https://pccs-server-url:8081/sgx/certification/v4/platforms

Request

Name	Type	Request Type	Required	Pattern	Description
admin-token	String	Header	True	String	The administrator token required to perform the request.
fmspc	String	Query	False	[fmspc1,fmspc2, …]	FMSPC array.

Response

An array of data structures defined below encoded as JSON in case of success (200 HTTP status code). The array contains a maximum of XXX platform elements. When the Queue Status is 0, information for all registered platforms is retrieved.

Response Format:

```
[
  {
    "qe_id": "xxxx",
    "pce_id": "xxxx",
    "cpu_svn": "xxxx",
    "pce_svn": "xxxx",
    "enc_ppid": "xxxx",
    "platform_manifest": "xxxx"
  },
  {
  }

]
```

Empty body otherwise.

Status codes

Code	Model	Headers	Description
200			Successfully completed
400			Invalid request parameters
401			Authentication failed
500			Internal server error occurred

Process

1. Checks if request parameters include fmspc.
    a. If fmspc is not included (will return platforms in registration queue):
        i. Gets the registration platforms list from cache db (see platformsRegDao.findRegisteredPlatforms).
    b. If fmspc is provided ():
        i. The format should be [fmspc1, fmspc2, …]. If not, returns the 400 error.
        ii. Gets cached platforms, whose fmspc is in the list (see platformsDao.getCachedPlatformsByFmspc).
2. Returns the JSON list in response body and platforms count in response header.

3.10. Put Platform Collateral to Cache

Administrators use this API to push the platform collateral for collected platforms from the Intel PCS into the caching service. This API is restricted to users with the access to the admin-token.

PUT https://pccs-server-url:8081/sgx/certification/v3/platformcollateral

PUT https://pccs-server-url:8081/sgx/certification/v4/platformcollateral

Request

Name	Type	Request Type	Required	Pattern	Description
Content-Type		Header		"application/json"	MIME type of the request body.
admin-token	String	Header	True	String	The administrator token required to perform the request.
platform_count	Integer	Query	True		Number of platforms in the PCK cert array.

Request body

A JSON object includes all the collaterals for the registered platforms. The format is defined as below. The v3 API will only accept v3 schema and the v4 API will only accept v4 schema.

```
{
  "platforms": [
    {
      "qe_id": "xxxx",
      "pce_id": "xxxx",
      "cpu_svn": "xxxx",
      "pce_svn": "xxxx",
      "enc_ppid": "xxxx",
      "platform_manifest": "xxxx"
    },
    {}
  ],
  "collaterals": {
    "version": "4",
    "pck_certs": [
      {
        "qe_id": "string",
        "pce_id": "string",
        "enc_ppid": "string",
        "platform_manifest": "string",
        "certs": []
      },
      {}
    ],
    "tcbinfos": [
      {
        "fmspc": "string",
        "sgx_tcbnfo": {},
        "tdx_tcbnfo": {}
      },
      {}
    ],
    "pckcacrl": "string",
    "qeidentity": "string",
    "tdqeidentity": "string",
    "qveidentity": "string",
    "certificates": {
      "SGX-PCK-Certificate-Issuer-Chain": "string",
      "TCB-Info-Issuer-Chain": "string",
      "SGX-Enclave-Identity-Issuer-Chain": "string"
    },
    "rootcacrl": "string"
  }
}
```

Response

Status codes

Code	Model	Headers	Description
200			Successfully completed
400			Invalid request parameters
401			Authentication failed
500			Internal server error occurred

Process

1. Validates the admin token(caclulate sha-512 hash of the token and compare the hash value with AdminTokenHash in the configuration file). If the validation fails, returns the 401 error to client.
2. Validates the request body with pre-defined collateral schema. If the validation fails, returns the 400 error to client.
3. For each platform in the list:
    a. Deletes old PCK certificates for this platform (pckcertDao.deleteCerts).

   b. Inserts all PCK certificates for this platform to **pck_cert** table(pckcertDao.upsertPckCert).
   c. Merges the raw TCBs in the request and cached **platform_tcbs** table to get a full raw tcb list.
   d. Extracts fmspc and ca value from any leaf cert (use the first cert for convenience).
   e. Finds the TCBInfo for this fmspc from the TCBInfos in the request.
   f. For each raw TCB in the raw tcb list:
      i. Gets the best cert with PCKCertSelectionTool using {cpusvn, pcesvn, pceid, TCB info, PCK certs }.
      ii. Updates **platform_tcbs** tables.
   g. Updates **platforms** table.
4. For each TCB info in the list:
   a. Updates **fmspc_tcbs** table.
5. Updates **pck_crl**, **qe_identities**, **qve_identities**, **pck_certchain**" if present.
6. Updates **pcs_certificates** and root CA CRL if present.

3.11. Cache Data Refresh

3.11.1. Refresh through HTTP Request

This API is for maintenance only. Refresh expired {TCB info, PCK CRLs, QE Identity, QvE Identity, Root CA CRL} or {PCK certs} in cache DB. This API is restricted to users with the access to the admin-token.

This API can be used when configured for with REQ or LAZY caching mode. It is not supported for OFFLINE caching mode.

[Deprecated] GET https://pccs-server-url:8081/sgx/certification/v3/refresh

POST https://pccs-server-url:8081/sgx/certification/v3/refresh

POST https://pccs-server-url:8081/sgx/certification/v4/refresh

Request

Name	**Type**	**Request Type**	**Required**	**Pattern**	**Description**
type		Query	False	"certs"	Refresh type. If not provided, TCB info, PCK CRLS, QE Identity, TDQE identity (v4 only) and QVE Identity will be refreshed. If type = "certs" and no fmspc is specified, all cached PCK certs will be refreshed.

admin-token	String	Header	True	String	The administrator token required to perform the request.
fmspc	String	Query	False	FMSPc1, FMSPC2, …, FMSPcn	Used with “type=certs”. If fmspc is provided, refresh only certs for those FMSPCs.

Response

A message shows the operation succeeded or failed.

Status codes

Code	Model	Headers	Description
200			Successfully completed
401			Operation failed
500			Internal server error occurred
502			Unable to retrieve the collateral from the Intel SGX PCS
503			Server is currently unable to process the request

Process

Default (type is not specified)

For each record in **pck_crls** table, contacts Intel PCS service to get the latest PCK CRL and CRL certchain, then updates the **pck_crls** table if necessary.

For each record in **fmspc_tcbs** table, contacts Intel PCS service to get the latest TCB Info, then updates the **fmspc_tcbs** table if necessary.

For each record in **enclave_identities** table, contacts Intel PCS service to get the latest QE Identity, QvE Identity and TDQE identity (v4 only) then updates the **enclave_identities** table if necessary.

Refresh root CA CRL: Get the root CA from **pcs_certificates** table, and parse it to get the CRL cdp uri, then contacts Intel PCS service to get the CRL and update the **pcs_certificates** table.

Refresh cached CRLs: Get all cached CRLs from **crl_cache** table, for each of them, download the CRL with the cdp_url as target url, and update the **crl_cache** table.

Type is “certs”

1. Gets all records for all the FMSPCs in the **platform_tcbs** table

(see platformTcbsDao.getPlatformTcbs(fmspc)). If the FMSPCs are not provided, updates all platforms that are already cached.

2. Sorts the records by {qeid, pceid} so that the TCB mapping records of the same platform are put together.
3. For each platform:
    a. Gets all PCK certs for this platform from Intel PCS with {encrypted_ppid, pceid} or {PLATFORM_MANIFEST,pceid}.
    b. Parses the first cert (X.509) in the array to get FMSPC value.
    c. Contacts Intel PCS again to get TCB info with the above FMSPC value.
    Note: This does not update the TCB info in the cache. The TCB info is only used with the PCK Cert Selection library to update the **platform_tcbs** table.
    d. Deletes old certificates and inserts new certificates for the platform.
4. For each raw TCB in the list of step 2, runs the PCK Cert Selection Tool to get the best certificate and update the cache.

3.11.2. Scheduled Cache Data Refresh

The PCCS can also be configured to refresh the cache data regularly, for example, once a day or once a week, etc. The scheduled task does not refresh PCK certificates because the network traffic overhead is large.

This refresh method is only supported in the LAZY cache mode.

It supports Cron-style Scheduling:

```
*    *    *    *    *    *

┬    ┬    ┬    ┬    ┬    ┬
│    │    │    │    │    │
│    │    │    │    │    └ day of week (0 - 7) (0 or 7 is Sun)
│    │    │    │    └───────── month (1 - 12)
│    │    │    └────────────────── day of month (1 - 31)
│    │    └─────────────────────────── hour (0 - 23)
│    └──────────────────────────────────── minute (0 - 59)
└───────────────────────────────────────────── second (0 - 59, OPTIONAL)
```

3.12. Get CRL by endpoint

Retrieve the X.509 Certificate Revocation List by the CRL endpoint.

GET https://pccs-server-url:8081/sgx/certification/v3/crl

GET https://pccs-server-url:8081/sgx/certification/v4/crl

Request

Name	Type	Request Type	Required	Pattern	Description
uri	String	Query	True	Example: https://certificates.trustedservices.intel.com/IntelSGXRootCA.der	URL to DER-encoded CRL (Root CA CRL, Platform/Package CA PCK CRL)

Response

DER-encoded representation of specified CRL.

Status codes

Code	Model	Headers	Description
200	Crl		Successfully completed
400			Invalid request parameters
500			Internal server error occurred
502			Unable to retrieve the collateral from the Intel SGX PCS
503			Server is currently unable to process the request

Process

1. Checks request parameters and returns the 400 error if the input parameter is invalid
2. Queries CRL with the uri as the key (see crlCacheDao.getCrl).
    a. If record exists:
        i. Returns the crl buffer.
        ii. Responds with the 200 status code.
    b. Else:
        i. If caching fill mode is not 'LAZY', returns the 404 (No cache data) error to client.
        ii. If caching fill mode is 'LAZY'
            1. Download the CRL with the uri address as target. If failed, returns the 404 error.
            2. Updates "**crl_cache**" table:
                a. Calls crlCacheDao.upsertCrl(uri, crl), crl is response body.
            3. Returns CRL in the response body.
            4. Responds with the 200 status code.

3.13. Get default platform policy

Retrieve the default platform policy provided by the platform owner.

GET https://pccs-server-url:8081/sgx/certification/v3/appraisalpolicy

GET https://pccs-server-url:8081/sgx/certification/v4/appraisalpolicy

Request

Name	Type	Request Type	Required	Pattern	Description
fmspc	String	Query	True	^[0-9a-fA-F]{12}	Base16-encoded FMSPC value

Response

The default platform policy for this fmspc in jwt format.

Status codes

Code	Model	Headers	Description
200			Successfully completed
400			Invalid request parameters
404			The default platform policy can't be found.
500			Internal server error occurred

Process

1. Checks request parameters and returns the 400 error if the input parameter is invalid
2. Queries the policy whose default flag is true for the specified fmspc (see appraisalPolicyDao. getDefaultAppraisalPolicies).
    a. If record exists:
        i. Returns the policy buffer.
        ii. Responds with the 200 status code.
    b. Else:
        i. Returns the 404 error.

3.14. Put appraisal policy

Upload an appraisal policy to the PCCS.

PUT https://pccs-server-url:8081/sgx/certification/v3/appraisalpolicy

PUT https://pccs-server-url:8081/sgx/certification/v4/appraisalpolicy

Request

Name	Type	Request Type	Required	Pattern	Description

Request body

```
 {
        "policy": policy string,
        “is_default”: default flag,
        "fmspc": fmspc value
}
```

Response

Status codes

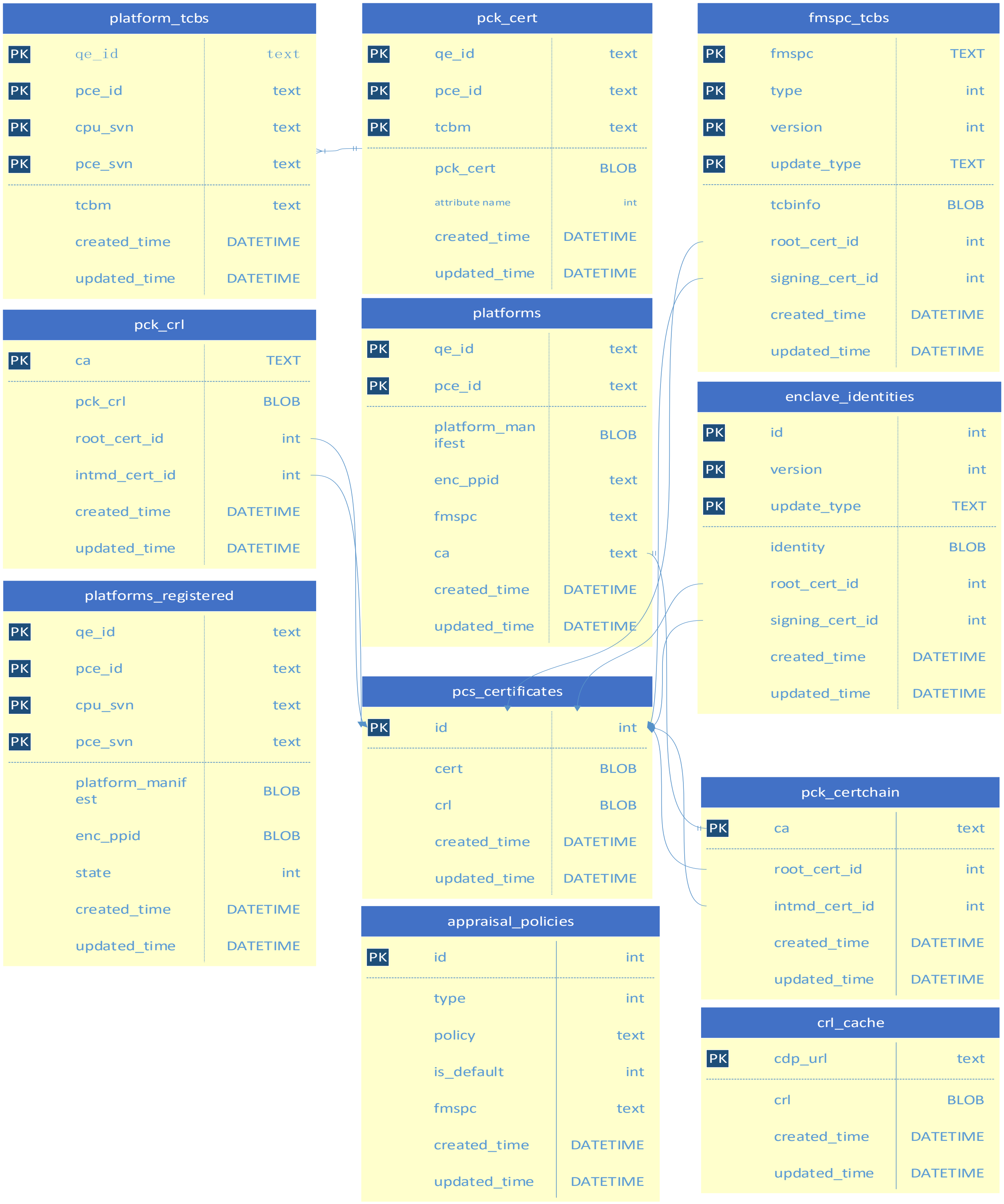
Code	Model	Headers	Description
200			Successfully completed
400			Invalid request parameters
401			Authentication failed
500			Internal server error occurred

Process

1. Validates the admin token(caclulate sha-512 hash of the token and compare the hash value with AdminTokenHash in the configuration file). If the validation fails, returns the 401 error to client.
2. Validates the request body with pre-defined policy schema. If the validation fails, returns the 400 error to client.
3. Calculate sha384 hash of the policy data, which will be used as the ID of the policy.
4. Parse the policy file to get policy type based on the class id. Currently only 3 types are supported: 0 – SGX; 1 – TDX 1.0; 2 – TDX 1.5;
5. Insert or update the **appraisal_policies** table (see appraisalPolicyDao).

4. Database

4.1.Schema Definition



platform_tcbs		
PK	qe_id	text
PK	pce_id	text
PK	cpu_svn	text
PK	pce_svn	text
	tcbm	text
	created_time	DATETIME
	updated_time	DATETIME

pck_crl		
PK	ca	TEXT
	pck_crl	BLOB
	root_cert_id	int
	intmd_cert_id	int
	created_time	DATETIME
	updated_time	DATETIME

platforms_registered		
PK	qe_id	text
PK	pce_id	text
PK	cpu_svn	text
PK	pce_svn	text
	platform_manifest	BLOB
	enc_ppid	BLOB
	state	int
	created_time	DATETIME
	updated_time	DATETIME

pck_cert		
PK	qe_id	text
PK	pce_id	text
PK	tcbm	text
	pck_cert	BLOB
	attribute name	int
	created_time	DATETIME
	updated_time	DATETIME

platforms		
PK	qe_id	text
PK	pce_id	text
	platform_manifest	BLOB
	enc_ppid	text
	fmspc	text
	ca	text
	created_time	DATETIME
	updated_time	DATETIME

pcs_certificates		
PK	id	int
	cert	BLOB
	crl	BLOB
	created_time	DATETIME
	updated_time	DATETIME

appraisal_policies		
PK	id	int
	type	int
	policy	text
	is_default	int
	fmspc	text
	created_time	DATETIME
	updated_time	DATETIME

fmspc_tcbs		
PK	fmspc	TEXT
PK	type	int
PK	version	int
PK	update_type	TEXT
	tcbinfo	BLOB
	root_cert_id	int
	signing_cert_id	int
	created_time	DATETIME
	updated_time	DATETIME

enclave_identities		
PK	id	int
PK	version	int
PK	update_type	TEXT
	identity	BLOB
	root_cert_id	int
	signing_cert_id	int
	created_time	DATETIME
	updated_time	DATETIME

pck_certchain		
PK	ca	text
	root_cert_id	int
	intmd_cert_id	int
	created_time	DATETIME
	updated_time	DATETIME

crl_cache		
PK	cdp_url	text
	crl	BLOB
	created_time	DATETIME
	updated_time	DATETIME

PCK_CERT: Cache table for all PCK certs of platforms. A **{qe_id, pce_id}** pair uniquely identifies a platform. There should be only one valid PCK cert for certain TCBm of the platform.

PLATFORM_TCBS: Stores the raw TCB to TCBm mapping.

PLATFORMS: Stores information of a specific platform identified by {qe_id, pce_id}.

PCK_CRL: Cache table for PCK CRL.

FMSPC_TCBS: Cache table for TCB Info.

QE_IDENTITIES: Cache table for QE Identity.

QVE_IDENTITIES: Cache table for QVE Identity.

PCS_CERTIFICATES:

ID(PK)	CERT	CRL	Description
1	BLOB	BLOB	Processor Root CA
2	BLOB	BLOB	Processor Intermediate CA
3	BLOB	BLOB	Processor Signing CA
4	BLOB	BLOB	Platform Intermediate CA

PLATFORMS_REGISTERED: Registration table for platforms.

PCK_CERTCHAIN: Cache table for PCK certificate chain IDs.

ca(PK)	root_cert_id	intmd_cert_id	Description
PROCESSOR	1	2	
PLATFORM	1	4	

4.2. Data Access Objects

4.2.1. platformsDao

- upsertPlatform(qe_id, pce_id, platform_manifest, enc_ppid, fmspc,ca)
  Inserts into or updates platforms table with {qe_id, pce_id, platform_manifest, enc_ppid, fmspc,ca}.

- getPlatform(qe_id, pce_id)
  Searches for a single platform instance by the {qe_id, pce_id} key pair.

- updatePlatform(qe_id, pce_id, platform_manifest, enc_ppid)
  Updates a single record of platforms table identified by the {qe_id, pce_id} key pair with new platform_manifest and enc_ppid values.

- getCachedPlatformsByFmspc(fmspc_arr)
  The input fmspc_arr is an array of fmspc values. This function queries all cached platforms based on the fmspc array.

```
select a.qe_id, a.pce_id, b.cpu_svn, b.pce_svn, a.enc_ppid, a.platform_manifest
      from platforms a, platform_tcbs b where a.qe_id=b.qe_id and a.pce_id = b.pce_id
             and a.fmspc in (:fmspc_arr)
```

4.2.2. pckcertDao

- getCert(qe_id, cpu_svn, pce_svn, pce_id)
  Queries PCK cert and PCK certchain based on the input keys.

```
select b.*,
(select cert from pcs_certificates e where e.id=d.root_cert_id) as root_cert,
(select cert from pcs_certificates e where e.id=d.intmd_cert_id) as intmd_cert
from platform_tcbs a, pck_cert b, platforms c left join pck_certchain d on c.ca=d.ca
where a.qe_id=$qe_id and a.pce_id=$pce_id and a.cpu_svn=$cpu_svn and a.pce_svn=$pce_svn
and a.qe_id=b.qe_id and a.pce_id=b.pce_id and a.tcbm=b.tcbm
and a.qe_id=c.qe_id and a.pce_id=c.pce_id
```

- getCerts(qe_id, pce_id)
  Queries all PCK certs associated with the input qe_id and pce_id.

- upsertPckCert(qe_id, pce_id, tcbm, cert)
  Inserts into or updates pck_cert table with the input paramerters.

- deleteCerts(qe_id, pce_id)
  Deletes all records associated with {qe_id, pce_id} from pck_cert table.

4.2.3. fmspcTcbDao

- upsertFmspcTcb(tcbinfoJson)
  Inserts into or updates fmspc_tcb table with the input tcbinfoJson:

Target Field	Value
type	tcbinfoJson.type
fmspc	tcbinfoJson.fmspc
version	tcbinfoJson.version
tcbinfo	tcbinfoJson.tcbinfo
root_cert_id	1
signing_cert_id	3

- getTcbInfo(type, fmspc, version)
  Queries TCB info and TCB Info issuer chain for given fmspc

```
select a.*,
        (select cert from pcs_certificates where id=a.root_cert_id) as root_cert,
        (select cert from pcs_certificates where id=a.signing_cert_id) as signing_cert
from fmspc_tcbs a where a.type=$type and a.fmspc=$fmspc and a.version=$version
```

- getAllTcbs()
  Queries all TCB info records from fmspc_tcb table.

4.2.4. pckCertchainDao

- upsertPckCertchain (ca)
  Inserts into or updates pck_certchain table with the below record:
  {id:1, root_cert_id:1, intmd_cert_id:2 for processor CA and 4 for platform CA}

- getPckCertchain (ca)
  Queries PCK certificate issuer chain for the input ca:

```
select a.*,
(select cert from pcs_certificates where id=a.root_cert_id) as root_cert,
(select cert from pcs_certificates where id=a.intmd_cert_id) as intmd_cert
from pck_certchain a
where a.ca=$ca
```

4.2.5. pckcrlDao

- getPckCrl(ca)
  Queries PCK CRL and PCK CRL certificate chain for given ca.

```
select a.*,
  (select cert from pcs_certificates where id=a.root_cert_id) as root_cert,
  (select cert from pcs_certificates where id=a.intmd_cert_id) as intmd_cert
from pck_crl a
where a.ca=$ca
```

- upsertPckCrl(ca, crl)
  Inserts into or updates pck_crl table with the input ca and crl:

Target Field	Value
ca	The input ca value
pck_crl	The input crl value
root_cert_id	1
intmd_cert_id	2 for processor CA and 4 for platform CA

4.2.6. pcsCertificatesDao

- upsertPcsCertificates(pcsCertJson)
  Inserts into or updates pcs_certificates table with the pcsCertJson object, which can be mapped to one record of the table.

- upsertPcsCertificates(id, cert)
  Inserts into or updates pcs_certificates table with the input id and cert values.

- upsertPckCertificateIssuerChain(ca, pck_certchain)
  Splits the input pck_certchain into SGX Intermediate CA Certificate and SGX Root CA Certificate, then updates/inserts them into pcs_certificates table. When ca is 'processor', use 2 for pcs_certificates.id, when ca is 'platform', use 4 for pcs_certificates.id.

- upsertPckCrlCertchain(ca, pck_crl_certchain)
  Splits the input pck_crl_certchain into SGX Intermediate CA Certificate and SGX Root CA Certificate, then updates/inserts them into pcs_certificates table. When ca is 'processor', use 2 for pcs_certificates.id, when ca is 'platform', use 4 for pcs_certificates.id.

- upsertTcbInfoIssuerChain (tcbinfo_certchain)
  Splits the input tcbinfo_certchain into SGX TCB Signing Certificate and SGX Root CA Certificate, then updates/inserts them into pcs_certificates table.

- upsertEnclaveIdentityIssuerChain (enclave_identity_certchain)
  Splits the input enclave_identity_certchain into SGX Enclave Signing Certificate and SGX Root CA Certificate, then updates/inserts them into pcs_certificates table.

- getCertificateById(ca_id)
  Finds a single instance identified by the ca_id from pcs_certificates table.

- upsertRootCACrl(rootcacrl)
  Updates the root certificate CRL(id=1) in pcs_certificates table.

4.2.7. platformsRegDao

- findRegisteredPlatform(regDataJson)
  Searches a single instance from platforms_registered table by {regDataJson}.

- findRegisteredPlatform()
  Finds all records whose status are NEW from platforms_registered table.

- registerPlatform(regDataJson, state)
  Adds a new record to platforms_registered table with the values in regDataJson and the state parameter.

- deleteRegisteredPlatforms()
  Updates all records with NEW state to DELETED state.

4.2.8. platformTcbsDao

- upsertPlatformTcbs(qe_id, pce_id, cpu_svn, pce_svn, tcbm)
  Inserts into or updates platform_tcbs table with the input parameters.

- getPlatformTcbs(fmspc)
  Gets all cached platform_tcbs for the input fmspc. It also needs to return the enc_ppid value. If fmspc is null, then all cached platform_tcbs are returned.

```
select a.*,b.enc_ppid as enc_ppid from platform_tcbs a, platforms b where a.qe_id=b.qe_id and
a.pce_id=b.pce_id and b.fmspc=$fmspc
```

- getPlatformTcbsById (qe_id, pce_id)
  Gets all records associated with the input qe_id and pce_id from platform_tcbs table.

4.2.9. enclaveIdentityDao

- upsertEnclaveIdentity(id, identity, version)
  Updates or inserts into enclave_identities table.

Target Field	Value
id	1
version	The input version paramert
identity	The input identity parameter
root_cert_id	1
signing_cert_id	3

- getEnclaveIdentity(id, version)
  Gets the enclave identity and enclave identity issuer chain from enclave_identities and pcs_certificates table.

```
select a.*,
(select cert from pcs_certificates where id=a.root_cert_id) as root_cert,
(select cert from pcs_certificates where id=a.signing_cert_id) as signing_cert
from enclave_identities a
where a.id=$id and a.version=$version
```

4.2.10. crlCacheDao

- upsertCrl (cdp_url, crl)
  Updates or inserts into crl_cache table.

Target Field	Value
cdp_url	cdp_url
crl	crl

- getCrl(cdp_url)
  Gets the CRL for the specified cdp_url.

- getAllCrls()
  Get all cached CRLs.

4.2.11. appraisalPolicyDao

- upsertAppraisalPolicy(apJson)
  Updates or inserts into appraisal_policies table.

Target Field	Value
id	Sha384(apJson.policy)
type	Type of the policy
policy	apJson.policy
Is_default	apJson.is_default
fmspc	apJson.fmspc

- getDefaultAppraisalPolicies (fmspc)
  Gets the default policy for the fmspc from appraisal_policies table.

```
AppraisalPolicy.findAll({
    where: {
      is_default: true,
      fmspc: fmspc
    },
    attributes: ['policy']
  })
```

5.Cache Fill Mode

When a new server platform is introduced to the data center or the cloud service provider that requires Intel® SGX remote attestation, the PCCS needs to import the platform's Intel® SGX attestation collateral retrieved from Intel PCS. This collateral is used for both generating and verifying ECDSA quotes during runtime flows. Currently, the PCCS supports three caching fill methods:

- **LAZY Mode**

  In this method of filling the cache, the PCCS gets a retrieval request (PCK Cert, TCB, etc.) at runtime, it looks for the collaterals in its database to see if they are already in the cache. If they do not exist, it contacts the Intel PCS to retrieve the collateral. This mode only works when internet connection is available.

- **REQ Mode**

  In this method of filling the cache, the PCCS creates a platform database entry when the PCCS receives the platform registration requests during platform deployment/provisioning. The PCK Cert ID retrieval tool sends platform registration information to the PCCS using the Put Platforms for Registration API. It does not return any data to the caller but contacts the Intel PCS to retrieve the platform's collaterals if they are not in the cache. It saves the retrieved collateral in cache database for use during runtime. This mode requires the PCCS to have an internet connection at deployment/provisioning time. During runtime the PCCS uses cache data only and does not contact Intel PCS.

- **OFFLINE Mode**

  In this method of filling the cache, the PCCS does not have access to the Intel PCS on the internet. Instead, the PCK Cert ID retrieval tool sends platform registration information to the PCCS's **platforms_registered** table using the Put Platforms for Registration API. The registration information can then be collected from the **platforms_registered** table using the Get Platforms API called by an administrator tool. The administrator tool runs on a platform that does have access to the internet. It can fetch platform collateral from Intel PCS and push it to the PCCS using the Put Platform Collateral to Cache API.

6.Configuration File

The PCCS can be configured using a configuration file, default.json, located in the 'config' sub-directory under the PCCS installation directory.

- **HTTPS_PORT**
  The port you want the PCCS to listen on. The default listening port is 8081.
- **hosts**
  The hosts that will be accepted for connections. Default is localhost only. To accept all connections, use 0.0.0.0
- **uri**
  The URL of Intel Provisioning Certificate Service. The current URL is https://api.trustedservices.intel.com/sgx/certification/v4/
- **ApiKey**
  The PCCS use the API key to request collateral from Intel Provisioning Certificate Service. You need to subscribe first to obtain an API key. For how to subscribe to the Intel Provisioning Certificate Service and receive an API key, go to Intel PCS API Portal and click 'Subscribe' (**Note:** You need an Intel® Developer Zone (IDZ) account to register).
- **Proxy**
  Specify the proxy server for internet connection, for example, "http://192.168.1.1:80". Leave it blank for no proxy or system proxy.
- **RefreshSchedule**
  cron-style refresh schedule for the PCCS to refresh cached artifacts including CRL/TCB Info/QE Identity/QVE Identity. The default setting is "0 0 1 * * *", which means refresh at 1:00 am every day.
- **UserTokenHash**
  Sha512 hash of the user token for the PCCS client user to register a platform. For example, PCK Cert ID retrieval tool uses the user token to send platform information to PCCS. Required by the Put platforms API.
- **AdminTokenHash**
  Sha512 hash of the administrator token for the PCCS administrator to perform a manual refresh of cached artifacts. It is required by these APIs: Get registered platforms, Put platform collateral to cache and Cache data refresh.

  Note: For Windows you need to set the UserTokenHash and the AdminTokenHash manually. You can calculate SHA512 hash with the help of OpenSSL:

  ```
  <nul: set /p password="mytoken" | openssl dgst -sha512
  ```
- **CachingFillMode**
  The method used to fill the cache DB. Can be one of the following: REQ/LAZY/OFFLINE. See section Caching Fill Mode
- **LogLevel**
  Log level. Use the same levels as npm: error, warn, info, http, verbose, debug, silly. Default is info. Log messages are written to <PCCS Install Directory>/logs/pccs server.log.
- **DB_CONFIG**

You can choose Sqlite or Mysql and many other DBMSes. For Sqlite, you do not need to change anything. For other DBMSes, you need to set database connection options correctly. Normally you need to change database, username, password, host, and dialect to connect to your DBMS.

Configuration file example:

```
{
    "HTTPS_PORT": 8081,
    "hosts": "127.0.0.1",
    "uri": "https://api.trustedservices.intel.com/sgx/certification/v3/",
    "ApiKey": "aaaaacaefe0a48a4bb36a49a48656788",
    “proxy”: “”,
    “RefreshSchedule”: “0 0 1 * * *”,
    “UserTokenHash”: “”,
    “AdminTokenHash”: “”,
    “CachingFillMode”:””,
    "LogLevel" : "info",
    “DB_CONFIG”:”sqlite”,
    “sqlite”: {
        “database”: “database”,
        “username”: “username”,
        “password”: “password”,
        “options”: {
            “host”: “localhost”,
            “dialect”: “sqlite”,
            “storage”: “pckcache.db”
        }
    }
}
```

7. Administration Tool (Admin Tool)

The admin tool is a Python* script that contains a set of commands to allow an administrator to manage the cached data for registered platforms. It is meant to be used when the cache filling mode is set to OFFLINE and the PCCS has no connection to the internet. The admin tool is used to collect information about new platforms registered with the PCCS using the Put platforms for registration REST API. The tool collects the new platform information then uses that information to retrieve platform attestation collateral/endorsements from the Intel PCS. The tool can also be used to refresh the collateral before they expire as well as refresh the data for a TCB Recovery.

The script commands are:

1. Get registration data from PCCS service

   pccsadmin.py get [-h] [-u URL] [-o OUTPUT_FILE] [-s SOURCE]

   Optional arguments:

   -h, --help

   Show this help message and exit.

   -u URL, --url URL

   The URL of the PCCS's GET platforms API; default: https://localhost:8081/sgx/certification/v3/platforms

   -o OUTPUT_FILE, --output_file OUTPUT_FILE

   The output file name for platform list; *default: platform_list.json*

   -s SOURCE, --source SOURCE

   reg - Get platforms from registration table.(default)

   reg_na - Get platforms whose PCK certs are currently not available from registration table.

   [FMSPC1,FMSPC2,...] - Get platforms from cache based on the FMSPC values.

   [] - Get all cached platforms.

2. Fetch platform collateral data from Intel PCS based on the registration data

   pccsadmin.py fetch [-h] [-u URL] [-i INPUT_FILE] [-o OUTPUT_FILE]

   Optional arguments:

   -h, --help

   Show this help message and exit.

   -u URL, --url URL

   The URL of the Intel PCS service; default: https://api.trustedservices.intel.com/sgx/certification/v3/

-i INPUT_FILE, --input_file INPUT_FILE

The input file name for platform list;
default: platform_list.json

-o OUTPUT_FILE, --output_file OUTPUT_FILE

The output file name for platform collaterals.
default: platform_collaterals.json

3. Put platform collateral data to PCCS cache database

   pccsadmin.py put [-h] [-u URL] [-i INPUT_FILE]

   Optional arguments:

   -h, --help

   Show this help message and exit

   -u URL, --url URL

   The URL of the PCCS service.
   default: https://localhost:8081/sgx/certification/v3/platformcollateral

   -i INPUT_FILE, --input_file INPUT_FILE

   The input file name for platform collaterals.
   default: platform_collaterals.json

4. Collect platform data that was retrieved by PCK ID retrieval tool into one json file. This file can be used as input of "fetch" command.

   pccsadmin.py collect [-h] [-d DIRECTORY] [-o OUTPUT_FILE]

   Optional arguments:

   -h, --help

   Show this help message and exit

   -d DIRECTORY, --directory DIRECTORY
   The directory which stores the platform data(*.csv) retrieved by PCK ID retrieval tool; default: ./

   -o OUTPUT_FILE, --output_file OUTPUT_FILE

   The output json file name; default: platform_list.json
5. Request PCCS to refresh certificates or collateral in cache database

   pccsadmin.py refresh [-h] [-u URL] [-f fmspc]

   Optional arguments:

   -**h, --help**

   Show this help message and exit.

   -u URL, --url URL
   The URL of the PCCS's refresh API; default:
   https://localhost:8081/sgx/certification/v3/refresh

-f FMSPCs, --fmspc FMSPCs

If this argument is not provided, then it will require PCCS to refresh quote verification collateral.

all - Refresh all cached certificates.

FMSPC1,FMSPC2,... - Refresh certificates of specified fmspc values.

The scripts can be found in the DCAP project on GitHub*. The latest usage information is in the README.txt file in the same directory.

8. Cache Management Flows

8.1. Platform Registration

The PCCS maintains a queue of platform information called the **platforms_registered** table. Datacenter and CSP owners add new Intel® SGX platforms to this queue using the Put Platforms IDs API. Platform registration is expected to happen before runtime workloads are executed by tenants. The queue contains all of the platform information provided in the Put Platforms IDs API. The PCCS adds the platform ID to the queue if any of the following are true:

1. An entry for that platform is not already in the PCCS's caching database.
    a. The platform is already in the database if the EncPPID and the PLATFORM_MANIFEST in the Put Platforms IDs request is the same as the platform ID already in PCCS's caching database.
2. The PLATFORM_MANIFEST in the registration request is different than the platform's PLATFORM_MANIFEST in the database.
3. Any of the collaterals (PCK Certs + verification collateral) for that platform entry is not already in the database.

- **Cache Mode is LAZY or REQ**

  When the PCCS gets the Put Platforms IDs request and adds a platform to the queue, it contacts the Intel PCS to retrieve the platform's PCK Certs. If the request contains the PLATFORM_MANIFEST, it uses the Intel PCS API to retrieve multiple PCK Certs using the {PLATFORM_MANIFEST,PCEID} tuple. If the PLATFORM_MANIFEST is not present, it uses the PCS API to retrieve multiple PCK Certs using the {EncPPID,PCEID} tuple. Once the PCCS retrieves the PCK Certs, the other verification collateral is retrieved:

  1. TCBInfo:
      a. Extract the FMSPC from the PCK Leaf Cert.
      b. If the TCBInfo for that FMSPC is not already in the database, the PCCS retrieves the TCBInfo from the Intel PCS.
  2. Intel PCK Cert CA CRL:
      a. Extract the FMSPC from the PCK Leaf Cert.
      b. If the TCBInfo for that FMSPC is not already in the database, the PCCS retrieves the TCBInfo from the Intel PCS.
  3. QEIdentity – If not already in the database.
  4. QvEIdentity – If not already in the database.
  5. Intel SGX Root CA CRLL:
      a. The URL for the Intel SGX Root CA CRL needs to be extracted from the CDP field of one of the intermediate certs or the Intel SGX Root Cert.

  If all the PCK Certs and their associated collaterals are successfully retrieved (or already in the database), the PCCS creates the platform entry in the database. Otherwise, the PCCS returns the Put Platforms IDs **Unable to retrieve the collateral from the Intel SGX PCS** in response to the request.

- **Cache Mode is OFFLINE**

  The platform IDs remain in the queue until the platform IDs are returned in response to the Get Platform IDs request. The Platform PCK Certs and verification collateral for those platforms need to be fetched from the Intel PCS in an offline manner. The Admin Tool can be used to fetch this data. Then that data needs to be imported back to the PCCS using the Put Platform Collateral to Cache request.

Note: When the caching server is configured for the LAZY mode, this registration step is not required. When in LAZY mode, the PCCS retrieves the respective platform collateral in response to the runtime request APIs:

a. Get PCK Certificate
b. Get PCK Cert CRL
c. Get TCB Info
d. Get Intel's QE Identity
e. Get Intel's QvE Identity
f. Get Root CA CRL

This requires that the PCCS have access to the internet during runtime workloads initiated by a tenant VM without requiring access to the PCCS admin token.

8.2. Handling TCB Recoveries

When a security vulnerability is found and it affects one of the components in the Intel® SGX TCB, an Intel® SGX TCB Recovery Event process starts. Part of this process can include generating new Intel® SGX PCK Certificates and the verification collateral (TCBInfo, QEIdentity and QVEIdentity structures) available. When a TCB Recovery Event is publicly announced, the PCCS needs to be updated with the new PCK Certs (for TCB Recovery events that affect those platforms) and all verification collaterals (for every TCB Recovery Event regardless of which platforms are affected).

- **Cache Mode is LAZY or REQ**

  In these modes, the PCCS has an internet connection. An administrator can request an update in two ways:

  1. List of affected FMSPC's **is not** known:

     a. Re-register all platforms in the CSP or database using the Put Platforms IDs request. This ensures all platforms that can generate a new PLATFORM_MANIFEST on TCB Recoveries get the PCK Certs associated with this TCB Recovery.

     b. Send the Refresh through HTTP Request with the optional 'type' field 'certs' option and without the optional 'fmspc' field. This will refresh all the PCK Certs in the cache for all registered platforms.

     c. Send the Refresh through HTTP Request without the optional 'type' filed option and without the optional 'fmspc' field. This will refresh all the verification collateral in the cache.

  2. List of affected FMSPC's **is** known:

     a. Send the Refresh through HTTP Request with the optional 'type' field 'certs' option and the optional 'fmspc' to the list of FMSPCs affected by the TCB Recovery. This will refresh all the PCK Certs in the cache for all registered platforms.

b. Send the Refresh through HTTP Request without the optional 'type' field option and without the optional 'fmspc' field. This will refresh all the verification collateral in the cache.

- **Cache Mode is OFFLINE**

  In this mode the PCCS does not have an internet connection. All collateral should be pushed to the PCCS through using the Administration Tool (Admin Tool). The admin tool must run on a machine that have an internet connection.

  1. Re-register all platforms in the CSP or database using the Put Platforms IDs request. This ensures all platforms that can generate a new PLATFORM_MANIFEST on TCB Recoveries get the PCK Certs associated with this TCB Recovery.
  2. Use the PCCS Admin Tool to retrieve the registered platform IDs in the cache.
     python pccsadmin.py get -s [FMSPC1,FMSPC2, …] -t YOUR_PCCS_ADMIN_TOKEN
     Use empty brackets **[]** to get the platform IDs for all FMSPCs.
  3. Use the PCCS Admin Tool to fetch collateral from the Intel PCS.
     python pccsadmin.py fetch -k YOUR_INTEL_PCS_API_KEY
  4. Use the PCCS Admin Tool to put the collateral into the cache.
     python pccsadmin.py put -t YOUR_PCCS_ADMIN_TOKEN

8.2.1. Special handling of Multi-Package TCB Recovery

There may be cases where a TCB Recovery requires a microcode patch. There is no guarantee that a given platform will be patched with the new microcode patch before Intel announces the TCB Recovery. In this case, the Intel PCS will not be able to generate all PCK Certificates required for this TCB Recovery. When this happens, the 'Get PCK Certs' API of the Intel PCS will return a "Not available" string instead of the PEM encoded PCK Certificate for the affected TCB's levels. Until the missing PCK Certificates exist in the PCCS database, the PCCS will return the PCK Certificate with the highest TCB for the requesting platform.

The PCCS and the PCCS Admin Tool will provide information to the PCCS administrator when this occurs. This allows the administrator to retry the TCB recovery refresh for the affected platforms. The method of indication depends on the cache mode.

- **Cache Mode is LAZY**

  The PCCS will automatically handle this case. The PCCS will continue to request PCK Certs from the PCS for each affected platform when requested by that platform until it receives a complete set of PCK Certificates.

  PCCS should provide the requester with the 'best' PCK Cert it can use by running the PCK Cert Selection Library with the available PCK Certs for the platform. It should only fail to provide a PCK cert to the requester if the PCK Cert Selection Library fails. –

- **Cache Mode is REQ**

  The PCCS will store the affected Platform IDs in the database's cached platform table. The administrator retrieves the PCK Certificates from the database using the administration tool's 'pccsadmin.py get' command with the 'source' parameter set to '-reg_na'. The administrator can then request the PCK Certs from the PCS and import them to the PCCS for these platforms using the administration tool's 'pccsadmin.py fetch' and 'pccsadmin.py fetch' command.

- **Cache Mode is Offline**

  The administration tool's 'python pccsadmin.py fetch' will output a warning that some platform's PCK Certificates were not available from the PCS. It will prompt the administration to store the platform IDs in a file. The administration can then retry the 'fetch' command for these files once the platforms have been patched.

8.3. Refreshing Expiring Collateral

Some of the cached collateral have short lifetimes whereas PCK Certs are long-lived. The datacenter or CSP needs to set up a policy for refreshing this collateral. It is possible to retrieve all the collaterals including PCK Certs (similar to the process described in Handling TCB Recoveries), but every platform has a unique set of PCK Certs, and refreshing PCK Certs can incur a lot of overhead. To refresh expiring collateral without refreshing PCK Certs, these flows can be implemented:

- **Cache Mode is LAZY**

  In this mode, the PCCS has an internet connection. An administrator can request only the verification collateral update in two ways:

  1. Set up a timed refresh using the method described in Scheduled Cache Data Refresh.
  2. Use the method described below for Cache Mode is REQ.

- **Cache Mode is REQ**

  In this mode, the PCCS has an internet connection. An administrator can request an update as follows:

  > Send the Refresh through HTTP Request without the optional 'type' filed option and without the optional 'fmspc' field. This refreshes all the verification collateral in the cache without refreshing the PCK Certs.

- **Caching Mode is OFFLINE**

  1. Create the platform_list.json file with a text editor and simply put a pair of brackets in it: [].
  2. Use PCCS Admin Tool to fetch collateral from the Intel PCS.
     ```
     python pccsadmin.py fetch -k YOUR_INTEL_PCS_API_KEY
     ```
  3. Use PCCS Admin Tool to put the collateral into the cache.
     ```
     python pccsadmin.py put -t YOUR_PCCS_ADMIN_TOKEN
     ```

8.4. Database migration

8.4.1. Migrating from v2 caching database (DCAP v1.8 and before) to v3 caching database (DCAP v1.9 and after)

Versions of PCCS from v1.8 and earlier do not support automatic database migration. Administrator of the PCCS service may want to import platform data from v2 PCCS caching database after he/she setup the v3 caching service. To ensure the certificate and collateral data is up to date, we suggest the administrator to get cached platforms list from V2 PCCS service first with the help of Admin Tool, then use “fetch” command to retrieve platform collateral from live Intel PCS service, and finally use “put” command to upload the data to V3 PCCS service.

1. Use the PCCS Admin Tool to retrieve the cached platform IDs in the cache.
   ```
   python pccsadmin.py get -s [] -u V2_PCCS_SERVICE_URL -t YOUR_PCCS_ADMIN_TOKEN
   ```
2. Use the PCCS Admin Tool to fetch collateral from the **V3** Intel PCS.
   ```
   python pccsadmin.py fetch -k YOUR_INTEL_PCS_API_KEY
   ```
3. Use the PCCS Admin Tool to put the collateral into the cache.
   ```
   python pccsadmin.py put -u V3_PCCS_SERVICE_URL -t YOUR_PCCS_ADMIN_TOKEN
   ```

8.4.2. Migrating database from DCAP v1.9 to newer versions of DCAP

The PCCS installation supports automatic database migration starting from DCAP v1.9. Database backup is recommended before performing the upgrade and the installer will provide a warning before automatic migration occurs.