Intel® Trust Domain Extensions Data Center Attestation Primitives (Intel® TDX DCAP): Quote Generation Library and Quote Verification Library

Rev 0.9
2025-May

intel

Intel® Trust Domain Extensions Data Center Attestation Primitives (Intel® TDX DCAP): Quote Generation Library and Quote Verification Library

Table of Contents

1 Introduction

Attestation is a process used by software to demonstrate to a remote party that the software has been properly instantiated on a platform. The Intel® Trust Domain Extensions (TDX) attestation allows a remote party to ensure that a Virtual Machine (VM) is hardware-isolated and protected using Intel® TDX, i.e., it is a so-called Trust Domain (TD)

The attestation process starts by securely generating an Attestation Key inside an enclave. As such, the Attestation Key is created and owned by the platform. Then, the Attestation Key gets certified by an Intel® SGX–rooted key whose certificate is distributed by Intel. After this setup, a TD can generate attestation evidence, the so-called TD Quote. Any off-platform entity having retrieved the certificate from Intel can verify the attestation evidence, i.e., the TD Quote. After a successful verification, the off-platform entity can be sure that a TD is running with Intel® TDX protections on a trusted Intel® TDX–enabled platform, that the platform is patched to a certain level, and that the identity of a TD is accurate.

This specification describes the API surface for Intel-provided libraries that can be used to generate and verify attestation evidence for TDs. The released versions of the libraries use ECDSA-based Attestation Keys.

1.1 Terminology

Trust Domain Extensions (TDX)	An Intel CPU mode and ISA extension that supports operation and management of Trust Domains (TDs).
Trust Domain (TD)	Trust Domains (TDs) are hardware-isolated virtual machines (VMs) protected by Intel® Trust Domain Extensions (Intel® TDX).
TD Quote	Data structure used to provide proof to an off-platform entity that a TD is running with Intel® TDX protections on a trusted Intel® TDX–enabled platform, that the platform is patched to a certain level, and that the identity of a TD is accurate.
TD Report	Hardware report generated by the Intel® TDX HW that provides identity and measurement information of the TD and the platform. It can be MAC'd with a key available to an SGX enclave on the same platform.
Software Guard Extensions (SGX)	An Intel CPU mode and ISA extensions that support operation and management of SGX enclaves.
TD Quoting Enclave (TDQE)	By default, an Intel® SGX enclave signed by Intel, which generates TD Quotes for which the trust is rooted in Intel. Optionally, the TDQE can be provided and signed by any authority trusted by the attestation infrastructure owner. In this case, the trust is not rooted in Intel.
Attestation Key (AK)	Asymmetric key generated by the TD Quoting Enclave (TDQE). The TDQE uses the private key–part to sign TD Quotes. The TDQE also generates an SGX Report

	containing the hash of the public key–part. This SGX Report and the public key–part is provided to the Provisioning Certification Enclave (PCE), which verifies the SGX Report, the hash of the AK public key-part, and then signs the SGX Report with the private key-part of the PCK.
TDQE Report	SGX Report generated by the TD Quoting Enclave (TDQE). It contains a hash of the private key–part of the Attestation Key (AK) and it is provided to the Provisioning Certification Enclave (PCE).
Quote Verification Enclave (QvE)	By default, an Intel® SGX enclave signed by Intel, which verifies TD Quotes for which the trust of the verification results is rooted in Intel. Optionally, the QvE can be provided and signed by any authority trusted by the attestation infrastructure owner. In this case, the trust is not rooted in Intel.
Provisioning Certification Enclave (PCE)	Intel® SGX enclave signed by Intel that uses a Provisioning Certification Key (PCK) to sign SGX Report structures for Provisioning or Quoting Enclaves. The signed SGX Reports provide the attestation root of trust for TDs and Enclaves.
Elliptic Curve Digital Signature Algorithm (ECDSA)	Algorithm to generate digital signatures as described in FIPS 186-4.
Quote Generation Service (QGS)	Software service or daemon running in an Intel® SGX–capable operating system containing the PCE and TDQE used to produce TD Quotes.
Intel® TDX Quote Generation Library (QGL)	Library used by TD tenant SW to generate TD Quotes and TD Reports. It abstracts the communication channel between TD and QGS and abstracts the Attestation Key (AK) used to generate the TD Quote.
Ring3 Attestation Abstraction Library (R3AAL)	Current name of the library binary containing the Intel® TDX Quote Generation Library.
Intel® TDX Quote Verification Library (QVL)	Library used by relying parties to verify TD Quotes.
Provisioning Certification Key (PCK)	An asymmetric key unique to the processor package (or platform), the HW TCB, the signer of the PCE, and the Security Version Number (SVN) of the PCE. The PCK can be requested using SGX's EGETKEY instruction.
Provisioning Certification Key Certificate (PCK Cert)	X.509 certificate chain signed and distributed by Intel for each Intel® SGX–enabled platform. The leaf certificate contains the public key–part of the Provisioning Certification Key (PCK) generated by the PCE.

	A PCK Cert is used by the Intel® TDX Quote Verification Library to verify that Quotes were generated by a valid TDQE on a trusted Intel® SGX platform at a particular TCB level.
Security Version Number (SVN)	Version number that gets increased whenever security relevant updates to any component in the TDX Trusted Computing Base (TCB) occurred. New versions of the component can have increased functional versions without incrementing the SVN.
TDX Module	Module that enforces security properties for hosting TDs on an Intel® TDX platform
Intel® SGX/TDX DCAP	Intel® Software Guard Extensions/Trust Domain Extensions Data Center Attestation Primitives. These are the attestation primitives (software and services) to support remote attestation for Intel® Xeon processors supporting SGX and TDX.

Table 1-1: Terminology

2 Overview

Before a Trust Domain (TD) can be trusted by an off-platform entity, the TD must prove that it's running with Intel® TDX protections on an Intel® TDX–enabled platform. Once trusted, an off-platform entity or a relying party can provide secrets or access to trusted services. Establishing this trust can be separated into a quote setup phase, a quote generation phase, and a quote verification phase. We summarized all phases in the following subsections.

Note that this document is mainly about two libraries that can be used by Linux* OS–based software to generate and verify TD Quotes: the Intel® TDX Quote Generation Library and the Intel® TDX Quote Verification Library. Sections 3 and 4 introduce the APIs of these libraries. The main purpose of the following subsections is to explain in which part of the TD Quote generation and verification flow these libraries are used.

2.1 Quote Setup Phase

Involved Components

For Intel® TDX, the quote collateral setup phase involves an Intel® SGX–capable component, the platform's PCK Certificate, and the TDX-enabled hardware. The VMM just passes through the exchanged messages. Figure 1 shows the entire attestation flow and the components involved in quote setup phase are highlighted.

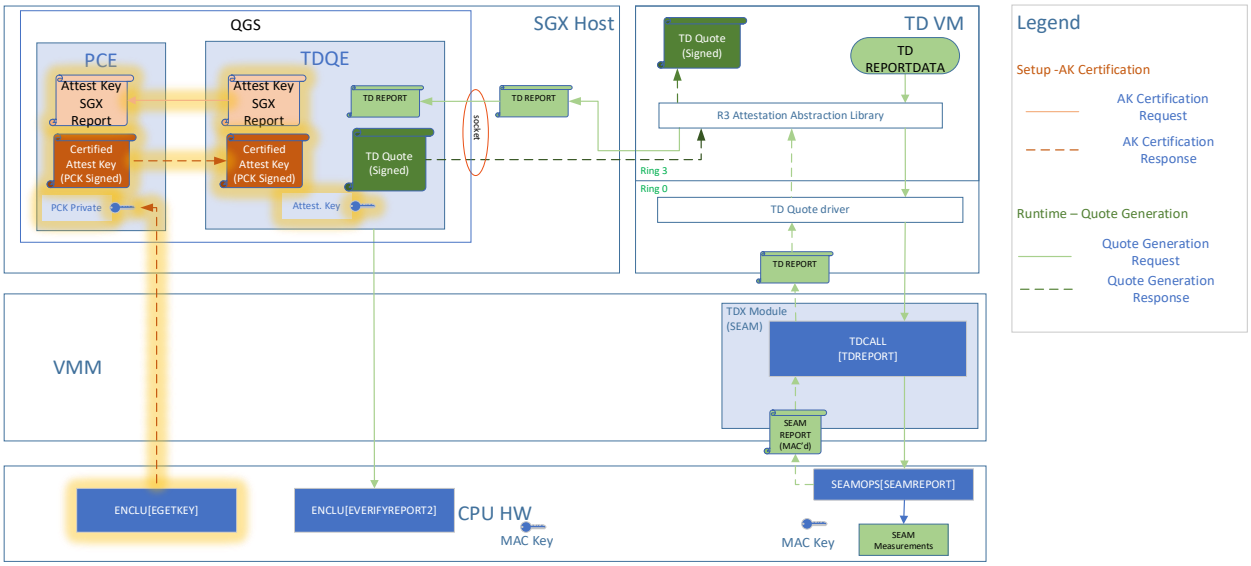


Figure 1: Attestation Overview – Quote Setup

The SGX–capable component can be an SGX-capable host (as depicted in Figure 1) or an SGX-capable VM. Inside this component, the Quote Generation Service (QGS) is executed. It consists of at least two SGX enclaves: the Provision Certification Enclave (PCE) and the TD Quoting Enclave (TDQE). Both enclaves are developed and signed by Intel.

The PCK Certificate for the platform is published by the Intel® SGX/TDX Provisioning Certification Service (PCS). It is a service provided by Intel that provides TD Quote generation and TD Quote verification collateral needed for the attestation process. The platform owner will cache the PCK Certificates for each

of its platforms. For more information on the Intel® SGX/TDX PCS, see https://api.portal.trustedservices.intel.com/provisioning-certification.

Quote Setup Flow

As a first step, the TDQE uses a FIPS 186-4 and RFC 6090 compliant algorithm to generate the Attestation Key (AK)—a 256-bit, asymmetric ECC key on the p256 curve. Then, the TDQE generates an SGX Report, which we will denote by TDQE Report in the following. The TDQE Report contains the identity of the TDQE, the identity of the TDQE's signer, and a hash of the public key–part of the AK. To guarantee the integrity of the TDQE Report, the TDQE uses the SGX HW to generate a Message Authentication Code (MAC) over the TDQE Report. The key needed to verify the TDQE Report's MAC is only accessible by the SGX enclave to which this SGX Report should be transferred. In this case, the targeted enclave is the PCE on the same platform. The inter-enclave attestation on one platform is referred to as SGX local attestation. Accordingly, the TDQE sends the TDQE Report to the PCE along with the public key–part of the AK.

Once the PCE successfully verified the MAC of the incoming SGX Report (i.e., the TDQE Report), the PCE can be sure that the TDQE is running on the same platform. Then, the PCE uses the SGX-provided EGETKEY instruction to derive the private key–part of the Provisioning Certification Key (PCK) using the TCB level described in the PCK Certificate. The PCE is the only SGX enclave that can derive the PCK, which is an asymmetric key unique to the processor package (or platform), the TCB of the PCK Certificate, the signer of the PCE, and the Security Version Number (SVN) of the PCE. The PCE verifies that the hash of the received public key (belonging to the AK) matches the hash contained in the TDQE Report. Finally, the PCE uses the PCK to sign the TDQE Report and sends back the result. Note that the signed TDQE Report contains the hash of the public key–part of the AK, because it was added to the TDQE Report by the TDQE. As a result, the PCK cryptographically certifies the public key–part of the AK.

The Provisioning Certification Service (PCS) can be used to retrieve PCK Certs. Each PCK Cert contains the public key corresponding to one specific PCK. In turn, each PCK Cert is unique to the processor package (or platform), the HW TCB, the signer of the PCE, and the Security Version Number (SVN) of the PCE. In more detail, the PCK Cert is an X.509 certificate chain rooted in an Intel® CA. To generate TD Quotes in the following phase, the QGS needs a PCK Cert with a TCB level equal-to or lower-than the actual TCB level of the platform. The TD Quotes will be signed with the PCK at a level proving that the platform's actual TCB level is at least as high as the TCB level described in the corresponding PCK Certificate.

In summary, the setup phase ends in the following state:

- The TDQE has generated an Attestation Key (AK) for which the private key–part is only known to the TDQE.
- The PCE has signed the public key–part of the AK using the PCK.
- An Intel-provided PCK Cert containing the public key corresponding to the PCK signature is available to the QGS.
- The PCK Cert's trust is rooted in an Intel® CA.

2.2 Quote Generation Phase

Involved Components

The quote generation phase involves a TD, an Intel® SGX–capable component on the same platform, the TDX Module, and TDX-enabled hardware. Figure 2 shows the entire attestation flow and the components involved in quote generation phase are highlighted.

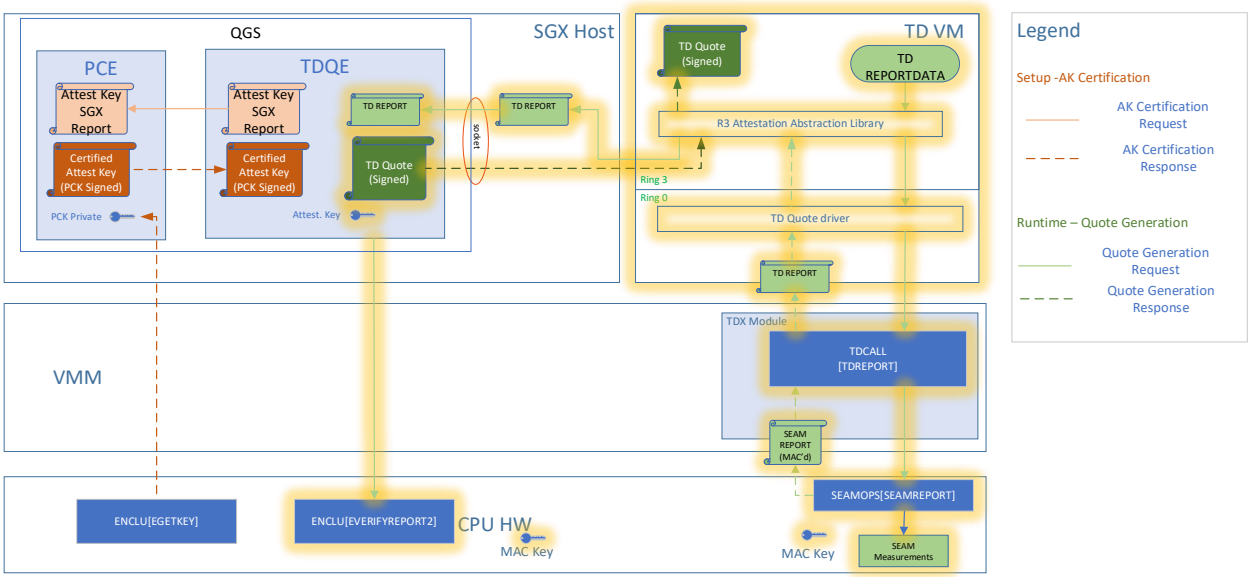


Figure 2: Attestation Overview – Quote Generation

Inside the TD, the Intel® TDX Quote Generation Library (QGL) provides an API to TD tenant workloads. This API abstracts the communication channel created by the platform owner to generate TD Quotes. The following subsection presents the currently supported communication channels. The QGL also abstracts the type of Quote generated by the platform. Note that the current name of the library binary containing the Intel® TDX Quote Generation Library is Ring3 Attestation Abstraction Library (R3AAL).

To support the generation of TD Quotes and TD Reports, the guest kernel/operating system provides a Ring0 interface to the TDX Module. This is depicted as the TD Quote Driver in Figure 2.

As already stated in Section 2.1, the SGX–capable component can be an SGX-capable host (as depicted in Figure 2) or an SGX-capable VM. This component contains a Quote Generation Service (QGS), which consists of at least two SGX enclaves: the Provision Certification Enclave (PCE) and the Trust Domain Quoting Enclave (TDQE). Both enclaves are developed and signed by Intel.

Note that SGX enclaves cannot be started inside the TD, because TD's do not support SGX. The platform owner is responsible for providing the QGS and the communication channel between the TD and the QGS. The construction and deployment of the QGS may differ for different platform owners and is only described in this document to understand the overall TD Quote generation flow.

The TDX Module is a software module running in SEAM mode inside the VMM. The SEAM mode isolates the TDX Module from accesses outside of the TD. For more information on the TDX Module, see TDX Module documentation.

Section 3 of this document focuses on the Quote generation elements running in the TD ('TD' block in Figure 2). The Ring3 Attestation Abstraction Library (R3AAL) provides an API to TD tenant workloads. This API abstracts the communication channel created by the platform owner to generate TD Quotes. It also abstracts the type of Quote generated by the platform.

Quote Generation Flow

Whenever a TD tenant workload wants to generate a TD Quote, it uses the API provided by the QGL. The API request must contain report data that will be part of the TD Report. The QGL contacts the TD Quote driver, which will use the TDCALL denoted as "TDREPORT" provided by the TDX Module. Subsequently,

the TDX Module collects the TD measurements, uses the report data, and uses a CPU-provided SEAM instruction denoted as "SEAMREPORT" to finalize the TD Report with the measurements of the TDX Module itself. The TD Report is MAC'd by a HW key available to the TDX HW. The HW key necessary to verify the MAC is only accessible to a valid SGX enclave (such as the TDQE) on the same platform. Finally, the TDX Module returns the TD Report to the QGL.

As a next step, the QGL passes the TD Report to the TDQE, which verifies the TD Report using the SGX instruction EVERIFYREPORT2. This instruction verifies the MAC and contents of the TD Report. Then, the TDQE signs the TD Report with the private key–part of the AK. The resulting data structure is denoted as TD Quote. Note that this TD Quote contains at least the public key–part of the AK, the PCK Cert, the measurement of the TD, the measurements of the TDX Module, the measurements of the TDQE, and the signature of the PCK. See Appendix Version 4 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID) for details of the TD Quote contents.

2.2.1 Communication Channels

The Intel® TDX Quote Generation Library (QGL) abstracts the communication channel used by the TD to request a TD Quote. Currently, the Intel-provided library binary (i.e., the R3AAL) supports three communication channels:

- Virtio-vsock† – Needs the TD to be configured to use vsock when the TD is launched.
- Configfs-tsm – Needs guest kernel v6.7 and later with CONFIG_TSM_REPORTS enabled.
- TD Call – Always available and implemented via the TD Quote Driver (as of the writing of this document, Linux upstreaming for the TD Quote Driver to support TD Calls is in progress)

The R3AAL uses the following logic to decide which communication channel it uses:

- If there is a valid port number defined inside the TD in the file /etc/tdx-attest.conf, the R3AAL will attempt to use virio-vsock-virtio channel.
    - The configuration file is not generated automatically. Thus, the user needs to create the file if virtio-vsock is the preferred communication channel.
- If configfs-tsm is supported, the R3AAL will attempt to use configfs-tsm channel.
    - Optional. Export env `DCAP_TDX_QUOTE_CONFIGFS_PATH` and the value is an exist and valid path. It's expected to be /sys/kernel/config/tsm/report/$name, where $name is chosen by user. User needs to set env `DCAP_TDX_QUOTE_CONFIGFS_PATH` and create the directory before using R3AAL.
- If the R3AAL fails to connect via virtio-vsock or configfs-tsm, the R3AAL will use the TD Call communication channel.

2.3 Quote Verification Phase

Involved Components

The quote verification phase can be done by any party, which we denote as quote verifier in the following section. For example, the quote verifier might be an external Relying Party or a service at a Cloud Service Provider (CSP).

The quote verification can be done in two variants: (1) in an unprotected manner or (2) inside an Intel SGX enclave. In variant 1, the quote verification involves a non-SGX application running at the quote verifier, a 'Usage Quote Verification Library' that links the Intel® TDX Quote Verification Library (QVL) and potentially contains additional validation checks (see Sections Quote Verification Trust Policies and Extended TD Checks), the Platform Quote Provider Library (see Section 5.1), and a Collateral Management Server. In variant 2, the application contains an SGX enclave and the QVL uses the Quote Verification Enclave (QvE). The main benefit of variant 2 is that the quote verifier can get a cryptographically verified verification result using SGX local attestation.

Quote Verification Flow

The detailed quote verification flow of variant 1 is depicted in Appendix Non-QvE Based Quote Verification. The quote verifier runs an application (without SGX capability), which the quote verifier uses to verify TD Quotes. This application passes TD Quotes to a 'Usage Quote Verification Library', which links the Intel® TDX Quote Verification Library (QVL). To verify a TD Quote, the QVL needs verification collateral, which at least includes the root Intel CA certificate of the Intel® CA that signed the PCK Cert and the reference values, the Certificate Revocation Lists (CRLs), and reference values for components in the platform TCB. For more details, see the definition of `tdx_ql_qv_collateral_t` in Appendix Quote Library Data Structures. This verification collateral can either be passed to the QVL or the QVL can fetch the verification collateral automatically using the Platform Quote Provider Library (see Section 5.1) and a Collateral Management Server. With this verification collateral, the QVL performs at least the following checks on the TD Quote:

- Check the PCK Cert (signature chain).
- Check if the PCK Cert is on the CRL.
- Check the verification collaterals' cert signature chain, including PCK Cert Chain, TCB info chain and QE identity chain
- Check if verification collaterals are on the CRL.
- Check the TDQE Report signature and the contained AK hash using the PCK Cert.
- Check the measurements of the TDQE contained in the TDQE Report.
- Check the signature of the TD Quote using the public key–part of the AK. Implicitly, this validated the TD and TDX Module measurements.
- Evaluate the TDX TCB information contained in the TD Quote.

Optionally, the quote verifier can instead use variant 2 to receive cryptographically verified verification results. Appendix QvE-based Quote Verification depicts a detailed sequence diagram of this variant. In this case, the quote verifier has to do the verification in an Intel® SGX–enabled environment. Compared to variant 1, the quote verifier's application contains an SGX-enclave to verify the QVL results. This application passes information about its SGX enclave with the TD Quote to a 'Usage Quote Verification Library', which still links the same Intel® TDX Quote Verification Library (QVL). For the actual TD Quote verification, the QVL uses an SGX enclave signed and developed by Intel – the Quote Verification Enclave (QvE). The QvE generates the verification result as an SGX Report, called the QvE Report, that the quote verifier application's SGX-enclave can verify. The MAC used to protect the integrity of the QvE Report is only accessible by the SGX enclave executed by the quote verifier. In other words, the verification result is protected using SGX's local attestation.

Whether variant 1 or variant 2 is used, the quote verifier can be certain that the TD described in the TD Quote is running on an Intel TDX–enabled platform at the evaluated TCB level.

Note: The quote verifier is responsible for verifying the measurements and identity of the TD are the expected values (see Section Extended TD Checks).

2.3.1 Quote Verification Trust Policies

The Intel® TDX Quote Verification Library (QVL) provides the ability to perform a simple/strict TD Quote verification trust policy as it was described in the last Section 2.3 The function `tee_verify_quote` (see Section Set Enclave Load Policy
When the TD quote needs to be verified inside QvE instead of TD VM, you need to know the proper enclave loading policy. The library may be linked with a long-lived process, such as a service, where it can load the enclaves and leave them loaded (persistent). This better ensures that the enclaves will be available upon quote requests and not subject to EPC limitations if loaded on demand. However, if the Quoting library is linked with an application process, there may be many applications with the Quote Verification Library and a better utilization of EPC is to load and unloaded the quoting verification enclave on-demand (ephemeral). The library will be shipped with a default policy of loading the enclave and unloading it on-demand. (SGX_QL_EPHEMERAL).

If the policy is set to default SGX_QL_EPHEMERAL, then the QvE will be loaded and unloaded on-demand. If the enclave is already loaded when the policy is change to SGX_QL_EPHEMERAL, the enclave will be unloaded before returning.

To enhance quote verification performance, we introduced new polices **SGX_QL_EPHEMERAL_QVE_MULTI_THREAD** and **SGX_QL_PERSISTENT_QVE_MULTI_THREAD** from DCAP 1.18 release. Each thread will have its own QvE instance, so please make sure you have enough EPC to load QvE in this mode. We recommend using the 2 new policies to get better performance if your system has enough EPC.

Note that the QvE loading policy can only be changed once in one process.

Syntax

```
quote3_error_t sgx_qv_set_enclave_load_policy(sgx_ql_request_policy_t policy);
```

Parameters

policy[In]

SGX_QL_EPHEMERAL - Default policy. QvE is initialized and terminated on every quote verification function call.

SGX_QL_PERSISTENT - All the threads will share single QvE instance, and QvE is initialized on first use and reused until process ends.

SGX_QL_EPHEMERAL_QVE_MULTI_THREAD - QvE is loaded per thread and be unloaded before function exit.

SGX_QL_PERSISTENT_QVE_MULTI_THREAD - QvE is loaded per thread and only be unloaded before thread exit.

Return Values

SGX_QL_SUCCESS:
Successfully set the enclave loading policy for the quoting library's enclaves.

SGX_QL_UNSUPPORTED_LOADING_POLICY:
The selected policy is not supported, or it has been set once.

Verify TD Quote) performs this simple/strict TD Quote verification, which includes the validation of the TD Quote, the TCB status, and the expiration status (assuming a trusted time passed to the function).

Additionally, the QVL allows the quote verifier to apply more complex, custom trust policies using supplemental data that optionally can be returned by `tee_verify_quote`. These custom trust policies enhance the verification checks done by `tee_verify_quote` – depending on the need of the quote verifier. For the content of the optional supplemental data, see the data structure `sgx_ql_qv_supplemental_t` in Appendix Quote Library Data Structures.

The quote verifier can, for example, use a custom trust policy in the following cases:

- When the quote verifier is willing to accept an outdated platform up to a certain point in time. This might be necessary when the platform owner that generates the TD Quote hasn't patched all platforms before the TCB Recovery Event date, i.e., before the date when Intel publishes a new TCB level requiring patches. In this case, `tee_verify_quote` will return an out-of-date status. The quote verifier can still accept the verification if the TCB date of the evaluated TCB level with an out-of-date result provided in the supplemental data isn't older than some allowed grace period.
- When the quote verifier is willing to accept an out-of-date TCB level from the platform if the platform contains certain patches. This might be necessary when the owner of the platform that generates the TD Quote hasn't patched all platforms before the TCB Recovery Event date. In this case, `tee_verify_quote` will return an out-of-date status. The quote verifier can compare the TCB date contained in the supplemental data to the public notification date of a particular Intel Security Advisory (SA). If the TCB date is greater-than-or-equal-to the SA's public notification date, then that platform has been patched for that SA and all SA's published before it.
- When the quote verifier does not have access to a trusted time source that is needed for `tee_verify_quote`. In this case, the quote verifier can use the supplemental data to compare 'tcbEvalDataSetNumber' or 'crlNum' contained in the verification collateral against some reference value. As a result, the caller can be sure that the expected collateral is used.

As before, the quote verifier can execute the custom trust policy check either (1) in an unprotected manner or (2) inside an Intel SGX enclave.

In variant 1, the verification flow is as follows:

1. The quote verifier calls the function `tee_get_supplemental_data_version_and_size` (see Section Get Quote Verification Supplemental Version and Size) to get the version and size of the supplemental data.
2. The quote verifier allocates a buffer for the supplemental data and calls `tee_verify_quote` passing a pointer to the allocated buffer. The function will perform simple/strict TD Quote verification and fill the supplemental data buffer.

3. The quote verifier checks the result of `tee_verify_quote` and passes the supplemental data to a function implemented by the quote verifier for custom checks. We denote this function as `tee_verify_quote_supplemental` and provide an example in Section Verify Quote with Supplemental Data

In variant 2, the entire flow is executed inside an SGX enclave. In this case, `tee_verify_quote` uses the QvE, which generates a QvE Report dedicated for this SGX enclave. The secure transmission of the QvE report is guaranteed by SGX's local attestation. Note that the supplemental data returned by `tee_verify_quote` is cryptographically bound to the result of the quote verification. The remaining steps do not change.

For multi-package platforms (i.e., Xeon Scalable platforms), the supplemental data will contain `sgx_type=1`. The security properties with an `sgx_type` of 1 may be different from the standard `sgx_type` of 0. The following supplemental data values are valid only when `sgx_type=1`. Otherwise, the values are undefined. For more information on attestation for multi-package platforms, see Remote Attestation for Multi-Package Platforms using Intel(R) SGX Datacenter Attestation Primitives (DCAP).

- `platform_instance_id`
  - Value of Platform Instance ID for this multi-package platform.
- `dynamic_platform`
  - Indicates whether a platform can be configured to enable the add package flow.
- `cached_keys`
  - Indicates whether the encrypted platform keys are stored by Intel's Registration Service. This is determined by using direct or indirect registration. Only in the first case, the encrypted root keys are stored.
- `smt_enabled`
  - Indicate whether the platform has SMT (simultaneous multithreading) enabled.

2.3.2 Extended TD Checks

Intel® TDX Quote Generation Library fills the TD Quote with several measurements, attributes, and other information. All this data becomes part of the TD Quote Body (see Appendix TD Quote Body).

The Intel® TDX Quote Verification Library (QVL) verifies that the TD Quote was generated by an TD running with TDX protections and that the TCB level is up to date. However, the QVL does not verify the data contained in the TD Quote Body. The quote verifier is responsible to define what data needs to be checked to establish trust in a TD.

Data contained in the TD Quote Body that potentially needs to be checked:

1. Identity of TD, i.e., the contained measurement:
   a. Verify the values assigned by the creator of the TD are as expected: MROWNER and MROWNERCONFIG
   b. Verify the software assigned ID: MRCONFIGID
   c. Verify the measurement of the initial contents of the TD: MRTD
   d. Verify kernel measurements provided in RTMR[0] and RTMR[1]
      i. By convention, RTMR[0] and RTMR[1] are updated by the TD virtual firmware/BIOS (TDVF). The measurements and the log file may differ depending on the TDVF vendor. For more information on the measurements in RTMR[0] and RTMR[1], contact your TDVF vendor.
   e. Verify any expected runtime generated measurements in RTMR[2] and RTMR[3]

i. By convention, RTMR[2] and RTMR[3] are measurements generated by the OS and runtime code. For more information on the RTMR[2] measurement, contact your OS vendor. For RTMR[3], contact the TD workload owner.

2. TD Attributes:
    a. Verify that all TD Under Debug flags (i.e., the `TDATTIBUTES.TUD` field in the TD Quote Body) are set to zero. If any flag is non-zero, the TD should not be trusted and thus should not be provisioned with production secrets.
    b. Verify that all security relevant attributes (i.e., the `TDATTIBUTES.SEC` field in the TD Quote body) are set as expected. Depending on the software stack used inside the TD (e.g., vBIOS, OS kernel), the value of these attributes may impact the security of a TD. For example, setting SEPT_VE_DISABLE attribute for the Linux kernel**.** Refer to the TDX Module documentation for more information on the latest security relevant attributes.

3. XFAM (eXtended Features Available Mask)

4. Report data:
    a. The TD can use the report data to provide specific data to the quote verifier. The following is a list of important examples:
        i. To provide replay protection, the report data can contain a nonce (provided by the quote verifier) or some other data that proves the freshness of the TD Quote.
        ii. To guarantee the integrity of some data inside the TD, the report data can contain a hash of this data.
        iii. To start the establishment of a secure channel between the Relying Party and the TD, the report data can contain a public key corresponding to a private key that has been generated within the TD. This public key can be used in a next step for a secure session key establishment.

3 Intel® TDX Quote Generation Library

This chapter and its subsections describe the C-like API used by TD tenant workloads to generate TD Quotes. The API abstracts the communication channel between the TD and the host OS.

Intel provides an implementation of the Intel® TDX Quote Generation Library (QGL). The provided binary of this library is called Ring 3 Attestation Abstraction Library (R3AAL), which should be installed in the TD and this library exposes the API described in this section. The rest of this document will refer to this library using use the name Intel® TDX Quote Generation Library.

3.1 TD Tenant Quote Generation API Definitions

3.1.1 Get TD Quote

Description

Request a TD Quote (in the data format defined in Appendix Version 4 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID)A.3 or Version 5 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID)A.4, depends on the platform and TD configuration) for the calling TD.

Using a buffer and the `p_td_report_data` parameter as pointer to the buffer, the caller can provide data that is cryptographically bound to the resulting TD Quote. Note that only the integrity, not the confidentiality is protected for this data.

A given platform can create TD Quotes using different cryptographic algorithms, different vendors' code/enclaves, and/or different formats. The so-called attestation key ID uniquely identifies a particular TD Quote type. Note that different attestation key IDs, e.g., might use different algorithms that result in a TD Quote of the same format, or they might use the same algorithms, but result in different TD Quote formats. The attestation key ID is represented as a Universally Unique Identifier (UUID).

Using a buffer and the `p_att_key_id_list` parameter as a pointer to the buffer, the caller can provide a list of attestation key IDs the caller requires from the platform. Typically, the list is provided by the party that eventually verifies the TD Quote. How the caller of this function obtains this list is outside the scope of this document. The function compares the provided list to the attestation key IDs that the platform supports and uses the first match. If the platform supports none of the attestation key IDs in the list, the API will return an error (`TDX_ATT_UNSUPPORTED_ATT_KEY_ID`). If the `p_att_key_id_list` parameter is `NULL`, the platform will use a default attestation key ID. The selected attestation key ID is accessible in a buffer pointed to by the `p_att_key_id` parameter.

When the function returns successfully, `p_quote` will point to a buffer containing the resulting TD Quote. This buffer is allocated by this function. The caller is expected to use the `tdx_att_free_quote` function to free the buffer pointed to by `p_quote`.

Syntax

```
tdx_att_error_t tdx_att_get_quote(
                            const tdx_report_data_t *p_tdx_report_data,
                            const tdx_uuid_t *p_att_key_id_list,
                            uint32_t list_size,
                            tdx_uuid_t *p_att_key_id,
```

```
                                uint8_t **p_quote,
                                uint32_t *p_quote_size,
                                uint32_t flags);
```

Parameters

`p_td_report_data` [In]

Pointer to data that the TD wants to cryptographically bind to the TD Quote. See REPORTDATA in TD_Quote_Body for more information. If the pointer is not NULL, the corresponding data will be present in the REPORTDATA field of the TD Quote body. Must not be NULL.

`p_att_key_id_list` [In]

Pointer to a list (array) of the attestation key IDs. If the pointer is not NULL, the function compares the attestation key IDs in the list to the attestation key IDs that the platform supports and uses the first match. If the pointer is NULL, the function uses the platform's default attestation key ID. The selected attestation key ID is accessible in a buffer pointed to by the `p_att_key_id` parameter when the function returns.

`list_size` [In]

The number of entries in the list of attestation key IDs pointed to by `p_att_key_id_list`. Currently, only a list size of 1 is supported.

`p_att_key_id` [Out]

The selected attestation key ID when the function returns. May be NULL.

`p_quote` [Out]

Pointer to a buffer containing the resulting TD Quote as byte (uint8_t) array. The function also allocates this buffer. The called is expected to use the `tdx_att_free_quote` function to free this buffer.

`p_quote_size` [Out]

Pointer to a uint32_t buffer, which will contain the size of the resulting TD Quote in bytes. May be NULL.

`flags` [In]

Reserved, must be zero.

Return Values

`TDX_ATTEST_SUCCESS`:

Successfully generated the TD Quote.

`TDX_ATTEST_ERROR_UNEXPECTED`:

An unexpected internal error occurred.

`TDX_ATTEST_ERROR_INVALID_PARAMETER`:

Returned under the following conditions:

```
p_quote == NULL
att_key_id_list == NULL && list_size != 0
att_key_id_list != NULL && list_size == 0
flags != 0
list_size > 1
```

TDX_ATT_UNSUPPORTED_ATT_KEY_ID:
The platform does not support any of the attestation key IDs described in the list pointed to by p_att_key_id_list.

TDX_ATTEST_ERROR_OUT_OF_MEMORY:
Heap memory allocation error in Quote Generation Library.

TDX_ATTEST_ERROR_BUSY:
Exceed supported number of concurrent, asynchronous calls of the function.

TDX_ATTEST_ERROR_DEVICE_FAILURE:
Failed to communicate to the TDX Module.

TDX_ATTEST_ERROR_VSOCK_FAILURE:
Error in the vsock communication channel

TDX_ATTEST_ERROR_NOT_SUPPORTED:
Preferred communication channel is not supported.

TDX_ATTEST_ERROR_QUOTE_FAILURE:
General error code when failed to generate TD Quote.

3.1.2 Free TD Quote

Description

Free the buffer of the TD Quote allocated by the tdx_att_get_quote function.

Syntax

```
tdx_att_error_t tdx_att_free_quote(
                              uint8_t *p_quote);
```

Parameters

p_quote [In]
The pointer to the buffer of the TD Quote returned by tdx_att_get_quote.

Return Values

TDX_ATT_SUCCESS:

Successfully freed the buffer of the TD Quote .

TDX_ATT_ERROR_UNEXPECTED:
An unexpected internal error occurred.

3.1.3 Get TD Report

Description

Request a TD Report of the calling TD. The caller can provide data intended to be cryptographically bound to the resulting TD Report. Note that only the integrity, not the confidentiality is protected for this data.

Syntax

```
tdx_att_error_t tdx_att_get_report(
                            const tdx_report_data_t *p_tdx_report_data,
                            tdx_report_t *p_tdx_report);
```

Parameters

`p_tdx_report_data` [In]
: Pointer to data that the TD wants to cryptographically bind to the TD Quote See REPORTDATA in TD_Quote_Body for more information. If the pointer is not NULL, the corresponding data will be present in the REPORTDATA field of the TD Quote body. Must not be NULL.

`p_tdx_report` [Out]
: Pointer to a buffer that will contain the generated TD Report. Must not be NULL.

Return Values

`TDX_ATTEST_SUCCESS`:
: Successfully generated the TD Report.

`TDX_ATT_ERROR_UNEXPECTED`:
: An unexpected internal error occurred.

`TDX_ATTEST_ERROR_REPORT_FAILURE`:
: There was an error retrieving the TD Report from the TDX Module.

`TDX_ATTEST_ERROR_INVALID_PARAMETER`:
: Returned under the following conditions:
`p_td_report == NULL`

`TDX_ATT_OUT_OF_MEMORY`:
: Heap memory allocation error in library or enclave.

`TDX_ATTEST_ERROR_DEVICE_FAILURE`:
: There was an error opening the interface to the TDX Module.

3.1.4 Get Platform Supported TD Attestation Keys

Description

Used to retrieve the list of attestation key IDs supported by the platform.
Specify `p_att_key_id_list = NULL` to learn the number of entries in the list.

Currently, the only supported TD attestation key id is:

```
TDX_SGX_ECDSA_ATTESTATION_ID {
    0xe8, 0x6c, 0x04, 0x6e, 0x8c, 0xc4, 0x4d, 0x95,
    0x81, 0x73, 0xfc, 0x43, 0xc1, 0xfa, 0x4f, 0x3f
}
```

Syntax

```
tdx_att_error_t tdx_att_get_supported_att_key_ids(
                            tdx_uuid_t *p_att_key_id_list,
                            uint32_t *p_list_size);
```

Parameters

`p_att_key_id_list` [Out]

List of the attestation key IDs that the platform supports. If set to `NULL`, the function will return the amount of attestation key IDs supported by the platform in the `uint32_t` pointed to by f`p_list_size`. This is useful to determine how big the output buffer pointed to by `p_att_key_id_list` has to be allocated for a subsequent call.

`p_list_size` [In/Out]

As input, when `p_att_key_id_list` is not `NULL`, `p_list_size` is a pointer to a `uint32_t` specifying the number of attestation key ID entries stored in the list pointed to by `p_att_key_id_list`.

As output, when `p_att_key_id_list` is `NULL`, `p_list_size` is a pointer to a `uint32_t` specifying the number of attestation key IDs entries supported by the platform.

Return Values

`TDX_ATT_SUCCESS`:

`p_att_key_id_list` points to a list of attestation key IDs and `p_list_size` points to a `uint32_t` that indicates the number of entries in the list of attestation key IDs. If `p_att_key_id_list` is set to NULL when calling the function, the function returns the amount of attestation key IDs supported by the platform in the `uint32_t` pointed to by `p_list_size.`

`TDX_ATT_ERROR_INVALID_PARAMETER`:

Returned under the following conditions:

```
p_list_size == NULL
*p_list_size < required size (required size will still be returned in
   this case)
p_att_key_id_list != NULL && *p_list_size == NULL
p_att_key_id_list == NULL && *p_list_size != NULL
```

`TDX_ATT_ERROR_UNEXPECTED`:

Unexpected internal error.

4 Intel® TDX Quote Verification Library (QVL)

This chapter presents the C-like API that allows applications to verify a TD using either the Quote Verification Enclave (QvE) when SGX is available or with an untrusted implementation when SGX is not available. Both implementations are contained in the Intel® TDX Quote Verification Library (QVL).

This library is delivered as a dynamically linked library (`.so/.dll`).

4.1 TD Quote Verification API Definitions

4.1.1 Set Enclave Load Policy

When the TD quote needs to be verified inside QvE instead of TD VM, you need to know the proper enclave loading policy. The library may be linked with a long-lived process, such as a service, where it can load the enclaves and leave them loaded (persistent). This better ensures that the enclaves will be available upon quote requests and not subject to EPC limitations if loaded on demand. However, if the Quoting library is linked with an application process, there may be many applications with the Quote Verification Library and a better utilization of EPC is to load and unloaded the quoting verification enclave on-demand (ephemeral). The library will be shipped with a default policy of loading the enclave and unloading it on-demand. (SGX_QL_EPHEMERAL).

If the policy is set to default SGX_QL_EPHEMERAL, then the QvE will be loaded and unloaded on-demand. If the enclave is already loaded when the policy is change to SGX_QL_EPHEMERAL, the enclave will be unloaded before returning.

To enhance quote verification performance, we introduced new polices **SGX_QL_EPHEMERAL_QVE_MULTI_THREAD** and **SGX_QL_PERSISTENT_QVE_MULTI_THREAD** from DCAP 1.18 release. Each thread will have its own QvE instance, so please make sure you have enough EPC to load QvE in this mode. We recommend using the 2 new policies to get better performance if your system has enough EPC.

Note that the QvE loading policy can only be changed once in one process.

Syntax

```
quote3_error_t sgx_qv_set_enclave_load_policy(sgx_ql_request_policy_t policy);
```

Parameters

policy[In]

SGX_QL_EPHEMERAL - Default policy. QvE is initialized and terminated on every quote verification function call.

SGX_QL_PERSISTENT - All the threads will share single QvE instance, and QvE is initialized on first use and reused until process ends.

SGX_QL_EPHEMERAL_QVE_MULTI_THREAD - QvE is loaded per thread and be unloaded before function exit.

SGX_QL_PERSISTENT_QVE_MULTI_THREAD - QvE is loaded per thread and only be unloaded before thread exit.

Return Values

SGX_QL_SUCCESS:
Successfully set the enclave loading policy for the quoting library's enclaves.

SGX_QL_UNSUPPORTED_LOADING_POLICY:
The selected policy is not supported, or it has been set once.

4.1.2 Verify TD Quote

Description

This function verifies a TD Quote (in the data format defined in Appendix Version 4 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID) and in Appendix Version 5 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID) generated by the TD Quoting Enclave (TDQE). The same function can be used to verify SGX Quotes, but we concentrate on TD Quotes in this document.

The Quote verification can be done in two ways: (1) in an unprotected manner or (2) inside an SGX enclave, the so-called Quote Verification Enclave (QvE). The Intel® TDX Quote Verification Library (QVL) supports both ways, but the second way required an SGX-protected environment. If the caller does not want to use the QvE, the `p_qve_report_info` pointer must be set to `NULL`. As a result, the verification result cannot be cryptographically authenticated. If the caller wants to use the QvE, the caller must run another SGX enclave that expects the result. Besides, the caller has to fill a `sgx_ql_qe_report_info_t` data structure (as defined in Appendix Core Generic Quote Wrapper Structures) and provide a pointer to this data structure in `p_qve_report_info`. This data structure must contain a nonce and the target info of the caller's enclave that expects the result. Once the QvE-based quote verification is complete, the `sgx_ql_qe_report_info_t` will contain an SGX Report with the verification result targeted to the caller's enclave.

The caller may provide the necessary Quote verification collateral by filling a `sgx_ql_qve_collateral_t` data structure (as defined in Appendix Quote Library Data Structures) and providing a pointer to this data structure in `p_quote_collateral`. If `p_quote_collateral` is `NULL`, the QVL will attempt to retrieve the verification collateral from the Platform Quote Provider Library (see Section 5.1). If the retrieval fails, this function will return `SGX_QL_PLATFORM_LIB_UNAVAILABLE`.

Independent of whether the Quote collateral is provided by the caller or successfully loaded by the Quote Verification Library, the QVL verifies the format and the certificate chains of all the collateral passed in. The function will return the appropriate error if the verification collateral format or signature checks fail. The current version of the QVL only supports version 1 of `sgx_ql_qve_collateral_t` data structure.

Once the function returns, the expiration status of the passed Quote verification collateral will be present in an `uint32_t` buffer pointed to by `p_collateral_expiration_status`. To evaluate the expiration,

the function expects a date passed via the `expiration_check_date` parameter. This date is compared to X.509 '`Not After`' fields, X.509 CRL '`Next Update`' fields, and the JSON '`nextUpdate`' fields in the passed Quote verification collateral. If any of these fields contains a date earlier than defined in `expiration_check_date`, the value pointed to by `p_collateral_expiration_status` will have a non-zero value. Note that the expiration check alone will not cause this function to return an error.

If all the input parameters are correctly formatted and signed, this function will return `SGX_QL_SUCCESS`. This function will also indicate the verification result via the `p_quote_verification_result` pointer pointing to a `sgx_ql_qv_result_t` enum (as defined in Appendix Quote Library Data Structures). This enum can have the following values:

1. `SGX_QL_QV_RESULT_OK` – *Non-terminal*
2. `SGX_QL_QV_RESULT_SW_HARDENING_NEEDED` – *Non-Terminal*
3. `SGX_QL_QV_RESULT_CONFIG_NEEDED` – *Non-terminal*
4. `SGX_QL_QV_RESULT_CONFIG_AND_SW_HARDENING_NEEDED` – *Non-Terminal*
5. `SGX_QL_QV_RESULT_OUT_OF_DATE` – *Non-terminal*
6. `SGX_QL_QV_RESULT_OUT_OF_DATE_CONFIG_NEEDED` – *Non-Terminal*
7. `SGX_QL_QV_RESULT_TD_RELAUNCH_ADVISED` – *Non-Terminal*
8. `SGX_QL_QV_RESULT_TD_RELAUNCH_ADVISED_CONFIG_NEEDED` – *Non-Terminal*
9. `SGX_QL_QV_RESULT_INVALID_SIGNATURE` – *Terminal*
10. `SGX_QL_QV_RESULT_REVOKED` – *Terminal*
11. `SGX_QL_QV_RESULT_UNSPECIFIED` – *Terminal*

If the caller wants to adhere to a strict compliance verification based on Intel's latest verification policy, the caller should input a trusted current time for `expiration_check_data` and only accept results when the function returns `SGX_QL_SUCCESS`, `*p_expiration_status` is 0, and `*p_quote_verification_result` is `SGX_QL_QV_RESULT_OK`. The other non-terminal verification results need more analysis before establishing trust in the attesting enclave.

- `SGX_QL_QV_RESULT_CONFIG_NEEDED`: The TDX platform firmware and TDX platform software are at the latest security patching level but there are platform hardware configurations that may expose the enclave to vulnerabilities. If these vulnerabilities are mitigated with the appropriate platform configuration changes, the verification result will change to `SGX_QL_QV_RESULT_OK`. See Intel communications on Security Advisories (SA) for information on configuration requirements (https://www.intel.com/content/www/us/en/security-center/default.html).

- `SGX_QL_QV_RESULT_SW_HARDENING_NEEDED`: The TDX platform firmware and TDX platform software are at the latest security patching level but there are certain vulnerabilities that can only be mitigated with software mitigations implemented by the TD. The TD Identity Policy (see Section Extended TD Checks) needs to indicate whether the TD has implemented these mitigations.

- `SGX_QL_QV_RESULT_CONFIG_AND_SW_HARDENING_NEEDED`: The TDX platform firmware and TDX platform software are at the latest security patching level but there are platform hardware configurations that may expose the TD to vulnerabilities. Additionally, there are certain vulnerabilities that can only be mitigated with software mitigations implemented by the TDe. If the platform configuration is changed appropriately, the verification result will change to `SGX_QL_QV_RESULT_SW_HARDENING_NEEDED`. See Intel communications on Security Advisories (SA) for information on configuration requirements (https://www.intel.com/content/www/us/en/security-center/default.html). The TD Identity Policy (see Section Extended TD Checks) needs to indicate whether the enclave has implemented these software mitigations.

- SGX_QL_QV_RESULT_OUT_OF_DATE: The TDX platform firmware and TDX platform software are ***not*** at the latest security patching level. The platform needs to be patched with firmware and/or software patches in order to produce an SGX_QL_QV_RESULT_OK verification result. See Intel communications on Security Advisories (SA) for information on patching requirements (https://www.intel.com/content/www/us/en/security-center/default.html).

- SGX_QL_QV_RESULT_OUT_OF_DATE_CONFIG_NEEDED: The TDX platform firmware and TDX platform software are ***not*** at the latest security patching level and there are platform hardware configurations that may expose the enclave to vulnerabilities. The platform needs to be patched with firmware and/or software patches in order to produce an SGX_QL_QV_RESULT_CONFIG_NEEDED verification result. If, additionally, the platform configuration is changed appropriately, the verification result will change to SGX_QL_QV_RESULT_OK. See Intel communications on Security Advisories (SA) for information on patching and configuration requirements (https://www.intel.com/content/www/us/en/security-center/default.html).

- SGX_QL_QV_RESULT_TD_RELAUNCH_ADVISED: The TDX platform firmware and TDX platform software are at the latest security patching level. However, the TD was launched prior to the application of new TDX TCB components using a TD Preserving update. For more information on a TD Preserving update, see Intel® TDX Module v1.5 Base Architecture Specification. This result indicates a TD originally ran under a TDX TCB with a lower security version but it is now executing with the latest TDX TCB. The relying party should make a trust decision on whether any secrets or data needs to be re-provisioned due to this exposure. Re-launching the TD will change the attestation result.

- SGX_QL_QV_RESULT_TD_RELAUNCH_ADVISED_CONFIG_NEEDED: The TDX platform firmware and TDX platform software are at the latest security patching level but there are platform hardware configurations that may expose the TD to vulnerabilities. In addition, the TD was launched prior to the application of new TDX TCB components using a TD Preserving update. For more information on a TD Preserving update, see Intel® TDX Module v1.5 Base Architecture Specification. This result indicates a TD originally ran under a TDX TCB with a lower security version but it is now executing with the latest TDX TCB. The relying party should make a trust decision on whether any secrets or data needs to be re-provisioned due to this exposure. Re-launching the TD will change the attestation result.

There may be cases where the caller cannot abide by the 'strict' compliance verification applied by this function. See Section Quote Verification Trust Policies for more details.

When this function returns a status other than SGX_QL_SUCCESS or *p_quote_verification_result is a terminal error, no alternative verification policy should be performed.

Syntax

```
quote3_error_t tee_verify_quote(
                    const uint8_t *p_quote,
                    uint32_t quote_size,
                    const uint8_t *p_quote_collateral,
                    const time_t expiration_check_date,
                    uint32_t *p_collateral_expiration_status,
                    sgx_ql_qv_result_t *p_quote_verification_result,
```

```
    sgx_ql_qe_report_info_t *p_qve_report_info,
    uint8_t *p_supp_data_descriptor);
```

Parameters

p_quote [In]

Pointer to the Quote that should be verified. The QVL only supports Quotes in the format corresponding to version 4 (see Appendix Version 4 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID) for details). Both the Intel-signed QvE and the Intel-provided Quote Verification Library only verify Quotes generated by the Intel-Signed TDQE. Currently, the QVL only supports Quotes with QE Certification Data.CertDataType = 5. This type of certification data contains the PCK in the Quote.

quote_size [In]

Size of the buffer pointed to by p_quote (in bytes).

p_quote_collateral [In]

This parameter is optional. If not-NULL, this is a pointer to a caller-provided Quote verification collateral in the sgx_ql_qve_collateral_t data structure (as defined in Appendix Quote Library Data Structures). Currently, the Quote verification collateral includes the TCBInfo (TDX), the QEIdentity (TDX), and CRL, which are necessary to verify a Quote (see https://api.portal.trustedservices.intel.com/provisioning-certification for more information on these collateral). If this parameter is NULL, the Quote Verification Library will attempt to retrieve the collateral from the Platform Quote Provider Library (see Section 5.1). If the Platform Quote Provider Library is not available or the verification collateral cannot be retrieved, this function will return SGX_QL_PLATFORM_LIB_UNAVAILABLE or SGX_QL_UNABLE_TO_GET_COLLATERAL, respectively.

expiration_check_date [In]

This is the date that the QVL uses to determine if any of the Quote collateral has expired. The expectation is that the caller has access to a trusted current time source or the caller can use a hardcoded threshold date.

p_collateral_expiration_status [Out]

Pointer to a buffer containing the expiration status. This parameter must not be NULL. If none of the Quote collateral has expired (compared to the inputted expiration_check_date), the expiration status will be 0. If one or more of the Quote collateral has expired (compared to the inputted expiration_check_date), the expiration status will be non-zero. The value at this address will contain a non-zero value if the function returns a value other than SGX_QL_SUCCESS.

p_quote_verification_result [In/Out]

Pointer to a buffer containing the quote verification result. The value at this address will contain SGX_QL_QV_RESULT_UNSPECIFIED if the API returns a value other than SGX_QL_SUCCESS.

p_qve_report_info [In/Out]

This parameter is optional. If the caller wants to use the QvE for Quote verification, the caller must fill a sgx_ql_qe_report_info_t data structure (as defined in Appendix A.1) and provide a pointer to this data structure with this parameter. This data structure must contain a nonce and target info of the enclave that will be used to verify the resulting SGX report. The QvE will generate an SGX Report (i.e., the QvE Report) using this provided target info. The report data of the resulting SGX Report will be the following: QVE.REPORT.REPORT_DATA = (SHA256_HASH[nonce||quote||expiration_check_date||expiration_status||verification_result||

supplemental_data]||32-0x00's). If the caller does not want to use the QvE or does not have an SGX-capable platform, the parameter must be NULL. The QVL will still verify the Quote, but the results cannot be cryptographically verified.

p_supp_data_descriptor [In/Out]

If pointer is NULL, no supplemental data is returned. Otherwise, this is a pointer to buffer containing supplemental data after the function successfully returned. The supplemental data is formatted in the tee_supp_data_descriptor_t data structure defined in Appendix Quote Library Data Structures. Caller can specify the major version of supplemental data by setting p_supp_data_descriptor->major_version. Setting the major version to 0 will result in the latest version of the tee_supp_data_descriptor_t data structure. Setting the major version to anything below the latest supported version, will result in the latest minor version of the tee_supp_data_descriptor_t data structure associated with the given major version. Any invalid major version will result in the error SGX_QL_SUPPLEMENTAL_DATA_VERSION_NOT_SUPPORTED.

Return Values

SGX_QL_SUCCESS:

Successfully evaluated the Quote.

SGX_QL_INVALID_PARAMETER:

One of the input parameters is invalid.

SGX_QL_QUOTE_FORMAT_UNSUPPORTED:

The format of the inputted Quote is not supported. Either because the header information is not supported or the Quote is malformed in some way.

SGX_QL_QUOTE_CERTIFICATION_DATA_UNSUPPORTED:

The QVL/QvE doesn't support the certification data in the Quote. Currently, the QVL (and with it the QvE) only support QE Report Certification Data.CertType = 5.

SGX_QL_QE_REPORT_UNSUPPORTED_FORMAT:

The QVL doesn't support the format of the TDQE Report in the Quote.

SGX_QL_QE_REPORT_INVALID_SIGNATURE:

The signature over the TDQE Report is invalid.

SGX_QL_QE_REPORT_ATT_KEY_MISMATCH:

The Attestation Key contained in the Quote was not generated by the TDQE described in the Quote.

SGX_QL_PCK_CERT_UNSUPPORTED_FORMAT:

The format of the PCK Cert is unsupported.

SGX_QL_PCK_CERT_CHAIN_ERROR:

There was an error verifying the certificate chain contained in the PCK Cert. This error can also be returned while validating the PCK Cert revocation.

SGX_QL_TCBINFO_UNSUPPORTED_FORMAT:

The format of the TCBInfo structure is unsupported.

SGX_QL_TCBINFO_CHAIN_ERROR:

There was an error verifying the signature chain of the TCBInfo. This error can also be returned while validating the TCBInfo revocation.

SGX_QL_TCBINFO_MISMATCH:
PCK Cert FMSPc does not match the TCBInfo FMSPc.

SGX_QL_QEIDENTITY_UNSUPPORTED_FORMAT:
The format of the QEIdentity structure is unsupported.

SGX_QL_QEIDENTITY_MISMATCH:
The identity of the TDQE contained in the Quote does not match the provided QEIdentity.

SGX_QL_QEIDENTITY_CHAIN_ERROR:
There was an error verifying the signature chain of the QEIdentity. This error can also be returned while validating QEIdentity revocation.

SGX_QL_OUT_OF_MEMORY:
Heap memory allocation error in the QVL/QvE.

SGX_QL_ENCLAVE_LOAD_ERROR:
Unable to load the QvE. Could be due to file I/O error, loading infrastructure error or insufficient enclave memory.

SGX_QL_ENCLAVE_LOST:
Enclave lost after power transition or used in child process created by linux:fork().

SGX_QL_INVALID_REPORT:
Report MAC check failed on TDQE Report.

SGX_QL_PLATFORM_LIB_UNAVAILABLE:
The Quote Verification Library could not locate the Platform Quote Provider Library (see Section 5.1) or one of Platform Quote Provider Library's required function.

SGX_QL_UNABLE_TO_GENERATE_REPORT:
The QvE was unable to generate an SGX Report about itself targeting the enclave that should verify the verification result.

SGX_QL_NO_QUOTE_COLLATERAL_DATA:
The Platform Quote Provider Library available, but it could not retrieve the data.

SGX_QL_ERROR_QVL_QVE_MISMATCH:
Only returned when the Quote Verification Library supports both, the unprotected quote verification and the QvE-backed quote verification. This error indicates that the version of the QVL does not match the version of the QvE. Most likely caused by using a QvE that does not match the version of the DCAP QVL installed.

SGX_QL_ERROR_UNEXPECTED:
An unexpected internal error occurred.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:
QVL/QVE received unexpected message while retrieving verification collateral.

SGX_QL_ERROR_MESSAGE_PARSING_ERROR:
Generic message parsing error from the attestation infrastructure while retrieving the platform data.

SGX_QL_PLATFORM_UNKNOWN:
The Platform Quote Provider Library was not able to collect the TDTCBInfo for this platform.

SGX_QL_TDX_MODULE_MISMATCH:
The security level of the TDX Module installed on the platform cannot be evaluated because it is not represented in the TDTCBInfo. This can happen when an old TDTCBInfo is used in the verification.

4.1.3 Get Quote Verification Supplemental Version and Size

Description

If the quote verifier wants to implement custom quote verification checks using supplemental data (see Section Quote Verification Trust Policies), the quote verifier must call this function (i.e., `tee_get_supplemental_data_version_and_size`) to retrieve the size and version of the supplemental data. Afterwards, the quote verifier can use the values to allocate a buffer and pass it as input parameter to `tee_verify_quote`. If custom quote verification checks should not be done, this function is not relevant.

This function is designed to support multiple versions of the supplemental data (see specification of `sgx_ql_qv_supplemental_t` in Appendix Quote Library Data Structures). The caller can specify which version of the supplemental data that the caller supports or request the latest version of the supplemental data.

Syntax

```
quote3_error_t tee_get_supplemental_data_version_and_size(
                                    const uint8_t *p_quote,
                                    uint32_t quote_size,
                                    uint32_t *p_version,
                                    uint32_t *p_data_size);
```

Parameters

`p_quote` [In]
Pointer to a Quote. Must not be `NULL`.

`quote_size` [In]
Size of the buffer pointed to by `p_quote` (in bytes).

`p_version` [In/Out]:
As input, a pointer to 0 requests the latest version of the supplemental data. A pointer to any other positive number requests the corresponding version of the supplemental data. The pointer must not be Null. As output, the value pointed to contains the version for which the data size is returned by the p_data_size pointer

`p_data_size` [Out]:
Pointer to hold the size of the buffer in bytes required to contain all of the supplemental data. Must not be NULL. The caller uses this size to allocate a buffer to contain the supplemental data before calling `tee_verify_quote`.

Return Values

SGX_QL_SUCCESS:

Successfully calculated the required supplemental data size. The required size in bytes is returned in the memory pointed to by `p_data_size`.

`SGX_QL_ERROR_INVALID_PARAMETER`:

Returned under the following conditions:

- `p_quote == NULL`
- `p_data_size == NULL`
- `p_version` == NULL
- `quote_size == 0`
- `p_quote is an unsupported type.`

`SGX_QL_ERROR_QVL_QVE_MISMATCH`:

Only returned when the Quote Verification Library supports both, the unprotected quote verification and the QvE-backed quote verification. This error indicates that the version of the QVL does not match the version of the QvE. Most likely caused by using a QvE that does not match the version of the DCAP QVL installed.

`SGX_QL_ERROR_UNEXPECTED`:

Unexpected internal error.

4.1.4 Extract the FMSPc from the Quote

Description

This function will extract the FMSPc (Family-Model-Stepping-Platform-CustomSKU) from the PCK Certificate in the certification data of the Quote. It will only work for TD Quotes in the format corresponding to version 4 (see Appendix Version 4 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID) for details) with QE Certification Data.CertDataType = 5. This type of certification data contains the PCK certificate in the Quote.

Typically, the function will be called when the caller needs to fetch the TDTCBInfo from the caching service or the Intel® SGX and Intel® TDX Provisioning Certification Service (PCS).

The caller needs to allocate a buffer for p_fmspc_from_quote, which is equal or larger than 6 bytes. This function will copy the FMSPc value to the memory allocated. The outputted FMSPc is a hex-encoded buffer (6 bytes). Finally, the caller needs free the FMSPc buffer.

Syntax

```
quote3_error_t tee_get_fmspc_from_quote(
                                    const uint8_t *p_quote,
                                    uint32_t quote_size,
                                    uint8_t *p_fmspc_from_quote,
                                    uint32_t fmspc_from_quote_size);
```

Parameters

`p_quote` [In]

Pointer to the Quote from which the FMSPc will be extracted. The function only supports Quotes in the format corresponding to version 4 (see Appendix Version 4 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID) for details). Currently, the QVL only supports Quotes with QE Certification Data.CertDataType = 5. This type of certification data contains the PCK Certificate in the Quote.

`quote_size` [In]
Size of the buffer pointed to by `p_quote` (in bytes).

`p_fmspc_from_quote` [In/Out]:
Pointer to the buffer that will contain the FMSPc upon successful completion.

`fmspc_from_quote_size` [In]:
Size in bytes of the buffer at `p_fmspc_from_quote`.

Return Values

`SGX_QL_SUCCESS`:
Successfully extracted the FMSPc from the PCK Certificate in the Quote certification data.

`SGX_QL_ERROR_INVALID_PARAMETER`:
Returned under the following conditions:

```
p_quote == NULL
p_data_size < QUOTE_MIN_SIZE
p_fmspc_from_quote == NULL
fmspc_from_quote_size < FMSPC_SIZE
p_quote is missing parameters.
```

`SGX_QL_ERROR_OUT_OF_MEMORY:`
Heap memory allocation error in the QVL/QvE.

`SGX_QL_QUOTE_CERTIFICATION_DATA_UNSUPPORTED:`

QE Certification Data.CertDataType not supported.

`SGX_QL_QUOTE_FORMAT_UNSUPPORTED:`
The format of the Quote is not supported.

`SGX_QL_PCK_CERT_CHAIN_ERROR:`
Cannot parse the PCK certificate chain, or root certificate is not trusted.

`SGX_QL_ERROR_UNEXPECTED`:
Unexpected internal error.

4.1.5 Verify Quote with Supplemental Data

Description

All custom Quote verification checks using the supplemental data have to be implemented by the quote verifier (see Section Quote Verification Trust Policies). ***There are currently no plans to implement custom verification checks in the QVL binary.***
In the following, we describe a simple custom verification check and the declaration of the corresponding function. We present a simple example, but the quote verifier may also consider using the QvE report to

verify the QvE inputs and results. Besides, the quote verifier may also perform the verification in an enclave and return an SGX Report of its own that can be validated by an enclave of another party.

For this example, we assume that `tee_verify_quote` returns `SGX_QL_SUCCESS` and `*p_quote_verification_result` contains a non-terminal verification result. Besides, we assume that the following custom data structure is defined:

```
typedef struct _tee_supp_freshness_t {
    uint32_t   min_tcb_eval_dataset_num;
    uint32_t   min_crl_num;
} tee_supp_freshness_t;
```

The custom verification check performs one (or both) of the following checks:

- Verify if a platform is vulnerable to a specific Security Advisory (SA):
    1. Find the specific SA at https://www.intel.com/content/www/us/en/security-center/default.html and record the 'Original Release' date.
    2. Platform is not vulnerable to specific SA if the supplemental data reports a TCB date greater-than-or-equal to date recorded in step 1.
- Verify if the collateral used by `tee_verify_quote` is newer than some date even if the collateral is expired.
    1. On a TCB Recovery Event, retrieve the new TCBInfo and QEIdentity and record the `tcbEvalDataSetNumber` from either collateral (as `min_tcb_eval_dataset_num` in a `tee_supp_freshness_t` data structure). Note that the `tcbEvalDataSetNumber` of latest TCBInfo and QEIdentity will always be in sync.
    2. When a PCK Cert CRL is updated, retrieve the new CRL and record the `CRLNum` (as `min_crl_num` in a `tee_supp_freshness_t` data structure).
    3. Determining freshness:
        i. Compare `tcb_eval_dataset_num` from the supplemental data to the recorded `min_tcb_eval_dataset_num` from the TCBInfo/QEidentity from step 1. If `tcb_eval_dataset_num` is greater-than-or-equal-to the `min_tcb_eval_dataset_number`, then the TCBInfo and QEIdentity are 'fresh'.
        ii. Compare the `pck_crl_num` from the supplemental data to the recorded `min_crl_num` from step 2. If `pck_crl_num` is greater-than-or-equal–to the `min_crl_num`, then the PCK CRL is 'fresh'.

The following is an example declaration for the custom verification check.

Syntax

```
quote3_error_t tee_verify_quote_supplemental(
                    tee_supp_data_descriptor_t *p_supp_data_descriptor,
                    const tee_supp_freshness_t *p_freshness_data,
                    const time_t *p_vulnerability_date);
```

Parameters

`p_supp_data_descriptor` [In]

Pointer to the supplemental data returned by `tee_verify_quote`. Must not be `NULL`. If `tee_verify_quote` returned a terminal verification result, the supplemental data will be invalid.

`freshness_data` [In]

Optional parameter. If NULL, vulnerability_date must not be NULL. If not NULL, all collateral used in the tee_verify_quote must have a tcb_eval_dataset_num greater than or equal to the min_tcb_eval_dataset_num and a pck_crl_num greater than or equal to the min_crl_num. If any of the collaterals' 'nums' are lower than the 'min nums', this function will return SGX_QL_COLLATERAL_NOT_FRESH.

vulnerability_date [In]

Optional parameter. If NULL, freshness_date must not be NULL. If not NULL, the platform must not be vulnerable to any SAs published before the vulnerability_date. If the platform is vulnerable to any SAs published before this date, then this function will return SGX_QL_PLATFORM_TCB_PREDATES_VULNERABILITY_DATE.

Return Values

SGX_QL_SUCCESS:

The quote supplemental date checks pass. All collateral has issue dates later than or equal the inputted freshness date and/or the platform is not vulnerable to an SGX SA published before the inputted date.

SGX_QL_SUPP_DATA_FORMAT_UNSUPPORTED:

The quote supplemental date checks pass.

SGX_QL_SUPP_DATA_INVALID:

The quote supplemental date checks pass.

SGX_QL_INVALID_PARAMETER:

Returned under the following conditions:

- `p_supplemental_data == NULL`
- `supplemental_data_size` is incorrect
- `freshness_data == NULL && vulnerability == NULL`

SGX_QL_COLLATERAL_NOT_FRESH:

The issue date of at least one of the collateral precedes the freshness date.

SGX_QL_PLATFORM_TCB_PREDATES_VULNERABILITY_DATE:

The platform is vulnerable to an SA published before the vulnerability date.

5 Dependent Libraries

In this section, we list and describe the libraries on which the Intel® TDX Quote Verification Library depend on.

5.1 Platform Quote Provider Library

The Platform Quote Provider Library provides an API that allows the Quote Verification Library to use platform specific services. This library is needed, when the Quote collateral is not explicitly provided as input to the Quote verification (see Section 4.1.1).

The Intel® TDX Quote Verification Library uses the same Platform Quote Provider Library as the one use by Intel® SDX and further details are provided in section 'Platform Quote Provider Library' in the following document: Intel_SGX_ECDSA_QuoteLibReference_DCAP_API.pdf.

A. Data Structures

A.1. Core Generic Quote Wrapper Structures

```
/** Structure definition used to provide target information of an enclave to the QvE when the
caller to the DCAP Quote Verification Library requires an SGX Report from the QvE. This
structure is only used when the caller requires the QvE to perform the Quote verification. */
typedef struct _sgx_ql_qe_report_info_t {
    sgx_quote_nonce_t   nonce;
    sgx_target_info_t   app_enclave_target_info;
    sgx_report_t        qe_report;
} sgx_ql_qe_report_info_t;
```

A.2. Quote Library Data Structures

```
/** Enum containing possible quote verification result values. */
typedef enum _sgx_ql_qv_result_t
{
   SGX_QL_QV_RESULT_OK = 0,          ///< The Quote verification passed and the platform
                                     ///< is patched to the latest TCB level.
   SGX_QL_QV_RESULT_CONFIG_NEEDED,   ///< The Quote verification passed and
                                     ///< the platform is patched to the
                                     ///< latest TCB level
                                     ///< but there are platform hardware configurations
                                     ///< that may expose the enclave to vulnerabilities.
                                     ///< The hardware configurations should be changed
                                     ///< appropriately.
   SGX_QL_QV_RESULT_OUT_OF_DATE,     ///< The Quote verification passed but the TCB
                                     ///< level of the platform is out of date.
                                     ///< The platform needs to be patched.
   SGX_QL_QV_RESULT_OUT_OF_DATE_CONFIG_NEEDED,   ///< The Quote verification passed
                                     ///< but the TCB level of the platform is out of
                                     ///< date and there are platform hardware
                                     ///< configurations that may expose the enclave to
                                     ///< vulnerabilities.
                                     ///< The platform needs to be patched and
                                     ///< the hardware configurations should be changed
                                     ///< appropriately.
   SGX_QL_QV_RESULT_INVALID_SIGNATURE,     ///< The signature over the
                                     ///< Quote is invalid.
   SGX_QL_QV_RESULT_REVOKED,         ///< The attestation key or platform
                                     ///< has been revoked.
   SGX_QL_QV_RESULT_UNSPECIFIED,     ///< The Quote verification failed due
                                     ///< to an unspecified error in processing the
                                     ///< Quote.
   SGX_QL_QV_RESULT_SW_HARDENING_NEEDED,   ///< The TCB level of the platform is
                                     ///< up to date, but software hardening is needed.
   SGX_QL_QV_RESULT_CONFIG_AND_SW_HARDENING_NEEDED,   ///< The TCB level of the
                                     ///< platform is up to date, but there are platform
                                     ///< hardware configurations that may expose the
                                     ///< enclave to vulnerabilities and software
                                     ///< hardening is needed.
                                     ///< The hardware configurations should be changed
                                     ///< appropriately.
```

```
} sgx_ql_qv_result_t;

/** This is the data provided to the quote verifier by the verifying platform software. They
are NULL terminated strings.  This data will need to be marshalled into the QVE as byte
buffers. PCK Cert chain is in the Quote thus there is no need to provide it in the collateral.
*/
typedef struct _sgx_ql_qve_collateral_t
{
    union {
        uint32_t version;                ///<‘version’ is the backward compatible legacy
                                         ///< representation
       struct {
           uint16_t major_version;       ///< For PCS V1 and V2 APIs, the
           uint16_t minor_version;       ///< major_version = 1 and minor_version = 0
       }                                 ///< the CRLs will be formatted in PEM. For
    };                                   ///< PCS V3 APIs, the major_version = 3 and the
                                         ///< minor_version can be either 0 or 1.  A
                                         ///< minor_verion of 0 indicates the CRL’s are
                                         ///< formatted in Base16 encoded DER.  A minor
                                         ///< version of 1 indicates the CRL’s are
                                         ///< formatted in raw binary DER.

    uint32_t tee_type;                   ///< 0x00000000: SGX or 0x00000081: TDX

    char *pck_crl_issuer_chain;         ///< concatenated - the order
                                        ///< as it is returned from PCS (root ca +
                                        ///< signing cert).  ‘version’ 1 collateral has
                                        ///< CRL encoded in PEM format.  ‘version’ 3
                                        ///< collateral has CRL encoded in DER format.
    uint32_t pck_crl_issuer_chain_size; ///< Size in bytes of the
                                        ///< pck_crl_issuer_chain string.  Size
                                        ///< includes the terminating NULL
                                        ///< character.
    char *root_ca_crl;                  ///< CRL for certs signed by root cert.
                                        ///< Version 1.0: PEM
                                        ///< Version 3.0: Base16 DER
                                        ///< Version 3.1: Raw Binary DER
    uint32_t root_ca_crl_size;          ///< Size in bytes of the
                                        ///< root_ca_crl string.  Size includes
                                        ///< the terminating NULL
    char *pck_crl;                       ///< CRL for PCK leaf certs.
                                        ///< Version 1.0: PEM
                                        ///< Version 3.0: Base16 DER
                                        ///< Version 3.1: Raw Binary DER
    uint32_t pck_crl_size;               ///< Size in bytes of the
                                        ///< pck_crl string.  Size includes the
                                        ///< terminating NULL
    char *tcb_info_issuer_chain;        ///< concatenated PEM format - the order
                                        ///< as it is returned from PCS (root ca +
                                        ///< signing cert)
    uint32_t tcb_info_issuer_chain_size; ///< Size in bytes of the
                                        ///< tcb_info_issuer_chain string.  Size
                                        ///< includes the terminating NULL
    char *tcb_info;                      ///< TCB Info structure
    uint32_t tcb_info_size;              ///< Size in bytes of the
                                        ///< tcb_info string.  Size includes the
                                        ///< terminating NULL
    char *qe_identity_issuer_chain;     ///< concatenated PEM format - the order
                                        ///< as it is returned from PCS (root ca +
                                        ///< signing cert)
```

```
    Uint32_t qe_identity_issuer_chain_size; ///< Size in bytes of the
                                            ///< qe_identity_issuer_chain string. Size
                                            ///< includes the terminating NULL
    char *qe_identity;                      ///< QE Identity Structure
    uint32_t qe_identity_size;              ///< Size in bytes of the
                                            ///< qe_identity string. Size includes the
                                            ///< terminating NULL
} sgx_ql_qve_collateral_t;

/** Contains data that will allow an alternative quote verification policy. */
typedef struct _sgx_ql_qv_supplemental_t
{
    union {
        uint32_t version;               ///< 'version' is the backward compatible
                                        ///< legacy representation
        struct {
            uint16_t major_version;     ///< If this major version doesn't change,
                                        ///< the size of the structure may change
                                        ///< and new fields appended to the end but
                                        ///< old minor version structure can still
                                        ///< be 'cast'
                                        ///< If this major version does change,
                                        ///< then the structure has been modified in
                                        ///< a way that makes the older definitions
                                        ///< non-backwards compatible. i.e. You
                                        ///< cannot 'cast' older definitions
            uint16_t minor_version;     ///< If this version changes, new fields
                                        ///< have been appended to the end of
                                        ///< the previous minor version definition
                                        ///< of the structure
                                        ///< Set to 1 to support SA_List.  Set to 0
                                        ///< to support everything except the SA List
        };
    };

    time_t earliest_issue_date;         ///< Earliest issue date of all the
                                        ///< collateral (UTC)
    time_t latest_issue_date;           ///< Latest issue date of all the
                                        ///< collateral (UTC)
    time_t earliest_expiration_data;    ///< Earliest expiration date of all the
                                        ///< collateral (UTC)
    time_t tcb_level_date_tag;          ///< The SGX platform that
                                        ///< generated the quote has the mitigations
                                        ///< prescribed for all Security Advisories with
                                        ///< an SGX TCB impact, released
                                        ///< on or before this date.
                                        ///< See Intel Security Center Advisories.
    uint32_t pck_crl_num;               ///< CRL Num from PCK Cert CRL
    uint32_t root_ca_crl_num;           ///< CRL Num from Root CA CRL
    uint32_t tcb_eval_dataset_num;      ///< Lower number of the TCBInfo's and
                                        ///< QEIdentity's tcbEvalDataSetNumber
    uint8_t root_key_id[48];            ///< ID of the collateral's root signer
                                        ///< (hash of Root CA's public key SHA-384
    sgx_key_128bit_t pck_ppid;          ///< PPID from remote platform.  Can be used
                                        ///< for platform ownership checks.
    sgx_cpu_svn_t tcb_cpusvn;           ///< CPUSVN of the remote platform's PCK
                                        ///< Cert
    sgx_isv_svn_t tcb_pce_isvsvn;       ///< PCE_ISVNSVN of the remote platform's
                                        ///< PCK CERT
    uint16_t pce_id;                    ///< PCE_ID of the remote platform
```

```
    uint8_t sgx_type;                   ///< Indicate the type of memory protection
                                        ///< available on the platform, it should be
                                        ///< one of Standard (0) and Scalable (1)
                                        ///< Multi-Package PCK cert related flags, they
                                        ///< are only relevant to PCK Certificates
                                        ///< issued by PCK Platform CA
    uint8_t platform_instance_id[PLATFORM_INSTANCE_ID_SIZE]; ///< Value of Platform
                                                             ///< Instance ID,16bytes
    pck_cert_flag_enum_t dynamic_platform;///< Indicate whether a platform can be
                                          ///< extended with additional packages
                                          ///< - via Package Add calls to SGX
                                          ///< Registration Backend
     pck_cert_flag_enum_t cached_keys;    ///< Indicate whether platform root
                                          ///< keys are cached by SGX
                                          ///< Registration Backend
     pck_cert_flag_enum_t smt_enabled;    ///< Indicate whether a plat form has
                                          ///< SMT (simultaneous multithreading)
                                          ///< enabled
     char sa_list[MAX_SA_LIST_SIZE];      ///< String of comma separated list
                                          ///< of Security Advisory IDs
} sgx_ql_qv_supplemental_t;
This data needs to be hashed in QvE.Report.ReportData.


/** Descriptor of the data structure used to request supplemental data from the Quote
Verification Library. */
typedef struct _tee_supp_data_descriptor_t
{
     uint16_t major_version;         ///< Input. Major version of supplemental
                                     ///< data.
                                     ///< If major_version = 0, the library will return
                                     ///< the latest version of the
                                     ///< sgx_ql_qv_supplemental_t structure.
                                     ///< If major_version <= latest supported, the
                                     ///< library will return the latest minor version
                                     ///< associated with that that major version.
                                     ///< If major_version > latest, return an error
                                     ///<(SGX_QL_SUPPLEMENTAL_DATA_VERSION_NOT_SUPPORTED)

    uint32_t data_size;              ///< Input. Size of buffer for supplemental data p_,
                                     ///< data. The size can be requested using
                                     ///< tee_get_supplemental_data_version_and_size
    uint8_t *p_data;                 ///< Output. Pointer to supplemental data.
} tee_supp_data_descriptor_t;


#define TDX_UUID_SIZE 16
/** UUID structure defined in TDX as a 16 byte array. */
typedef struct _tdx_uuid_t
{
    uint8_t d[TDX_UUID_SIZE];
} tdx_uuid_t;

/** 64 byte array that will contain the TD provided TDREPORT.REPORTDATA */
#define TDX_REPORT_DATA_SIZE 64
typedef struct _tdx_report_data_t
{
    uint8_t d[TDX_REPORT_DATA_SIZE];
} tdx_report_data_t;
```

A.3. Version 4 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID)

This section describes the format of Quotes in version 4. This format is used for TD Quotes, but also applies to SGX Quotes.

Endianess: *Little Endian (applies to all integer fields).*

Name	Size (bytes)	Type	Description
Quote Header	48	TD Quote Header	Header of *Quote* data structure. This field is ***transparent***, i.e., the user knows its internal structure. Rest of the *Quote* data structure can be treated as ***opaque***, i.e., hidden from the user.
TD Quote Body	584	TD Quote Body	Report of the attested TD. The REPORTDATA contained in this field is defined by the TD developer. See the description of the field for example usages.
Quote Signature Data Len	4	Integer	Size of the Quote Signature Data structure
Quote Signature Data	Variable	Signature Dependent	Variable-length data containing the signature and supporting data. For instance, an ECDSA P-256 Signature

A.3.1. TD Quote Header

Name	Size (bytes)	Type	Description
Version	2	Integer	Version of the *Quote* data structure. • *Value*: 4
Attestation Key Type	2	Integer	Type of the *Attestation Key* used by the *Quoting Enclave*. • *Supported values*: 2 (ECDSA-256-with-P-256 curve) 3 (ECDSA-384-with-P-384 curve) (*Note: currently not supported*) (Note: 0 and 1 are reserved, for when EPID is moved to version 4 quotes.)
TEE Type	4	Integer	TEE for this Attestation 0x00000000: SGX

			0x00000081: TDX
RESERVED	2	Byte Array	Zero
RESERVED	2	Byte Array	Zero
QE Vendor ID	16	UUID	Unique identifier of the QE Vendor. *Value*: 939A7233F79C4CA9940A0DB3957F0607 (Intel® SGX QE Vendor) *Note: Each vendor that decides to provide a customized Quote data structure should have unique ID.*
User Data	20	Byte Array	Custom user-defined data. For the Intel® SGX and TDX DCAP Quote Generation Libraries, the first 16 bytes contain a Platform Identifier that is used to link a PCK Certificate to an Enc(PPID). This identifier is consistent for every quote generated with this QE on this platform.

A.3.2. TD Quote Body

Name	Size (bytes)	Type	Description
TEE_TCB_SVN	16	Byte Array	Describes the TCB of TDX.
MRSEAM	48	SHA384	Measurement of the TDX Module.
MRSIGNERSEAM	48	SHA384	Zero for the Intel® TDX Module.
SEAMATTRIBUTES	8	Byte Array	Must be zero for TDX 1.0
TDATTRIBUTES	8	Byte Array	TD Attributes
XFAM	8	Byte Array	XFAM (eXtended Features Available Mask) is defined as a 64b bitmap, which has the same format as XCR0 or IA32_XSS MSR.
MRTD	48	SHA384	Measurement of the initial contents of the TD. See TDX Module definitions here: TDX Module documentation
MRCONFIGID	48	Byte Array	Software-defined ID for non-owner-defined configuration of the TD, e.g., runtime or OS configuration.
MROWNER	48	Byte Array	Software-defined ID for the TD's owner
MROWNERCONFIG	48	Byte Array	Software-defined ID for owner-defined configuration of the TD, e.g., specific to the workload rather than the runtime or OS.

RTMR0	48	SHA384	Runtime extendable measurement register
RTMR1	48	SHA384	Runtime extendable measurement register
RTMR2	48	SHA384	Runtime extendable measurement register
RTMR3	48	SHA384	Runtime extendable measurement register
REPORTDATA	64	Byte Array	Each TD Quote is based on a TD Report. The TD is free to provide 64 bytes of custom data to a TD Report. For instance, this space can be used to hold a nonce, a public key, or a hash of a larger block of data. Note that the signature of a TD Quote covers the REPORTDATA field. As a result, the integrity is protected with a key rooted in an Intel CA.

A.3.3. TEE_TCB_SVN

TEE_TCB_SVN		
Byte location	**Name**	**Description**
0	Tdxtcbcomp01	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[0]
1	Tdxtcbcomp02	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[1]
2	Tdxtcbcomp03	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[2]
3	Tdxtcbcomp04	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[3]
4	Tdxtcbcomp05	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[4]
5	Tdxtcbcomp06	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[5]
6	Tdxtcbcomp07	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[6]
7	Tdxtcbcomp08	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[7]
8	Tdxtcbcomp09	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[8]

9	Tdxtcbcomp10	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[9]
10	Tdxtcbcomp11	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[10]
11	Tdxtcbcomp12	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[11]
12	Tdxtcbcomp13	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[12]
13	Tdxtcbcomp14	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[13]
14	Tdxtcbcomp15	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[14]
15	Tdxtcbcomp16	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[15]

A.3.4. TD Attributes

Group	Bits	Description	Bits	Field	Description
TUD	7:0	TD Under Debug flags. If any of the bits in this group are set to 1, the TD is untrusted.	0	DEBUG	Defines whether the TD runs in TD debug mode (set to 1) or not (set to 0). In TD debug mode, the CPU state and private memory are accessible by the host VMM.
			7:1	RESERVED	Reserved for future TUD flags – must be 0
SEC	31:8	Attributes that may impact the security of the TD	27:8	RESERVED	Reserved for future SEC flags – must be 0
			28	SEPT_VE_DISABLE	Disable EPT violation conversion to #VE on TD access of PENDING pages
			29	RESERVED	Reserved for future SEC flags – must be 0
			30	PKS	TD is allowed to use Supervisor Protection Keys.
			31	KL	TD is allowed to use Key Locker.

OTHER	63:32	Attributes that do not impact the security of the TD	62:32	RESERVED	Reserved for future OTHER flags – must be 0
			63	PERFMON	TD is allowed to use Perfmon and PERF_METRICS capabilities.

A.3.5. ECDSA P-256 Signature

Name	Size (bytes)	Type	Description
Signature	64	Byte Array	ECDSA signature, the r component followed by the s component, 2 x 32 bytes.

A.3.6. ECDSA P-256 Public Key

Name	Size (bytes)	Type	Description
Public Key	64	Byte Array	EC KT-I Public Key, the x-coordinate followed by the y-coordinate (on the RFC 6090 P-256 curve), 2 x 32 bytes.

A.3.7. QE Authentication Data

Name	Size (bytes)	Type	Description
Size	2	Integer	Size of the 'Data' array. 0 is a valid value.
Data	Variable	Byte Array	Data to be additionally 'signed' by the certification key.

A.3.8. ECDSA 256-bit Quote Signature Data Structure – Version 4

Name	Size (bytes)	Type	Description
Quote Signature	64	ECDSA P-256 Signature	ECDSA signature over the *Header* and the *TD Quote Body* calculated using the private part of the *Attestation Key* generated by the *Quoting Enclave*.

ECDSA Attestation Key	64	ECDSA P-256 Public Key	Public part of the *Attestation Key* generated by the *Quoting Enclave*.
QE Certification Data	Variable	QE Certification Data – Version 4	Data required to verify the signature over QE Report and the *Attestation Key*.

A.3.9. QE Certification Data – Version 4

Name	Size (bytes)	Type	Description
Certification Data Type	2	Integer	Determines type of data required to verify the QE Report Signature in the Quote Signature Data structure. *Supported values*: 1 (PCK identifier: PPID in plain text, CPUSVN, and PCESVN) 2 (PCK identifier: PPID encrypted using RSA-2048-OAEP, CPUSVN, and PCESVN) 3 (PCK identifier: PPID encrypted using RSA-3072-OAEP, CPUSVN, and PCESVN) 4 (PCK Leaf Certificate in plain text; currently not supported) 5 (Concatenated PCK Cert Chain) 6 (QE Report Certification Data) 7 (PLATFORM_MANIFEST; currently not supported)
Size	4	Integer	Size of Certification Data field.
Certification Data	Variable	Byte Array	Data required to verify the QE Report Signature depending on the value of the *Certification Data Type*: 1: Byte array that contains concatenation of PPID, CPUSVN, PCESVN (LE), PCEID (LE). 2: Byte array that contains concatenation of PPID encrypted using RSA-2048-OAEP, CPUSVN, PCESVN (LE), PCEID (LE). 3: Byte array that contains concatenation of PPID encrypted using

			RSA-3072-OAEP, CPUSVN, PCESVN (LE), PCEID (LE). 4: PCK Leaf Certificate 5: Concatenated PCK Cert Chain (PEM formatted). PCK Leaf Cert|| Intermediate CA Cert|| Root CA Cert 6: QE Report Certification Data 7: PLATFORM_MANIFEST

A.3.10. Enclave Report Body

Name	Size (bytes)	Type	Description
CPU SVN	16	Byte Array	Security Version of HW components in the TDX TCB (raw value).
MISCSELECT	4	Integer	SSA Frame extended feature set. Reports what SECS.MISCSELECT settings are used in the enclave. You can limit the allowed MISCSELECT settings in the sigstruct using MISCSELECT/MISCMASK.
Reserved	28	Byte Array	Reserved field.
Attributes	16	Byte Array	Set of flags describing attributes of the enclave. The ISV can limit what SECS.ATTRIBUTES can be used when loading the enclave through parameters to the SGX Signtool. The Signtool will produce a SIGSTRUCT with ATTRIBUTES and ATTRIBUTESMASK which determine allowed ATTRIBUTES. For each set bit in the SIGSTRUCT's ATTRIBUTESMASK, the corresponding bit in the SECS.ATTRIBUTES must match the same bit in SIGSTRUCT.ATTRIBUTES.
MRENCLAVE	32	Byte Array	Hash of enclave measurement.
Reserved	32	Byte Array	Reserved field.
MRSIGNER	32	Byte Array	Hash of the key used to sign the enclave.
Reserved	96	Byte Array	Reserved field.
ISV ProdID	2	Integer	Enclave Product ID. The ISV should configure a unique ISVProdID for each product that the same enclave signing key. Enclaves that use the same enclave signing key will share the same

			MRSIGNER. If these enclaves don't want to share sealing keys, they should have unique ISVProdIDs.
ISV SVN	2	Integer	Security Version of the enclave.
Reserved	60	Byte Array	Reserved field.
Report Data	64	Byte Array	Additional report data. The enclave is free to provide 64 bytes of custom data to the report. For instance, this space can be used to hold a nonce, a public key, or a hash of a larger block of data. Note that the signature of the report covers the report data field. As a result, the integrity is protected with a key rooted in an Intel CA.

A.3.11. QE Report Certification Data

Name	Size (bytes)	Type	Description
QE Report	384	Enclave Report Body	SGX Report of the *Quoting Enclave* that generated an *Attestation Key*. • *Report Data*: SHA256(*ECDSA Attestation Key* || *QE Authentication Data*) || 32-0x00's Note: The 'QE Report' field in the Quote structure is the value of the QE Report when the PCE certifies it. The Attestation Key certification step may happen before generating a Quote. Therefore, CPUSVN and ISVSVN in this field may be older than the currently loaded QE.
QE Report Signature	64	ECDSA P-256 Signature	ECDSA signature over the *QE Report* calculated using the *Provisioning Certification Key (PCK)*.
QE Authentication Data	Variable	QE Authentication Data	Variable-length data chosen by the *Quoting Enclave* and signed by the *Provisioning Certification Key* (as a part of the *Report Data* in the *QE Report*). It can be used by the *QE* to add additional context to the *Attestation Key* utilized by the *QE*. For example, the authentication data may indicate the customer, geography, network, or anything pertinent to the identity of the QE.

			Size should be set to 0 if there is no additional data.
QE Certification Data	Variable	QE Certification Data – Version 4	Data required to verify the QE Report Signature.

A.3.12. Full TD Quote in v4

The following table summarized the structure of a TD Quote in version 4. The difference to the generic structure presented in beginning of Appendix Version 4 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID) is the variable-length Quote Signature Data field.

Name	Size (bytes)	Type	Description
Quote Header	48	TD Quote Header	Header of *Quote* data structure. This field is ***transparent*** (the user knows its internal structure). Rest of the *Quote* data structure can be treated as ***opaque*** (hidden from the user).
TD Quote Body	584	TD Quote Body	Report of the attested TD. The REPORTDATA contained in this field is defined by the TD developer. See the description of the field for example usages.
Quote Signature Data Len	4	Integer	Size of the Quote Signature Data structure
Quote Signature Data	Variable	ECDSA 256-bit Quote Signature Data Structure – Version 4	Version 4 of the ECDSA 256Bit Signature Data Structure. Contains the TD Quote Body Signature, the TDQE attestation key, Certification Data Type = 6 (QE Report Certification Data)
QE Report Certification Data	Variable	QE Report Certification Data	Contains the TDQE REPORT, the signature over the TDQE REPORT (PCE's signature for TDX 1.0), TDQE Authentication data and the TDQE Certification Data. Certification Data Type = 5 (PCK Cert Chain)
PCK Cert Chain	Variable	PCK Cert Chain	Concatenated PCK Cert Chain (PEM formatted). PCK Leaf Cert||

					Intermediate CA Cert|| Root CA Cert

A.4. Version 5 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID)

This section describes the format of Quotes in version 5. This format is used for TD Quotes, but also applies to SGX Quotes. This version only supports both TDX Module 1.0 and TDX Module 1.5.

Endianess: *Little Endian (applies to all integer fields).*

Name	Size (bytes)	Type	Description
Quote Header	48	TD Quote Header	Header of *Quote* data structure. This field is ***transparent***, i.e., the user knows its internal structure. Rest of the *Quote* data structure can be treated as ***opaque***, i.e., hidden from the user.
TD Quote Body Descriptor	584	TD Quote Body Descriptor	Report of the attested TD. *Quote Body Type* inside the TD Quote Body Descriptor structure defines what specific version of the Quote Body it contains. The REPORTDATA contained in this field is defined by the TD developer. See the description of the field for example usages.
Quote Signature Data Len	4	Integer	Size of the Quote Signature Data structure
Quote Signature Data	Variable	Signature Dependent	Variable-length data containing the signature and supporting data. For instance, an ECDSA P-256 Signature

A.4.1. TD Quote Header

Name	Size (bytes)	Type	Description
Version	2	Integer	Version of the *Quote* data structure. • *Value*: 5
Attestation Key Type	2	Integer	Type of the *Attestation Key* used by the *Quoting Enclave*. • *Supported values*: 2 (ECDSA-256-with-P-256 curve)

			3 (ECDSA-384-with-P-384 curve) (*Note: currently not supported*) (Note: 0 and 1 are reserved, for when EPID is moved to version 4 quotes.)
TEE Type	4	Integer	TEE for this Attestation 0x00000000: SGX 0x00000081: TDX
RESERVED	2	Byte Array	Zero
RESERVED	2	Byte Array	Zero
QE Vendor ID	16	UUID	Unique identifier of the QE Vendor. *Value*: 939A7233F79C4CA9940A0DB3957F0607 (Intel® SGX QE Vendor) *Note: Each vendor that decides to provide a customized Quote data structure should have unique ID.*
User Data	20	Byte Array	Custom user-defined data. For the Intel® SGX and TDX DCAP Quote Generation Libraries, the first 16 bytes contain a Platform Identifier that is used to link a PCK Certificate to an Enc(PPID). This identifier is consistent for every quote generated with this QE on this platform.

A.4.2. TD Quote Body Descriptor

Name	Size (bytes)	Type	Description
TD Quote Body Type	2	Unsigned Integer	Determines type of Quote body (TEE report). Architecturally supported values: - 1 (Future SGX support) - 2 (TD Quote Body for TDX 1.0) - 3 (TD Quote Body for TDX 1.5)
Size	4	Unsigned Integer	Size of *Quote Body* field.
TD Quote Body	Variable	Byte Array	Data conveyed as Quote Body. Its content depends on the value of *Quote Body Type*: - 1: Future SGX support - 2: Byte array that contains TD Report for TDX 1.0.

			- 3: Byte array that contains TD Report for TDX 1.5.

A.4.3. TD Quote Body for TDX 1.0

Name	Size (bytes)	Type	Description
TEE_TCB_SVN	16	Byte Array	Describes the TCB of TDX when the TD was launched.
MRSEAM	48	SHA384	Measurement of the TDX Module
MRSIGNERSEAM	48	SHA384	Zero for the Intel® TDX Module
SEAMATTRIBUTES	8	Byte Array	Must be zero for TDX 1.0
TDATTRIBUTES	8	Byte Array	TD Attributes
XFAM	8	Byte Array	XFAM (eXtended Features Available Mask) is defined as a 64b bitmap, which has the same format as XCR0 or IA32_XSS MSR.
MRTD	48	SHA384	Measurement of the initial contents of the TD. See TDX Module definitions here: TDX Module documentation
MRCONFIGID	48	Byte Array	Software-defined ID for non-owner-defined configuration of the TD, e.g., runtime or OS configuration.
MROWNER	48	Byte Array	Software-defined ID for the TD's owner
MROWNERCONFIG	48	Byte Array	Software-defined ID for owner-defined configuration of the TD, e.g., specific to the workload rather than the runtime or OS.
RTMR0	48	SHA384	Runtime extendable measurement register
RTMR1	48	SHA384	Runtime extendable measurement register
RTMR2	48	SHA384	Runtime extendable measurement register
RTMR3	48	SHA384	Runtime extendable measurement register
REPORTDATA	64	Byte Array	Each TD Quote is based on a TD Report. The TD is free to provide 64 bytes of custom data to a TD Report. For instance, this space can be used to hold a nonce, a public key, or a hash of a larger block of data. Note that the signature of a TD Quote covers the REPORTDATA field. As a result, the

			integrity is protected with a key rooted in an Intel CA.

A.4.4. TD Quote Body for TDX 1.5

Name	Size (bytes)	Type	Description
TEE_TCB_SVN	16	Byte Array	Describes the TCB of TDX when the TD was launched
MRSEAM	48	SHA384	Measurement of the TDX Module
MRSIGNERSEAM	48	SHA384	Zero for the Intel® TDX Module
SEAMATTRIBUTES	8	Byte Array	Must be zero for TDX 1.5
TDATTRIBUTES	8	Byte Array	TD Attributes
XFAM	8	Byte Array	XFAM (eXtended Features Available Mask) is defined as a 64b bitmap, which has the same format as XCR0 or IA32_XSS MSR.
MRTD	48	SHA384	Measurement of the initial contents of the TD. See TDX Module definitions here: TDX Module documentation
MRCONFIGID	48	Byte Array	Software-defined ID for non-owner-defined configuration of the TD, e.g., runtime or OS configuration.
MROWNER	48	Byte Array	Software-defined ID for the TD's owner
MROWNERCONFIG	48	Byte Array	Software-defined ID for owner-defined configuration of the TD, e.g., specific to the workload rather than the runtime or OS.
RTMR0	48	SHA384	Runtime extendable measurement register
RTMR1	48	SHA384	Runtime extendable measurement register
RTMR2	48	SHA384	Runtime extendable measurement register
RTMR3	48	SHA384	Runtime extendable measurement register
REPORTDATA	64	Byte Array	Each TD Quote is based on a TD Report. The TD is free to provide 64 bytes of custom data to a TD Report. For instance, this space can be used to hold a nonce, a public key, or a hash of a larger block of data. Note that the signature of a TD Quote covers the REPORTDATA field. As a result, the integrity is protected with a key rooted in an Intel CA.

TEE_TCB_SVN_2	16	Byte Array	Describes the current TCB of TDX. This value may will be different than TEE_TCB_SVN by loading a new version of the TDX Module using the TD Preserving update capability)
MRSERVICETD	48	SHA384	Measurement of the initial contents of the Migration TD

A.4.5. TEE_TCB_SVN

TEE_TCB_SVN		
Byte location	**Name**	**Description**
0	Tdxtcbcomp01	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[0] Contains the TDX Module minor SVN
1	Tdxtcbcomp02	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[1] Contains the TDX Module major version
2	Tdxtcbcomp03	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[2] Contains the microcode SVN on a non-TD Preserving update of the TDX Module
3	Tdxtcbcomp04	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[3]
4	Tdxtcbcomp05	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[4]
5	Tdxtcbcomp06	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[5]
6	Tdxtcbcomp07	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[6]
7	Tdxtcbcomp08	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[7]
8	Tdxtcbcomp09	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[8]
9	Tdxtcbcomp10	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[9]
10	Tdxtcbcomp11	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[10]

11	Tdxtcbcomp12	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[11]
12	Tdxtcbcomp13	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[12]
13	Tdxtcbcomp14	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[13]
14	Tdxtcbcomp15	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[14]
15	Tdxtcbcomp16	QVL compares with TCBInfo.TCBLevels.tcb.tdxtcbcomponents.svn[15]

A.4.6. TD Attributes

Group	Bits	Description	Bits	Field	Description
TUD	7:0	TD Under Debug flags. If any of the bits in this group are set to 1, the TD is untrusted.	0	DEBUG	Defines whether the TD runs in TD debug mode (set to 1) or not (set to 0). In TD debug mode, the CPU state and private memory are accessible by the host VMM.
			7:1	RESERVED	Reserved for future TUD flags – must be 0
SEC	31:8	Attributes that may impact the security of the TD	27:8	RESERVED	Reserved for future SEC flags – must be 0
			28	SEPT_VE_DISABLE	Disable EPT violation conversion to #VE on TD access of PENDING pages
			29	RESERVED	Reserved for future SEC flags – must be 0
			30	PKS	TD is allowed to use Supervisor Protection Keys.
			31	KL	TD is allowed to use Key Locker.
OTHER	63:32	Attributes that do not impact	62:32	RESERVED	Reserved for future OTHER flags – must be 0

		the security of the TD	63	PERFMON	TD is allowed to use Perfmon and PERF_METRICS capabilities.

A.4.7. ECDSA P-256 Signature

Name	Size (bytes)	Type	Description
Signature	64	Byte Array	ECDSA signature, the r component followed by the s component, 2 x 32 bytes.

A.4.8. ECDSA P-256 Public Key

Name	Size (bytes)	Type	Description
Public Key	64	Byte Array	EC KT-I Public Key, the x-coordinate followed by the y-coordinate (on the RFC 6090 P-256 curve), 2 x 32 bytes.

A.4.9. QE Authentication Data

Name	Size (bytes)	Type	Description
Size	2	Integer	Size of the 'Data' array. 0 is a valid value.
Data	Variable	Byte Array	Data to be additionally 'signed' by the certification key.

A.4.10. ECDSA 256-bit Quote Signature Data Structure – Version 4

Name	Size (bytes)	Type	Description
Quote Signature	64	ECDSA P-256 Signature	ECDSA signature over the *Header* and the *TD Quote Body* calculated using the private part of the *Attestation Key* generated by the *Quoting Enclave*.
ECDSA Attestation Key	64	ECDSA P-256 Public Key	Public part of the *Attestation Key* generated by the *Quoting Enclave*.

QE Certification Data	Variable	QE Certification Data – Version 4	Data required to verify the signature over QE Report and the *Attestation Key*.

A.4.11. QE Certification Data – Version 4 and Version 5

Name	Size (bytes)	Type	Description
Certification Data Type	2	Integer	Determines type of data required to verify the QE Report Signature in the Quote Signature Data structure. *Supported values*: 1 (PCK identifier: PPID in plain text, CPUSVN, and PCESVN) 2 (PCK identifier: PPID encrypted using RSA-2048-OAEP, CPUSVN, and PCESVN) 3 (PCK identifier: PPID encrypted using RSA-3072-OAEP, CPUSVN, and PCESVN) 4 (PCK Leaf Certificate in plain text; currently not supported) 5 (Concatenated PCK Cert Chain) 6 (QE Report Certification Data) 7 (PLATFORM_MANIFEST; currently not supported)
Size	4	Integer	Size of Certification Data field.
Certification Data	Variable	Byte Array	Data required to verify the QE Report Signature depending on the value of the *Certification Data Type*: 1: Byte array that contains concatenation of PPID, CPUSVN, PCESVN (LE), PCEID (LE). 2: Byte array that contains concatenation of PPID encrypted using RSA-2048-OAEP, CPUSVN, PCESVN (LE), PCEID (LE). 3: Byte array that contains concatenation of PPID encrypted using RSA-3072-OAEP, CPUSVN, PCESVN (LE), PCEID (LE). 4: PCK Leaf Certificate

			5: Concatenated PCK Cert Chain (PEM formatted). PCK Leaf Cert|| Intermediate CA Cert|| Root CA Cert 6: QE Report Certification Data 7: PLATFORM_MANIFEST

A.4.12. Enclave Report Body

Name	Size (bytes)	Type	Description
CPU SVN	16	Byte Array	Security Version of HW components in the TDX TCB (raw value).
MISCSELECT	4	Integer	SSA Frame extended feature set. Reports what SECS.MISCSELECT settings are used in the enclave. You can limit the allowed MISCSELECT settings in the sigstruct using MISCSELECT/MISCMASK.
Reserved	28	Byte Array	Reserved field.
Attributes	16	Byte Array	Set of flags describing attributes of the enclave. The ISV can limit what SECS.ATTRIBUTES can be used when loading the enclave through parameters to the SGX Signtool. The Signtool will produce a SIGSTRUCT with ATTRIBUTES and ATTRIBUTESMASK which determine allowed ATTRIBUTES. For each set bit in the SIGSTRUCT's ATTRIBUTESMASK, the corresponding bit in the SECS.ATTRIBUTES must match the same bit in SIGSTRUCT.ATTRIBUTES.
MRENCLAVE	32	Byte Array	Hash of enclave measurement.
Reserved	32	Byte Array	Reserved field.
MRSIGNER	32	Byte Array	Hash of the key used to sign the enclave.
Reserved	96	Byte Array	Reserved field.
ISV ProdID	2	Integer	Enclave Product ID. The ISV should configure a unique ISVProdID for each product that the same enclave signing key. Enclaves that use the same enclave signing key will share the same MRSIGNER. If these enclaves don't want to share sealing keys, they should have unique ISVProdIDs.

ISV SVN	2	Integer	Security Version of the enclave.
Reserved	60	Byte Array	Reserved field.
Report Data	64	Byte Array	Additional report data. The enclave is free to provide 64 bytes of custom data to the report. For instance, this space can be used to hold a nonce, a public key, or a hash of a larger block of data. Note that the signature of the report covers the report data field. As a result, the integrity is protected with a key rooted in an Intel CA.

A.4.13. QE Report Certification Data

Name	Size (bytes)	Type	Description
QE Report	384	Enclave Report Body	SGX Report of the *Quoting Enclave* that generated an *Attestation Key*. • *Report Data*: SHA256(*ECDSA Attestation Key* || *QE Authentication Data*) || 32-0x00's Note: The 'QE Report' field in the Quote structure is the value of the QE Report when the PCE certifies it. The Attestation Key certification step may happen before generating a Quote. Therefore, CPUSVN and ISVSVN in this field may be older than the currently loaded QE.
QE Report Signature	64	ECDSA P-256 Signature	ECDSA signature over the *QE Report* calculated using the *Provisioning Certification Key (PCK)*.
QE Authentication Data	Variable	QE Authentication Data	Variable-length data chosen by the *Quoting Enclave* and signed by the *Provisioning Certification Key* (as a part of the *Report Data* in the *QE Report*). It can be used by the *QE* to add additional context to the *Attestation Key* utilized by the *QE*. For example, the authentication data may indicate the customer, geography, network, or anything pertinent to the identity of the QE. Size should be set to 0 if there is no additional data.

QE Certification Data	Variable	QE Certification Data – Version 4	Data required to verify the QE Report Signature.

A.4.14. Full TD Quote in v5

The following table summarized the structure of a TD Quote in version 4. The difference to the generic structure presented in beginning of Appendix Version 5 Quote Format (TDX-ECDSA, SGX-ECDSA, and SGX-EPID) is the variable-length Quote Signature Data field.

Name	Size (bytes)	Type	Description
Quote Header	48	TD Quote Header	Header of *Quote* data structure. This field is ***transparent*** (the user knows its internal structure). Rest of the *Quote* data structure can be treated as ***opaque*** (hidden from the user).
TD Quote Body Descriptor	584	TD Quote Body Descriptor	Report of the attested TD. The REPORTDATA contained in this field is defined by the TD developer. See the description of the field for example usages.
Quote Signature Data Len	4	Integer	Size of the Quote Signature Data structure
Quote Signature Data	Variable	ECDSA 256-bit Quote Signature Data Structure – Version 4	Version 4 of the ECDSA 256Bit Signature Data Structure. Contains the TD Quote Body Signature, the TDQE attestation key, Certification Data Type = 6 (QE Report Certification Data)
QE Report Certification Data	Variable	QE Report Certification Data	Contains the TDQE REPORT, the signature over the TDQE REPORT (PCE's signature for TDX 1.0), TDQE Authentication data and the TDQE Certification Data. Certification Data Type = 5 (PCK Cert Chain)
PCK Cert Chain	Variable	PCK Cert Chain	Concatenated PCK Cert Chain (PEM formatted). PCK Leaf Cert|| Intermediate CA Cert|| Root CA Cert

B. Result Code Mappings

B.1. Quote Verification Result Mapping (sgx_ql_qv_result_t)

SGX_QL_QV_RESULT_OK	0x0000
SGX_QL_QV_RESULT_CONFIG_NEEDED	0xa001
SGX_QL_QV_RESULT_OUT_OF_DATE	0xa002
SGX_QL_QV_RESULT_OUT_OF_DATE_CONFIG_NEEDED	0xa003
SGX_QL_QV_RESULT_INVALID_SIGNATURE	0xa004
SGX_QL_QV_RESULT_REVOKED	0xa005
SGX_QL_QV_RESULT_UNSPECIFIED	0xa006
SGX_QL_QV_RESULT_SW_HARDENING_NEEDED	0xa007
SGX_QL_QV_RESULT_CONFIG_AND_SW_HARDENING_NEEDED	0xa008

B.2. Quote Generation and Verification Libraries API Return Result Mapping (quote3_error_t)

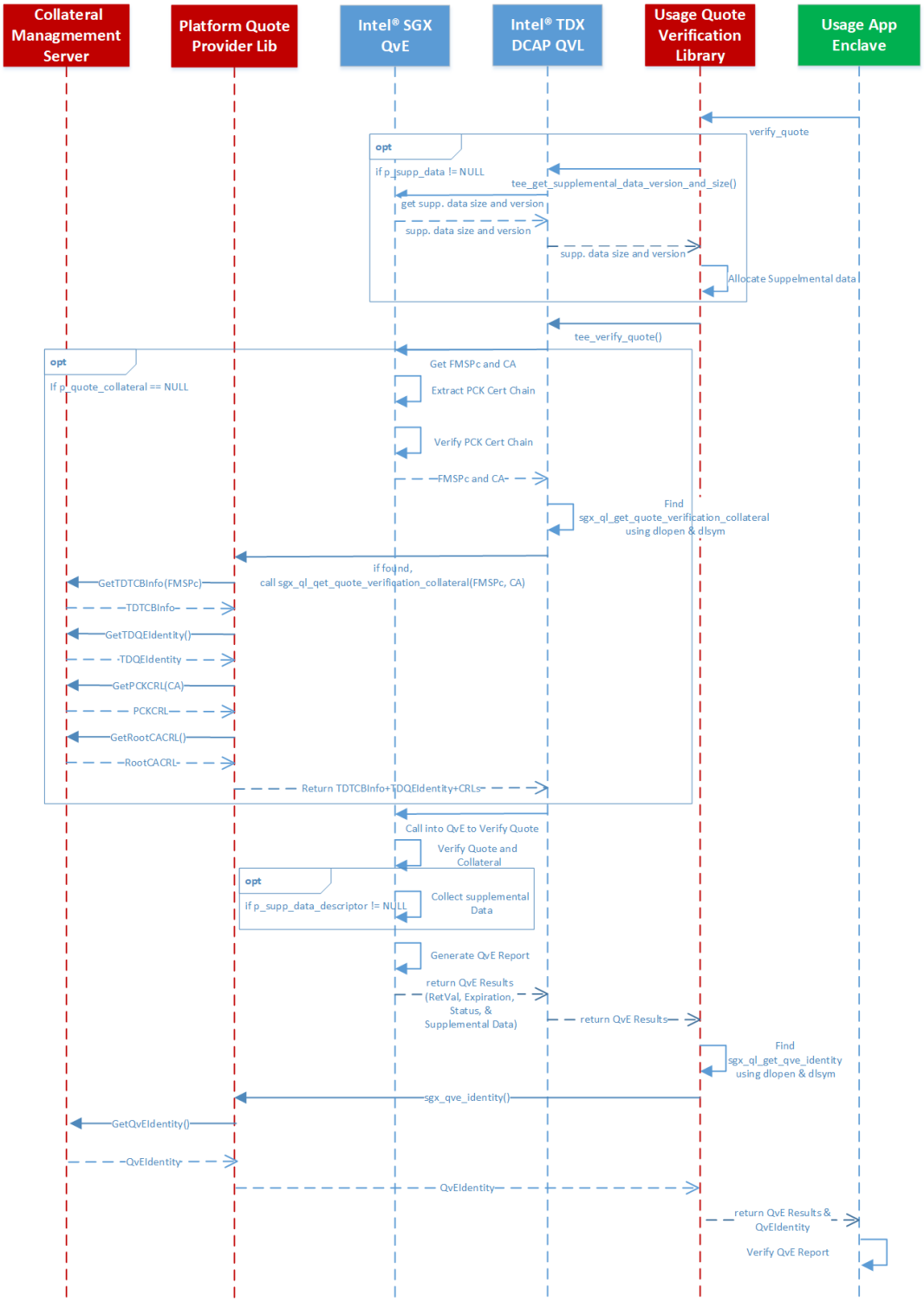
SGX_QL_SUCCESS	0x0000
SGX_QL_ERROR_UNEXPECTED	0xe001
SGX_QL_ERROR_INVALID_PARAMETER	0xe002
SGX_QL_ERROR_OUT_OF_MEMORY	0xe003
SGX_QL_ERROR_ECDSA_ID_MISMATCH	0xe004
SGX_QL_PATHNAME_BUFFER_OVERFLOW_ERROR	0xe005
SGX_QL_FILE_ACCESS_ERROR	0xe006
SGX_QL_ERROR_STORED_KEY	0xe007
SGX_QL_ERROR_PUB_KEY_ID_MISMATCH	0xe008
SGX_QL_ERROR_INVALID_PCE_SIG_SCHEME	0xe009
SGX_QL_ATT_KEY_BLOB_ERROR	0xe00a
SGX_QL_UNSUPPORTED_ATT_KEY_ID	0xe00b
SGX_QL_UNSUPPORTED_LOADING_POLICY	0xe00c
SGX_QL_INTERFACE_UNAVAILABLE	0xe00d
SGX_QL_PLATFORM_LIB_UNAVAILABLE	0xe00e
SGX_QL_ATT_KEY_NOT_INITIALIZED	0xe00f
SGX_QL_ATT_KEY_CERT_DATA_INVALID	0xe010

SGX_QL_NO_PLATFORM_CERT_DATA	0xe011
SGX_QL_OUT_OF_EPC	0xe012
SGX_QL_ERROR_REPORT	0xe013
SGX_QL_ENCLAVE_LOST	0xe014
SGX_QL_INVALID_REPORT	0xe015
SGX_QL_ENCLAVE_LOAD_ERROR	0xe016
SGX_QL_UNABLE_TO_GENERATE_QE_REPORT	0xe017
SGX_QL_KEY_CERTIFCATION_ERROR	0xe018
SGX_QL_NETWORK_ERROR	0xe019
SGX_QL_MESSAGE_ERROR	0xe01a
SGX_QL_NO_QUOTE_COLLATERAL_DATA	0xe01b
SGX_QL_QUOTE_CERTIFICATION_DATA_UNSUPPORTED	0xe01c
SGX_QL_QUOTE_FORMAT_UNSUPPORTED	0xe01d
SGX_QL_UNABLE_TO_GENERATE_REPORT	0xe01e
SGX_QL_QE_REPORT_INVALID_SIGNATURE	0xe01f
SGX_QL_QE_REPORT_UNSUPPORTED_FORMAT	0xe020
SGX_QL_PCK_CERT_UNSUPPORTED_FORMAT	0xe021
SGX_QL_PCK_CERT_CHAIN_ERROR	0xe022
SGX_QL_TCBINFO_UNSUPPORTED_FORMAT	0xe023
SGX_QL_TCBINFO_MISMATCH	0xe024
SGX_QL_QEIDENTITY_UNSUPPORTED_FORMAT	0xe025
SGX_QL_QEIDENTITY_MISMATCH	0xe026
SGX_QL_TCB_OUT_OF_DATE	0xe027
SGX_QL_TCB_OUT_OF_DATE_CONFIGURATION_NEEDED	0xe028
SGX_QL_SGX_ENCLAVE_IDENTITY_OUT_OF_DATE	0xe029
SGX_QL_SGX_ENCLAVE_REPORT_ISVSVN_OUT_OF_DATE	0xe02a
SGX_QL_QE_IDENTITY_OUT_OF_DATE	0xe02b
SGX_QL_SGX_TCB_INFO_EXPIRED	0xe02c
SGX_QL_SGX_PCK_CERT_CHAIN_EXPIRED	0xe02d
SGX_QL_SGX_CRL_EXPIRED	0xe02e
SGX_QL_SGX_SIGNING_CERT_CHAIN_EXPIRED	0xe02f
SGX_QL_SGX_ENCLAVE_IDENTITY_EXPIRED	0xe030

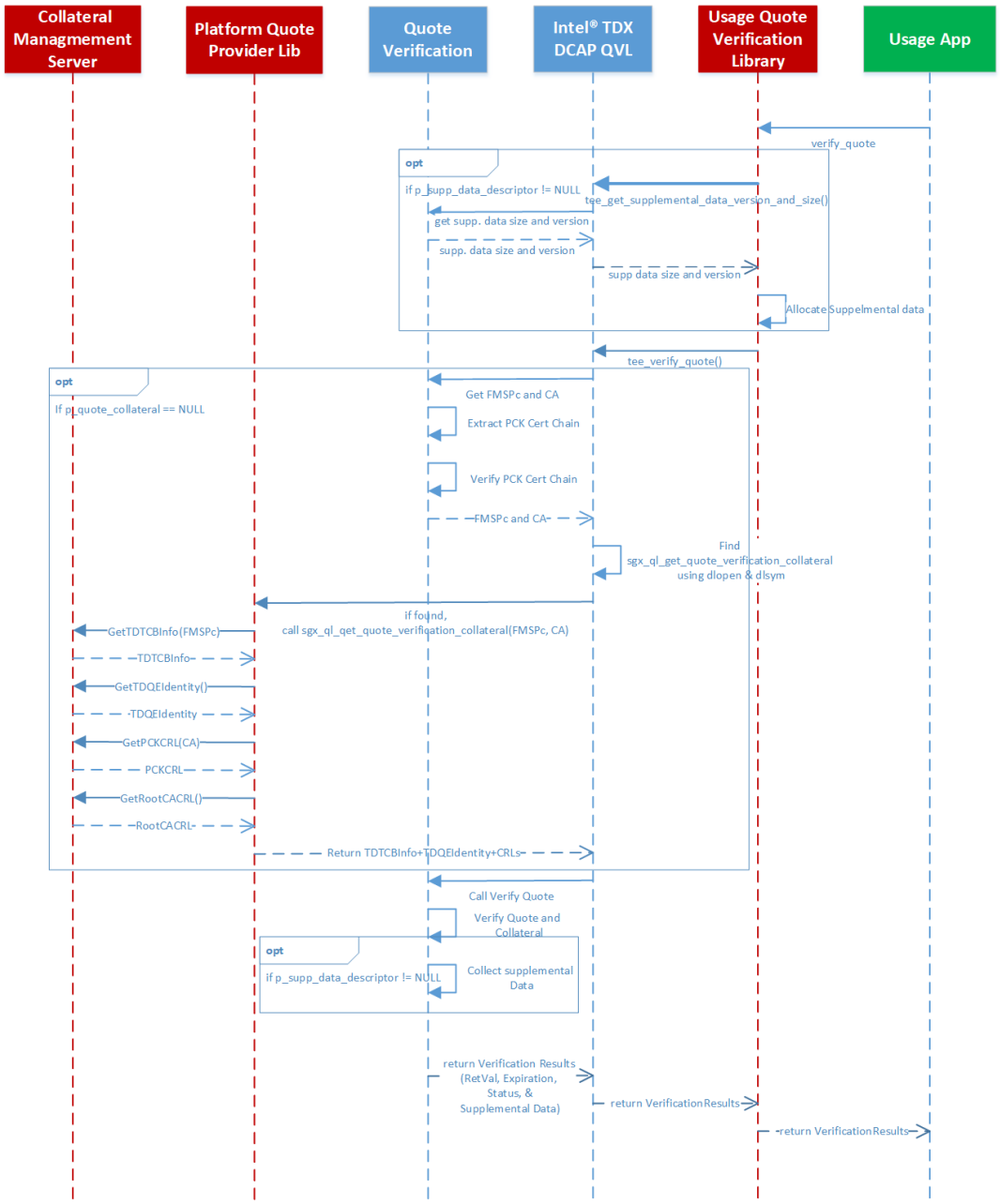
SGX_QL_PCK_REVOKED	0xe031
SGX_QL_TCB_REVOKED	0xe032
SGX_QL_TCB_CONFIGURATION_NEEDED	0xe033
SGX_QL_UNABLE_TO_GET_COLLATERAL	0xe034
SGX_QL_ERROR_INVALID_PRIVILEGE	0xe035
SGX_QL_NO_QVE_IDENTITY_DATA	0xe037
SGX_QL_CRL_UNSUPPORTED_FORMAT	0xe038
SGX_QL_QEIDENTITY_CHAIN_ERROR	0xe039
SGX_QL_TCBINFO_CHAIN_ERROR	0xe03a
SGX_QL_ERROR_QVL_QVE_MISMATCH	0xe03b
SGX_QL_TCB_SW_HARDENING_NEEDED	0xe03c
SGX_QL_TCB_CONFIGURATION_AND_SW_HARDENING_NEEDED	0xe03d
SGX_QL_UNSUPPORTED_MODE	0xe03e
SGX_QL_NO_DEVICE	0xe03f
SGX_QL_SERVICE_UNAVAILABLE	0xe040
SGX_QL_NETWORK_FAILURE	0xe041
SGX_QL_SERVICE_TIMEOUT	0xe042
SGX_QL_ERROR_BUSY	0xe043
SGX_QL_UNKNOWN_MESSAGE_RESPONSE	0xe044
SGX_QL_PERSISTENT_STORAGE_ERROR	0xe045
SGX_QL_ERROR_MESSAGE_PARSING_ERROR	0xe046
SGX_QL_PLATFORM_UNKNOWN	0xe047
SGX_QL_QVEIDENTITY_MISMATCH	0xe050
SGX_QL_QVE_OUT_OF_DATE	0xe051
SGX_QL_PSW_NOT_AVAILABLE	0xe052
SGX_QL_COLLATERAL_VERSION_NOT_SUPPORTED	0xe053
SGX_QL_TDX_MODULE_MISMATCH	0xe060
SGX_QL_QEIDENTITY_NOT_FOUND	0xe061
SGX_QL_TCBINFO_NOT_FOUND	0xe062
SGX_QL_INTERNAL_SERVER_ERROR	0xe063
SGX_QL_SUPPLEMENTAL_DATA_VERSION_NOT_SUPPORTED	0xe064
SGX_QL_ROOT_CA_UNTRUSTED	0xe065

C. Sample Sequence Diagrams

C.1. QvE-based Quote Verification



C.2. Non-QvE Based Quote Verification



D. Disclaimer and Legal Information

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Intel technologies features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at Intel.com, or from the OEM or retailer.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, Xeon, and Xeon Phi are trademarks of Intel Corporation in the U.S. and/or other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

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