Intel® Software Guard Extensions (Intel® SGX) Data Center Attestation Primitives: ECDSA Quote Library API

Rev Production
May, 2025

Intel® Software Guard Extensions (Intel® SGX) Data Center Attestation Primitives: ECDSA Quote Library API

Table of Contents

1. Introduction

Attestation is a process of demonstrating that a software executable has been properly instantiated on a platform. The Intel® Software Guard Extensions (Intel® SGX) attestation allows a remote party to ensure that a particular software is securely running within an enclave on an Intel® SGX enabled platform.

This specification describes the API surface for the libraries that allows the software to both generate an attestation evidence for an Intel® SGX enclave of an application and to verify that attestation evidence. The Intel® Software Guard Extensions Data Center Attestation Primitives (Intel® SGX DCAP) version of the libraries generate the attestation evidence using an ECDSA Attestation Key to sign an identity Report of an Intel® SGX enclave of an application. The signed Report is called an attestation Quote. The ECDSA Attestation key is created and owned by the owner of the remote attestation infrastructure but is certified by an Intel® SGX rooted key whose certificate is distributed by Intel®. The Intel® SGX rooted certificate proves that the platform running the Intel® SGX enclave is valid and in good standing.

1.1.Terminology

SGX Quote	Data structure used to provide proof to an off-platform entity that an application enclave is running with Intel® SGX protections on a trusted Intel® SGX enabled platform.
Report (EREPORT)	Hardware report generated by the Intel® SGX HW that provides identity and measurement information of the enclave and the platform. It can be MAC'd with a key available to another enclave on the same platform.
Quoting Enclave (QE)	Intel signed enclave that is trusted by the attestation infrastructure owner to sign and issue Quotes or attestations about other enclaves.
Quote Verification Enclave (QvE)	Intel signed enclave that is trusted by the attestation infrastructure owner to verify Intel generated Quotes.
Elliptic Curve Digital Signature Algorithm (ECDSA)	Signing cryptographic algorithm as described in FIPS 186-4.
Attestation Key (AK)	Key used by the Quoting Enclave (QE) to sign Quotes that describe the measurements and identity of an application enclave.
Provisioning Certification Enclave (PCE)	Intel® SGX architectural enclave that uses a Provisioning Certification Key (PCK) to sign QE REPORT structures for Provisioning or Quoting Enclaves. These signed REPORTS contain the ReportData indicating that attestation keys or provisioning protocol messages are created on genuine hardware.
Provisioning Certification Key (PCK)	Signing key available to the Provisioning Certification Enclave for signing certificate-like QE REPORT structures.

	The key is unique to a processor package or platform instance, the HW TCB, and the PCE version (PSVN).
Provisioning Certification Key Certificate (PCK Cert)	The x.509 Certificate chain signed and distributed by Intel for every Intel® SGX enabled platform. This certificate is used by Quote verifiers to verify that the QE generating quotes is valid and running on a trusted Intel® SGX platform at a particular PSVN. It matches the private key generated by the PCE.
Platform Provisioning ID (PPID)	Provisioning ID for a processor package or platform instance. PPID is not TCB dependent.
Security Version Number (SVN)	Version number that indicates when security relevant updates occurred. New versions can have increased functional versions without incrementing the SVN.
Platform Security Version Numbers (PSVN)	Set of SVNs for all components in the Intel® SGX attestation Trusted Computing Base (TCB) including the PCE SVN.
Enclave Page Cache (EPC)	Amount of memory on the platform allocated to enclave code and data storage.
Intel® SGX Provisioning TCB	Trusted Computing Base of Intel® SGX provisioning that includes the platform HW TCB and the PCE SVN.
PCEID	Identifies the version of the PCE used to generate the PPID and PCK signing key.
Intel® SGX DCAP	Intel® Software Guard Extensions Data Center Attestation Primitives
LE	Launch Enclave. Generates the launch token needed to load and initialize another enclave. The LE does not need a launch token to load but its signing key (MRSIGNER) must match the CPU configuration. See more in the Launch Control documents.
FLC	Flexible Launch Control. An Intel® SGX feature that allows arbitrary LE to generate Launch Tokens. The default Launch Control Policy adheres Intel® SGX client based Launch Policy List. FLC exposes a set of MSRs that allow a platform owner to change the default LE MRSINGER to a different MRSIGNER to enable LE to generate Launch Tokens. Not all platforms or all BIOSs support FLC.

Table 1-1: Terminology

2. Overview

Before an application enclave can be trusted by an off-platform entity, the application must prove that its enclave is running with Intel® SGX protections on an Intel® SGX platform in good standing. Once trusted, the off-platform entity or a relying party can provide secrets or trusted services. Each enclave can generate a hardware rooted identity REPORT MAC'd with a symmetric key that another enclave on the same platform can then verify. This is called an Intel® SGX Report based local attestation. This REPORT can then be verified and signed by an asymmetric private key owned by a special enclave called the Quoting Enclave (QE). The QE is running on the same platform as the application enclave. The resulting data structure is called a Quote and the asymmetric signing key is called an attestation key. Any relying parties that have access to a public portion of the attestation key can check the Quote signature, the application enclave identity and the TCB of the platform to establish trust in the application enclave.

Intel will develop a libraries for the Linux* OS based software that will generate quotes for application enclaves as well as verify those quotes for a verifier. These libraries will not depend on any specific platform software, such as the Intel® SGX PSW, but will rely on a set of APIs provided by the environment in which the library runs. This will allow the libraries to load the Intel signed enclaves required to generate the quotes and to verify the quotes. This allows the libraries to be designed and distributed to work in different environments. For example, they can be linked into the Intel® SGX PSW AESM or they can exist in another system service. They can also be linked as a part of an application allowing them to run in the application process. See section Quote Library Dependent APIs for the dependent system APIs.

2.1. Intel® SGX ECDSA Quote Generation Library

The ECDSA Quoting library contains an ECDSA-based Quoting Enclave (QE) that uses a FIPS 186-4 and RFC 6090 compliant algorithm to generate a 256 bit ECC signing key. The key is on the p256 curve. The QE is developed and signed by Intel.

The ECDSA attestation key generated by the QE needs to be certified by an Intel® SGX key rooted to the platform HW fuses. Intel develops and signs an enclave called the Provisioning Certification Enclave (PCE). The key generated by the PCE to certify (sign) attestation keys is rooted to the CPU HW fuses. This key is called the Provisioning Certification Key (PCK) private key. Intel will also generate and publish a public key that matches the signing key (PCK) generated by the PCE. The public key is published as an X.509 certificate format called the Provision Certification Key Certificate (PCK Cert). The PCE will provide an interface to retrieve the PCK Certificate identifier (EncPPID+TCB+PCEID) used by a verifier to find the matching PCK Cert. The PCE also provides a mechanism to sign another enclave (i.e. QE) REPORT using the PCK private key. For Intel® SGX DCAP, the QE will generate the ECDSA Attestation Key (AK) and include a hash of the AK in the QE.REPORT.ReportData. Only the PCE can produce the PCK private key. This PCE certification data will ultimately be embedded in the ECDSA Quote generated by the QE. The AK is then used to signed application enclave Reports to prove that the enclave is running with Intel® SGX protections at a given TCB. This is called the ECDSA Quote. The Attestation infrastructure owner can verify the ECDSA attestation key using the PCK Certificate. The Intel® SGX DCAP ECDSA Quoting Library described in this document will be shipped with the PCE library and will use the PCE APIs internally. The applications will use the APIs described in this document to generate Quotes for its enclave.

2.2. Intel® SGX ECDSA Quoting Verification Library Overview

The Intel® SGX ECDSA Quote Verification Library contains a Quote Verification Enclave (QvE) that can verify the Quote generated by the ECDSA-based Quoting Enclave (QE). The QvE is developed and signed by Intel.

The Intel SGX ECDSA Quote Verification Library also supports quote verification without using the QvE. But the results cannot be cryptographically verified. This model supports quote verification on a non-SGX platform.

The Intel® SGX ECDSA Quote Verification Library may be wrapped by a ‘usage’ library to meet the requirements for a particular usage. These usages may be for Intel® SGX DCAP or the Intel® SGX AESM. In those cases, the library released may need to be dynamically or statically linked by the ‘usage’. The applications will use the APIs described in this document to verify Quotes generated for an enclave.

3. Intel® SGX DCAP Quote Libraries

3.1. Quote Generation Library API's

This chapter presents a set of C-like APIs that allow applications to request an ECDSA Quote. The Intel® SGX DCAP usage exposes a set of quote generation APIs that simplify the quoting interface to support a single ECDSA attestation key specific to that platform.

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

3.1.1. Process Model

There are 2 process modes available for Quote Generation Library. The default mode is in-process mode where the Quote Generation Library and its dependencies will be loaded into the application's process. In this mode, the application can use the enclave load policies described in Set Enclave Load Policy and Cleanup Enclaves by Policy. Another mode is the out-of-process mode. To use this mode, users need to create an environment variable, SGX_AESM_ADDR, before loading the Quote Generation Library to switch to out-of-process mode. In this mode, the SGX AESM service installed with the Intel SGX Platform Software will manage the loading and unloading of QE and PCE. As a result, APIs related to the enclave load policy described in 3.1.2 and Cleanup Enclaves by Policy are not available in the out-of-process mode. Multiple applications that use Quote Generation Library in out-of-process mode share one instance of QE and PCE in memory. To switch between these 2 modes, users need to reload the Quote Generation Library.

3.1.2. Set Enclave Load Policy

When the Quoting Library is linked to a process, it needs 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 enclave interfaces are available upon quote requests and not subject to Intel® SGX memory (EPC) limitations when loaded on demand. However, if the Quoting library is linked within an application process, there may be many applications with the Quoting library and a better utilization of EPC is to load and unload the enclaves on demand (ephemeral). The library will be shipped with a default policy of loading enclaves and leaving them loaded until the library is unloaded (SGX_QL_PERSISTENT).

If the policy is set to SGX_QL_EPHEMERAL, then the QE and PCE are loaded and unloaded on demand. If an enclave is already loaded when the policy is changed to SGX_QL_EPHEMERAL, the enclaves are unloaded before returning.

This function only works when the Quote Generation Library is linked into the application process. If the platform is configured to use the out-of-process implementation of quote generation (i.e. the environment variable "SGX_AESM_ADDR" is set), the API will return SGX_QL_UNSUPPORTED_MODE.

Syntax

```
quote3_error_t sgx_qe_set_enclave_load_policy(sgx_ql_request_policy_t policy);
```

Parameters

policy[In]

Sets the requested enclave loading policy to SGX_QL_PERSISTENT, SGX_QL_EPHEMERAL, or SGX_QL_DEFAULT.

Return Values

SGX_QL_SUCCESS:

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

SGX_QL_UNSUPPORTED_LOADING_POLICY:

Selected policy is not supported by the quoting library.

SGX_QL_ERROR_UNEXPECTED:

Unexpected error occurred.

SGX_QL_UNSUPPORTED_MODE:

The platform has been configured to use the out-of-process implementation of quote generation.

3.1.3. Get QE Target Info

Description

This API allows the calling code to retrieve the target info of the QE. The loading of the QE and the PCE follows the selected loading policy. The application enclave uses the returned QE target info when generating its Report.

During this API execution, the Quoting Library generates and certifies the attestation key. The key and certification data is stored in process memory for the sgx_qe_get_quote_size() and sgx_qe_get_quote() APIs to use. Generating and certifying the keys at this point make the following APIs more efficient. If the following APIs return the SGX_QL_ATT_KEY_NOT_INITIALIZED error, this API needs to be called again to regenerate and recertify the key.

Syntax

```
quote3_error_t sgx_qe_get_target_info(sgx_target_info_t *p_target_info);
```

Parameters

p_target_info [Out]

Pointer to the buffer that contains the QE target information. This is used by an application enclave to generate a REPORT verifiable by the QE. Must not be NULL.

Return Values

SGX_QL_SUCCESS:

Retrieved the p_target_info.

SGX_QL_ERROR_INVALID_PARAMETER:

p_target_info must not be NULL.

SGX_QL_ERROR_UNEXPECTED:
Unexpected internal error occurred.

SGX_QL_ENCLAVE_LOAD_ERROR:
Unable to load the enclaves required to initialize the attestation key. Could be due to file I/O error or some other loading infrastructure errors.

SGX_QL_OUT_OF_MEMORY:
Heap memory allocation error occurred in a library or an enclave.

SGX_QL_ERROR_OUT_OF_EPC:
Not enough EPC memory to load one of the enclaves needed to complete this operation.

SGX_QL_ATTESTATION_KEY_CERTIFCATION_ERROR:
Failed to generate and certify the attestation key. Typically, this may happen if the TCB used to request PCE signing is higher than the platform TCB.

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

SGX_QL_NO_PLATFORM_CERT_DATA:
The platform quote provider library doesn't have the platform certification data for this platform.

SGX_QL_NO_DEVICE:
Can't open SGX device. This error happens only when running in out-of-process mode.

SGX_QL_SERVICE_UNAVAILABLE:
Indicates AESM didn't respond or the requested service is not supported. This error happens only when running in out-of-process mode.

SGX_QL_NETWORK_FAILURE:
Network connection or proxy setting issue is encountered. This error happens only when running in out-of-process mode.

SGX_QL_SERVICE_TIMEOUT:
The request to out-of-process service has timed out. This error happens only when running in out-of-process mode.

SGX_QL_ERROR_BUSY:
The requested service is temporarily not available. This error happens only when running in out-of-process mode.

SGX_QL_UNSUPPORTED_ATT_KEY_ID:
Unsupported attestation key ID.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:
Unexpected error from the attestation infrastructure while retrieving the platform data.

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

SGX_QL_PLATFORM_UNKNOWN
This platform is an unrecognized SGX platform.

3.1.4. Get Quote Size

The application needs to call this API before generating a quote. The quote size varies depending on the type of certification data used to describe how the ECDSA AK is certified. Once the application calls this API, it uses the returned `p_quote_size` in bytes to allocate a buffer to hold the quote. A pointer to this allocated buffer is provided to the `sgx_qe_get_quote()` API.

If the key is not available, this API returns an error (SGX_QL_ATT_KEY_NOT_INITIALIZED). In this case, you must call `sgx_qe_get_target_info()` to re-generate and re-certify the attestation key.

The size returned in this API indicates the size of the quote buffer required in the `sgx_qe_get_quote()` API.

Syntax

```
quote3_error_t sgx_qe_get_quote_size(
                     uint32_t *p_quote_size)
```

Parameters

p_quote_size[Out]:

Pointer to the size of the buffer in bytes required to contain the full quote. This value is passed in to the `sgx_qe_get_quote()` API. You need to allocate a buffer large enough to contain the quote.

Return Values

SGX_QL_SUCCESS:

Successfully calculated the required quote size. The required size in bytes is returned in the memory pointed to by p_quote_size.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error occurred.

SGX_QL_ERROR_INVALID_PARAMETER:

Invalid parameter. `p_quote_size` must not be NULL.

SGX_QL_ATT_KEY_NOT_INITIALIZED:

Platform quoting infrastructure does not have the attestation key available to generate quotes. Call `sgx_qe_get_target_info()` again.

SGX_QL_ATT_KEY_CERT_DATA_INVALID:

Data returned by the platform quote provider library's `sgx_ql_get_quote_config()` is invalid (see section Platform Quote Provider Library).

SGX_QL_ERROR_OUT_OF_EPC:

Not enough EPC memory to load one of the quote library enclaves needed to complete this operation.

SGX_QL_OUT_OF_MEMORY:

Heap memory allocation error occurred in a library or an enclave.

SGX_QL_ENCLAVE_LOAD_ERROR:
Unable to load one of the quote library enclaves required to initialize the attestation key. Could be due to file I/O error or some other loading infrastructure errors.

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

SGX_QL_ATT_KEY_CERT_DATA_INVALID:
Certification data retrieved from the platform quote provider library is invalid.

SGX_QL_NO_PLATFORM_CERT_DATA:
The platform quote provider library doesn't have the platform certification data for this platform.

SGX_QL_NO_DEVICE:
Can't open SGX device. This error happens only when running in out-of-process mode.

SGX_QL_SERVICE_UNAVAILABLE:
Indicates AESM didn't respond or the requested service is not supported. This error happens only when running in out-of-process mode.

SGX_QL_NETWORK_FAILURE:
Network connection or proxy setting issue is encountered. This error happens only when running in out-of-process mode.

SGX_QL_SERVICE_TIMEOUT:
The request to out-of-process service has timed out. This error happens only when running in out-of-process mode.

SGX_QL_ERROR_BUSY:
The requested service is temporarily not available. This error happens only when running in out-of-process mode.

SGX_QL_UNSUPPORTED_ATT_KEY_ID:
Unsupported attestation key ID.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:
Unexpected error from the attestation infrastructure while retrieving the platform data.

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

SGX_QL_PLATFORM_UNKNOWN
This platform is an unrecognized SGX platform.

3.1.5. Get Quote

Description

Finally, the application calls this API to generate a quote. The function takes the application enclave REPORT as input and converts it into a quote once the QE verifies the REPORT. Once verified, it signs it with the ECDSA AK of the Intel® SGX DCAP QE. If the key is not available, this API returns an error (SGX_QL_ATT_KEY_NOT_INITIALIZED). In this case, call `sgx_qe_get_target_info`() to re-generate and re-certify the attestation key.

For Intel® SGX DCAP, the Quote.Header.UserData[0..15] (see Quote Format) contains the 128bit platform identifier (QE_ID) based on the QE Seal Key at TCB 0 (see QE ID Derivation). This allows the attestation infrastructure to link a quote generated on the platform with the platform PCK Cert.

To allow the application to remain agnostic to the type of the attestation key used generate the quote, the application should not try to parse the quote.

Syntax

```
quote3_error_t sgx_qe_get_quote(
                    const sgx_report_t *p_app_report,
                    uint32_t quote_size
                    uint8_t *p_quote);
```

Parameters

p_app_report [In]

Pointer to the application enclave REPORT that requires a quote. The report needs to be generated using the QE target info returned by the sgx_qe_get_target_info() API. Must not be NULL.

quote_size [In]

Size of the buffer that p_quote points to (in bytes).

p_quote [Out]

Pointer to the buffer that will contain the generated quote. Must not be NULL.

Return Values

SGX_QL_SUCCESS:

Successfully generated the quote.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error occurred.

SGX_QL_ERROR_INVALID_PARAMETER:

Invalid parameter.

SGX_QL_ATT_KEY_NOT_INITIALIZED:

Platform quoting infrastructure does not have the attestation key available to generate quotes. Call init_quote() again.

SGX_QL_ATT_KEY_CERT_DATA_INVALID:

Data returned by the platform quote provider library's sgx_ql_get_quote_config() is invalid.

SGX_QL_ERROR_OUT_OF_EPC:

Not enough EPC memory to load one of the Architecture Enclaves needed to complete this operation.

SGX_QL_OUT_OF_MEMORY:

Heap memory allocation error occurred in a library or an enclave.

SGX_QL_ENCLAVE_LOAD_ERROR:

Unable to load the enclaves required to initialize the attestation key. Could be due to file I/O error or some other loading infrastructure errors.

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

SGX_QL_INVALID_REPORT:
Report MAC check failed on an application report.

SGX_QL_NO_PLATFORM_CERT_DATA:
The platform quote provider library doesn't have the platform certification data for this platform.

SGX_QL_NO_DEVICE:
Can't open SGX device. This error happens only when running in out-of-process mode.

SGX_QL_SERVICE_UNAVAILABLE:
Indicates AESM didn't respond or the requested service is not supported. This error happens only when running in out-of-process mode.

SGX_QL_NETWORK_FAILURE:
Network connection or proxy setting issue is encountered. This error happens only when running in out-of-process mode.

SGX_QL_SERVICE_TIMEOUT:
The request to out-of-process service has timed out. This error happens only when running in out-of-process mode.

SGX_QL_ERROR_BUSY:
The requested service is temporarily not available. This error happens only when running in out-of-process mode.

SGX_QL_UNSUPPORTED_ATT_KEY_ID:
Unsupported attestation key ID.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:
Unexpected error from the attestation infrastructure while retrieving the platform data.

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

SGX_QL_PLATFORM_UNKNOWN
This platform is an unrecognized SGX platform.

3.1.6. Cleanup Enclaves by Policy

Description

This method is primarily a hint for the Quote library that it can release the QE and the PCE it cached for efficiency. In the mainline case, sgx_qe_get_targetinfo(), sgx_qe_get_quote_size(), and sgx_qe_get_quote() are called in succession. If the Quote library keeps the enclaves loaded between sgx_qe_get_targetinfo() and sgx_qe_get_quote_size(), they may not be unloaded if the process using the quote library fails prior to sgx_qe_get_quote(). sgx_cleanup_qe_by_policy() informs the Quote Library that it should clean up the QE and the PCE since it cannot depend on the sgx_qe_get_quote() to unload them. If SGX_QE_PERSISTENT is the default policy, it can choose to no-op.

This function only works when the Quote Generation Library is linked into the application process. If the platform is configured to use the out-of-process implementation of quote generation (i.e. the environment variable "SGX_AESM_ADDR" is set), the API will return SGX_QL_UNSUPPORTED_MODE.

Syntax

```
quote3_error_t sgx_qe_cleanup_by_policy();
```

Parameters

None

Return Values

SGX_QL_SUCCESS:

Successfully completed.

SGX_QL_UNSUPPORTED_MODE:

The platform has been configured to use the out-of-process implementation of quote generation.

3.1.7. Set Quote Generation Enclave and Dependent Library Directory Paths

Description

This API can be used to set the location and filename of SGX ECDSA Quote Enclave (QE3), the SGX Provisioning Certification Enclave (PCE) and the Platform Quote Provider Library (QPL) library (see Platform Quote Provider Library). The function takes the path_type ENUM and the corresponding directory path plus filename as input. The user can change the default directory path and filename used by the Quote Generation Library to find the enclaves and the QPL independently with this API.

If this API is not called, it will load the PCE and QE in the local directory of the process and use the dlopen search path for the platform quote provider library (see Platform Quote Provider Library).

This function only works when the Quote Generation Library is linked into the application process. If the platform is configured to use the out-of-process implementation of quote generation (i.e. the environment variable "SGX_AESM_ADDR" is set), the API will return SGX_QL_UNSUPPORTED_MODE.

Syntax

```
quote3_error_t sgx_ql_set_path(
                   sgx_ql_path_type_t path_type,
                   const char *p_path);
```

Parameters

path_type [In]

The entity whose directory path and filename is specified in p_path. It can be SGX_QL_QE3_PATH, SGX_QL_PCE_PATH, or SGX_QL_QPL_PATH

p_path [In]

The directory path and filename of the entity specified in path_type specified as a NUL terminated string.

Return Values

SGX_QL_SUCCESS:

Successfully completed.

SGX_QL_ERROR_INVALID_PARAMETER:

Invalid parameter.

SGX_QL_UNSUPPORTED_MODE:

The platform has been configured to use the out-of-process implementation of quote generation.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error occurred.

3.1.8. Register a Logging Function with SGX and DCAP Libraries

Description

The Quoting Library provides a function to register a logging function. This allows a process with logging capability to register its logging function with all SGX and DCAP libraries (except Platform Quote Provider Library). This allows the Quoting Library writer to utilize the registered logging function to output debug logs of SGX and DCAP libraries.

Syntax

```
quote3_error_t sgx_ql_set_trace_callback(
                            sgx_ql_logging_callback_t logger,
                            sgx_ql_log_level_t loglevel);
```

Parameters

logger [In]

Function pointer to the logging function. If NULL, default logging function is used.

It must have this prototype:

```
typedef void (*sgx_ql_logging_callback_t)(
                            sgx_ql_log_level_t level,
                             const char* message);
```

The logging function takes the string ‘message’ and a logging ‘level’ defined as:

```
typedef enum _sgx_ql_log_level_t
{
    SGX_QL_LOG_ERROR,
   SGX_QL_LOG_INFO,
    SGX_QL_LOG_DEBUG,
    SGX_QL_LOG_TRACE,
} sgx_ql_log_level_t;
```

loglevel [In]

Log level.

Return Values

SGX_QL_SUCCESS:

The logging function was successfully registered.

SGX_QL_ERROR_INVALID_PARAMETER:

The logger parameter was NULL.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

3.2. Enclave Loading

The Quote libraries load and unload the Intel-signed and formatted QE, PCE, and QVE enclaves as needed and as specified by the enclave loading policy. The Quote Libraries load the QE, PCE and QVE using a library called the modified URTS (Untrusted Runtime Service) exposing APIs that are compatible with the enclave loading APIs exposed by the Intel® SGX SDK. The modified URTS library is shipped with the Quote Libraries or has it statically linked. The modified URTS library uses the Intel® Enclave Common Abstraction Layer Library (see 'Enclave Common Loader API Reference' document for API descriptions).

3.2.1. Enclave Launch Policy Implications

To use a Quoting Library that supports ECDSA quote generation, the platform that runs the Quote Library must support Flexible Launch Control (FLC). FLC allows the platform owner to choose which Launch Enclave (LE) can generate launch tokens or whether enclaves need launch tokens at all. Using an LE allows the platform owner to enforce the enclave launch control policy by limiting which enclaves can launch. Other platform owners may allow enclaves to launch without a launch token and therefore will not have an LE. FLC is not available on all platforms. For platforms that support FLC, BIOS must also support FLC.

Some environments may inherently allow the PCE and the QE based on the enclave MRSIGNER (hash of the enclave signing key) and the <ProvisionKey> attribute. For example, the out-of-tree Intel® SGX DCAP driver only allows the Intel-signed PCE and QE enclaves to launch with the <ProvisionKey> attribute set to 1 by default to maintain legacy behavior. In this case, launching any other enclave with the <ProvisionKey> attribute set to 1, the application process launching the enclave must request permission from the OS. For the current DCAP driver, the application process or user running the process must get file access permissions to the 'sgx_prv' group. For example:

```
$ sudo usermod -a -G sgx_prv <username>
```

Future versions of the DCAP driver will drop this legacy behavior and all process that use the quote generation library will need to file access permissions to the 'sgx-prv' group as well as any user loading their own <ProvisionKey> enclaves.

Also note that the Intel signed PCE will not provide certification information or Report signing to enclaves with the <ProvisionKey> attribute to 0.

3.3. Quote Library Dependent APIs

The Quoting library looks for these APIs when needed and expects them to be available from a library dynamically linked with the Quoting Library.

3.3.1. Platform Quote Provider Library

The platform quote provider library provides a set of APIs that allow the Quote libraries to get platform specific services. They are not required for the Quote Libraries to function but they may be required to properly generate quotes in a given platform environment.

The Quote libraries looks for a library named ***libdcap_quoteprov.so*** using dlopen search path during runtime. The Quote generation library does not require the platform quote provider library to generate Quotes but the generated Quotes may not be verifiable. The Quote verification library does require the platform quote provider library if the verification collateral is not passed in to the Verify Quote API.

Both the Quote generation library and the Quote verification library provide APIs to override the default platform quote provider library's directory path and file name. (See Set Quote Generation Enclave and Dependent Library Directory Paths and Set Quote Verification Enclave and Dependent Library Directory Paths).

3.3.1.1. Initialize and Cleanup the platform quote provider library

Description

The sgx_qpl_global_init() function sets up the program environment that libdcap_quoteprov needs. Think of it as an extension of the library loader.

This function must be called at least once within a program before the program calls any other function in libdcap_quoteprov. The environment it sets up is constant for the life of the program and is the same for every program, so multiple calls have the same effect as one call.

The sgx_qpl_global_cleanup function releases resources acquired by sgx_qpl_global_init. You should call it after you are done using libdcap_quoteprov.

Important: Old versions of libdcap_quoteprov do not include these two APIs. To maintain backward compatibility, it is necessary to verify the existence of these APIs before calling them. Only call the APIs if they are present to ensure proper functionality.

Syntax

```
quote3_error_t sgx_qpl_global_init();
```

Return Values

SGX_QL_SUCCESS:

The libdcap_quoteprov library was initialized successfully and you can call other functions afterward.

SGX_QCNL_CONFIG_INVALID_JSON:

The config file (sgx_default_qcnl.conf) is in JSON format but has a format error.

Syntax

```
quote3_error_t sgx_qpl_global_cleanup();
```

Return Values

SGX_QL_SUCCESS:

The libdcap_quoteprov library was cleaned up successfully.

3.3.1.2. Get PCK Certification Information

Description

For ECDSA quote generation, the Quote generation library by default generates an attestation key certified by the PCE using the raw Intel® SGX TCB of the platform. This may not work for all attestation environments. The TCB used to generate the PCE signature over the ECDSA AK needs a matching Intel generated x.509 PCK Certificate for that TCB. This is problematic for Xeon-E and client platforms that do not have Intel Generated PCK Certs for all TCB levels. It also causes problems when the attestation infrastructure caches PCK Certs and may not have certs for all platforms due to restrictions on contacting Intel hosted services during runtime. In these cases, the Quote Library needs to get the TCB from the platform software to generate the proper PCE signature. The Quote Library first requests the TCB information from the platform quote provider library if it is available and submits that value for PCE signing. In addition, the platform quote provider library responds with the associated Certification Data to append to the ECDSA Quote. See the definitions for `sgx_ql_pck_cert_id_t` and `sgx_ql_config_t`.

If the platform quote provider library cannot be found, the sgx_ql_get_quote_config() symbol is not found within the platform quote provider library or it returns an error, the platform quote library uses the raw-TCB of the platform to certify the key and use the certification type PPID_RSA3072_ENCRYPTED as the Quote's Certification Data Type to identify platform. If the API is found, the API returns 2 pieces of information:

1. The TCB to use when requesting the PCE to certify the attestation key. This matches the TCB of the PCK Certificate that the quote verifier uses to certify the attestation key.
2. The certification data for the associated PCK Cert to be added to the quote when the quote is generated.

The data returned by this API is used to determine the ultimate size of the quote. The currentrelease of the Quote Library only supports the sgx_ql_config_t.version of SGX_QL_CONFIG_VERSION_1 (0x0001). For this version of the sgx_ql_config_t data structure returned by the sgx_ql_get_quote_config(), the sgx_ql_config_t.p_cert_data is expected to point to the PCK Cert chain as defined by the certification type PCK_CERT_CHAIN (5) for the Quote Certification Data Type. The Quote Library uses this data to replace the default certification data type generated by the QE. Because of this, the 'Quote Signature Data Len' and the 'QE Certification Data' fields in the Quote are not signed by the AK.

Future versions of the sgx_ql_config_t data structure may support more Certification Data Types.

The functionality of the API is not limited to just ECDSA attestation. The data inputted and outputted from this function only pertains to the TCB to use for generating the PCE signature and the data needed to locate the associated PCK Certificate. It can be considered independent of the type of AK used for quote signing and may apply to other attestation environments.

Syntax

```
quote3_error_t sgx_ql_get_quote_config(
                        const sgx_ql_pck_cert_id_t *p_pck_cert_id,
                          sgx_ql_config_t **pp_cert_config);
```

Parameters

p_pck_cert_id [In]

The Quoting Library passes a pointer to the PCK Certificate ID structure. The platform quote provider library will use this information to find the proper TCB and Quote Certification Data. If the Quoting Library does not support reporting the optional field, encrypted PPID, when this call is made, then p_encrypted_ppid is NULL, encrypted_ppid_size is 0 and crypto_suite is 0.

pp_cert_config [Out]

Pointer to a pointer to the PCK certification data needed for quote generation. The platform quote provider library allocates this buffer and it is expected that the Quote Library frees it with the platform quote provider library sgx_ql_free_quote_config() API. If the platform does not yet have the configuration data available, the SGX_QL_NO_PLATFORM_CERT_DATA error is returned and the PCK Signature is generated using the raw TCB of the platform. If the library does not support the data or there is a problem with the format, the Quoting library returns SGX_QL_ATT_KEY_CERT_DATA_INVALID.

Return Values

SGX_QL_SUCCESS:

Platform has the certification data available and returned it in the p_quote_config buffer.

SGX_QL_NO_PLATFORM_CERT_DATA:

Platform quote provider library cannot provide the platform's certification data.

SGX_QL_ERROR_INVALID_PARAMETER:

Platform quote provider library rejected the input.

SGX_QL_NETWORK_ERROR:

If the platform quote provider library uses the network to retrieve the verification collateral, this error will be returned when it encounters network connectivity problems.

SGX_QL_MESSAGE_ERROR:

If the platform quote provider library uses message protocols to retrieve the verification collateral, this error will be returned when it encounters any protocol problems.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error occurred.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:

Unexpected error from the attestation infrastructure while retrieving the platform data.

SGX_QL_ERROR_MESSAGE_PARSING_ERROR

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

SGX_QL_PLATFORM_UNKNOWN

This platform is an unrecognized SGX platform.

SGX_QL_QEIDENTITY_NOT_FOUND

The server cannot find the QE identity.

SGX_QL_NO_QVE_IDENTITY_DATA

The server cannot find the QvE identity.

SGX_QL_TCBINFO_NOT_FOUND

The server cannot find the TCB info requested.

SGX_QL_ERROR_STATUS_SERVICE_UNAVAILABLE

The server is currently busy to response.

3.3.1.3. Free PCK Certification Information

Description

This API frees the PCK Cert configuration data buffer allocated by the platform quote provider library `sgx_ql_get_quote_config()` API.

Syntax

```
quote3_error_t sgx_ql_free_quote_config(
                       sgx_ql_config_t *p_cert_config);
```

Parameters

p_cert_config [Out]

Pointer to the PCK certification that the `sgx_ql_get_quote_config()` API allocated.

Return Values

SGX_QL_SUCCESS:

Pointer is successfully freed or the input pointer is NULL.

3.3.1.4. Store Persistent Data

Not required and does not need to be implemented by the platform quote provider library. If implemented, it is expected that the platform quote provider library stores the data to the file specified in the input.

Syntax

```
quote3_error_t sgx_ql_write_persistent_data(
                              const uint8_t *p_buf,
                              uint32_t buf_size,
                              const char *p_label);
```

Parameters

p_buf [In]

Pointer to the data to be written. Must not be NULL.

buf_size [In]

Size of the data in bytes that p_buf points to.

p_label [In]

Pointer to the string label of the data to be stored. Must not be NULL and must be a valid string.

Return Values

SGX_QL_SUCCESS:

Data was written successfully.

SGX_QE_PLATFORM_LIB_UNAVAILABLE:

Platform quote provider library was not found.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error occurred.

SGX_QL_ERROR_INVALID_PARAMETER:

One of the pointers is NULL

SGX_QL_FILE_ACCESS_ERROR:

Not able to find the 'label' or there was a problem writing the data.

3.3.1.5. Retrieve Persistent Data

Description

Not required and does not need to be implemented by the platform quote provider library. If implemented, it is expected that the platform quote provider library loads the data from the file specified in the input.

Syntax

```
quote3_error_t sgx_ql_read_persistent_data(
                         const uint8_t *p_buf,
                         uint32_t *p_buf_size,
                         const char *p_label);
```

Parameters

p_buf [In/Out]

Pointer to the buffer to store the data.

p_buf_size [In/Out]

Pointer to the size in the buffer.   If the p_buff is NULL, the API returns the required size.   Must not be NULL.

p_label[In]

Pointer to the string label of the data to be stored.   Must not be NULL and must be a valid string.

Return Values

SGX_QL_SUCCESS:

Data was read successfully.

SGX_QE_PLATFORM_LIB_UNAVAILABLE:

Platform quote provider library was not found.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error occurred.

SGX_QL_ERROR_INVALID_PARAMETER:

If all pointers are not NULL, the size of the inputted buffer is too small. Otherwise, one of the mandatory input pointers is NULL

SGX_QL_FILE_ACCESS_ERROR:

Not able to find the 'label' or there was a problem retrieving the data.

3.3.1.6. Get SGX Quote Verification Collateral

Description

For ECDSA quote verification, this API is called when the platform needs to retrieve the remote platform's quote verification collateral. This is the data required to complete quote verification. It includes:

- The root CA Cert
- The root CA CRL
- The PCK Cert CRL
- The PCK Cert CRL signing chain
- The signing cert chain for the TCBInfo structure
- The signing cert chain for the QEIdentity structure
- The TCBInfo structure
- The QEIdentity structure

See the sgx_ql_qve_collateral_t definition.

The 'version' field of the sgx_ql_qve_collateral_t structure reflects the version of the PCCS/PCS API used to retrieve the collateral. For V1 and V2 APIs of the PCS/PCCS, the 'version' field has a value of 1.0. For V3 APIs of the PCS/PCCS, the 'version' field has the value of either 3.0 or 3.1.

- Collateral.version = 1.0 have CRL's formatted in PEM (string).
- Collateral.version = 3.0 have the CRL's formatted in Base16 DER (string).
- Collateral.version = 3.1 have the CRL's formatted in raw binary DER.

Syntax

```
quote3_error_t sgx_ql_get_quote_verification_collateral(
                                        const uint8_t *fmspc,
                                        const uint16_t fmspc_size,
                                        const char *pck_ca,
                                        sgx_ql_qve_collateral_t **pp_quote_collateral);
```

Parameters

fmspc [In]

Base 16-encoded representation of FMSPC. (currently defined to be 6 bytes).

fmpsc_size [In]

Number of bytes in the buffer pointed by *fmspc*.

ca [In]

Null terminated string identifier of the PCK Cert CA that issued the PCK Certificates. Allowed values:

- "processor" – indicates PCK Certificate was issued by the Intel SGX Processor CA.
- "platform" – indicates PCK Cert was issued by the Intel SGX Platform CA.

pp_quote_collateral [Out]

Pointer to a pointer to the PCK quote collateral data needed for quote verification. The platform quote provider library will allocate this buffer and it is expected that the Quote Verification Library will free it using the platform quote provider library's sgx_ql_free_quote_verification_collateral() API. If the platform quote provider library cannot be found, the error SGX_QL_PLATFORM_LIB_UNAVAILABLE will be returned. If the platform quote provider library cannot retrieve the data, the SGX_QL_NO_QUOTE_COLLATERAL_DATA error will be returned.

Return Values

SGX_QL_SUCCESS:

Data was read successfully.

SGX_QL_NO_QUOTE_COLLATERAL_DATA:

The platform quote provider library does not have the quote verification collateral data available.

SGX_QL_ERROR_INVALID_PARAMETER:

The platform quote provider library rejected the input.

SGX_QL_ERROR_OUT_OF_MEMORY:

Heap memory allocation error in library or enclave.

SGX_QL_NETWORK_ERROR:

If the platform quote provider library uses the network to retrieve the verification collateral, this error will be returned when it encounters network connectivity problems.

SGX_QL_MESSAGE_ERROR:

If the platform quote provider library uses message protocols to retrieve the verification collateral, this error will be returned when it encounters any protocol problems.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:

Unexpected error from the attestation infrastructure while retrieving the platform data.

SGX_QL_ERROR_MESSAGE_PARSING_ERROR

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

SGX_QL_PLATFORM_UNKNOWN

This platform is an unrecognized SGX platform.

SGX_QL_QEIDENTITY_NOT_FOUND

The server cannot find the QE identity.

SGX_QL_NO_QVE_IDENTITY_DATA

The server cannot find the QvE identity.

SGX_QL_TCBINFO_NOT_FOUND

The server cannot find the TCB info requested.

SGX_QL_ERROR_STATUS_SERVICE_UNAVAILABLE

The server is currently busy to response.

3.3.1.7. Get SGX Quote Verification Collateral with Request Parameters

Description

This API performs the same function as the API described in Get Quote Verification Collateral but allows for the caller to specify additional parameters that may be required to the verification collateral in a specific attestation environment. This API is not required and does not need to be implemented by the platform quote provider library.

Syntax

```
quote3_error_t sgx_ql_get_quote_verification_collateral_with_parameters(
                                    const uint8_t *fmspc,
                                    const uint16_t fmspc_size,
                                    const char *pck_ca,
                                    const void* custom_param,
                                    const uint16_t custom_param_length,
                                    sgx_ql_qve_collateral_t **pp_quote_collateral);
```

Parameters

fmspc [In]

Base 16-encoded representation of FMSPC. (currently defined to be 6 bytes).

Fmpsc_size [In]

Number of bytes in the buffer pointed by *fmspc*.

Ca [In]

Null terminated string identifier of the PCK Cert CA that issued the PCK Certificates. Allowed values:

- "processor" – indicates PCK Certificate was issued by the Intel SGX Processor CA.
- "platform" – indicates PCK Cert was issued by the Intel SGX Platform CA.

custom_param[In]

Address of the buffer of additional information needed by a give attestation infrastructure. The definition and format of this data is dependent on the implementer of the Platform Quote Provider Library. For example, the reference Platform Quote Provider will attempt to encode this data in Base64 format to be used when fetching the collateral from the network. If this parameter is not NULL and `custom_param_length is 0,` the API will return an error. If this parameter is NULL and c `custom_param_length is greater than 0,` the API will return an error.

custom_param_length[In]

Length, in bytes, of the buffer pointed to by `custom_param`.

pp_quote_collateral [Out]

Pointer to a pointer to the PCK quote collateral data needed for quote verification. The platform quote provider library will allocate this buffer and it is expected that the Quote Verification Library will free it using the platform quote provider library's sgx_ql_free_quote_verification_collateral() API. If the platform quote provider library cannot be found, the error SGX_QL_PLATFORM_LIB_UNAVAILABLE will be returned. If the platform quote provider library cannot retrieve the data, the SGX_QL_NO_QUOTE_COLLATERAL_DATA error will be returned.

Return Values

SGX_QL_SUCCESS:

Data was read successfully.

SGX_QL_NO_QUOTE_COLLATERAL_DATA:

The platform quote provider library does not have the quote verification collateral data available.

SGX_QL_ERROR_INVALID_PARAMETER:

The platform quote provider library rejected the input.

SGX_QL_ERROR_OUT_OF_MEMORY:

Heap memory allocation error in library or enclave.

SGX_QL_NETWORK_ERROR:

If the platform quote provider library uses the network to retrieve the verification collateral, this error will be returned when it encounters network connectivity problems.

SGX_QL_MESSAGE_ERROR:

If the platform quote provider library uses message protocols to retrieve the verification collateral, this error will be returned when it encounters any protocol problems.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:

Unexpected error from the attestation infrastructure while retrieving the platform data.

SGX_QL_ERROR_MESSAGE_PARSING_ERROR

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

SGX_QL_PLATFORM_UNKNOWN

This platform is an unrecognized SGX platform.

SGX_QL_QEIDENTITY_NOT_FOUND

The server cannot find the QE identity.

SGX_QL_NO_QVE_IDENTITY_DATA

The server cannot find the QvE identity.

SGX_QL_TCBINFO_NOT_FOUND

The server cannot find the TCB info requested.

SGX_QL_ERROR_STATUS_SERVICE_UNAVAILABLE

The server is currently busy to response.

3.3.1.8. Free SGX Quote Verification Collateral

Description

Called to free the quote verification collateral data buffer allocated by the platform quote provider library's sgx_ql_get_quote_verification_collateral() API.

Syntax

```
quote3_error_t sgx_ql_free_quote_verification_collateral(
                       sgx_ql_qve_collateral_t *p_quote_collateral);
```

Parameters

p_quote_collateral [Out]

Pointer to the quote verification collateral date provided by the `sgx_ql_get_quote_verification_collateral()` API.

Return Values

SGX_QL_SUCCESS:

The pointer was successfully freed or the input pointer was NULL.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

3.3.1.9.Get TDX Quote Verification Collateral

Description

For TDX quote verification, this API is called when the platform needs to retrieve the remote platform's quote verification collateral. This is the data required to complete quote verification. It includes:

- The root CA Cert
- The root CA CRL
- The PCK Cert CRL
- The PCK Cert CRL signing chain
- The signing cert chain for the TDX TCBInfo structure
- The signing cert chain for the TD_QE Identity structure
- The TDX TCBInfo structure
- The TD_QE Identity structure

See the sgx_ql_qve_collateral_t definition.

The 'version' field of the sgx_ql_qve_collateral_t structure reflects the version of the PCCS/PCS API used to retrieve the collateral. For V1 and V2 APIs of the PCS/PCCS, the 'version' field has a value of 1.0. For V3 APIs of the PCS/PCCS, the 'version' field has the value of either 3.0 or 3.1.

- Collateral.version = 1.0 have CRL's formatted in PEM (string).
- Collateral.version = 3.0 have the CRL's formatted in Base16 DER (string).
- Collateral.version = 3.1 have the CRL's formatted in raw binary DER.

Syntax

```
quote3_error_t tdx_ql_get_quote_verification_collateral(
                                                     const uint8_t *fmspc,
                                                     const uint16_t fmspc_size,
                                                     const char *pck_ca,
                                                     sgx_ql_qve_collateral_t **pp_quote_collateral);
```

Parameters

fmspc [In]

Base 16-encoded representation of FMSPC. (currently defined to be 6 bytes).

fmpsc_size [In]

Number of bytes in the buffer pointed by *fmspc*.

ca [In]

Null terminated string identifier of the PCK Cert CA that issued the PCK Certificates. Allowed values:

- "processor" – indicates PCK Certificate was issued by the Intel SGX Processor CA.
- "platform" – indicates PCK Cert was issued by the Intel SGX Platform CA.

pp_quote_collateral [Out]

Pointer to a pointer to the PCK quote collateral data needed for quote verification. The platform quote provider library will allocate this buffer and it is expected that the Quote Verification Library will free it using the platform quote provider library's sgx_ql_free_quote_verification_collateral() API. If the platform quote provider library cannot be found, the error SGX_QL_PLATFORM_LIB_UNAVAILABLE will be returned. If the platform quote provider library cannot retrieve the data, the SGX_QL_NO_QUOTE_COLLATERAL_DATA error will be returned.

Return Values

SGX_QL_SUCCESS:

Data was read successfully.

SGX_QL_NO_QUOTE_COLLATERAL_DATA:

The platform quote provider library does not have the quote verification collateral data available.

SGX_QL_ERROR_INVALID_PARAMETER:

The platform quote provider library rejected the input.

SGX_QL_ERROR_OUT_OF_MEMORY:

Heap memory allocation error in library or enclave.

SGX_QL_NETWORK_ERROR:

If the platform quote provider library uses the network to retrieve the verification collateral, this error will be returned when it encounters network connectivity problems.

SGX_QL_MESSAGE_ERROR:

If the platform quote provider library uses message protocols to retrieve the verification collateral, this error will be returned when it encounters any protocol problems.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:

Unexpected error from the attestation infrastructure while retrieving the platform data.

SGX_QL_ERROR_MESSAGE_PARSING_ERROR

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

SGX_QL_PLATFORM_UNKNOWN

This platform is an unrecognized SGX platform.

SGX_QL_QEIDENTITY_NOT_FOUND

The server cannot find the QE identity.

SGX_QL_NO_QVE_IDENTITY_DATA

The server cannot find the QvE identity.

SGX_QL_TCBINFO_NOT_FOUND

The server cannot find the TCB info requested.

SGX_QL_ERROR_STATUS_SERVICE_UNAVAILABLE

The server is currently busy to response.

3.3.1.10.Free TDX Quote Verification Collateral

Description

Called to free the quote verification collateral data buffer allocated by the platform quote provider library's tdx_ql_get_quote_verification_collateral() API.

Syntax

```
quote3_error_t tdx_ql_free_quote_verification_collateral(
                        sgx_ql_qve_collateral_t *p_quote_collateral);
```

Parameters

p_quote_collateral [Out]

Pointer to the quote verification collateral date provided by the `sgx_ql_get_quote_verification_collateral()` API.

Return Values

SGX_QL_SUCCESS:

The pointer was successfully freed or the input pointer was NULL.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

3.3.1.11. Get Quote Verification Enclave Identity (QVEIdentity)

Description

When the SGX ECDSA quote is verified by the Quote Verification Enclave, the caller can request that the QVE return an SGX REPORT. This allows the caller to cryptographically verify that the quote verification results were generated by the QVE. The caller uses REPORT based local attestation to perform this verification. Once the REPORT has been verified using local attestation, the caller needs to verify that the REPORT was generated by the expected Intel QVE. Intel signs and publishes a JSON data structure with the QVE identity information (MRSIGNER, ProdID, ISVSVN, etc) called the QVEIdentity. The platform quote provider library implements this API to provide the QVEIdentity to the caller. If the platform quote provider library cannot retrieve the data, the SGX_QL_NO_QVE_IDENTITY_DATA error will be returned.

Syntax

```
quote3_error_t sgx_ql_get_qve_identity(
                                     char **pp_qve_identity,
                                     uint32_t *p_qve_identity_size,
                                     char **pp_qve_identity_issuer_chain,
                                     uint32_t *p_qve_identity_issuer_chain_size);
```

Parameters

pp_qve_identity [Out]

Pointer to a pointer to the UTF-8 encoded JSON string containing the QVE Identity structure. The platform quote provider library will allocate this buffer and it is expected that the caller will free it using the platform quote provider library's sgx_ql_free_qve_identity() API.

p_qve_identity_size [Out]

The length of the string in bytes in the buffer pointed by **pp_qve_identity* including the terminating null character.

pp_qve_identity_issuer_chain [Out]

Pointer to a pointer to the UTF-8 encoded string containing the QVE Identity issuer certificate chain for SGX QVE Identity. It consists of SGX Root CA Certificate and SGX TCB Signing Certificate. The platform quote provider library will allocate this buffer and it is expected that the caller will free it using the platform quote provider library's sgx_ql_free_qve_identity() API.

p_qve_identity_issuer_chain_size [Out]

The length of the string in bytes in the buffer pointed by `*pp_qve_identity_issuer_chain` including the terminating null character.

Return Values

SGX_QL_SUCCESS:

Data was read successfully.

SGX_QL_NO_QVE_IDENTITY_DATA:

The platform quote provider library does not have the QVE identity data available.

SGX_QL_ERROR_INVALID_PARAMETER:

The platform quote provider library rejected the input.

SGX_QL_PLATFORM_LIB_UNAVAILABLE:

The Quote Library could not locate the provider library.

SGX_QL_ERROR_OUT_OF_MEMORY:

Heap memory allocation error in library or enclave.

SGX_QL_NETWORK_ERROR:

If the platform quote provider library uses the network to retrieve the QVE Identity, this error will be returned when it encounters network connectivity problems.

SGX_QL_MESSAGE_ERROR:

If the platform quote provider library uses message protocols to retrieve the QVE Identity collateral, this error will be returned when it encounters any protocol problems.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

SGX_QL_NO_QVE_IDENTITY_DATA

The server cannot find the QvE identity.

SGX_QL_ERROR_STATUS_SERVICE_UNAVAILABLE

The server is currently busy to response.

3.3.1.12. Free Quote Verification Enclave Identity

Description

Called to free the quote verification enclave identity data buffer allocated by the platform quote provider library's sgx_ql_get_qve_identity() API.

Syntax

```
quote3_error_t sgx_ql_free_qve_identity(
                            char *p_qve_identity,
                            char *p_qve_identity_issuer_chain);
```

Parameters

p_qve_identity [Out]

Pointer to the QVE identity that the `sgx_ql_get_qve_identity()` API returned.

p_qve_identity_issuer_chain [Out]

Pointer to the QVE identity issuer chain that the `sgx_ql_get_qve_identity()` API returned.

Return Values

SGX_QL_SUCCESS:

The pointer was successfully freed or the input pointer was NULL.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

3.3.1.13. Get the Root CA CRL

Description

Syntax

```
quote3_error_t sgx_ql_get_root_ca_crl (
                                   char **pp_root_ca_crl,
                                   uint16_t *p_root_ca_crl_size);
```

Parameters

pp_root_ca_crl [Out]

Pointer to a pointer to the UTF-8 encoded JSON string containing the x.509 SGX Root CA CRL. The platform quote provider library will allocate this buffer and it is expected that the caller will free it using the platform quote provider library's sgx_ql_free_root_ca_crl () API.

p_root_ca_crl_size [Out]

The length of the string in bytes in the buffer pointed by **pp_root_ca_crl* including the terminating null character.

Return Values

SGX_QL_SUCCESS:

The data was read successfully.

SGX_QL_NO_QUOTE_COLLATERAL_DATA:

The platform quote provider library does not have the root ca crl data available.

SGX_QL_ERROR_INVALID_PARAMETER:

The platform quote provider library rejected the input.

SGX_QL_ERROR_OUT_OF_MEMORY:

Heap memory allocation error in library or enclave.

SGX_QL_NETWORK_ERROR:

If the platform quote provider library uses the network to retrieve the root ca crl, this error will be returned when it encounters network connectivity problems.

SGX_QL_MESSAGE_ERROR:

If the platform quote provider library uses message protocols to retrieve the root ca crl, this error will be returned when it encounters any protocol problems.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

3.3.1.14. Free the Root CA CRL

Description

Syntax

```
quote3_error_t sgx_ql_free_root_ca_crl (
                                   char *p_root_ca_crl);
```

Parameters

p_root_ca_crl [Out]

Pointer to the root CA CRL that the `sgx_ql_get_root_ca_crl()` API returned.

Return Values

SGX_QL_SUCCESS:

The pointer was successfully freed or the input pointer was NULL.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

3.3.1.15. Register a Logging Function with the Platform Quote Provider Library

Description

The Platform Quote Provider Library provides a function to register a logging function. This allows a process with logging capability to register its logging function with the Platform Quote Provider Library. This allows the Platform Quote Provider Library writer to utilize the registered logging function to output its debug logs.

Syntax

```
quote3_error_t sgx_ql_set_logging_callback(
                            sgx_ql_logging_callback_t logger,
                            sgx_ql_log_level_t loglevel = SGX_QL_LOG_ERROR);
```

Parameters

logger [In]

Function pointer to the logging function. Must not be NULL.

It must have this prototype:

```
typedef void (*sgx_ql_logging_callback_t)(
                            sgx_ql_log_level_t level,
                             const char* message);
```

The logging function takes the string 'message' and a logging 'level' defined as:

```
typedef enum _sgx_ql_log_level_t
{
    SGX_QL_LOG_ERROR,
    SGX_QL_LOG_INFO,
    SGX_QL_LOG_DEBUG,
    SGX_QL_LOG_TRACE,

} sgx_ql_log_level_t;
```

loglevel [In]

Log level. Default is SGX_QL_LOG_ERROR.

Return Values

SGX_QL_SUCCESS:

The logging function was successfully registered.

SGX_QL_ERROR_INVALID_PARAMETER:

The logger parameter was NULL.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

3.3.1.16. Clear local cache files

Description

The Platform Quote Provider Library may cache files in local hard drive when the caching option is enabled. This API is used to clear the cached files.

Syntax

```
quote3_error_t sgx_qpl_clear_cache (uint32_t cache_type);
```

Parameters

cache_type [In]

Specify the types of cache that you want to clear. It can be a specific 'sgx_qpl_cache_type_t' value, or you can select a combination of individual values using the bitwise OR operator '|'.

Return Values

SGX_QL_SUCCESS:

The specified cache files was cleared successfully.

3.3.1.17. Get default platform policy for appraisal engine

Description

The default platform policy is provided by platform owners. Use this API to retrieve the default platform policy from PCCS.

Syntax

```
quote3_error_t tee_get_default_platform_policy(const uint8_t *fmspc, const
uint16_t fmspc_size, uint8_t **pp_platform_policy, uint32_t
*p_platform_policy_size);
```

Parameters

fmspc [In]

Base 16-encoded representation of FMSPC. (currently defined to be 6 bytes).

fmpsc_size [In]

Number of bytes in the buffer pointed by *fmspc*.

pp_platform_policy [Out]

Number of bytes in the buffer pointed by *fmspc*.

p_platform_policy_size [Out]

Pointer to a pointer to the default platform policy for the specified fmspc in jwt format. The platform quote provider library will allocate this buffer and it is expected that the caller will free it using the platform quote provider library's tee_free_platform_policy() API.

Return Values

SGX_QL_SUCCESS:

Data was read successfully.

SGX_QL_NO_QUOTE_COLLATERAL_DATA:

The platform quote provider library cannot get a default platform policy for this platform.

SGX_QL_ERROR_INVALID_PARAMETER:

The platform quote provider library rejected the input.

SGX_QL_ERROR_OUT_OF_MEMORY:

Heap memory allocation error in library or enclave.

SGX_QL_NETWORK_ERROR:

Network error.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

SGX_QL_SERVICE_UNAVAILABLE

The server is currently busy to response.

3.3.1.18. Free default platform policy

Description

Called to free the platform policy data buffer allocated by the platform quote provider library's tee_get_default_platform_policy`()` API.

Syntax

```
Quote3_error_t tee_free_platform_policy(uint8_t *p_platform_policy);
```

Parameters

p_platform_policy [In]

Pointer to the buffer that the tee_get_default_platform_policy API allocated.

Return Values

SGX_QL_SUCCESS:

The pointer was freed successfully.

3.3.2. Intel® SGX Enclave Loading Library

Provided by the modified 'urts' library that uses the Intel® SGX Enclave Loading Abstraction Layer Library (see Enclave Common Loader API Reference document). This 'modified-sgx-urts' library exposes the same enclave loading API as provided by the legacy Intel® SGX SDK.

3.4. Deployment Tool for PCK Certificate Chain Retrieval for Intel® SGX DCAP

Some attestation environments do not provision each platform with its PCK Cert or the PCK Cert TCB at platform bring-up and then store it persistently for each VM when the VM starts up. Also, some environments do not permit access to the external Intel hosted PCK Certificate Service during runtime. These environments retrieve the PCK Cert Chain from Intel during platform bring-up (Deployment) and store the PCK Cert chain in a PCK Cert Proxy/PCK Cert Inventory hosted within their attestation infrastructure. Then, the PCK Cert chain is provided to the VM when the VM starts up (run-time) based on a request from the VM to the Proxy Service. Or, it provides the PCK Cert TCB at VM start up and then retrieve the PCK Cert when the Quote is verified. There is a need to identify the PCK Cert retrieved during deployment to download the TCB or PCK Cert to the VM at runtime.

The PPID in the PCK Cert could be used for this purpose but the Intel® SGX protects the PPID privacy by encrypting the PPID with a PCK Server owned public key. The RSA-OAEP algorithm used to encrypt the PPID changes the encrypted PPID value between successive requests to the PCE. The Enc(PPID) generated during deployment does not match the Enc(PPID) used during runtime.

The *PCKRetrievalTool* will be released along with the Quote Library release. The production version of the Quote Library encrypts the PPID with a 3072bit RSA-OAEP key owned by the Intel hosted PCK Certification Service. The PCKRetrievalTool output can be used to request a PCK Cert from the service.

The PCK Retrieval Tool will output a Base16 (Hex) encoded text file in CSV format:

```
EncryptedPPID(384 BE byte array),PCE_ID(LE 16 bit integer),CPUSVN(16 byte BE
byte array),PCE ISVSVN (LE 16 bit integer),QE_ID (16 byte BE byte array)
```

To request PCK Certs Online, follow the onboarding and RESTful API described in the PCK Service documentation.

Note: The EncPPID changes each time the tool is called while the QE_ID does not. The user of the tool should keep a link between the QE_ID and EncPPID to properly link the Platform to its PCK Cert Chain.

Note: You may get more than one PCK Cert Chain for each platform depending on the number of active TCB levels for that platform and the PCK Certificate Service API used.

3.5. Key Derivations

3.5.1. QE_ID Derivation

The QE_ID is a platform ID that is not associated with a particular SVN but is dependent on the Quoting Enclave (QE) MRSIGNER and its Seal Key. The QE_ID is designed to be dependent on the seal key, which depends on the platform OWNER_EPOCH value. The OWNER_EPOCH value is set by the platform owner in the BIOS configuration. If the BIOS non-volatile memory (FLASH) is wiped, then the QE_ID changes even if generated by the same QE. This prevents the QE_ID from being a true HW ID. A true HW ID cannot be modified by the platform owner.

1) QE_ID-Seed = EGETKEY(KEYNAME=SEAL_KEY,
   KEY_POLICY=MRSIGNER,
   KEY_ID = 0,
   CPUSVN=0,
   ISVSVN = 0)
2) QE_ID = AES128-CMAC(QE_ID-Seed, 16 bytes below)

Byte Position	Value
0	0x00
1-9	“QE_ID_DER” (ascii encoded)
10-13	0x00000000
14-15	0x0080 (Big Endian)

3.5.2. ECDSA Attestation Key Derivation

3.5.2.1. ECDSA Attestation Key Derivation using QE Seal Key (Intel® SGX DCAP Solution)

The ECDSA Attestation key is derived from the QE seal key at the current TCB level. This allows the QE to regenerate the same attestation key without requiring persistent storage. However, the key changes when any of the QE TCB components change (CPUSVN, PCE_ISVSVN or the QE_ ISVSVN). This extends the lifetime of the QE attestation key beyond the time the QE library exists in process memory.

The QE attestation key is used by the QE to sign reports from application enclaves. It is a 256 bit ECC signing key using NIST curve secp256r1.

The QE attestation key derivation is rooted in the HW key. EGETKEY does a series of AES-base derivations resulting in a Provisioning Key unique to the HW, the current TCB (CPUSVN), and the QE identify (MRSIGNER, ISVPRODID, ISVSVN). The AK private key is derived from 320 random bits (providing 128 bits of entropy) derived from the Seal Key using the following flow:

1) Sealing Key = EGETKEY(KEYNAME = SEAL_KEY,
   KEY_POLICY =MRSIGNER,
   KEY_ID = 0,
   Current CPUSVN,
   Current ISVSVN)
2) Block 1 = AES-CMAC(Sealing Key, QE string with Counter = 0x01)
3) Block 2 = AES-CMAC(Sealing Key, QE string with Counter = 0x02)

4) Block 3 = AES-CMAC(Sealing Key, QE string with Counter = 0x03)

5) QE ATT Seed = most significant 320 bits of (Block 1 || Block 2 || Block 3).

6) QE ATT key pair is generated using NIST SP 186-4 section B 4.1 “Key Pair Generation Using Extra Random Bits.” AE ATT Seed are used for the random bits.

Byte Position	Value
0	Counter (See Description)
1-10	“QE_KEY_DER” (ascii encoded)
11-13	0x000000
14-15	0x0140 (Big Endian)

3.6.Quote Verification Library

This chapter presents a set of C-like APIs that allow applications to verify an Intel® SGX ECDSA Quote as defined in Quote Format using either Quote Verification Enclave (QVE) when SGX is available or with a untrusted implementation when SGX is not available. Both implementations are built with the Quote Verification Library (QVL) published with DCAP.This library is delivered as a dynamically linked library (.so/.dll)

3.6.1. Set Enclave Load Policy

When the Quote Verification Library is linked to a process, it needs 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 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.

3.6.2. Verify Quote

Description

This API will verify an SGX ECDSA Quote as defined in Quote_Format generated by an SGX ECDSA Quoting Enclave. This API is designed to work with verification code running in an enclave as well as verification code running in an untrusted environment. A non-NULL p_qve_report_info input parameter indicates the verifying platform is SGX compatible and should use an SGX ECDSA Quote Verification Enclave (QVE) to verify the Quote. The platform software will know if the platform is SGX-capable or not. The Intel provided DCAP library will support both the QVE based verification as well as the non-QVE based verification.   When p_qve_report is non-NULL, the user needs to provide the application enclave's target info and the QVE will return a REPORT which targets the calling enclave in the p_qve_report_info structure. Then calling application's enclave can use SGX DCAP TVL library to verify the QvE has the expected identity. When p_qve_report_info is NULL, the quote verification will be performed but the results cannot be cryptographically authenticated.

The caller may provide the p_quote_collateral as defined in sgx_ql_qve_collateral_t.   If p_quote_collateral is NULL, then the quote library will attempt to retrieve the collateral from the platform quote provider library. If the quote library fails to retrieve the data from the platform quote provider library, it will return SGX_QL_PLATFORM_LIB_UNAVAILABLE . The QVE will verify the format and the certificate chains of all the collateral passed in.   The API will return the appropriate error if the collateral format or signature check fails. The current version of the QVE only supports version 1 of the sgx_ql_qve_collateral_t.

The expiration_check_date is used to compare to the x.509 'Not After' field, x.509 CRL 'Next Update' and the JSON 'nextUpdate' field. If any of these collateral's 'Not After' or 'nextUpdate' field has a date earlier than the expiration_check_date input parameter, the value pointed by p_collateral_expiration_status will have a non-zero value. But this check alone will not cause the Verify Quote API to return an error.

If all the input parameters, the quote verification collateral, and the quote format are correct, the API will return SGX_QL_SUCCESS. It will also indicate the verification result in p_quote_verifcation_result. The verification result can have the following values (see sgx_ql_qve_result_t):

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_INVALID_SIGNATURE - *Terminal*
8. SGX_QL_QV_RESULT_REVOKED - *Terminal*
9. SGX_QL_QV_RESULT_UNSPECIFIED - *Terminal*

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

- SGX_QL_QV_RESULT_CONFIG_NEEDED – The SGX platform firmware and SW are at the latest security patching level but there are platform hardware configurations that may expose the enclave to vulnerabilities. These vulnerabilities can be mitigated with the appropriate platform configuration changes that will produce an SGX_QL_QV_RESULT_OK verification result.

- SGX_QL_QV_RESULT_SW_HARDENING_NEEDED – The SGX platform firmware and SW are at the latest security patching level but there are certain vulnerabilities that can only be mitigated with software mitigations implemented by the enclave. The enclave identity policy needs to indicate whether the enclave has implemented these mitigations.

- SGX_QL_QV_RESULT_CONFIG_AND_SW_HARDENING_NEEDED – The SGX platform firmware and SW are at the latest security patching level but there are certain vulnerabilities that can only be mitigated with software mitigations implemented by the enclave. The enclave identity policy needs to indicate whether the enclave has implemented these mitigations. There are also platform hardware configurations that may expose the enclave to vulnerabilities. These configuration vulnerabilities can be mitigated with the appropriate platform configuration changes that will produce an SGX_QL_QV_RESULT_SW_HARDENING_NEEDED verification result.

- SGX_QL_QV_RESULT_OUT_OF_DATE – The SGX platform firmware and SW 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.
- SGX_QL_QV_RESULT_OUT_OF_DATE_CONFIG_NEEDED – The SGX platform firmware and SW are ***not*** at the latest security patching level. The platform needs to be patched with firmware and/or software patches. There are also platform hardware configurations that may expose the enclave to vulnerabilities. These configuration vulnerabilities can be mitigated with the appropriate platform configuration changes. Applying both the updated patches and the appropriate platform configuration changes will produce an SGX_QL_QV_RESULT_OK verification result.

Multiple thread support: By default, the quote verification API exhibits limited scalability in multithreaded environments, leading to increased processing time with concurrent requests. To potentially enhance multithreaded performance specifically for QvE enclave based quote verification, users **must** explicitly set the QvE load policy by calling API "sgx_qv_set_enclave_load_policy". The supported values for improved concurrency are "SGX_QL_EPHEMERAL_QVE_MULTI_THREAD" or "SGX_QL_PERSISTENT_QVE_MULTI_THREAD". **It is important to set this policy before calling the API in multithreaded contexts, even when using QVL-based quote verification, although the policy primarily affects QvE mode.**

There may be cases where the caller cannot abide by the strict compliance verification applied by this API. For example, the caller may not have access to a trusted time source for the expiration_time or the platform owner may not be able to patch all platforms prior to Intel's latest TCB level. In these cases, the caller can make use of the optional supplemental data returned by this API. This supplemental data will allow the caller to implement a different quote verification policy. When this API returns an error other than SGX_QL_SUCCESS or the *p_quote_verification_result is a terminal error, no alternative verification policy should be performed. See Verify Quote - Supplemental for more information.

Syntax

```
quote3_error_t sgx_qv_verify_quote(
                    const uint8_t *p_quote,
                    uint32_t quote_size,
                    const sgx_ql_qve_collateral_t *p_quote_collateral,
                    const time_t expiration_check_date,
                    uint32_t *p_collateral_expiration_status,
                    sgx_qv_result_t *p_quote_verification_result,
                    sgx_ql_qe_report_info_t *p_qve_report_info,
                    uint32_t supplemental_data_size,
                    uint8_t *p_supplemental_data);
```

Parameters

p_quote [In]

Pointer to an SGX Quote. The QVE only supports version 3 of the SGX ECDSA Quote. Currently, the QVE only supports Quotes with CertType = 5. The Intel signed QVE will only verify Quotes generated by an Intel Signed QE. This type of certification data contains the PCK Certificate Chain 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 the Quote Certification Collateral provided by the caller. The quote collateral structure contains a version number. This is the data that is required to fully verify the quote. Such as the TCBInfo, QEIdentity and CRL structures, etc. If it is NULL, the DCAP library will attempt to retrieve the collateral from the platform quote provider library if available. If the platform quote provider library is not available or the collateral cannot be retrieved, this API 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 QVE will use to determine if any of the inputted collateral have expired. The expectation is that the caller has access to a trusted current source or uses a hardcoded threshold date.

p_collateral_expiration_status [Out]

Address of the outputted expiration status. This input must not be NULL. When this API returns a 0 at this address, none of the inputted collateral has expired as compared to the inputted expiration_check_date. This API will return a non-zero value when one or more of the inputted collateral has expired according to the inputted expiration_check_date. This value will contain a non-zero value if the API returns a value other than SGX_QL_SUCCESS.

p_quote_verification_result [In/Out]

Address of the outputted quote verification result. This value 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 not NULL, the user needs to provide a ‘nonce’ and application enclave’s target info, then the QVE will generate a report using the target_info provided in the sgx_ql_qe_report_info_t structure . The QVE.REPORT.REPORT_DATA = (SHA256_HASH[nonce||quote||expiration_check_date||expiration_status||verification_result||supplemental_data] ||32-0x00’s).
If NULL, the quote can still be verified on a non-SGX capable platform or by an untrusted QVL but the results cannot be cryptographically verified.

supplemental_data_size [In]

Size of the buffer pointed to by p_supplemental_data (in bytes). The value should match the value returned by the sgx_qv_get_quote_supplemental_data_size(). If the caller does not need the supplemental data, the parameter should be 0 and the p_supplemental_data should be NULL. SGX_QL_QUOTE_INVALID_PARAMETER if the size in not large enough to return the supplemental data.

p_supplemental_data [Out]

The parameter is optional. If it is NULL, supplemental_data_size must be 0. This data can be used by the CSP or Relying Party to enforce a different quote verification policy than enforced by this API.

Return Values

SGX_QL_SUCCESS:

Successfully evaluated the quote.

SGX_QL_INVALID_PARAMETER:
One of the input parameters value is invalid

SGX_QL_QUOTE_FORMAT_UNSUPPORTED:
The inputted quote format 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 quote verifier doesn't support the certification data in the Quote. Currently, the Intel QVE only supported CertType = 5.

SGX_QL_APP_REPORT_UNSUPPORTED_FORMAT:
The quote verifier doesn't support the format of the application REPORT the Quote.

SGX_QL_QE_REPORT_UNSUPPORTED_FORMAT:
The quote verifier doesn't support the format of the application REPORT the Quote.

SGX_QL_QE_REPORT_INVALID_SIGNATURE:
The signature over the QE Report is invalid.

SGX_QL_QE_REPORT_ATT_KEY_MISMATCH:
The attestation key provided in the Quote was not produced by the QE 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 PCK Cert signature chain including 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 TCBInfo signature chain including 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 Quote's QE doesn't match the inputted expected QEIdentity.

SGX_QL_QEIDENTITY_CHAIN_ERROR:
There was an error verifying the QEIdentity signature chain including QEIdentity revocation.

SGX_QL_OUT_OF_MEMORY:
Heap memory allocation error in library or enclave.

SGX_QL_ENCLAVE_LOAD_ERROR:
Unable to load the enclaves required to initialize the attestation key. 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 application report.

SGX_QL_PLATFORM_LIB_UNAVAILABLE:

The Quote Library could not locate the platform quote provider library or one of its required APIs.

SGX_QL_UNABLE_TO_GENERATE_REPORT:

The QVE was unable to generate its own report targeting the application enclave because there is an enclave compatibility issue.

SGX_QL_NO_QUOTE_COLLATERAL_DATA :

The Quote Library was available, but the quote library could not retrieve the data.

SGX_QL_ERROR_QVL_QVE_MISMATCH:

Only returned when the quote verification library supports both the untrusted mode of verification and the QvE backed mode of verification. This error indicates that the 2 versions of the verification modes are different. Most caused by using a QvE that does not match the version of the DCAP installed.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:

Unexpected error from the attestation infrastructure while retrieving the platform data.

SGX_QL_ERROR_MESSAGE_PARSING_ERROR

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

SGX_QL_PLATFORM_UNKNOWN

This platform is an unrecognized SGX platform.

3.6.3. Get Quote Verification Supplemental Size

Description

If the owner of the quote verification needs to provide a different quote verification policy beyond the policy enforced by the sgx_qv_verify_quote() API, the caller can request the sgx_qv_verify_quote() API to return supplemental data. This supplemental data can be used to implement that alternate policy.

For example, if the owner of the quote verification does not have access to a trusted time source to reliably enforce the expiration date check, they can use the supplemental data to check that the verification collateral's 'tcbEvalDataSetNumber' or 'crlNum' against some reference value to ensure old collateral is not used in the sgx_qv_verify_quote() API.

Or, for example, if the platform owner that generates the quote hasn't patched all platforms before the TCB Recovery Event date, a verifier using the latest verification collateral (TCBInfo, QEIdentity and QVEIdentity) will get an out-of-date status returned from sgx_qv_verify_quote(). If the quote verifier wants to provide a longer grace period, they can use the supplemental data to make sure that the platform has the mitigations prescribed for a specific threat published by Intel in the past. The supplemental data will provide a data associated with the platform's out-of-date TCB. This date can be compared to the public notification date of a particular Intel Security Advisory (SA). If the supplemental TCB date is greater-then-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.

The supplemental data input parameter provided to the sgx_qv_verify_quote() API is optional. If the caller of sgx_qv_verify_quote() wants to implement an alternative verification policy, they must call this API first to get the proper size of the alternative data, allocate the buffer and pass it in to sgx_qv_verify_quote(). If they do not need an alternative verification policy, they do not need to call this API.

Syntax

```
quote3_error_t sgx_qv_get_quote_supplemental_data_size(
                     uint32_t *p_data_size)
```

Parameters

p_data_size[Out]:

Pointer to hold the size of the buffer in bytes required to contain all the supplemental data. This value is passed in to the `sgx_qv_verify_quote`() API. The caller is responsible for allocating a buffer large enough to contain the supplemental data.

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:

Invalid parameter. p_data_size must not be NULL.

SGX_QL_ERROR_QVL_QVE_MISMATCH:

Only returned when the quote verification library supports both the untrusted mode of verification and the QvE backed mode of verification. This error indicates that the 2 versions of the verification modes are different. Most caused by using a QvE that does not match the version of the DCAP installed.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error.

3.6.4.Get latest supplemental data version and size

Description

This is an extension of API `` `sgx_qv_get_quote_supplemental_data_size()` ``, the owner of the quote verification can use this API to get latest supplemental data version and size, then decide the allocated buffer size based on returned size, decide which supplemental version based on returned latest version.

For example, if latest major version is 4, then caller can specify major version 3 or 4 in structure 'tee_supp_data_descriptor_t ' when calling API `` `tee_verify_quote()` ``.
If set major version to 0, then API `` `tee_verify_quote()` `` will always return latest version 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 an SGX Quote. The QVE only supports version 3 of the SGX ECDSA Quote. Currently, the QVE only supports Quotes with CertType = 5. The Intel signed QVE will only verify Quotes generated by an Intel Signed QE. This type of certification data contains the PCK Certificate Chain in the Quote.

quote_size [In]

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

p_version[Out]:

Pointer to hold the latest version of of the supplemental data. The caller can specify the major supplemental data version based on this value. Note that the minimal support major version is 3.

p_data_size[Out]:

Pointer to hold the size of the buffer in bytes required to contain all the supplemental data. This value is passed into the quote verification API. The caller is responsible for allocating a buffer large enough to contain the supplemental data.

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:

Invalid parameter.   p_data_size must not be NULL.

SGX_QL_ERROR_QVL_QVE_MISMATCH:

Only returned when the quote verification library supports both the untrusted mode of verification and the QvE backed mode of verification.   This error indicates that the 2 versions of the verification modes are different. Most caused by using a QvE that does not match the version of the DCAP installed.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error.

3.6.5. Verify Quote Unified Version

Description

This is an updated version of API `sgx_qv_verify_quote()`, which merge SGX and TDX quote verification into one single API, also add additional SA (Intel® Adversary ID) list in supplemental data. Users are recommended to use this new API to verify quote.

This usage of this API is almost same with API `sgx_qv_verify_quote()` except supplemental data. We introduced another structure `tee_supp_data_descriptor_t` to allow user to specify supplemental data major version, then user can always get specific supplemental data version even the structure has been updated.

For multiple thread support, user still needs to set QvE load policy before calling this API, details please refer to section `sgx_qv_verify_quote()`.

```
/** Descriptor of the supplemental data requestor structure. Used when requesting supplemental data
from the DCAP quote verification API */
typedef struct _tee_supp_data_descriptor_t
{
        uint16_t major_version;            ///< Input. Major version of supplemental data
                                            ///< If == 0, then return latest version of
                                            ///< the sgx_ql_qv_supplemental_t structure
                                            ///< If <= latest supported, return the latest minor
                                            ///< version associated with that major version
                                            ///< if larger than latest supported, return an
                                            ///<SGX_QL_SUPPLEMENTAL_DATA_VERSION_NOT_SUPPORTED

        uint32_t data_size;                ///< Input. Supplemental data size of `p_data`, which returned
                                            ///< by API `tee_get_supplemental_data_version_and_size()`
        uint8_t *p_data;                   ///< Output. Pointer to supplemental data
}tee_supp_data_descriptor_t;
```

For example, if latest major version is 4, but caller always want the API returns supplemental version 3 in his infrastructure, then caller can specify major version 3 in structure 'tee_supp_data_descriptor_t' when calling this API.
If set major version to 0, then this API will always return latest version supplemental data.

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_qv_result_t *p_quote_verification_result,
                    sgx_ql_qe_report_info_t *p_qve_report_info,
                    tee_supp_data_descriptor_t *p_supp_data_descriptor);
```

Parameters

p_quote [In]

Pointer to an SGX/TDX Quote. The QVE only supports version 3 of the SGX/TDX ECDSA Quote. Currently, the QVE only supports Quotes with CertType = 5. The Intel signed QVE will only verify Quotes generated by an Intel Signed QE. This type of certification data contains the PCK Certificate Chain 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 the Quote Certification Collateral provided by the caller. The quote collateral structure contains a version number. This is the data that is required to fully verify the quote. Such as the TCBInfo, QEIdentity and CRL structures, etc. If it is NULL, the DCAP library will attempt to retrieve the collateral from the platform quote provider library if available. If the platform quote provider library is not available or the collateral cannot be retrieved, this API 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 QVE will use to determine if any of the inputted collateral have expired. The expectation is that the caller has access to a trusted current source or uses a hardcoded threshold date.

p_collateral_expiration_status [Out]

Address of the outputted expiration status. This input must not be NULL. When this API returns a 0 at this address, none of the inputted collateral has expired as compared to the inputted expiration_check_date. This API will return a non-zero value when one or more of the inputted collateral has expired according to the inputted expiration_check_date. This value will contain a non-zero value if the API returns a value other than SGX_QL_SUCCESS.

p_quote_verification_result [In/Out]

Address of the outputted quote verification result. This value 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 not NULL, the user needs to provide a ‘nonce’ and application enclave’s target info, then the QVE will generate a report using the target_info provided in the sgx_ql_qe_report_info_t structure . The QVE.REPORT.REPORT_DATA = (SHA256_HASH[nonce||quote||expiration_check_date||expiration_status||verification_result||supplemental_data] ||32-0x00’s).
If NULL, the quote can still be verified on a non-SGX capable platform or by an untrusted QVL but the results cannot be cryptographically verified.

p_supp_data_descriptor[In/Out]

Pointer to tee_supp_data_descriptor_t structure. This parameter is optional, if it is NULL, then API will not return any supplemental data. If it’s not NULL, then user can specify the major version of supplemental data by setting p_supp_datal_descriptor->major_version

- If p_supp_datal_descriptor == NULL, no supplemental data is returned
- If p_supp_datal_descriptor->major_version == 0, then return the latest version of the sgx_ql_qv_supplemental_t structure
- If p_supp_datal_descriptor <= latest supported version, return the latest minor version associated with that major version
- If p_supp_datal_descriptor > latest supported version, return an 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 value is invalid

SGX_QL_QUOTE_FORMAT_UNSUPPORTED:
The inputted quote format 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 quote verifier doesn't support the certification data in the Quote. Currently, the Intel QVE only supported CertType = 5.

SGX_QL_APP_REPORT_UNSUPPORTED_FORMAT:
The quote verifier doesn't support the format of the application REPORT the Quote.

SGX_QL_QE_REPORT_UNSUPPORTED_FORMAT:
The quote verifier doesn't support the format of the application REPORT the Quote.

SGX_QL_QE_REPORT_INVALID_SIGNATURE:
The signature over the QE Report is invalid.

SGX_QL_QE_REPORT_ATT_KEY_MISMATCH:
The attestation key provided in the Quote was not produced by the QE 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 PCK Cert signature chain including 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 TCBInfo signature chain including 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 Quote's QE doesn't match the inputted expected QEIdentity.

SGX_QL_QEIDENTITY_CHAIN_ERROR:
There was an error verifying the QEIdentity signature chain including QEIdentity revocation.

SGX_QL_OUT_OF_MEMORY:
Heap memory allocation error in library or enclave.

SGX_QL_ENCLAVE_LOAD_ERROR:
Unable to load the enclaves required to initialize the attestation key. 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 application report.

SGX_QL_PLATFORM_LIB_UNAVAILABLE:

The Quote Library could not locate the platform quote provider library or one of its required APIs.

SGX_QL_UNABLE_TO_GENERATE_REPORT:

The QVE was unable to generate its own report targeting the application enclave because there is an enclave compatibility issue.

SGX_QL_NO_QUOTE_COLLATERAL_DATA :

The Quote Library was available, but the quote library could not retrieve the data.

SGX_QL_ERROR_QVL_QVE_MISMATCH:

Only returned when the quote verification library supports both the untrusted mode of verification and the QvE backed mode of verification. This error indicates that the 2 versions of the verification modes are different. Most caused by using a QvE that does not match the version of the DCAP installed.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

SGX_QL_UNKNOWN_MESSAGE_RESPONSE:

Unexpected error from the attestation infrastructure while retrieving the platform data.

SGX_QL_ERROR_MESSAGE_PARSING_ERROR

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

SGX_QL_PLATFORM_UNKNOWN

This platform is an unrecognized SGX platform.

3.6.6. Get Quote Verification Collateral

Description

In current quote verification implementation, verification owner can ask QVL (Quote Verification Library) to retrieve collateral automatically internally or provide collateral buffer explicitly.
If the owner of the quote verification needs to get collaterals before quote verification, e.g. Internet is not available during quote verification. Then this API will help verification owner to connect PCCS (Or Intel PCS) to retrieve collateral.

provide a different quote verification policy beyond the policy enforced by the sgx_qv_verify_quote() API, the caller can request the sgx_qv_verify_quote() API to return supplemental data. This supplemental data can be used to implement that alternate policy.

Syntax

```
quote3_error_t tee_qv_get_collateral(const uint8_t *p_quote,
                      uint32_t quote_size,
                      uint8_t **pp_quote_collateral,
                      uint32_t *p_collateral_size);
```

Parameters

p_quote [In]

Pointer to an SGX Quote. The QVE only supports version 3 of the SGX ECDSA Quote. Currently, the QVE only supports Quotes with CertType = 5. The Intel signed QVE will only verify Quotes generated by an Intel Signed QE. This type of certification data contains the PCK Certificate Chain in the Quote.

quote_size [In]

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

pp_quote_collateral[Out]:

Pointer to the pointer which hold the quote verification collateral buffer, including CRL, TCB info, QE identity etc. Please refer to structure ` sgx_ql_qve_collateral_t` for detail. API internal logic will try to allocate enough memory to hold the collateral buffer, and you need to call API `tee_qv_free_collateral()` to free the buffer after use.

p_collateral_size[Out]:

Pointer to hold the collateral buffer size in bytes.

Return Values

SGX_QL_SUCCESS:

Successfully to free the collateral buffer

SGX_QL_ERROR_INVALID_PARAMETER:

Invalid parameter. p_quote, pp_quote_collateral and p_collateral_size must not be NULL. Or quote type is not correct, only support SGX and TDX quote so far.

SGX_QL_OUT_OF_MEMORY:

Heap memory allocation error in library or enclave.

SGX_QL_PLATFORM_LIB_UNAVAILABLE:

Cannot find quote provider library on the system.

SGX_QL_NO_QUOTE_COLLATERAL_DATA:

Cannot get quote collateral from PCCS or Intel PCS.

3.6.7. Free Quote Verification Collateral

Description

Free the quote verification collateral buffer allocated by API tee_qv_get_collateral() API.
This API was added to the QVL (Quote Verification Library) to allow user to retrieve and free quote verification collateral buffer, instead of asking QVL to handle the collateral buffer internally.

Syntax

```
quote3_error_t tee_qv_free_collateral(uint8_t *p_quote_collateral);
```

Parameters

p_quote_collateral[In]:

Pointer to hold the collateral buffer in bytes which allocated by API `tee_qv_get_collateral`().

Return Values

SGX_QL_SUCCESS:

Successfully to free the collateral buffer

SGX_QL_ERROR_INVALID_PARAMETER:

Invalid parameter. p_quote_collateral must not be NULL.

SGX_QL_QUOTE_FORMAT_UNSUPPORTED:

The collateral buffer type is not supported. Only SGX (0x0) and TDX (0x81) are supported now.

3.6.8. Extract FMSPC from Quote

Description

This function takes SGX/TDX quote as input and extract the FMSPC (Family-Model-Stepping-Platform-CustomSKU) from PCK certificate chain in cert_type 5 quote. You need to allocate a buffer which equal or larger than 6 bytes, then this function will copy the FMSPC value to the memory you allocated. Note that the out FMSPC is hex-encoded string (6 bytes). Finally, you should 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 an SGX/TDX Quote. Only Quote with CertType = 5 which contains PCK Certificate Chain is supported.

quote_size[In]:

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

p_fmspc_from_quote[In/Out]:

Pointer to hold the FMSPC buffer, which allocated by user.

fmspc_from_quote_size[In]:

Size of the buffer pointed to p_fmspc_from_quote, must equal or larger than 6 bytes.

Return Values

SGX_QL_SUCCESS:

Successfully to extract the FMSPC from quote, which is hex-encoded string representation (6 bytes).

SGX_QL_ERROR_INVALID_PARAMETER:

Invalid parameter. p_quote and p_fmspc_from_quote must not be NULL, and fmspc_from_size must larger or equal to 6 bytes.

SGX_QL_PCK_CERT_CHAIN_ERROR:

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

SGX_QL_QUOTE_CERTIFICATION_DATA_UNSUPPORTED:

The certification type is not supported in the quote, which means PCK certificate chain doesn't exist or not correct.

3.6.9. Verify Quote with Supplemental Data

Description

This is an example API that will implement an alternative verification policy when the sgx_qv_verify_quote() return SGX_QL_SUCCESS and *p_quote_verification_result contains a non-terminal verification result**.** ***There are currently no plans to implement this in the DCAP library***.

If the quote verifier (CSP or relying party) needs to implement a different verification policy than the one provided in the sgx_qv_verify_quote(), they can implement an API similar to this one. This API is a simple example, but the implementer may consider inputting the QVE report to verify the QVE inputs and results. It may also consider returning a REPORT of its own so that the caller can verify that an enclave owned by the alternate verifier generated the alternative verification result.

- A quote verifier that only needs to know if a platform may be vulnerable to a targeted SA, check:
    1. Find the targeted SA at https://www.intel.com/content/www/us/en/security-center/default.html and record the 'Original Release' date.
    2. Platform may be vulnerable to the SA if the platform's TCB evaluation has a 'tcbDate' less-than-or-equal to date recorded in step 1.

- A quote verifier that only needs to know if the collateral used 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'. (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'.
    3. Determining freshness:
        i. Compare the supplemental 'tcb_eval_dataset_num' to the recorded 'tcbEvalDataSetNumber' from the TCBInfo/QEidentity from step 1. If 'tcb_eval_dataset_num' is greater-than-or-equal-to the 'tcbEvalDataSetNumber', then the TCBInfo and QEIdentity are 'fresh'

ii. Compare the supplemental ‘pck_crl_num’ to the recorded CRLNum from step 2, if ‘pck_crl_num’ is greater-than-or-equal–to the CRLNum, then the PCK CRL is ‘fresh’

Syntax

```
quote3_error_t sgx_qv_verify_quote_supplemental(
                     const uint8_t *p_supplemental_data,
                     uint32_t supplemental_data_size,
                     const time_t *p_freshness_date,
                     const time_t *p_vulnerability_date);
```

Parameters

p_supplemental_data [In]

Pointer to the supplemental data returned by the passed in by the sgx_qv_verify_quote() API. Must not be NULL. If the sgx_qv_verify_quote() API returned a terminal verification result, the data returned in p_supplemental_data will be invalid.

supplemental_data_size [In]

Number of bytes in the buffer pointed to by p_supplemental_data.

freshness_date [In]

Optional parameter. If NULL, vulnerability_date must not be NULL. If not NULL, all collateral used in the sgx_qv_verify_quote() must have an issue date later than or equal to the freshness date. If any of the collaterals’ issue dates precedes the freshness date, this API 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 have the prescribed mitigations for all SGX SAs published before the vulnerability_date. If the platform doesn’t have the prescribed mitigations for all SGX SAs published before this date, then this API will return SGX_QL_PLATFORM_TCB_PREDATES_VULNERABILITY_DATE

Return Values

SGX_QL_SUCCESS:

The quote supplemental date checks pass. All collateral have issue dates later than or equal the inputted freshness date and

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:

Invalid parameter. p_supplemental_data is NULL, supplemental_data_size is incorrect, or both freshness_data and vulnerability date are NULL.

SGX_QL_COLLATERAL_NOT_FRESH:

At least one of the collaterals’ issue date precedes the freshness date.

SGX_QL_PLATFORM_TCB_PREDATES_VULNERABILITY_DATE

The platform may be vulnerable to one or more SGX SAs published before the vulnerability date.

3.6.10. Get QvE Identity Structure

Description

When the SGX ECDSA quote is verified by the Quote Verification Enclave, the caller can request that the QVE return an SGX REPORT. This allows the caller to cryptographically verify that the quote verification results were generated by the QVE. The caller uses REPORT based local attestation to perform this verification. Once the REPORT has been verified using local attestation, the caller needs to verify that the REPORT was generated by the expected Intel QVE. Intel signs and publishes a JSON data structure with the QVE identity information (MRSIGNER, ProdID, ISVSVN, etc) called the QVEIdentity.

This API will use the quote provider library (see Get Quote Verification Enclave Identity (QVEIdentity)) to retrieve the QVEIdentity to the caller. If the platform quote provider library cannot be found, the error SGX_QL_PLATFORM_LIB_UNAVAILABLE will be returned. If the platform quote provider library cannot retrieve the data, the SGX_QL_NO_QVE_IDENTITY_DATA error will be returned.

This API was added to the Quote Verification Library to prevent the application from having to link directly to the Platform Quote Provider Library.

Syntax

```
quote3_error_t sgx_qv_get_qve_identity (
                     uint8_t **pp_qveid,
                     uint32_t *p_qveid_size,
                     uint8_t **pp_qveid_issue_chain,
                     uint32_t *p_qveid_issue_chain_size,
                     uint8_t **pp_root_ca_crl,
                     uint16_t *p_root_ca_crl_size);
```

Parameters

pp_qve_identity [Out]

Pointer to a pointer to the UTF-8 encoded JSON string containing the QVE Identity structure. The platform quote provider library will allocate this buffer and it is expected that the caller will free it using the platform quote provider library’s sgx_ql_free_qve_identity() API.

p_qve_identity_size [Out]

The length of the string in bytes in the buffer pointed by **pp_qve_identity* including the terminating null character.

pp_qve_identity_issuer_chain [Out]

Pointer to a pointer to the UTF-8 encoded string containing the QVE Identity issuer certificate chain for SGX QVE Identity. It consists of SGX Root CA Certificate and SGX TCB Signing Certificate. The platform quote provider library will allocate this buffer and it is expected that the caller will free it using the platform quote provider library’s sgx_ql_free_qve_identity() API.

p_qve_identity_issuer_chain_size [Out]

The length of the string in bytes in the buffer pointed by `*pp_qve_identity_issuer_chain` including the terminating null character.

Return Values

SGX_QL_SUCCESS:

Data was read successfully.

SGX_QL_NO_QVE_IDENTITY_DATA:

The platform quote provider library does not have the QVE identity data available.

SGX_QL_ERROR_INVALID_PARAMETER:

The platform quote provider library rejected the input.

SGX_QL_PLATFORM_LIB_UNAVAILABLE:

The Quote Verification Library could not locate the provider library.

SGX_QL_ERROR_OUT_OF_MEMORY:

Heap memory allocation error in library or enclave.

SGX_QL_NETWORK_ERROR:

If the platform quote provider library uses the network to retrieve the QVE Identity, this error will be returned when it encounters network connectivity problems.

SGX_QL_MESSAGE_ERROR:

If the platform quote provider library uses message protocols to retrieve the QVE Identity collateral, this error will be returned when it encounters any protocol problems.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

3.6.11. Free QvE Identity Structure

Description

Called to free the quote verification enclave identity data buffer allocated by the platform quote provider library's sgx_qv_get_qve_identity() API.

This API was added to the Quote Verification Library to prevent the application from having to link directly to the Platform Quote Provider Library.

Syntax

```
quote3_error_t sgx_qv_free_qve_identity(
                    uint8_t *p_qveid,
                    uint8_t *p_qveid_issue_chain,
                    uint8_t *p_root_ca_crl);
```

Parameters

p_qve_identity [Out]

Pointer to the QVE identity that the `sgx_ql_get_qve_identity()` API returned.

p_qve_identity_issuer_chain [Out]

Pointer to the QVE identity issuer chain that the `sgx_ql_get_qve_identity()` API returned.

Return Values

SGX_QL_SUCCESS:

The pointer was successfully freed or the input pointer was NULL.

SGX_QL_ERROR_UNEXPECTED:

An unexpected internal error occurred.

3.6.12. Set Quote Verification Enclave and Dependent Library Directory Paths

Description

This API can be used to set the location and filename of SGX ECDSA Quote Verification Enclave (QVE) and the Platform Quote Provider Library (QPL) library (see Platform Quote Provider Library). The function takes the path_type ENUM and the corresponding directory path plus filename as input. The user can change the default directory path and filename used by the Quote verification library to find the enclaves and the QPL independently with this API.

If this API is not called, it will load the QVE in the local directory of the process and use the dlopen search path for the platform quote provider library (see Platform Quote Provider Library).

Syntax

```
quote3_error_t sgx_qv_set_path(
                    sgx_qv_path_type_t path_type,
                    const char *p_path);
```

Parameters

path_type [In]

The entity whose directory path and filename is specified in p_path. It can be SGX_QV_QVE_PATH or SGX_QV_QPL_PATH

p_path [In]

The directory path and filename of the entity specified in path_type specified as a NUL terminated string.

Return Values

SGX_QL_SUCCESS:

Successfully completed.

SGX_QL_ERROR_INVALID_PARAMETER:

Invalid parameter.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error occurred.

3.7. Using Supplemental Data in Trust Decisions

The supplemental data structure has other values that the relying party can use to further evaluate a Quote. Section Verify Quote with Supplemental Data describes how to use the supplemental data to verify the quote when the verifier wants to implement a less strict verification policy. But the supplemental data has other values that can be used to get other important information from the Quote to further evaluate the enclave and the platform. See the full description of the supplemental data structure in Intel® SGX DCAP Quote Wrapper Structures.

New values were added for multi-package platforms only. Multi-package platforms will have and sgx_type = 1 (scalable). When sgx_type = 1 (scalable), the security properties may be different than when the sgx_type = 0 (standard). The following supplemental data values are valid onl.y when sgx_type=1 (scalable) otherwise they are undefined.

- 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 SGX Registration Authority Service (direct registration vs indirect registration)
- smt_enabled
  - Indicate whether the platform has SMT (simultaneous multithreading) enabled

When sgx_type = 1 (scalable), the security properties are different then when the sgx_type = 0 (standard).

3.8. Enclave Identity Checking

The Verify Quote API will verify that the application's enclave REPORT was generated by an SGX enclave running with SGX protections, but it does not identify whose enclave it is. The verifier calling the Verify Quote API is responsible for checking the identity of the application enclave's report.

There are several fields in the REPORT as defined in the Enclave Report Body. The verifier will use the various fields in the REPORT to ensure its enclave generated that REPORT according to an identity policy.

The verifier must choose an enclave identity policy.

- Verify Enclave Identity
  - Strict Enclave Modification Policy
    - The MRENCLAVE is the hash over the enclave pages loaded into the SGX protected memory. Whenever the contents of the signed enclave have changed, its MRENCLAVE will change. If the MRENCLAVE is used to identify an enclave, the verifier must be updated whenever a new modified version of the enclave is released.
  - Security Enclave Modification Policy
    - The identity of the enclave can also be established by verifying the MRSIGNER and the ProdID. The MRSIGNER is the hash of the public portion of the key used to sign the enclave. The ProdID is used to distinguish different enclaves signed with the same key. Using the MRSIGNER+ProdID as the enclave identity, the verifier enclave identity policy doesn't need to be updated each time the enclave is modified.

- Verify Attributes
    - Decide which enclave attributes are important and verify they are set properly.
    - ***Production enclaves should not have the REPORT.Attribute.Debug flag set to 1.*** When the Debug flag is set, a debugger can read the enclave's memory and should not be provisioned with production secrets.

- Verify SSA Frame extended feature set

- Verify the ISV_SVN level of the enclave
    - Whenever there is a security update to an enclave, the ISV_SVN value should be increased to reflect the higher security level. When verifying an enclave's REPORT, the verifier should check that the ISV_SVN in the REPORT has a minimum trusted value.

- Verify that the ReportData contains the expected value.
    - This can be used to provide specific data from the enclave or it can be used to hold a hash of a larger block of data which is provided with the quote. The verification of the quote signature confirms the integrity of the report data (and the rest of the REPORT body).
      Additional data which is provided with a report could be:
        - Nonce - provided to the enclave and then inserted tied to the REPORT to prevent a replay of a previously used Quote.
        - Public Key - corresponding to a Private key which has been generated within the enclave.

3.9. Trusted Verification Library

This chapter presents a set of C-like APIs in a library, sgx_dcap_tvl, that an application enclave can statically link with to verify the report and the identity of the Intel® SGX ECDSA QvE (Quote Verification Enclave). Also, the application enclave can include the trusted library's sgx_dcap_tvl.edl file to allow an application to make ECALLs directly to the trusted library. The library, header file, and EDL files are delivered with the Intel® SGX SDK installer.

3.9.1.QvE Report Verification and Identity Check

Description

This function allows a user's enclave to more easily verify the QvE REPORT returned in the p_qve_report_info paramer in the Verify Quote API was generated by the Intel QvE at an expected TCB level. This API will use SGX local attestation to verify the QvE REPORT. This API does not verify the Quote itself.
In order to verify QvEs' identity, caller needs to provide a QvE ISV SVN as a threshold. If the QvE ISV SVN in the REPORT is smaller than the threshold, the API will return error SGX_QL_QVE_OUT_OF_DATE. Intel will publish the latest QvE Identity JSON structure and its certificate chain in the Intel® PCS. The caller can get latest QvE ISV SVN from the JSON structure. The sgx_dcap_tvl will hardcode the Intel® SGX QvE enclave's identity values, including latest QvE ISVSVN, misc_select and misc_select_mast, attribute and attribute mask, MRSIGNER, PRODID within the API. (See Enclave Identity Checking for more information

on enclave identity checking). This API will check the QvE REPORT against these hardcoded values. If this API finds any identity mismatches in the QvE REPORT, e.g. the report includes a different MRSIGNER to hardcoded value, this API will return an error code.

In addition, the caller needs to cache the outputs of the Verify Quote API. The outputs should contain the QvE REPORT, the expiration_check_date, the expiration status of verification collateral, the quote verification result and the optional supplemental data. Then, the caller provides this QvE output, the ISV SVN threshold and ECDSA quote. The API will verify the QvE's identity info by following 3 steps.

a. Verify the QvE's REPORT and REPORT.report_data
b. Compare the QvE identity values in the REPORT with hardcoded identity values
c. Make sure the QvE's ISV SVN in the REPORT is greater or equal to threshold provided by caller

Syntax

```
quote3_error_t sgx_tvl_verify_qve_report_and_identity(
                     const uint8_t *p_quote,
                     uint32_t quote_size,
                     const sgx_ql_qe_report_info_t *p_qve_report_info,
                     time_t expiration_check_date,
                     uint32_t collateral_expiration_status,
                     sgx_ql_qv_result_t quote_verification_result,
                     const uint8_t *p_supplemental_data,
                     uint32_t supplemental_data_size,
                     sgx_isv_svn_t qve_isvsvn_threshold);
```

Parameters

p_quote [In]

Pointer to an SGX Quote. The QvE only supports version 3 of the SGX ECDSA Quote. Currently, the QVE only supports Quotes with CertType = 5. This type of certification data contains the PCK Certificate Chain in the Quote. The Intel signed QVE will only verify Quotes generated by an Intel Signed QE.

quote_size [In]

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

p_qve_report_info [In]

An output of the Verify Quote API.   It must contain QvE REPORT and a nonce.

expiration_check_date [In]

This is the date to verify QvE REPORT.report_data, you must use same value for this API to the "expiration_check_data" input provided to the Verify Quote API.

collateral_expiration_status [In]

An output of the Verify Quote API which indicates the quote verification collateral's expiration status.

quote_verification_result [In]

An output of the Verify Quote API which indicates the quote verification result.

p_supplemental_data [In]

An output of the Verify Quote API. It is a pointer to the supplemental data. If the Verify Quote API was provided a non-NULL pointer to p_supplemental_data, then this API must be provided the outputted data. This parameter is optional.

supplemental_data_size [In]

Size of the buffer pointed to by p_supplemental_data (in bytes). The value should match the value retuned by the Verify Quote API.

qve_isvsvn_threshold [In]

The threshold of QvE ISVSVN, the ISVSVN of QvE used to verify the Quote must be greater or equal to this threshold. You can get latest QvE ISVSVN from the QvE Identity (JSON) structure provided by the Intel PCS

Return Values

SGX_QL_SUCCESS:

Successfully evaluated the QvE report and Identity.

SGX_QL_INVALID_PARAMETER:

One of the input parameters value is invalid.

SGX_QL_ERROR_REPORT:

The QvE report can NOT be verified.

SGX_QL_ERROR_UNEXPECTED:

Unexpected error during verify QvE report and Identity.

SGX_QL_QVEIDENTITY_MISMATCH:

The QvE identity info from report doesn't match to value in sgx_dcap_tvl.

SGX_QL_QVE_OUT_OF_DATE:

The input QvE ISV SVN threshold is smaller than actual QvE ISV SVN.

A. Data Structures

A.1. Quote Library Data Structures

```
typedef enum {
    SGX_QL_PERSISTENT, ///< AEs areinitialized on first use and reused until process
                       ///< ends.
    SGX_QL_EPHEMERAL,  ///< AEs are initialized and terminated on every quote.
                       ///< If a previous QE exists, it is stopped & restarted before
                       ///< quoting.
    SGX_QL_DEFAULT = SGX_QL_PERSISTENT
} sgx_ql_request_policy_t;



/** Identifies the type of certification data used in the Quote */
typedef struct _sgx_ql_certification_data_t {
    uint16_t cert_key_type;  ///< The type of certification key used to sign the QE3
                             ///<Report and Att key hash (ECDSA_ID+Authentication
                             ///<Data).
    uint32_t size;           ///< Size of the data structure for the cert_key_type
                             ///<information.
    uint8_t certification_data[];  ///< Certification data associated with the
cert_key_type
} sgx_ql_certification_data_t;




/** Enumerates the different certification data types used to describe the signer of
the attestation key */
typedef enum {
    PPID_CLEARTEXT = 1,          ///< Clear PPID + CPU_SVN, PvE_SVN, PCE_SVN, PCE_ID
    PPID_RSA2048_ENCRYPTED = 2,  ///< RSA-2048-OAEP Encrypted PPID + CPU_SVN, PvE_SVN,
                                 ///< PCE_SVN, PCE_ID
    PPID_RSA3072_ENCRYPTED = 3,  ///< RSA-3072-OAEP Encrypted PPID + CPU_SVN, PvE_SVN,
                                 ///<PCE_SVN, PCE_ID
     PCK_CLEARTEXT = 4,          ///< Clear PCK Leaf Cert
     PCK_CERT_CHAIN = 5,         ///< Full PCK Cert chain
                                 ///< (trustedRootCaCert||intermediateCa||pckCert)
    ECDSA_SIG_AUX_DATA = 6,      ///< Indicates the contents of the
                                 ///< CERTIFICATION_INFO_DATA contains the
                                 ///< ECDSA_SIG_AUX_DATA of another Quote.
} sgx_ql_cert_key_type_t;
```

A.2. Core Generic Quote Wrapper Structures

```
/** Enumerates the different attestation key algorithms */
typedef enum {
    SGX_ALG_EPID = 0,       ///< EPID 2.0 - Anonymous
    SGX_ALG_RESERVED_1 = 1, ///< Reserved
    SGX_ALG_ECDSA_P256 = 2, ///< ECDSA-256-with-P-256 curve, Non - Anonymous
    SGX_ALG_ECDSA_P384 = 3, ///< ECDSA-384-with-P-384 curve, Non-Anonymous
    SGX_ALG_MAX = 4
} sgx_ql_attestation_algorithm_id_t;

/** Describes the header that contains the list of attestation keys supported by a
given verifier */
typedef struct _sgx_ql_att_key_id_list_header_t {
    uint16_t      id;           ///< Structure ID
    uint16_t      version;      ///< Structure version
    uint32_t      num_att_ids;  ///< Number of 'Attestation Key Identifier' Elements
}sgx_ql_key_id_list_header_t;
```

```
/** Describes a single attestation key.  Contains both QE identity and the attestation
algorithm ID. */
typedef struct _sgx_ql_att_key_id_t {
   uint16_t    id;                         ///< Structure ID
   uint16_t    version;                    ///< Structure version
   uint16_t    mrsigner_length;            ///< Number of valid bytes in
                                           ///< MRSIGNER.
   uint8_t     mrsigner[48];               ///< SHA256 or SHA384 hash of the
                                           ///< Public key that signed the QE.
                                           ///< The lower bytes contain
                                           ///< MRSIGNER.  Bytes beyond
                                           ///< mrsigner_length '0'
   uint32_t    prod_id;                    ///< Legacy Product ID of the QE
   uint8_t     extended_prod_id[16];       ///< Extended Product ID or the QE.
                                           ///< All 0s for legacy format
                                           ///< enclaves.
   uint8_t     config_id[64];              ///< Config ID of the QE.
   uint8_t     family_id[16];              ///< Family ID of the QE.
   sgx_ql_attestation_algorithm_id_t algorithm_id; ///< Identity of the attestation
                                                    ///<key algorithm.
}sgx_ql_att_key_id_t;

/** The full data structure passed to the platform by the verifier. It will list all
of the attestation algorithms and QE's supported by the verifier */
typedef struct _ sgx_ql_att_key_id_list_t {
   sgx_ql_att_key_id_list_header_t   header;   ///< Header for the attestation key
                                               ///< ID list provided by the quote
                                               ///< verifier.
   sgx_ql_att_key_id_t            id_list[];   ///< Place holder for the
                                               ///< attestation ID list.
}sgx_ql_att_key_id_list_t;


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.3. Intel® SGX DCAP Quote Wrapper Structures

```
/** Used to describe the PCK Cert for a platform */
typedef struct _sgx_ql_pck_cert_id_t
{
   uint8_t *p_qe3_id;                   ///< The QE_ID used to identify the platform
                                        ///< for PCK Cert Retrieval
   uint32_t qe_id_size;                 ///< The Size of the QE_ID (currently 16
                                        ///< bytes)
   sgx_cpu_svn_t *platform_cpu_svn;     ///< Pointer to the platform raw CPUSVN
   sgx_cpu_svn_t *platform_pce_isv_svn; ///< Pointer to the platform raw PCE ISVSVN
   uint8_t *p_encrypted_ppid;           ///< Pointer to the encrypted PPID (Optional)
   uint32_t encrypted_ppid_size;        ///< Size of encrypted PPID.
   uint8_t crypto_suite;                ///< Crypto algorithm used to encrypt the
                                        ///<PPID (currently only
                                        ///<PCE_ALG_RSA_OAEP_3072 = 1 supported)
   uint16_t pce_id;                     ///< Identifies the PCE-Version used to
                                        ///<generate the encrypted PPID.
}sgx_ql_pck_cert_id_t;

/** Contains valid versions of the sgx_ql_config_t data structure. */
typedef enum _sgx_ql_config_version_t
{
   SGX_QL_CONFIG_VERSION_1 = 1,
}sgx_ql_config_version_t;
```

```
/** Contains the certification data used by the quoting library to certify the
attestation key and the certification data required to generate the final quote. */
typedef struct _sgx_ql_config_t
{
   sgx_ql_config_version_t version;
   sgx_cpu_svn_t cert_cpu_svn;        ///< The CPUSVN used to generate the PCK
                                      ///< Signature that certifies the attestation
                                      ///< key.
   sgx_isv_svn_t cert_pce_isv_svn;    ///< The PCE ISVSVN used to generate the PCK
                                      ///<Signature that certifies the attestation
                                      ///<key.
   uint32_t p_cert_data_size;         ///< The size of the buffer that
                                      ///<p_cert_data points to
   uint8_t *p_cert_data;              ///< The certification data used for the quote.
}sgx_ql_config_t;

/** Contains the possible values of the quote verification result. */
typedef enum _sgx_ql_qv_result_t
{
  SGX_QL_QV_RESULT_OK = 0,               ///< The Quote verification passed and
                                         ///< is at 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 additional
                                         ///< configuration of the SGX platform
                                         ///< may be needed.
  SGX_QL_QV_RESULT_OUT_OF_DATE,          ///< The Quote is good but the TCB
                                         ///< level of the platform is out of
                                         ///< date.
                                         ///< The platform needs patching to be
                                         ///< at the latest TCB level.
  SGX_QL_QV_RESULT_OUT_OF_DATE_CONFIG_NEEDED, ///< The Quote is good but the TCB
                                         ///< level of the platform is out of
                                         ///< date and additional configuration
                                         ///< of the SGX Platform at its
                                         ///< current patching level may be
                                         ///< needed. The platform needs
                                         ///< patching to be at the latest TCB
                                         ///< level.
  SGX_QL_QV_RESULT_INVALID_SIGNATURE,    ///< The signature over the
                                         ///< application
                                         ///< report 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 error in processing the
                                         ///< Quote.
  SGX_QL_QV_RESULT_SW_HARDENING_NEEDED,///< The TCB level of the platform is
                                         ///< up to date, but SGX SW Hardening
                                         ///< of the enclave is needed
  SGX_QL_QV_RESULT_CONFIG_AND_SW_HARDENING_NEEDED, ///< The TCB level of the platform
                                         ///< is up to date, but additional
                                         ///< configuration of the platform
                                         ///< at its current patching level
                                         ///< may be needed. Moreover, SGX SW
                                         ///< Hardening of the enclave is also
                                         ///< needed

} 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;


typedef struct _tee_supp_data_descriptor_t
{
        uint16_t major_version;              ///< Input. Major version of supplemental data
                                              ///< If == 0, then return latest version of
                                              ///< the sgx_ql_qv_supplemental_t structure
                                              ///< If <= latest supported, return the latest minor
```

```
                                        ///< version associated with that major version
                                        ///< if larger than latest supported, return an
                                        ///<SGX_QL_SUPPLEMENTAL_DATA_VERSION_NOT_SUPPORTED

    uint32_t data_size;                ///< Input. Supplemental data size of `p_data`, which returned
                                        ///< by API `tee_get_supplemental_data_version_and_size()`
    uint8_t *p_data;                   ///< Output. Pointer to supplemental data
}tee_supp_data_descriptor_t;
```

```
/** */
typedef enum
{
    SGX_QL_QE3_PATH,                   ///< Indicate overriding the default QE3
                                       ///< location and filename
    SGX_QL_PCE_PATH,                   ///< Indicate overriding the default PCE
                                       ///< location and filename
    SGX_QL_QPL_PATH                    ///< Specify the location and file name of
                                       ///< The platform quote provider library
                                       ///< used by quote generation. Overrides the
                                       ///< dlopen search path and default
                                       ///< filename.
} sgx_ql_path_type_t;

/** */
typedef enum
{
    SGX_QV_QVE_PATH,                   ///< Indicate overriding the default QVE
                                       ///< location and filename
    SGX_QV_QPL_PATH                    ///< Specify the location and file name of
                                       ///< The platform quote provider library
                                       ///< used by quote generation. Overrides the
                                       ///< dlopen search path and default
                                       ///< filename.
} sgx_qv_path_type_t;
```

A.4. Quote Format

The new quote structure to support ECDSA will have a version number of 3. The existing Intel® Enhanced Privacy ID (EPID) Quote structure with a version number of 2 will still exist. The version 3 quote does not specifically support EPID but was designed so that the header is compatible based on size and the first 5 fields of the header.

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

Name	Size (bytes)	Type	Description
Quote Header	48	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).
ISV Enclave Report	384	Enclave Report Body	Report of the attested *ISV Enclave*. The CPUSVN and ISVSVN is the TCB when the quote is generated.

			The REPORT.ReportData is defined by the ISV but should provide quote replay protection if required.
Quote Signature Data Len	4	uint32_t	Size of the Quote Signature Data structure
Quote Signature Data	Variable	Signature Dependent	Variable-length data containing the signature and supporting data. E.g. ECDSA 256-bit Quote Signature Data Structure

Table 2: High-Level Quote Structure

Name	**Size (bytes)**	**Type**	**Description**
Version	2	Integer	Version of the *Quote* data structure. • *Value*: 3
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, EPID is moved to version 3 quotes.)
Reserved	4	Byte Array	Reserved field. • *Value*: 0
QE SVN	2	Integer	Security Version of the Quoting Enclave currently loaded on the platform.
PCE SVN	2	Integer	Security Version of the Provisioning Certification Enclave currently loaded on the platform.
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 DCAP library, the first 16 bytes contain a QE identifier that is used to link a PCK Cert to an

			Enc(PPID). This identifier is consistent for every quote generated with this QE on this platform.

Table 3: Quote Header

Name	**Size (bytes)**	**Type**	**Description**
ISV Enclave Report Signature	64	ECDSA P-256 Signature	ECDSA signature over the *Header* and the *Enclave Report* calculated using *ECDSA Attestation Key*.
ECDSA Attestation Key	64	ECDSA P-256 Public Key	Public part of the *ECDSA Attestation Key* generated by the *Quoting Enclave*.
QE Report	384	Enclave Report Body	Report of the *Quoting Enclave* that generated the *ECDSA 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 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*.
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 *ECDSA Attestation Key* utilized by the *QE*. For example, this may indicate the customer, geography, network, or anything pertinent to the identity of the Quoting Enclave. Size should be set to 0 if there is no additional data.
QE Certification Data	Variable	QE Certification Data	Data required to verify the QE Report Signature.

Table 4: ECDSA 256-bit Quote Signature Data Structure

Name	Size (bytes)	Type	Description
CPU SVN	16	Byte Array	Security Version of CPU (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. SECS.ATTRIBUTES used in 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 SIGSTRUCT.ATTRIBUTESMASK bit that is set, then 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 enclave signing key.
Reserved	96	Byte Array	Reserved field.
ISV ProdID	2	Integer	Enclave Product ID. The ISV should configure a unique ISVProdID for each product which may want to share sealed data between enclaves signed with a specific MRSIGNER. The ISV may want to supply different data to identical enclaves signed for different products.
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. This can be used to provide specific data from the

			enclave or it can be used to hold a hash of a larger block of data which is provided with the quote. The verification of the quote signature confirms the integrity of the report data (and the rest of the REPORT body).

Table 5: Enclave Report Body

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

Table 6: ECDSA P-256 Signature

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.

Table 7: ECDSA P-256 Public Key

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

Table 8: QE Authentication Data

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

			- 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: Intel® SGX Quote (currently not supported) - 7: PLATFORM_MANIFEST (currently not supported)

Table 9: QE Certification Data

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 Libraries API Return Result Mapping (quote3_error_t)

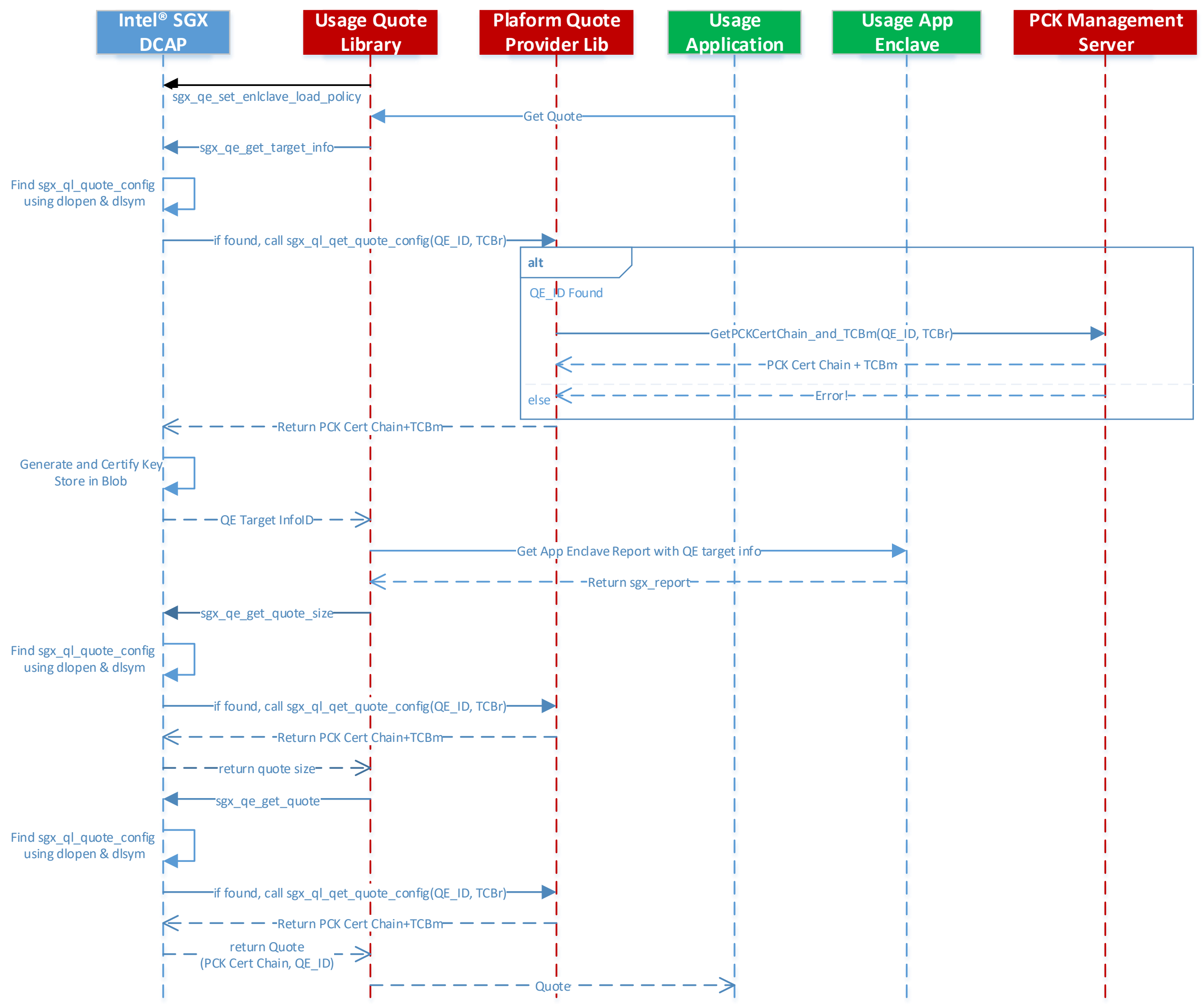
SGX_QL_SUCCESS	0x0000
SGX_QL_ERROR_UNEXPECTED	0xe001
SGX_QL_ERROR_INVALID_PARAMETER	0xe002
SGX_QL_ERROR_OUT_OF_MEMORY	0xe003
SGX_QL_ERROR_ECDSA_ID_MISMATCH	0xe004
SGX_QL_PATHNAME_BUFFER_OVERFLOW_ERROR	0xe005
SGX_QL_FILE_ACCESS_ERROR	0xe006
SGX_QL_ERROR_STORED_KEY	0xe007
SGX_QL_ERROR_PUB_KEY_ID_MISMATCH	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

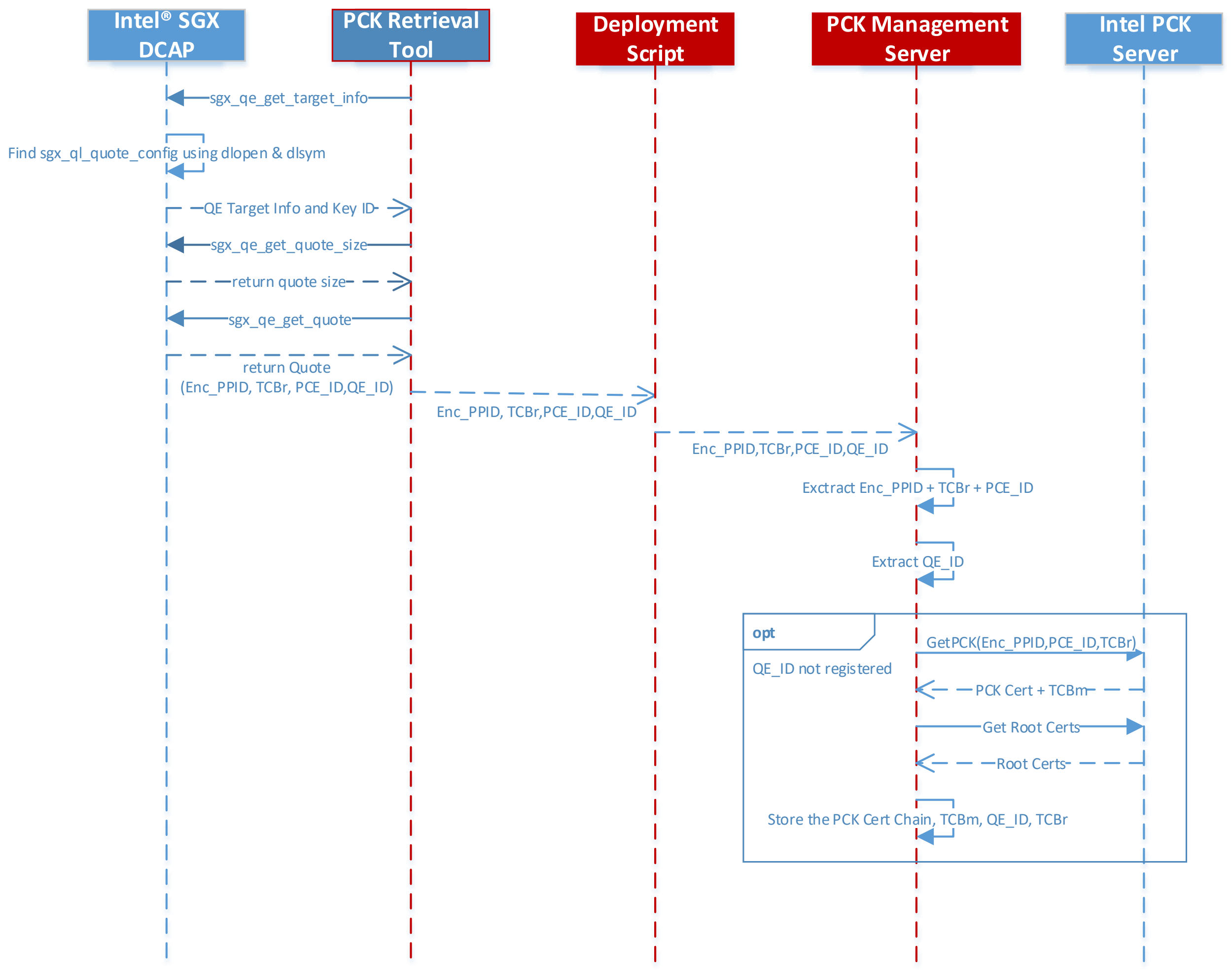
C. Sample Sequence Diagrams

C.1. Sample Quote Generation Sequence Diagram for the Intel® SGX DCAP APIs

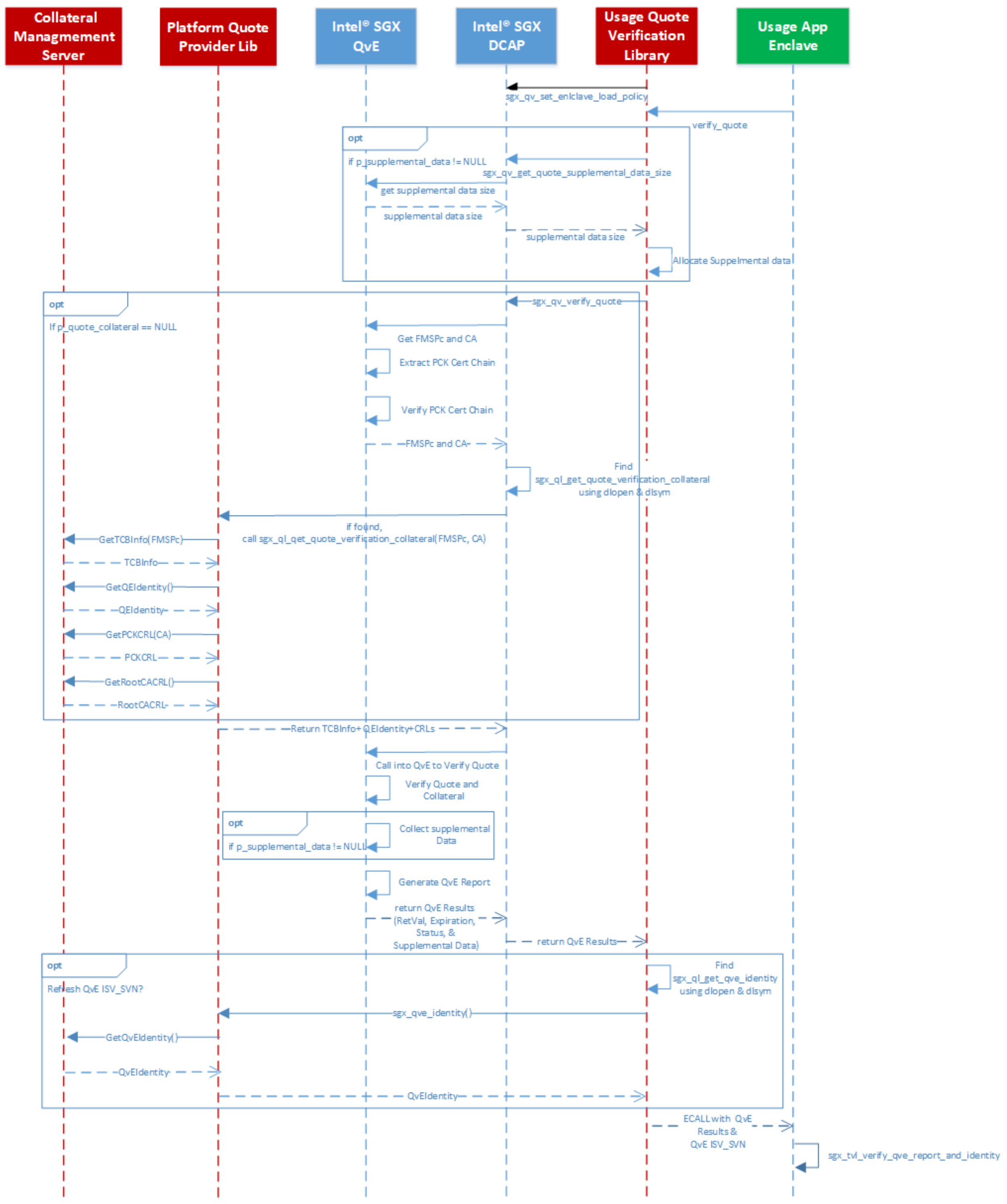


Intel® SGX DCAP
Usage Quote Library
Plaform Quote Provider Lib
Usage Application
Usage App Enclave
PCK Management Server
sgx_qe_set_enlclave_load_policy
Get Quote
sgx_qe_get_target_info
Find sgx_ql_quote_config using dlopen & dlsym
if found, call sgx_ql_get_quote_config(QE_ID, TCBr)
alt
QE_ID Found
GetPCKCertChain_and_TCBm(QE_ID, TCBr)
PCK Cert Chain + TCBm
else
Error!
Return PCK Cert Chain+TCBm
Generate and Certify Key, Store in Blob
QE Target InfoID
Get App Enclave Report with QE target info
Return sgx_report
sgx_qe_get_quote_size
Find sgx_ql_quote_config using dlopen & dlsym
if found, call sgx_ql_get_quote_config(QE_ID, TCBr)
Return PCK Cert Chain+TCBm
return quote size
sgx_qe_get_quote
Find sgx_ql_quote_config using dlopen & dlsym
if found, call sgx_ql_get_quote_config(QE_ID, TCBr)
Return PCK Cert Chain+TCBm
return Quote (PCK Cert Chain, QE_ID)
Quote

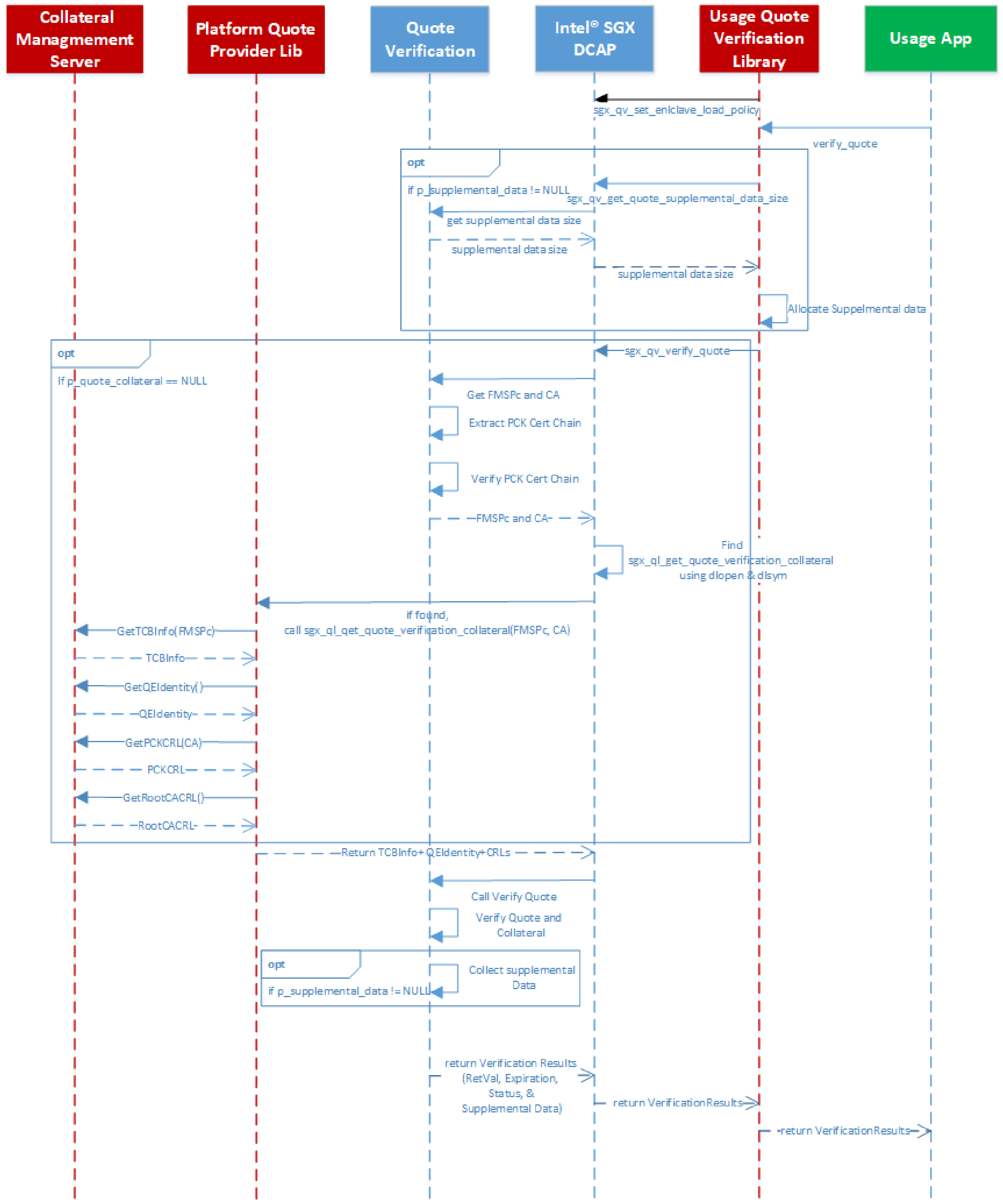
C.2. Deployment Phase PCK Retrieval Sequence Diagram


Intel® SGX DCAP
PCK Retrieval Tool
Deployment Script
PCK Management Server
Intel PCK Server
sgx_qe_get_target_info
Find sgx_ql_quote_config using dlopen & dlsym
QE Target Info and Key ID
sgx_qe_get_quote_size
return quote size
sgx_qe_get_quote
return Quote (Enc_PPID, TCBr, PCE_ID,QE_ID)
Enc_PPID, TCBr,PCE_ID,QE_ID
Enc_PPID,TCBr,PCE_ID,QE_ID
Exctract Enc_PPID + TCBr + PCE_ID
Extract QE_ID
opt
QE_ID not registered
GetPCK(Enc_PPID,PCE_ID,TCBr)
PCK Cert + TCBm
Get Root Certs
Root Certs
Store the PCK Cert Chain, TCBm, QE_ID, TCBr

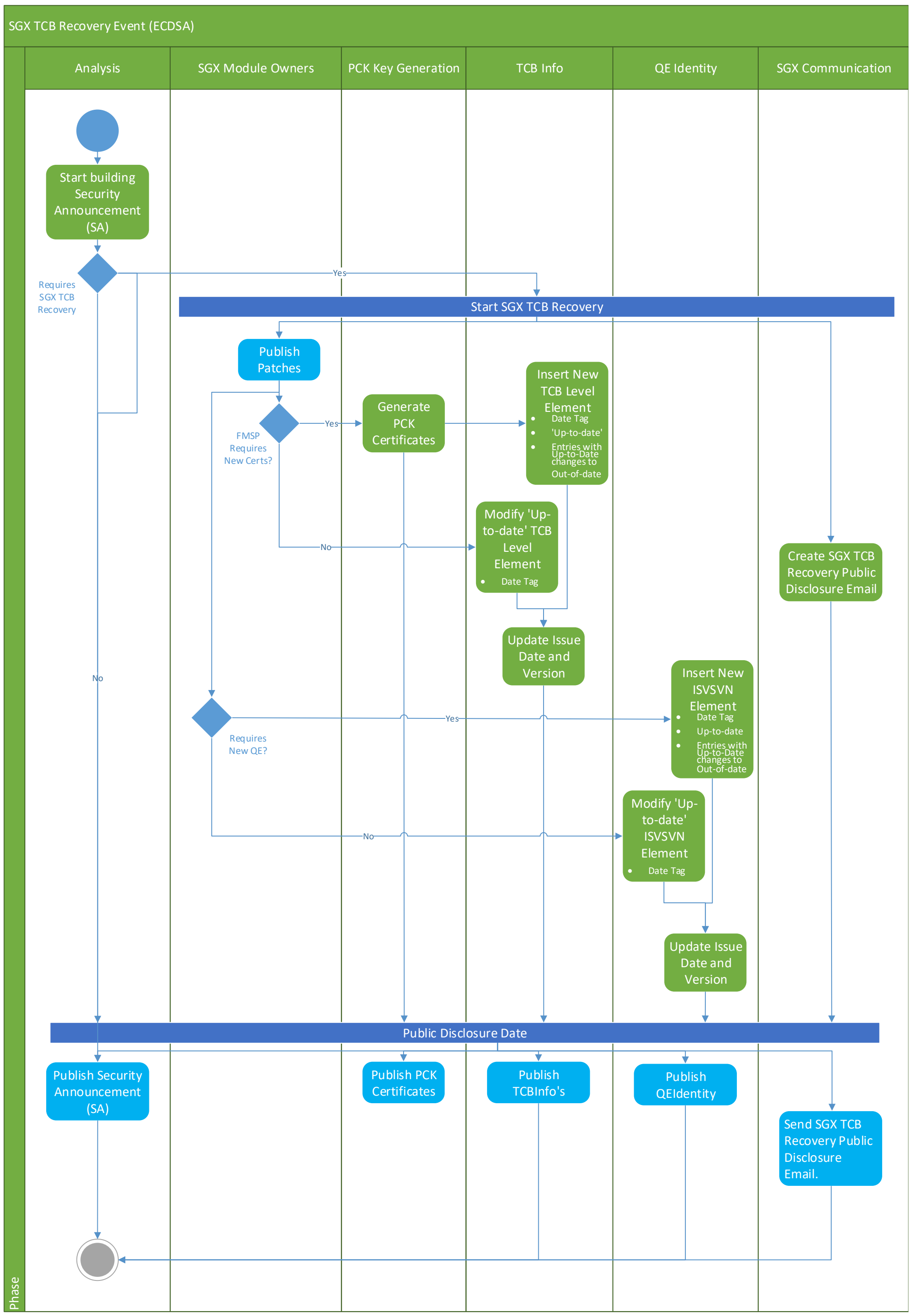
C.3. QvE Based Quote Verification Sequence Diagram


Collateral Managmement Server
Platform Quote Provider Lib
Intel® SGX QvE
Intel® SGX DCAP
Usage Quote Verification Library
Usage App Enclave
sgx_qv_set_enlclave_load_policy
verify_quote
opt
if p_supplemental_data != NULL
sgx_qv_get_quote_supplemental_data_size
get supplemental data size
supplemental data size
supplemental data size
Allocate Suppelmental data
opt
If p_quote_collateral == NULL
sgx_qv_verify_quote
Get FMSPc and CA
Extract PCK Cert Chain
Verify PCK Cert Chain
FMSPc and CA
Find sgx_ql_get_quote_verification_collateral using dlopen & dlsym
if found, call sgx_ql_get_quote_verification_collateral(FMSPc, CA)
GetTCBInfo(FMSPc)
TCBInfo
GetQEIdentity()
QEIdentity
GetPCKCRL(CA)
PCKCRL
GetRootCACRL()
RootCACRL
Return TCBInfo+QEIdentity+CRLs
Call into QvE to Verify Quote
Verify Quote and Collateral
opt
if p_supplemental_data != NULL
Collect supplemental Data
Generate QvE Report
return QvE Results (RetVal, Expiration, Status, & Supplemental Data)
return QvE Results
opt
Refresh QvE ISV_SVN?
Find sgx_ql_get_qve_identity using dlopen & dlsym
sgx_qve_identity()
GetQvEIdentity()
QvEIdentity
QvEIdentity
ECALL with QvE Results & QvE ISV_SVN
sgx_tvl_verify_qve_report_and_identity

C.4. Non-QvE Based Quote Verification Sequence Diagram


Collateral Managmement Server
Platform Quote Provider Lib
Quote Verification
Intel® SGX DCAP
Usage Quote Verification Library
Usage App
sgx_qv_set_enlclave_load_policy
verify_quote
opt
if p_supplemental_data != NULL
sgx_qv_get_quote_supplemental_data_size
get supplemental data size
supplemental data size
supplemental data size
Allocate Suppelmental data
opt
If p_quote_collateral == NULL
sgx_qv_verify_quote
Get FMSPc and CA
Extract PCK Cert Chain
Verify PCK Cert Chain
FMSPc and CA
Find sgx_ql_get_quote_verification_collateral using dlopen & dlsym
if found, call sgx_ql_get_quote_verification_collateral(FMSPc, CA)
GetTCBInfo(FMSPc)
TCBInfo
GetQEIdentity()
QEIdentity
GetPCKCRL(CA)
PCKCRL
GetRootCACRL()
RootCACRL
Return TCBInfo+QEIdentity+CRLs
Call Verify Quote
Verify Quote and Collateral
opt
if p_supplemental_data != NULL
Collect supplemental Data
return Verification Results (RetVal, Expiration, Status, & Supplemental Data)
return VerificationResults
return VerificationResults

C.5. TCB Recovery Intel Activity Diagram – Quote Verification Collateral


SGX TCB Recovery Event (ECDSA)
Analysis
SGX Module Owners
PCK Key Generation
TCB Info
QE Identity
SGX Communication
Phase
Start building Security Announcement (SA)
Requires SGX TCB Recovery
Yes
No
Start SGX TCB Recovery
Publish Patches
FMSP Requires New Certs?
Yes
No
Generate PCK Certificates
Insert New TCB Level Element
Date Tag
'Up-to-date'
Entries with Up-to-Date changes to Out-of-date
Modify 'Up-to-date' TCB Level Element
Date Tag
Update Issue Date and Version
Create SGX TCB Recovery Public Disclosure Email
Requires New QE?
Yes
No
Insert New ISVSVN Element
Date Tag
Up-to-date
Entries with Up-to-Date changes to Out-of-date
Modify 'Up-to-date' ISVSVN Element
Date Tag
Update Issue Date and Version
Public Disclosure Date
Publish Security Announcement (SA)
Publish PCK Certificates
Publish TCBInfo's
Publish QEIdentity
Send SGX TCB Recovery Public Disclosure Email.

🟦 External Actions

🟩 Internal Actions

4. Disclaimer and Legal Information