ETSI GS QKD 020 V1.1.1 (2026-06)

GROUP SPECIFICATION

Quantum Key Distribution (QKD); Protocol and data format of REST-based Interoperable Key Management System API

Disclaimer

Reference
DGS/QKD-020_InteropKMS

Keywords
API, protocol, quantum cryptography, quantum key distribution

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Quantum Key Distribution (QKD).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies a REST API that allows key management systems to interoperate to pass keys horizontally between two systems located in a common trusted node. The API enables QKD networks to serve applications that request shared secret keys from key management systems that are not linked by a contiguous chain of systems from the same vendor. It is beyond the scope of the present document to describe how the underlying QKD network agrees key material between nodes. URI formats, communication protocols (HTTPS), and the JSON data format encoding of posted parameters and responses (including key material) are described. An OpenAPI description of the API is available.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found in the ETSI docbox.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] IETF RFC 9110: STD 97 (June 2022): "HTTP Semantics".

[2] IETF RFC 8259: STD 90 (December 2017): "The JavaScript Object Notation (JSON) Data Interchange Format".

[3] IETF RFC 9112: STD 99 (June 2022): "HTTP/1.1".

[4] IETF RFC 8446 (August 2018): "The Transport Layer Security (TLS) Protocol Version 1.3".

[5] IETF RFC 3986: STD 66 (January 2005): "Uniform Resource Identifier (URI): Generic Syntax".

[6] IETF RFC 5234: STD 68 (October 2005): "Augmented BNF for Syntax Specifications: ABNF".

[7] IETF RFC 4648 (October 2006): "The Base16, Base32, and Base64 Data Encodings".

[8] IETF RFC 9562 (May 2024): "Universally Unique IDentifiers (UUIDs)".

[9] IETF RFC 9457 (July 2023): "Problem Details for HTTP APIs".

[10] OpenAPI Initiative: "OpenAPI Specification v3.1.2 (September 2025)".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] ETSI GR QKD 007: "Quantum Key Distribution (QKD); Vocabulary".

[i.2] ETSI GS QKD 014: "Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API".

[i.3] ETSI GS QKD 004: "Quantum Key Distribution (QKD); Application Interface".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GR QKD 007 [i.1] and the following apply:

NOTE: A term defined in the present document takes precedence over the definition of the same term, if any, in ETSI GR QKD 007 [i.1].

interworking node: node of more than one QKD network that contains key management entities from each QKD network interworking within the node

key management entity: entity that manages keys in a network in cooperation with one or more other key management entities

secure application entity: entity that requests one or more keys from a key management entity for one or more applications running in cooperation with one or more other secure application entities

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
HEXDIG	HEXadecimal DIGits
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IP	Internet Protocol
JSON	JavaScript Object Notation
KME	Key Management Entity
KMS	Key Management System
PEN	Private Enterprise Number
QKD	Quantum Key Distribution
REST	REpresentational State Transfer
SAE	Secure Application Entity
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF-8	UCS Transformation Format 8 bits
UUID	Universally Unique IDentifier

4 Description

4.1 Overview

The present document specifies a communication protocol and data format for Key Management Entities (KMEs) for use by Quantum Key Distribution (QKD) networks to exchange keys when located together within a protected operational environment. Use-cases include where:

- KMEs belong to different QKD networks and are located within an interworking node.
- KMEs belong to different policy domains of a QKD network.
- KMEs are otherwise incompatible, e.g. from different vendors.

Since use-cases include interworking between different network operators and policy domains, the information that KMEs are required to exchange is intentionally limited.

A REST (REpresentational State Transfer) API is specified as a simple, scalable, widely deployed approach that is familiar to a large developer community. The REST-based API specifies the format of the URIs, the communication protocol "HTTPS" i.e. HTTP with TLS, as well as the data format and encoding of request and response content, including key material.

4.2 Operation of the API

The primary purpose of QKD networks is to deliver shared secure random keys to authorized cryptographic applications called Secure Application Entities (SAEs). APIs have been specified in ETSI GS QKD 014 [i.2] and ETSI GS QKD 004 [i.3] for the delivery of keys from a KME to an SAE.

Where there is no direct QKD link between the KMEs from which SAEs are requesting to share keys, the KMEs within QKD networks typically relay keys. They can consume QKD keys to help achieve this securely. Different sets of KMEs might not be fully compatible between sets of KMEs. Sets of KMEs could also be managed separately, e.g. by different network operators that wish to limit unnecessary sharing of internal network information.

In such cases a standardized interface is required for KMEs to pass keys horizontally between different sets of KMEs that cannot otherwise interoperate due to incompatibility or policy, etc. The API described in the present document enables key transfers between KMEs within a secure operational environment. This can enable shared keys to be delivered to two (or more) SAEs connected to KMEs in different QKD networks, policy domains, etc.

There are two primary modes of operation for the API methods described in the present document: asynchronous mode and synchronous mode. In the asynchronous mode, requests are processed in a non-blocking manner, allowing a KME to accept a request without undue delay and close the request connection by issuing a "`202`" ('Request accepted') response. Actual key processing can then be performed in the background before the KME acknowledges the result of that key processing by issuing one or more requests back to the KME that made the initial request to provide details of the outcome. This mode is particularly advantageous in high-traffic environments, or where request processing can be time consuming. It avoids blocking the requestor and minimizes the number of simultaneous open connections between KMEs. As a result, this mode can help achieve scalability as well as predictable performance under anomalous conditions. An overview message sequence diagram for the asynchronous mode is provided in Figure 4.2-1.

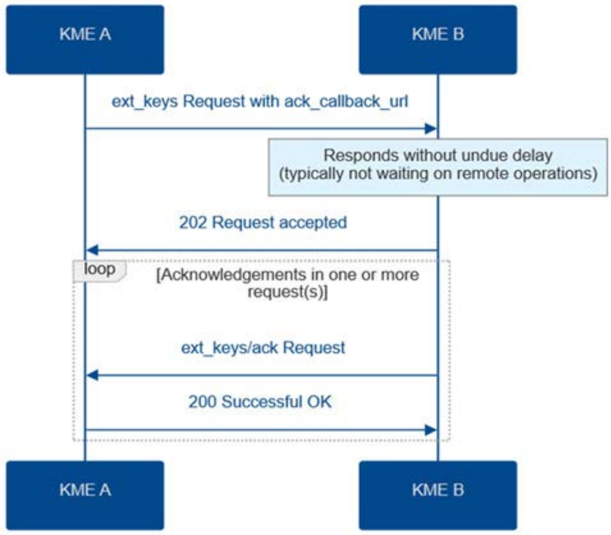

Figure 4.2-1: Overview message sequence diagram for successful required asynchronous mode

Conversely, in the synchronous mode the requestor has to wait for all parts of a request to either complete or fail before responding since all results are returned within a single response. This mode provides a more straightforward and direct interaction between KMEs and can be suitable where results can be returned almost immediately, but it can limit scalability in large or congested networks. A message sequence diagram for the optional synchronous mode is provided in Figure 4.2-2.

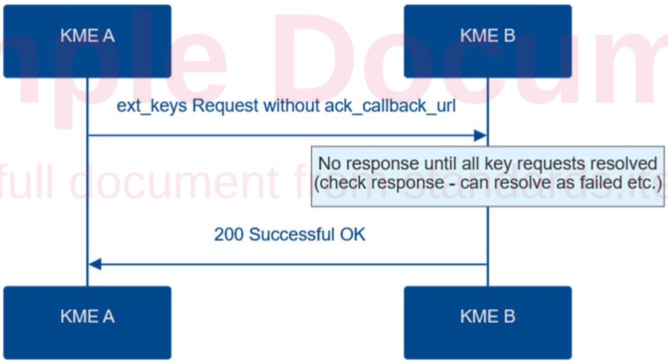

Figure 4.2-2: Overview message sequence diagram for successful optional synchronous mode

To ensure interoperability, all KMEs shall support the asynchronous mode of operation for the API methods defined in the present document.

KMEs may support the synchronous mode of operation as an optional mode in addition to the required asynchronous mode for specific implementation(s).

A KME shall not retain an association for any key in a request that arose from a call it received to "`ext_keys`" and for which it returned either a "`400`" ('Bad request format') or a "`401`" ('Unauthorised') response.

KMEs shall assume keys could have been associated with the "`initiator_sae_id`" and "`target_sae_id`" as a result of an "`ext_keys`" request unless either a "`400`" ('Bad request format') or "`401`" ('Unauthorised') response is returned.

Unless one of these two responses is successfully received, a KME shall assume that assignment(s) of the key(s) in the request could potentially have been created.

NOTE 1: Potential associations could reside within the KME or have been created by other entities it interacted with. Potential associations could have been created where a timeout occurs.

KMEs should not reassign any key that was in a "`ext_keys`" call without considering the potential assignments that could have been created.

NOTE 2: KMEs will typically not release a key to SAEs until they know that the key was successfully relayed to all intended "target_sae_id".

Where errors or timeouts other than a "400" ('Bad request format') or "401" ('Unauthorised') occur in an "ext_keys" request, KMEs shall either retry the failed requests for all impacted keys in subsequent "ext_keys" requests or void the impacted keys within their network(s) and by issuing "ext_keys/void" requests.

For "400" ('Bad request format') or a "401" ('Unauthorised') response KMEs may optionally retry the failed requests.

Where errors or timeouts occur in an "ext_keys/ack" request, KMEs shall either retry the failed requests for all impacted keys in subsequent "ext_keys/ack" requests or void the impacted keys within their network(s) and issue "ext_keys/void" requests.

NOTE 3: For "ext_keys/ack" the call to "ext_keys/void" is issued in the opposite direction to the KME that issued the related "ext_keys" request.

More comprehensive sequence diagrams are provided for the required asynchronous mode in Figure 4.2-3 and for the optional synchronous mode in Figure 4.2-4.

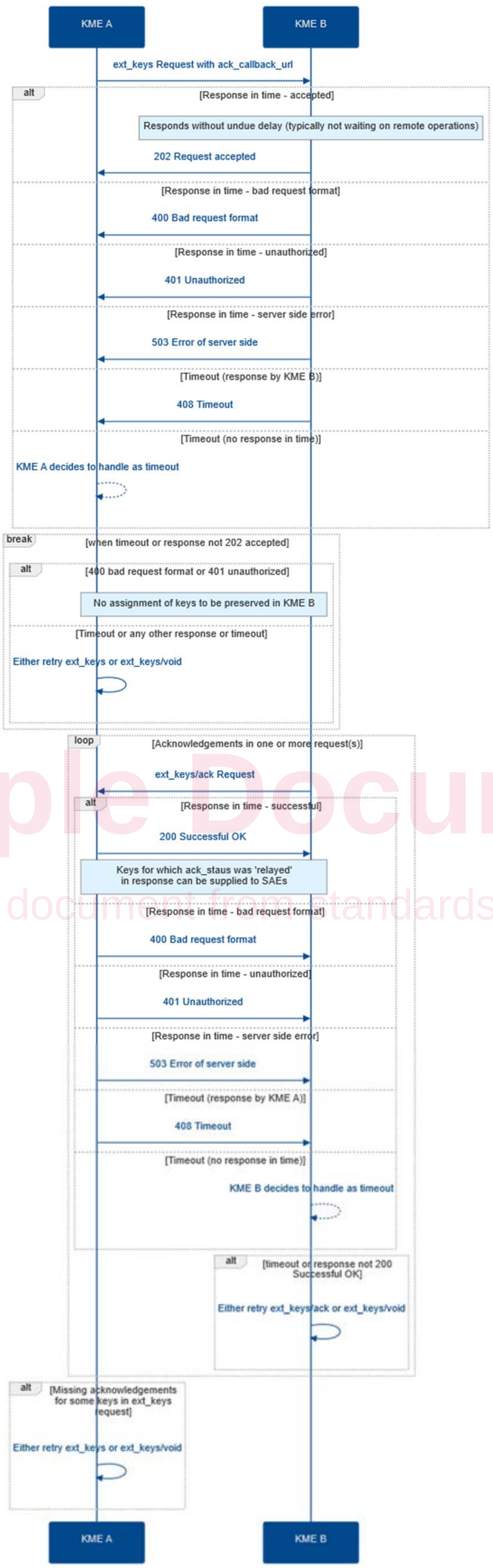

Figure 4.2-3: Message sequence diagram for required asynchronous mode

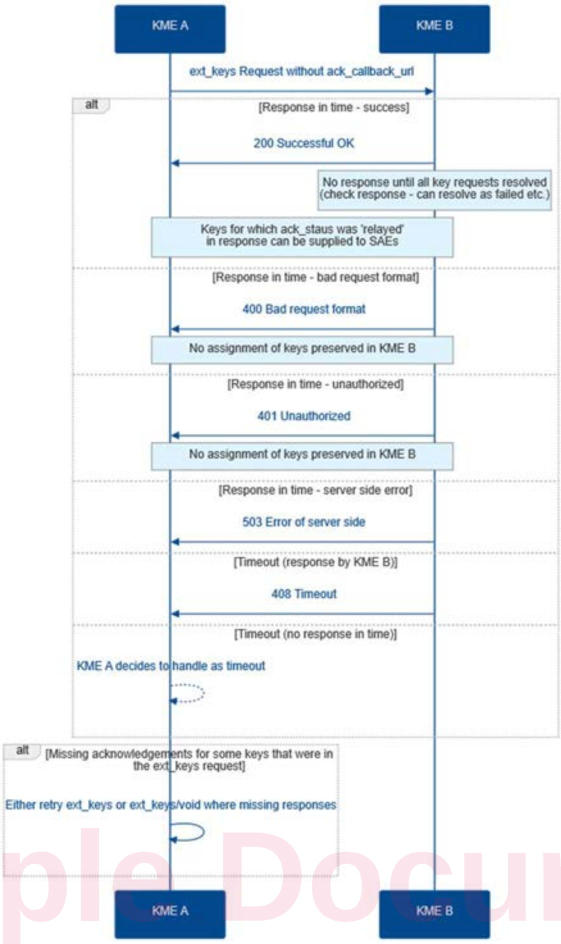

Figure 4.2-4: Message sequence diagram for optional synchronous mode

4.3 Connection specification and security scheme

The common connection specifications for the API are shown in Table 4.3-1.

Table 4.3-1: Common connection specifications

Name	Description
Communication Protocol	"HTTPS": HTTP/1.1 and optionally higher versions in addition; TLS v1.3 or higher versions
Content-Type	application/json; charset=UTF-8

All communication functions specified in this API shall be implemented using the HTTP/1.1 protocol as specified in IETF RFC 9110 [1] and IETF RFC 9112 [3] with Transport Layer Security (TLS).

NOTE 1: This combination is often referred to as an HTTPS protocol.

Higher versions of the HTTP protocol may be implemented in addition to HTTP/1.1.

TLS v1.3 as specified in IETF RFC 8446 [4] or higher versions shall be used.

At the connection establishment, mutual authentication between the KMEs shall be performed using certificates.

Each of the two KMEs shall verify the validity of the certificate of the other KME and shall confirm its identity from the certificate.

Where requests and responses include body content these shall be encoded using the "`application/json`".

NOTE 2: OpenAPI introduced support for specifying mutual TLS security in version 3.1. While other parts of this OpenAPI description can be easily converted to OpenAPI version 3.0 for use with older tools, it would be necessary to ensure that whatever security is specified in a Security Scheme Object is manually overridden according to this clause 4.3 in any implementation.

4.4 Security and operational environment

This API is for use between KMEs that are both located within the same site (an interworking node when used between two QKD networks). The KMEs and the communications the present document specifies between them should remain within a secure operational environment. Appropriate security measures should be considered for the operational environment.

General security requirements for key processing and management are outside the scope of the present document.

4.5 Extensions

Some methods allow for extensions to be provided. Extensions can be specified by vendors and KMEs shall follow the specification of any extension it claims to have implemented. Where relevant, this includes passing on without change any information indicated to be immutable in the specification of the extension.

Properties may be of any type, but their names shall be unique and start with a vendor prefix that shall be the capital letter "E" followed by an IANA Private Enterprise Number (PEN) followed by an underscore. The organization to which the PEN is assigned can use the extension for their own purposes, or can provide a specification for the extension to other organizations so they can also implement the extension. No organization shall use an extension starting with a vendor prefix with the PEN of any other organization without following a specification from the organization associated with the PEN.

ETSI might specify extensions in future but unless it does ETSI's IANA Private Enterprise Number "13019" is reserved and shall not be used.

NOTE: Extensions can originate from SAE requests that triggered key relaying, or can be added by a KME, e.g. to add metadata etc. Requirements in the specification of an extension can prohibit addition of an extension by a KME in subsequent calls. Extensions could be deserialized to a dictionary within a KME. Both property names and their values can be significant.

Relevant schema include "extension", "extension_mandatory", and "extension_optional".

4.6 SAE IDs

SAE IDs shall be unique across any set of connected QKD networks. In some circumstances, duplicate SAE IDs could result in keys being made available to an SAE other than the intended SAE(s). To obtain a key intended for another SAE in the presence of duplicate SAE IDs using the API in ETSI GS QKD 014 [i.2] from an otherwise securely-implemented KME, a malicious target SAE would need to specify a valid "key_id" associated with the other target SAE in an appropriately-timed request. However, even where potential mitigations apply in some circumstances (such as maintaining the "key_id" as confidential) reliable delivery can be compromised by duplicate SAE IDs.

How connected QKD networks ensure the uniqueness of SAE ID is beyond the scope of the present document.

SAE IDs ("initiator_sae_id" and "target_sae_id") shall not exceed 64 characters in length. Where SAEs are likely to want to communicate SAE IDs in restricted-length-parameters of other protocols connected QKD networks can consider further limiting SAE ID lengths.

SAE IDs shall not include characters that would not be allowed within a URI as specified in IETF RFC 3986 [5] with reference to IETF RFC 5234 [6]. SAE IDs are not required to be URIs.

NOTE: IETF RFC 3986 [5] allows unreserved characters "unreserved = (ALPHA / DIGIT / "-" / "." / "_" / "~")", reserved general delimiter characters "gen-delims = (":" / "/" / "?" / "#" / "[" / "]" / "@")", and sub-delimiter characters "sub-delims = ("!" / "$" / "&" / "'" / "(" / ")" / "*" / "+" / "," / ";" / "=")" as well as percent encoded characters "pct-encoded = "%" HEXDIG HEXDIG". See IETF RFC 5234 [6] for definitions of "ALPHA = (%x41-5A / %x61-7A; A-Z / a-z)", "DIGIT = %x30-39; 0-9", and "HEXDIG = (DIGIT / "A" / "B" / "C" / "D" / "E" / "F")".

4.7 Example use case for the API

Figure 4.7-1 shows a use-case in which the "ext_keys" operation is used to pass keys between KMEs in an interworking node of a three-node network. It is comprised of two networks X and Y that both have a presence in an interworking node. Key delivery to applications using ETSI GS QKD 014 [i.2] is considered in this use case.

The asynchronous key delivery between two QKD networks is uncoupled from any SAE protocol or workflow. This use case considers a reactive pattern in which keys are requested by an initiator SAE. Use cases following a proactive pattern are also possible in which a KME node acts as an initiator SAE to request keys to fill one or more shared buffers of keys that such KMEs can then use to delivery keys to client SAEs.

1) SAE A makes a local request within Site A to KME X1 of QKD network X for key(s) shared with SAE C, which is in a separate Site C served by another QKD network Y.

2) QKD network X performs operations using QKD keys to establish shared key(s) between KME X1 in SAE A's site and KME X2, which is in an interworking node at Site B. Communications can be proprietary to systems within QKD network X and details are not indicated in Figure 4.7-1.

3) Within Site B KME X2 issues a request to "ext_keys" on KME Y2 using the API specified in the present document.

4) KME Y2 responds immediately with HTTP status code "202" ('Request accepted').

5) QKD network Y performs operations using QKD keys to transfer the key(s) to KME Y1 in Site C. Communications can be proprietary to systems within QKD network Y, hence details are not indicated in Figure 4.7-1.

6) Once QKD network Y has ensured the key(s) are available in KME Y1, KME Y2 issues a request to "ext_keys/ack" on KME X2 to acknowledge that the request was successful.

7) KME X2 responds immediately with HTTP status code "200" ('Successful OK') to confirm that KME X2 has received the acknowledgement.

8) QKD network X communicates the successful result back to KME X1. Communications can be proprietary to systems within QKD network X and details are not indicated in Figure 4.7-1.

9) KME X1 returns the key(s) to SAE A by responding to the original request from SAE A.

10) SAE A communicated the key ID(s) to SAE C. Communications can be proprietary to systems of the application, hence details are not indicated in Figure 4.7-1.

11) SAE C uses the key ID(s) to request the key(s) from KME Y1 locally within Site C.

12) KME Y1 responds to SAE C with the key(s). Both SAE A and SAE C now share identical key(s) and can proceed to consume them for cryptographic purposes.

In this manner keys are shared between SAEs A and C via the use of "ext_keys" within the interworking node.

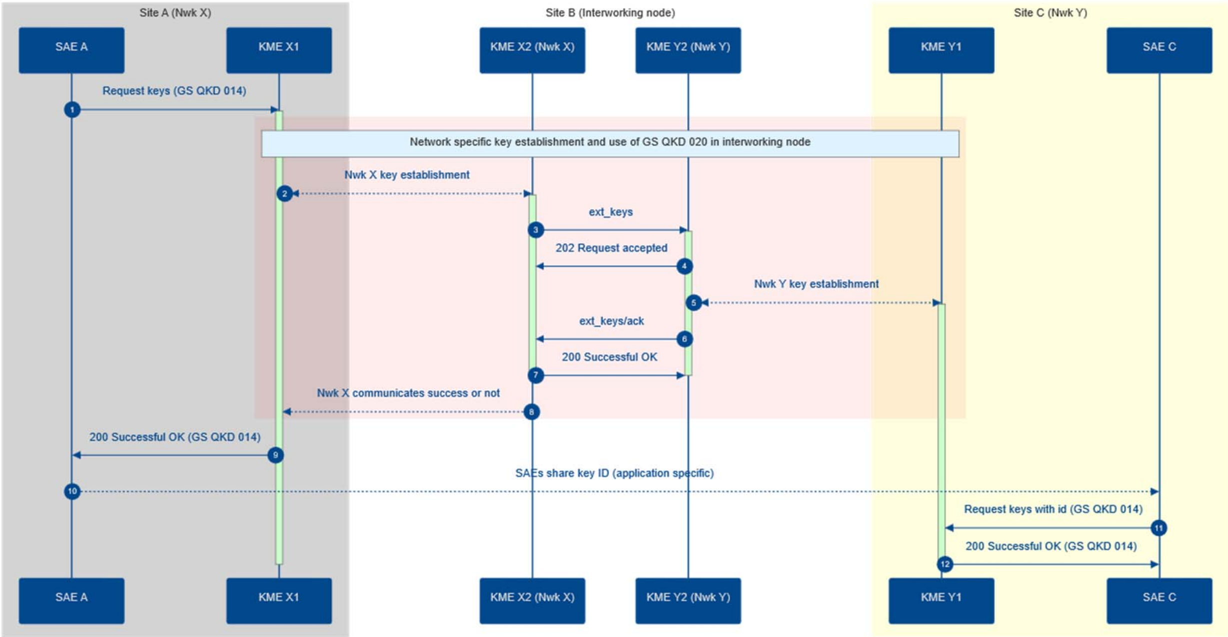

Figure 4.7-1: Sequence diagram for use case of real-time ETSI GS QKD 014 [i.2] key request established via interwork node

5 versions: API versions available

5.1 Tag description

Versionless path that can be used to query the available versions of this API on a KME.

5.2 GET /kmapi/versions: Get supported API versions

5.2.1 Description

Return list of versions of the API that are supported by the KME.

When an KME makes a request to another KME, it should use the most recent version of the API that the KMEs both support.

The SAE can use this end-point to determine which API versions the KME supports.

5.2.2 Responses

5.2.2.1 200 Successful OK response application/json

The response body shall match the "`version_container`" schema as shown in Table 5.2.2.1-1 encoded as media type "`application/json`" in JSON format as specified in IETF RFC 8259 [2].