ETSI GS QKD 004 V2.1.1 (2020-08)

Quantum Key Distribution (QKD); Application Interface

Disclaimer

The present document has been produced and approved by the Quantum Key Distribution (QKD) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
RGS/QKD-004ed2_ApplIntf

Keywords
API, quantum cryptography, quantum key distribution, security, use case

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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, 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.

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 is intended to specify an Application Programming Interface (API) between a QKD key manager and applications. The function of a QKD key manager is to manage the secure keys produced by an implementation of a QKD protocol and to deliver the identical set of keys, via this API, to the associated applications at the communication end points.

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 at https://docbox.etsi.org/Reference/.

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] IANA Media Type registry list of Directories of Content Types and Subtypes.

NOTE: Available at http://www.iana.org/assignments/media-types/.

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 are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] OASIS Standard (22 November 2017): "Key Management Interoperability Protocol Specification Version 1.4". Editor by Tony Cox.

NOTE: Available at http://docs.oasis-open.org/kmip/spec/v1.4/os/kmip-spec-v1.4-os.html.

[i.2] ETSI GS QKD 014 (V1.1.1): "Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

Application Programming Interface (API): interface implemented by a software program to be able to interact with other software programs

Key Management Interoperability Protocol (KMIP): protocol for the communication between enterprise key management systems and encryption system

NOTE: The KMIP is directed by the OASIS initiative.

key management layer: abstraction in a layered model including physically distributed key managers in two or more network nodes

NOTE: The key management layer sits between QKD modules and various applications. It manages the synchronization and deletion of keys, etc. as well as key delivery to applications.

link encryptor: device performing link encryption, i.e. the communication security process of encrypting / decrypting information between two peers on the data link level

Organization for the Advancement of Structured Information Standards (OASIS): global consortium that drives the development, convergence and adoption of e-business and web service standards, including KMIP

QKD application interface: interface between a QKD key manager and one or more application

QKD link: set of active and/or passive components that connect a pair of QKD modules to enable them to perform QKD and where the security of symmetric keys established does not depend on the link components under any of the one or more QKD protocols executed

QKD module: set of hardware and software and/or firmware components contained within a defined cryptographic boundary that implements part of one or more QKD protocol(s) to be capable of securely establishing symmetric keys with at least one other QKD module

QKD protocol: list of steps including the transport of quantum states that have to be performed by QKD modules to establish symmetric keys between remote parties with security based on quantum entanglement or the impossibility of perfectly cloning the transported quantum states

Quality of Service (QoS): description or measure of the overall performance of a service provided to an application after any management policies for prioritizing different applications or users have been applied

synchronization: function to ensure that symmetric keys in two key managers are identical

Transport Layer Security (TLS): cryptographic protocols used to encrypt the segments of network connections above the Transport Layer, using symmetric cryptography for privacy and a keyed message authentication code for message reliability

3.2 Symbols

Void.

3.3 Abbreviations

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

AAA	Authentication, Authorization and Accounting
API	Application Programming Interface
APPA	Application at host A
APPB	Application at host B
APPZ	Application at host Z
CORBA	Common Object Request Broker Architecture
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
ITS	Information Theoretical Secure
JSON	JavaScript Object Notation
KIMP	Key Management Interoperability Protocol
KM	Key Manager
KMIP	Key Management Interoperability Protocol
KSID	Key stream ID
OASIS	Organization for the Advancement of Structured Information Standards

PAP	Password Authentication Protocol
QKD	Quantum Key Distribution
QKDA	QKD key management layer peer at host A
QKDB	QKD key management layer peer at host B
QoS	Quality of Service
REST	Representational State Transfer
SDN	Software Defined Networking
SSL	Secure Socket Layer
TLS	Transport Layer Security
TTL	Time To Live
URI	Uniform Resource Identifier

4 Introduction

The present document is intended to specify an Application Programming Interface (API) between QKD Key Managers (KMs) and applications. The function of a QKD KM is to manage secure keys produced by an implementation of a QKD protocol and to deliver, on demand, identical sets of keys, via this API, to peer applications at the communication end points. It should be noted that there may be multiple levels of key managers (e.g. one at the QKD link level and another one at the QKD network level) and the API specified in the following clauses may be implemented in every QKD key manager level and that such key interchange is accomplished within the user's local security perimeter.

Manufacturers may implement this API wherever a QKD key manager is provided. Manufacturers may provide additional APIs and expanded functionality as they deem fit, realizing that these additions may be incompatible with versions provided by other vendors. Also, ther high level APIs can be built on top of this primary API, such as the REST API specified in ETSI GS QKD 014 [i.2].

5 QKD Application Interface Specification Description

The present document encompasses the ability to have multiple levels of key managers (e.g. one at the QKD link level and another one at the QKD network level), see figure 1 and figure 2, and the API specified in the following clauses may apply to every QKD key manager level and that this interchange is accomplished within the user's local security perimeter, referred to as sites. The methods by which a key manager accomplishes these functions is beyond the scope of the present document, but it is expected that external communication between key managers at different sites will be necessary. In addition, some communication between the peer applications may also be necessary to communicate a common key association.

The simplest example, figure 1, shows a single QKD link, with end points at Site A and Site B. Each site has a single application (shaded in yellow) and a single QKD module, enclosed by a blue box, which implements its part in a QKD protocol (shaded in red) to produce QKD keys that are managed by a QKD key manager peer (shaded in green). The single peer application in this case, uses the QKD application interface to acquire identical sets of secure keys in the two sites on demand.

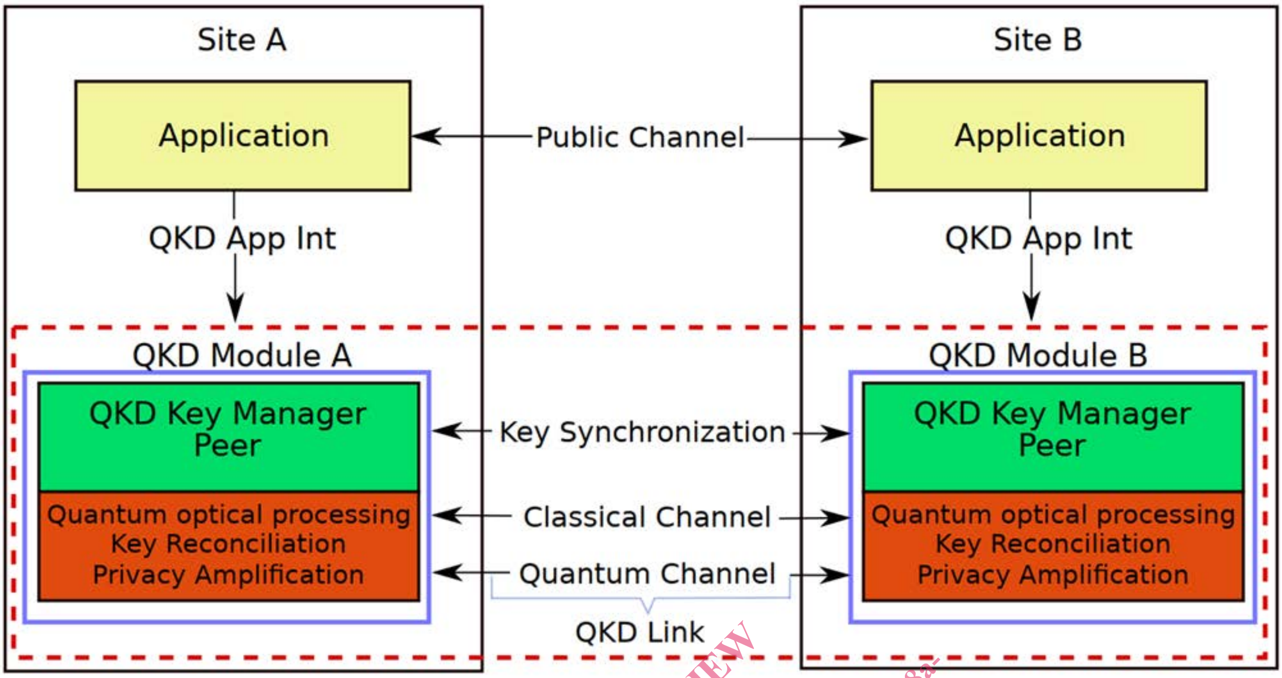

Figure 1: QKD Application Interface and peer relationships
Sites A and B represent security perimeters at each site

A more general example of two sites in a QKD network is shown in figure 2. Here two Sites of many in the network that contain a number of applications are shown. Site A and Site B are connected via a single QKD link between QKD Module A3 and QKD Module B1. The end points of the other QKD modules are not shown. In addition, network layer key managers, referred to as a QKD key servers are also depicted. The function of the QKD key servers is to manage keys between end points and deliver identical sets of keys to the applications at these end points. Note that the QKD application interface specified in the present document may be used to deliver keys from the QKD key management peers to the QKD key servers as well as to deliver keys from the QKD key servers to applications.

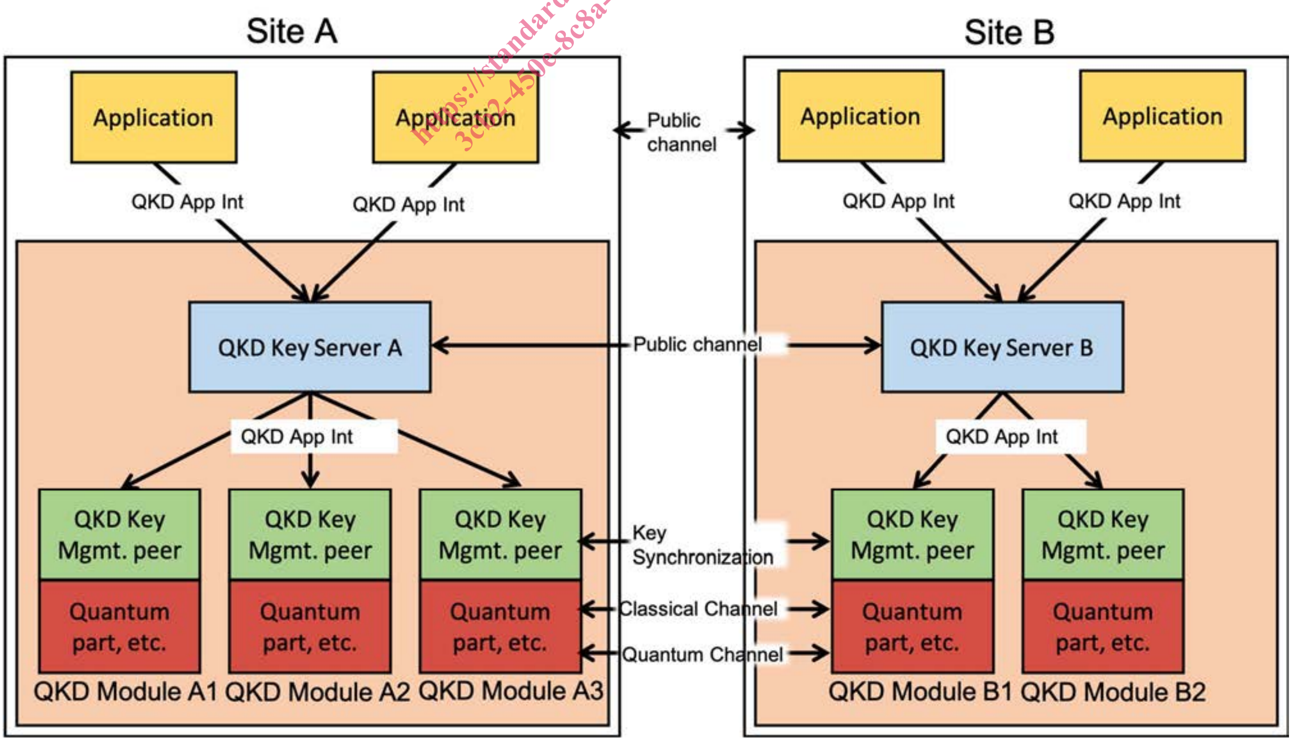

Figure 2: QKD Application Interface at two levels and peer relationships for
Sites within a QKD Network. Sites A and B represent security perimeters at each location

6 QKD Application Interface API Specification

6.1 General

A QKD key manager implementing this application interface shall provide the following API functions.

Table 1

Name	Description
OPEN_CONNECT	Reserve an association (Key_stream_ID) for a set of future keys at both ends of the QKD link through this distributed QKD key management layer and establish a set of parameters that define the expected levels of key service. This function shall block until the peers are connected or until the Timeout value in the QoS parameter is exceeded.
CLOSE	Terminate the association established for this Key_stream_ID. No further keys shall be allocated for this Key_stream_ID after the association has been closed. Due to timing differences at the other end of the link this peer operation will happen at some other time and any unused keys shall be held until that occurs and then discarded or the TTL (Time To Live QoS parameter) has expired.
GET_KEY	Obtain the required amount of key material requested for this Key_stream_ID. Each call shall return the fixed amount of requested key, in the key_buffer parameter, or an error message indicating why it failed. Together with the requested key, the QKD key manager peer shall return an index value, or allow the application to specify the position of the key to be retrieved for synchronization purposes via the index parameter. The key position will be the result of multiplying the index value by the Key_chunk_size value within the QoS parameters. The call will return information about the chunk of key in the Metadata_buffer parameter, if the Metadata_size parameter is not zero. If it is zero, no Metadata will be returned. If the Metadata_size is not zero and this information cannot fit into the buffer provided, the call will fail without providing any key, but will return the needed buffer size in the Metadata_size parameter and will set a flag in the status parameter. Thus, the application can reissue the call with the buffer size set correctly or to zero, without loss of key. This function may be called as often as desired, but the QKD key manager only needs to respond at the bit rate requested through the QoS parameters, or at the best rate the system can manage. The QKD key manager is responsible for reserving and synchronizing the keys at the two ends of the QKD link through communication with its peers. This function always returns with the status parameter indicating success or failure, depending on the request made via the OPEN_CONNECT function. The Timeout value for this function is specified in the OPEN_CONNECT() function.

The syntax of these functions are as follows:

```
Interface QKD{
    OPEN_CONNECT (in source, in destination, inout QOS, inout Key_stream_ID, out status);
    GET_KEY (in Key_stream_ID, inout index, out Key_buffer, inout Metadata, out status);
    CLOSE (in Key_stream_ID, out status);
}
```

NOTE: The parameter "Key_stream_ID" is an output parameter when the caller is initially opening the connection and the value passed in is "NULL". If the value is not "NULL" the value is assumed to be either a "Key_stream_ID" recently established by the peer end or a specifically desired/predefined "Key_stream_ID". For use in all other functions "Key_stream_ID" is an input. If the "Key_stream_ID" is already in use by some other application, the OPEN_CONNECT function will return with an error. On return, the structure QoS is the closest QoS that the system can provide (best effort). If on input QoS is "NULL" on a certain field of the structure, the system is free to provide the one that it considers most appropriate. In particular, if a KSID has been already registered, the QoS structure will be filled with the QoS values previously assigned to that KSID, independently of any value of the QoS in the OPEN_CONNECT call.

The function parameters are described in table 2.

Table 2: Description of API function parameters

Name		Description	Type	Comments
Key_stream_ID (KSID)		A unique identifier for the group of synchronized bits provided by the QKD Key Manager to the application	UUID_v4 16 bytes (128 bits)	It contains a reference to the necessary information to locate a key, but does not contain any key material. Key material cannot be derived from the Key_stream_ID. It shall be represented as a character array (char *), with the high order octet being the lowest significant octet. It may be passed to and stored in other applications at other sites. It shall be unique and both peers will use that same value to reference their application key stream. It can be previously agreed upon between the peer applications or sent between them by a public channel.
Key_buffer		Buffer containing the current stream of keys.	Array of bytes (octets)	Key buffer is an array of bits packed into octets (char*) ordered such that bit[0] of octet[0] is the 1^{st} bit and bit[7] of octet[n] is the $8*n+8^{th}$ bit.
Index		Position of the key to be accessed within the reserved key store for the application	32 bit unsigned integer	For client/server synchronization purposes, the index value allows one to specify which position within the reserved key stored is to be accessed. The actual position will be calculated as the multiplication of the index value by Key_chunk_size from the QoS parameters.
Source		Source identifier defined as a uniform resource identifier	URI	Identifier of the source application connecting to the QKD key management layer. The identifier is structured as a URI. If a client/server scheme is defined, the source will integrate the URI of the server.
Destination		Destination identifier defined as a uniform resource identifier	URI	Identifier of the destination application connecting to the QKD key management layer. The identifier is structured as a URI. If a client/server scheme is defined, the source will integrate the URI of the client.
QoS	Structure describing the characteristics of the requested key source			
	Key_chunk_size	Length of the key buffer, in Bytes, requested by the application	32 bit unsigned integer	If the requested key amount cannot be provided, an error shall be returned in the status parameter.
	Max_bps	Maximum key rate, in bps, requested by the application	32 bit unsigned integer	This is intended to provide guidance to the key management layer on expected maximum demand. If the peer cannot meet the requested rate, it may choose to do its best, which may be lower than the requested rate, or reject the request.
	Min_bps	Minimum key rate, in bps, required by the application	32 bit unsigned integer	This is intended to provide an estimation of the minimum key rate needed by the application to the QKD key management layer, so that an application could maintain its security channels. If the peer cannot meet the requested rate, it may reject the request.
	Jitter	Maximum expected deviation, in bps, for key delivery	32 bit unsigned integer	This value allows applications to specify flow variation of key bits with a minimum deviation for the delivery rate.
	Priority	Priority of the request	32 bit unsigned integer	This is intended to provide guidance to the QKD Key Management layer about the priority level of the request. The handling of this information is left to the specific implementation.
	Timeout	Time, in msec, after which the call will be aborted, returning an error.	32 bit unsigned integer	If this much time has passed and the one of the API functions has not completed, the function will return a TIMEOUT_ERROR in the status parameter This value shall be expressed in milliseconds.