ETSI GS QKD 018 V1.1.1 (2022-04)

Quantum Key Distribution (QKD); Orchestration Interface for Software Defined Networks

Reference

DGS/QKD-018OrchintSDN

Keywords

interface, management, orchestration, quantum cryptography, quantum key distribution, SDN, security

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 4
Foreword 4
Modal verbs terminology 4
Executive summary 4
Introduction 5
1 Scope 6
2 References 6
2.1 Normative references 6
2.2 Informative references 6
3 Definition of terms, symbols and abbreviations 7
3.1 Terms 7
3.2 Symbols 7
3.3 Abbreviations 8
4 SDN orchestration of QKD Overview 8
4.1 Use case of SDN orchestration 8
4.2 Functions of SDN orchestrator 9
5 SDN orchestration interface of QKD network 10
5.1 Discovery of QKD network topology 10
5.2 Monitoring of QKD network status and resource inventory 11
5.3 Monitoring of end-to-end QKD service status 13
5.4 End-to-end QKD service provisioning with path calculation 15
5.5 Notifications 17
6 Sequence diagrams and workflows 19
6.1 Introduction 19
6.2 QKD service request and end-to-end service provisioning across multi-domain networks 20
7 Security consideration 22
8 Protocol considerations 22
Annex A (normative): SDN orchestration interface YANG data models 23
A.1 General 23
A.2 YANG modules 23
Annex B (informative): Bibliography 24
Annex C (informative): Change History 25
History 26

Intellectual Property Rights

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.

Executive summary

The present document deals with the interface between an SDN orchestrator and an SDN controller of a QKD network. It describes the flow of information between both entities, the SDN controller performing as a server and the SDN orchestrator operating as a client. The information model is given in YANG and it is agnostic from the implementation by any vendors. This information model enables the SDN orchestrator to manage and configure the SDN controller of a QKD network, permitting it to orchestrate the QKD network and a classical optical transport network.

Introduction

Deploying an optical network where quantum channels and classical data channels can coexist is critical for adopting QKD networks in network 'operators' infrastructures. When network operators consider introducing QKD networks into their existing networks, there are a few implementation options to choose from, for example, the integration of quantum channels and classical data channels in a single optical fibre or separation of them in different optical fibres. There are two major technical issues with integrating quantum channels and classical data channels in a single optical fibre: differences between the transmitted optical powers and the achievable link distances of the channels. Network operators can optionally choose to separate such channels placing quantum channels in a QKD network and classical data channels in an Optical Transport Network (OTN). This can optimize performance of QKD networks in a fibre-rich environment. In this separated case, under the design principle of Software-Defined Network (SDN) architectures, the QKD network and OTN can be controlled by different SDN controllers.

However, if QKD-derived keys are to be supplied to secure application entities in an OTN for cryptographic use, network operators need to know which QKD nodes in the QKD network can supply QKD-derived keys to which secure application entities in an OTN. Each SDN controller has only resource information about the network it controls. Therefore, adopting an SDN orchestrator capable of controlling both the QKD and OTN networks is one option for achieving the end-to-end service provisioning of QKD-derived key generation in a QKD network and its use in secure application entities in an OTN. In this case, an SDN orchestrator plays the role of matching the addresses of QKD nodes in QKD network and ones of secure application entities in an OTN in order to supply secure application entities in OTN with QKD-derived keys generated in a QKD network.

In addition, with the introduction of an SDN orchestrator interface to the SDN controller of the QKD network, a network operator can operate and maintain a QKD network in terms of network topology, configuration, management policy and performance management.

So, adopting an SDN orchestrator for a QKD network can mitigate the burden of integrating the QKD network into network 'operators' communication networks where the network operators already have an SDN orchestrator for SDN controllers of their OTNs.

The information model for the interface between an SDN orchestrator and an SDN controller of QKD network is presented to simplify the management and configuration of QKD networks through the North Bound Interface (NBI) of the SDN controllers of the QKD networks.

1 Scope

The present document provides a definition of an orchestration interface between an SDN orchestrator and an SDN controller of a QKD network. This orchestration interface defines the abstract information models and workflows for QKD network resource management, configuration management, performance management, service provisioning, notifications and management of multi-domain QKD networks. Interfaces between an SDN orchestrator and SDN controllers of classical optical transport networks are out of scope.

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] IETF RFC 6020 (October 2010): "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)".

[2] IETF RFC 7950 (August 2016): "The YANG 1.1 Data Modeling Language".

[3] IETF RFC 6241 (June 2011): "Network Configuration Protocol (NETCONF)".

[4] IETF RFC 8040 (January 2017): "RESTCONF Protocol".

[5] ETSI GS QKD 004 (V2.1.1): "Quantum Key Distribution (QKD); Application Interface".

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

[7] ETSI GS QKD 015 (V1.2.1): "Quantum Key Distribution (QKD); Control Interface for Software Defined Networks".

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] ETSI GR QKD 007: "Quantum Key Distribution (QKD); Vocabulary Revision".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

NOTE: Where possible, the definitions from ETSI GR QKD 007 [i.1] are used.

entity: set of hardware, software or firmware components providing specific functionalities

Key Management Entity (KME): entity that manages keys in a network in cooperation with one or more other Key Management Entities

QKD application: entity consuming QKD-derived keys from the key management system

NOTE: They can be either external applications (similar to SAE, see below) or internal applications running in the QKD system.

QKD-derived key: secret key derived from QKD system(s) operating QKD protocol(s) over a QKD link

QKD interface: interface that is a high-level abstraction of a QKD system

NOTE: A QKD interface defines only attributes that are relevant from the point of view of the network. These attributes are revealed to a SDN controller to establish and manage QKD.

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, software or firmware components that implements part of one or more QKD protocol(s) to be capable of securely agreeing symmetric keys with at least one other QKD module

QKD network: network comprised of two or more QKD nodes

QKD node: set of QKD modules installed in the same location within the same security perimeter

QKD protocol: defined set of procedures performed by QKD modules (and optionally link modules) to agree shared secret bit strings by QKD

QKD system: pair of QKD modules connected by a QKD link designed to provide Quantum Key Distribution functionality using QKD protocols

quantum channel: communication channel for transmitting quantum signals

Quantum Key Distribution (QKD): procedure involving the transport of quantum states to agree shared secret bit strings between remote parties using a protocol with security based on quantum entanglement or the impossibility of perfectly cloning or measuring the unknown transported quantum states

SD-QKD node: logical and abstracted representation of the QKD resources under the responsibility of a single SDN agent

Secure Application Entity (SAE): entity that requests one or more keys from a Key Management System for one or more applications running in cooperation with one or more other Secure Application Entities

service link: logical key association link between two QKD nodes connected by a set of one or more physical QKD links (single hop or multi-hop)

3.2 Symbols

Void.

3.3 Abbreviations

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

FCAPS	Fault-management, Configuration, Accounting, Performance, and Security
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
JSON	JavaScript Object Notation
KME	Key Management Entity
NBI	North Bound Interface
OTN	Optical Transport Network
QKD	Quantum Key Distribution
QoS	Quality of Service
RFC	Request For Comments
SAE	Secure Application Entity
SDN	Software-Defined Network
SDNO	Software-Defined Network Orchestrator
SD-QKD	Software-Defined Quantum Key Distribution
SDQNC	Software-Defined Quantum Network Controller
URI	Uniform Resource Identifier
XML	Extensible Markup Language
YANG	Yet Another Next Generation

4 SDN orchestration of QKD Overview

4.1 Use case of SDN orchestration

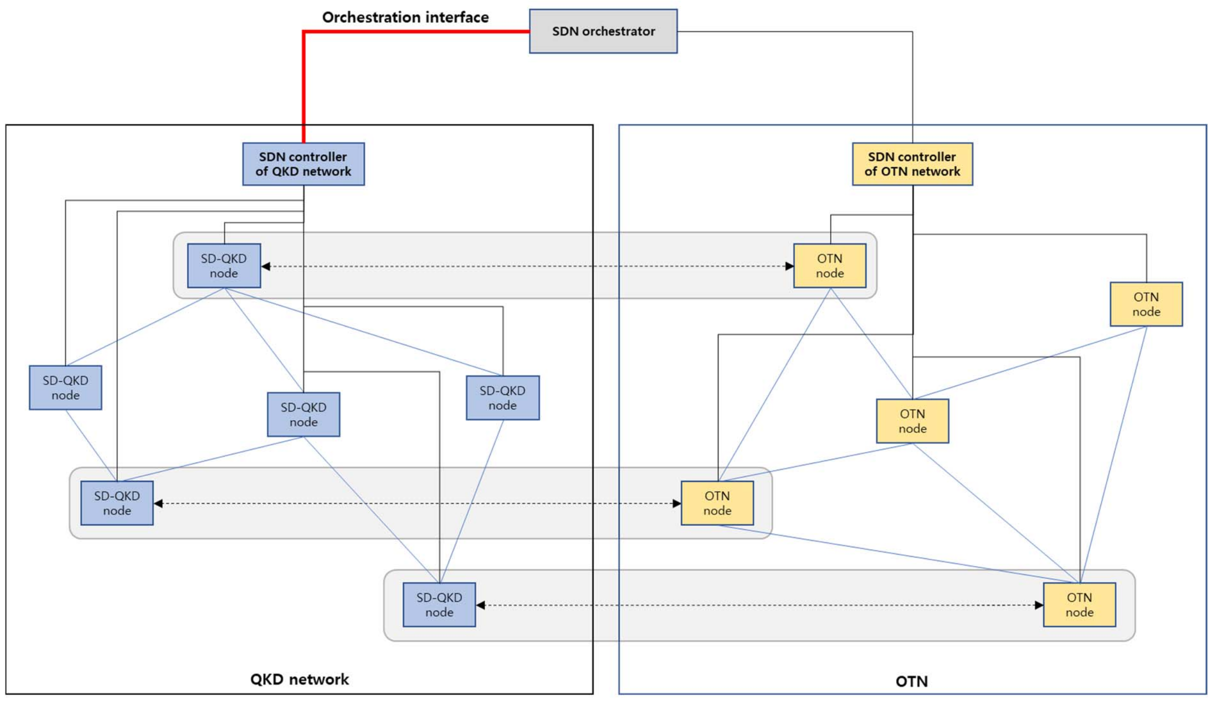
When network operators deploy a QKD network to secure data in a communication network, they need to consider how the QKD network will be incorporated into their classical communication network for QKD-derived 'keys' delivery to secure application entities in a classical communication network. Secure application entities can reside in various network domains within a classical communication network. While a QKD network domain and network domains containing secure application entities can be managed and configured independently via domain-specific SDN controllers, a network operator can introduce a multi-domain SDN orchestrator to orchestrate the whole network system.

For the use case of delivering QKD-derived keys to secure application entities in an Optical Transport Network (OTN), where the infrastructure is available to do so, network operators could choose to deploy a QKD network that is separate from a classical OTN and to operate and manage each network separately. Possible reasons for this include optimizing the performance of QKD links that depend upon fibre length and the presence of stray light, security isolation, or separation of responsibilities for management, etc. Both network domains need to be aware of nodes that belong to both network domains for QKD-derived keys to be delivered to secure application entities in the OTN. In particular, addresses for such nodes in the two network domains need to be matched. In this use case, the SDN orchestrator can perform this role with the information from the SDN controllers of the QKD network and the OTN.

In addition, a network operator can coordinate QKD networks and OTNs with an SDN orchestrator via each SDN controller to ensure end-to-end QKD service provisioning. The SDN orchestrator may be responsible for selecting network domains for a new service that is to be provisioned. Such domain selection is based on abstracted knowledge of intra- and inter-domain connectivity and topology. For the SDN orchestrator to coordinate between network domains, an interface between the SDN orchestrator and the QKD network SDN controller needs to be defined. This interface describes the flow of information between both entities, the SDN controller performing as a server and the SDN orchestrator operating as a client. The SDN orchestrator can orchestrate QKD networks through this interface in terms of network configuration and topology, management policy, performance management, and the address matching described above.

In this use case, a network operator starts to request end-to-end QKD service provisioning from the SDN orchestrator. Before the SDN orchestrator requests end-to-end QKD service provisioning from the SDN controller of the QKD network following a network operator's request, the SDN orchestrator should collect the topology and inventory information of the QKD network from the SDN controller of the QKD network as a previous step to check the status of QKD network. After collecting this information, the SDN orchestrator requests a service link and the candidate paths for the service link within a QKD network to transport QKD-derived keys for end-to-end QKD service provisioning from the SDN controller of the QKD network. After the SDN orchestrator receives the candidate paths from the SDN controller of the QKD network, the SDN orchestrator decides which candidate path in the QKD network it will use for end-to-end QKD service provisioning and requests to deploy a specific path from the SDN controller of the QKD network. In the YANG models included in the present document, the topology and inventory YANG models are separated from the connectivity YANG model to reflect this procedure.

With this configuration extended, an SDN orchestrator can orchestrate multi-QKD network domains from multi-vendors via each SDN controller of each QKD network as well as both QKD and classical network domains, as shown in Figure 1.



NOTE: The orchestration interface (solid red line) between the SDN orchestrator and SDN controller of the QKD network is shown. The key delivery API (dashed line) from ETSI GS QKD 014 [6] or remote function call from ETSI GS QKD 004 [5] can be used for Secure Application Entities (SAEs) in OTN nodes in the OTN network to retrieve keys from Key Management Entities (KMEs) in SD-QKD nodes in QKD network. SAEs in OTN nodes are located within the same security boundary as their connected KMEs in SD-QKD nodes.

Figure 1: Use case of SDN orchestrator for QKD network and OTN

4.2 Functions of SDN orchestrator

SDN orchestration can be defined as the continuing process of automatically coordinating the available resources according to optimization criteria to establish and release the end-to-end service provisioning through different network domains controlled by each SDN controller. SDN orchestration may be used to start the series of automated processes required to satisfy a customer service request generated via a customer website. An SDN orchestrator is a master entity that enables each SDN controller to establish and release multiple paths in its own network domain to meet end-to-end service provisioning requests from customers through different network domains.

To enable end-to-end service provisioning through different network domains, the SDN orchestrator has the following functions:

- Translation from the end-to-end QKD service provisioning requests from a customer to the configuration of the different network domains through each SDN controller and the allocation of secure application entities for this service provisioning.
- Establishment and release requests of the end-to-end QKD service provisioning with the inter-domain connections between secure application entities through orchestration interfaces.
- Identification of multi-domain path calculation across the different network domains, including inter-domain connections and endpoints for each request of the end-to-end service provisioning.
- Requests to change the established path calculation with constraints from a network operator.
- The QKD nodes and QKD links are discovered under each SDN controller in the QKD network and an abstracted view of the QKD network topology.
- Inventory monitoring of QKD-derived key resources available from the key management system in each QKD node and each QKD link.
- Monitoring of FCAPS management of the QKD network and notification of changes of FCAPS management in the QKD network.

The orchestration interface between the SDN orchestrator and the SDN controller of the QKD network needs to be defined to address the outlined functions of the SDN orchestrator. The SDN controller is a server through the orchestration interface, and the SDN orchestrator is a client in terms of communication between them.

5 SDN orchestration interface of QKD network

5.1 Discovery of QKD network topology

The information model includes the discovery of QKD nodes and each direct (physical) QKD link between QKD nodes under each SDN controller in the QKD network and an abstracted view of the QKD network topology.

An SDN orchestrator can compose the overall multi-domain network topology with the information from the underlying SDN controllers, one of which can expose its intra-domain QKD network topology. An SDN orchestrator does not need complete QKD physical network composition information but needs an abstracted view of the network domains and the inter-domain connectivity.

The discovery of QKD network topology can be done proactively or reactively. Proactively, the SDN orchestrator requests information about the QKD network topology from the SDN controller every time the SDN orchestrator needs to make a new path calculation request to the SDN controller. Reactively, the SDN orchestrator receives the information about the QKD network topology from the SDN controller whenever there is any change in the QKD network topology, for example, in case new QKD nodes are added, or the existing QKD nodes are removed from the QKD network.

In the discovery of the QKD network topology, the information about internal or external applications that consume QKD-derived keys for their own purpose is not incorporated. Therefore, QKD physical links that connect QKD nodes directly are displayed in the QKD network topology. After starting the operation of the QKD network and preparing end-to-end QKD service provisioning, the QKD network has set up QKD service links and information about the QKD service links is also given in terms of QKD service link connection.

The SDN controller of a QKD network shall provide the following parameters and values to the SDN orchestrator.

Table 1: SD-QKD node parameters for QKD network topology

Name	Type	Detail	Description
sdqkd_nodes	container	None	Container of SD-QKD nodes.
sdqkd_nodes/ qkdn	list	Key: "qkdn_id"	List of SD-QKD nodes.
qkdn/ qkdn_id	ietf-yang-types: uuid	None	Unique ID of the SD-QKD node.
qkdn/ qkd_interfaces	container	None	Container of the physical QKD modules of the SD-QKD node.
qkd_interfaces/ qkdi	list	Key: "qkdi_id"	List of the physical QKD modules of the SD-QKD node.
qkdi/ qkdi_id	uint32	None	Interface id. It is described as a locally unique number, which is globally unique when combined with the SD-QKD node ID.

Table 2: QKD link parameters for QKD network topology

Name	Type	Detail	Description
qkd_phys_links	container	None	Container of QKD physical links to directly connect SD-QKD nodes in the QKD network.
qkd_phys_links/ phys_link	list	Key: "phys_link_id"	List of QKD physical links to directly connect SD-QKD nodes in the network.
phys_link/ phys_link_id	ietf-yang-types: uuid	None	Universally Unique ID of the QKD physical link.
phys_link/ link_type	etsi-qkdn-types: QKD-LINK-TYPES	None	The QKD physical link type is included. The identity is PHYS according to ETSI GS QKD 015 [7].
phys_link/ local_qkdn_id	ietf-yang-types: uuid	None	Unique ID of the local SD-QKD node which is connected to the QKD physical link.
phys_link/ local_qkdi_id	uint32	None	Interface ID of the local SD-QKD node which is connected to the QKD physical link.
phys_link/ remote_qkdn_id	ietf-yang-types: uuid	None	Unique ID of the remote SD-QKD node which is connected to the QKD physical link.
phys_link/ remote_qkdi_id	uint32	None	Interface ID of the remote SD-QKD node which is connected to the QKD physical link.
qkd_svc_links	container	None	Container of QKD service links (end-to-end key association links) in the QKD network.
qkd_svc_links/ svc_link	list	Key: "svc_link_id"	List of QKD service links in the network.
svc_link/ svc_link_id	ietf-yang-types: uuid	None	Universally Unique ID of the QKD service link.
svc_link/ link_type	etsi-qkdn-types: QKD-LINK-TYPES	None	The QKD service link type is included. The identity is VIRT according to ETSI GS QKD 015 [7].
svc_link/ local_qkdn_id	ietf-yang-types: uuid	None	Unique ID of the local SD-QKD node which is connected to the QKD service link.
svc_link/ local_qkdi_id	uint32	None	Interface ID of the local SD-QKD node which is connected to the QKD service link.
svc_link/ remote_qkdn_id	ietf-yang-types: uuid	None	Unique ID of the remote SD-QKD node which is connected to the QKD service link.
svc_link/ remote_qkdi_id	uint32	None	Interface ID of the remote SD-QKD node which is connected to the QKD service link.

5.2 Monitoring of QKD network status and resource inventory

The information model includes monitoring QKD network status and QKD-derived key resources available from the key management system in each QKD node and each direct (physical) QKD link.

With the discovered QKD network topology based on QKD nodes and each direct (physical) QKD link between QKD nodes, the SDN orchestrator needs to monitor QKD network status and QKD-derived key resources for intra-domain QKD network operation and maintenance.

The information model of monitoring QKD network status includes QKD node status, QKD physical link status, and physical performance. The information model of monitoring resource inventory includes the QKD-derived key generation rate in each QKD physical link and the available QKD-derived key rate in each QKD physical link after internal consumption.

As this information is provided for the SDN orchestrator to operate and maintain the QKD network, it does not incorporate the internal or external application information at the initial operation of the QKD network. After starting the operation of the QKD network and preparing the end-to-end QKD service provisioning, the QKD network has set up QKD service links and the information of QKD service links is also given in terms of QKD service link status, available QKD-derived key resource, and maintenance policy when any failure happens in supporting QKD service link.

The SDN controller of a QKD network shall provide the following parameters and values to the SDN orchestrator:

Table 3: SD-QKD node parameters for QKD network status and inventory

Name	Type	Detail	Description
qkdn/ qkdn_version	string	None	Hardware or software version of the SD-QKD node.
qkdn/ qkdn_capabilities	container	None	Capabilities of the SD-QKD node.
qkdn_capabilities/ link_stats_support	boolean	Default: true	The SD-QKD node exposes link-related statistics (key generation rate, link consumption, status).
qkdn_capabilities/ application_stats_support	boolean	Default: true	The SD-QKD node exposes application-related statistics (application consumption, alerts).
qkdn_capabilities/ key_relay_mode_enable	boolean	Default: true	The SD-QKD node supports key relay (multi-hop) mode services.
qkdn/ location_id	string	None	Location of the secure area that contains the SD-QKD node.
qkdn/ KME_ip_port	container	None	The container for the IP address and port number of the key management entity in the SD-QKD node provides QKD keys to external applications.
KME_ip_port/ KME_ip	ietf-inet-types: ip-address	None	The IP address of the key management entity in the SD-QKD node, which provides QKD keys to external application.
KME_ip_port/ KME_port	ietf-inet-types: port-number	None	The port number of the key management entity in the SD-QKD node, which provides QKD keys to external application.
qkdn/ qkdn_status	etsi-qkdn-types: NODE-STATUS-TYPES	Config: false	Status of the SD-QKD node. (for monitoring QKD node operation and connection status to SDN controller).
qkdi/ role_support	etsi-qkdn-types: QKD-ROLE-TYPES	None	QKD system that can work as a transmitter or receiver.
qkdi/ model	string	None	Device model (vendor/device).
qkdi/ type	etsi-qkdn-types: QKD-TECHNOLOGY-TYPES	None	Interface type (QKD technology).
qkdi/ att_point	container	None	Interface attachment point to an optical switch.
att_point/ device	string	None	Unique ID of the optical switch (or passive component) to which the interface is connected.
att_point/ port	uint32	None	Port ID from the device to which the interface is connected.
qkdi/ qkdi_status	etsi-qkdn-types: IFACE-STATUS-TYPES	Config:false	Status of each QKD interface (module).

Table 4: QKD link parameters for QKD network status and inventory

Name	Type	Detail	Description
phys_link/ enable	boolean	Default: true	This value allows to enable or disable the key generation process for a given link.
phys_link/ wavelength	etsi-qkdn-types: wavelength	(if link_type=PHYS)	Wavelength (nm) to be used for the quantum channel.
phys_link/ performance	container	None	Performance of each QKD physical link.
performance/ expected_consumption	uint32	Config: false	Sum of all the applications' bandwidths (in bits per second) that are on this particular QKD physical link.
performance/ skr	uint32	Config: false	Secret key generation rate (in bits per second) of each QKD physical link.
performance/ eskr	uint32	Config: false	Effective secret key generation rate (in bits per second) of each QKD physical link available after all the applications' consumption.
performance/ phys_link_perf	list	(if link_type=PHYS) Config: false Key: "type"	List of physical performance parameters.
phys_link_perf/ type	etsi-qkdn-types: PHYS-PERF-TYPES	(if link_type=PHYS) Config: false	Type of the physical performance value to be exposed to the controller (QBER, SNR).
phys_link_perf/ value	decimal64 fraction-digits 3	(if link_type=PHYS) Config: false	Numerical value for the performance parameter type specified above.
phys_link/ phys_link_status	etsi-qkdn-types: LINK-STATUS-TYPES	Config: false	Status of each QKD physical link.
svc_link/ bandwidth	uint32	None	Required bandwidth (bps) for the key association link. Used to reserve bandwidth from the physical QKD links to support the service key association link as an internal application.
svc_link/ performance	container	None	Performance of QKD service link.
performance/ expected_consumption	uint32	Config: false	The bandwidth (in bits per second) of the external application that is consuming keys from QKD service link.
performance/ skr	uint32	Config: false	Secret key generation rate (in bits per second) of QKD service link.
performance/ eskr	uint32	Config: false	Effective secret key generation rate (in bits per second) of QKD service link after the external application's consumption.
svc_link/ path_restoration	container	None	Container for path restoration conditions when any failure happens in the end-to-end key relay path.
path_restoration/ type	PATH-RESTORATION-TYPE	None	Path restoration type of QKD service link. It may be AUTOMATIC or MANUAL.
path_restoration/ hold-off-time	uint32	None	Time to wait before attempting a restoration.
svc_link/ svc_link_status	etsi-qkdn-types: LINK-STATUS-TYPES	Config: false	Status of each QKD service link.

5.3 Monitoring of end-to-end QKD service status

The information model includes querying end-to-end QKD service status and path information used for QKD service links in the QKD network.

The end-to-end QKD service in an inter-domain network starts with external applications requesting QKD-derived keys from the SDN orchestrator. Thus, the SDN orchestrator needs information about external applications, local QKD node, remote QKD node, and QKD service links between local QKD node and remote QKD node for inter-domain end-to-end QKD service.

For the SDN orchestrator to operate and maintain ongoing end-to-end QKD service in inter-domain network, the information also incorporates QKD service link status and the available QKD-derived key rate in each QKD service link.

The SDN controller of the QKD network shall provide the following parameters and values to the SDN orchestrator.

Table 5: QKD application parameters for monitoring QKD service status

Name	Type	Detail	Description
qkd_service	container	None	Container for external applications that are currently registered in the SD-QKD node and request QKD-derived keys.
qkd_service/ qkd_app	list	Key: "app_id"	List of external applications that are currently registered in the SD-QKD node and request QKD-derived keys.
qkd_app/ app_id	ietf-yang-types: uuid	None	Unique ID that identifies a QKD application consisting of a set of entities that are allowed to receive keys shared with each other from the specific QKD service link.
qkd_app/ app_type	etsi-qkdn-types: QKD-APP-TYPES	None	Application's type. All use cases described in the present document use applications working as external and their identity is CLIENT.
qkd_app/ app_priority	uint32	None	Priority of the association/application. This might be defined by the user but it is usually handled by a network administrator.
qkd_app/ server_app_id	inet:URI	None	ID that identifies the entity that initiated the creation of the QKD application to receive keys shared with one or more specified target entity identified by client_app_id. It is a client in the interface to the SD-QKD node and the name server_app_id reflects that it requested the QKD application to be initiated.
qkd_app/ client_app_id	leaf-list: inet:URI	None	List of IDs that identifies the one or more entities that are allowed to receive keys from SD-QKD node(s) under the QKD application in addition to the initiating entity identified by server_app_id.
qkd_app/ app_status	etsi-qkdn-types: APP-STATUS-TYPES	Config:false	Status of the application.
qkd_app/ service_creation_time	ietf-yang-types: date-and-time	Config:false	Date and time of the service creation.
qkd_app/ service_expiration_time	ietf-yang-types: date-and-time	None	Date and time of the service expiration.
qkd_app/ app_statistics	container	None	Container for statistical information relating to a specific statistic period.
app_statistics/ end_time	ietf-yang-types: date-and-time	None	End time for the statistics collection period.
app_statistics/ start_time	ietf-yang-types: date-and-time	None	Start time for the statistics collection period.
app_statistics/ consumed_bits	uint32	None	Consumed secret key amount (bits) for a statistics collection period.
qkd_app/ app_mapped_service_link	container	None	Container for paths per QKD service link for app.
app_mapped_service_link/svc_link_and_qkdn_ids	container	None	Container for QKD service link and its local SD-QKD node and remote SD-QKD node.
svc_link_and_qkdn_ids/svc_link_id	ietf-yang-types: uuid	None	Universally Unique ID of the QKD service link which is providing QKD keys to the application.

Table 6: QKD service link parameters for monitoring QKD service status

Name	Type	Detail	Description
app_mapped_service_link/svc_link_and_qkdn_ids	container	None	Container for QKD service link and its local SD-QKD node and remote SD-QKD node.
svc_link_and_qkdn_ids/svc_link_id	ietf-yang-types: uuid	None	Universally Unique ID of the QKD service link which is providing QKD keys to the application.
svc_link_and_qkdn_ids/local_qkdn_id	ietf_yang_types:uuid	None	Unique ID of the local SD-QKD node which is providing QKD keys to the local application.
svc_link_and_qkdn_ids/local_qkdi_id	uint32	None	Interface ID of the local SD-QKD node which is connected to the QKD physical link.
svc_link_and_qkdn_ids/remote_qkdn_id	ietf_yang_types:uuid	None	Unique ID of the remote SD-QKD node that is providing QKD keys to the remote application. While unknown, the local SD-QKD will not be able to provide keys to the local application.
svc_link_and_qkdn_ids/remote_qkdi_id	uint32	None	Interface ID of the remote SD-QKD node, which is connected to the QKD physical link.
app_mapped_service_link/deploy_path	container	None	Container for a path deployed among candidate paths.
deploy_path/ path_id	uint32	None	Identifier of a path deployed for QKD service link. This value is chosen by SDNO from the path ids of candidate paths after SDNO has received candidate paths list from SDQNC.
deploy_path/ phys_links	leaf-list: ietf-yang-types: uuid	None	A sequence of Unique IDs of QKD physical links which constitute the deployed path.
svc_link/ bandwidth	uint32	None	Required bandwidth (bps) for the key association link. Used to reserve bandwidth from the physical QKD links to support the service key association link as an internal application.
svc_link/ performance	container	None	Performance of QKD service link.
performance/ expected_consumption	uint32	Config: false	The bandwidth (in bits per second) of the external application that is consuming keys from QKD service link.
performance/ skr	uint32	Config: false	Secret key generation rate (in bits per second) of QKD service link.
performance/ eskr	uint32	Config: false	Effective secret key generation rate (in bits per second) of QKD service link after the external application's consumption.
svc_link/ path_restoration	container	None	Container for path restoration conditions when any failure happens in the end-to-end key relay path.
path_restoration/ type	PATH-RESTORATION-TYPE	None	Path restoration type of QKD service link. It may be AUTOMATIC or MANUAL.
path_restoration/ hold-off-time	uint32	None	Time to wait before attempting a restoration.
svc_link/ svc_link_status	etsi-qkdn-types: LINK-STATUS-TYPES	Config: false	Status of each QKD service link.

5.4 End-to-end QKD service provisioning with path calculation

The information model includes the establishment and release requests of the end-to-end QKD service provisioning from the SDN orchestrator for secure application entities in the OTN.

For the end-to-end QKD-derived key delivery, the SDN orchestrator can request path calculations with constraints to the QKD network SDN controller. The SDN controller repeatedly provides a candidate paths list that satisfies the constraints received from the SDN orchestrator until the SDN orchestrator finally chooses a path and decides to deploy it for the QKD service link in the QKD network.

In case of any failure in the path for the QKD service link, the SDN orchestrator can request to change the established path with a new path which network operator chooses.

The SDN orchestrator can request to configure the following parameters and values from the SDN controller of the QKD network.

Table 7: QKD application parameters for QKD service provisioning

Name	Type	Detail	Description
qkd_app/ app_id	ietf-yang-types: uuid	None	Unique ID that identifies a QKD application consisting of a set of entities that are allowed to receive keys shared with each other from the specific QKD service link.
qkd_app/ app_type	etsi-qkdn-types: QKD-APP-TYPES	None	Application's type. All use cases described in the present document use applications working as external and their identity is CLIENT.
qkd_app/ app_priority	uint32	None	Priority of the association/application. This might be defined by the user but it is usually handled by a network administrator.
qkd_app/ server_app_id	inet:URI	None	ID that identifies the entity that initiated the creation of the QKD application to receive keys shared with one or more specified target entity identified by client_app_id. It is a client in the interface to the SD-QKD node and the name server_app_id reflects that it requested the QKD application to be initiated.
qkd_app/ client_app_id	leaf-list: inet:URI	None	List of IDs that identifies the one or more entities that are allowed to receive keys from SD-QKD node(s) under the QKD application in addition to the initiating entity identified by server_app_id.
qkd_app/ app_qos	container	None	Requested Quality of Service.
app_qos/ max_bandwidth	uint32	None	Maximum bandwidth (in bits per second) allowed for this specific application. Exceeding this value will raise an error from the local key store to the application and SDN orchestrator via the SDN controller. This value might be internally configured (or by an admin) with a default value.
app_qos/ min_bandwidth	uint32	None	This value is an optional QoS parameter that enables a minimum key rate (in bits per second) for the application.
app_qos/ jitter	uint32	None	This value allows to specify the maximum jitter (in millisecond) provided by the key delivery API for applications requiring fast rekeying. This value can be coordinated with the other QoS to provide a wide enough QoS definition.
app_qos/ ttl	uint32	None	This value is used to specify the maximum time (in seconds) that a key could be kept in the key store for a given application without being used.
local_qkdn_id	ietf_yang_types:uuid	None	Unique ID of the local SD-QKD node, which is providing QKD keys to the local application.
local_qkdn_id/ local_qkdi_id	uint32	None	Interface ID of the local SD-QKD node, which is connected to the QKD physical link.
remote_qkdn_id/	ietf_yang_types:uuid	None	Unique ID of the remote SD-QKD node which is providing QKD keys to the remote application. While unknown, the local SD-QKD will not be able to provide keys to the local application.
remote_qkdn_id/ remote_qkdi_id	uint32	None	Interface ID of the remote SD-QKD node, which is connected to the QKD physical link.
svc_link_id	ietf-yang-types: uuid	None	Universally Unique ID of the QKD service link, which is providing QKD keys to the application. This value is set by SDQNC and it can also be set by SDNO.

Table 8: QKD service link planning parameters for QKD service provisioning

Name	Type	Detail	Description
svc_link_and_qkdn_ids/svc_link_id	ietf-yang-types: uuid	None	Universally Unique ID of the QKD service link, which is providing QKD keys to the application.
app_mapped_svc_link/ path_constraints	container	None	Container for constraints when requesting a candidate paths calculation.
path_constraints/ include_nodes	list	Key: "include_qkdn_id include_qkdi_id"	List of SD-QKD nodes and their interfaces, which should be included in candidate paths list when SDQNC calculates a candidate paths list.
include_nodes/ include_qkdn_id	ietf-yang-types: uuid	None	Unique ID of the SD-QKD node which should be included in candidate paths list.
include_nodes/ include_qkdi_id	uint32	None	Interface ID of the SD-QKD node which should be included in a candidate paths list.
path_constraints/ exclude_nodes	list	Key: "exclude_qkdn_id exclude_qkdi_id"	List of SD-QKD nodes and their interfaces, which should be excluded in candidate paths list when SDQNC calculates a candidate paths list.
exclude_nodes/ exclude_qkdn_id	ietf-yang-types: uuid	None	Unique ID of the SD-QKD node, which should be excluded in candidate paths list.
exclude_nodes/ exclude_qkdi_id	uint32	None	Interface ID of the SD-QKD node, which should be excluded in candidate paths list.
app_mapped_svc_link/ candidate_paths	container	None	Container of end-to-end key relay paths per QKD service link between local SD-QKD node and remote SD-QKD node in the QKD network.
candidate_paths/ num_of_candidate_paths	uint32	None	The number of candidate paths which SDQNC calculates and provides.
candidate_paths/ candidate_path	list	Key: "path_id"	List of end-to-end key relay paths. Each path consists of QKD physical links from the local SD-QKD node to the remote SD-QKD node.
candidate_path/ path_id	uint32	None	Priority of the specific path among candidate paths. This value is defined by SDQNC.
candidate_path/ phys_links	leaf-list: ietf-yang-types: uuid	None	A sequence of Unique IDs of QKD physical links which constitute the specific end-to-end key relay path from the local SD-QKD node to the remote SD-QKD node.

Table 9: QKD service link deployment parameters for QKD service provisioning

Name	Type	Detail	Description
qkd_app/ app_id	ietf-yang-types: uuid	None	This value uniquely identifies a pair of external applications extracting keys from the specific QKD service link.
svc_link_and_qkdn_ids/svc_link_id	ietf-yang-types: uuid	None	Universally Unique ID of the QKD service link, which is providing QKD keys to the application.
app_mapped_service_link/deploy_path	container	None	Container for a path deployed among candidate paths.
deploy_path/ path_id	uint32	None	Identifier of a path deployed for QKD service link. This value is chosen by SDNO from the path ids of candidate paths after SDNO has received a candidate paths list from SDQNC.
deploy_path/ phys_links	leaf-list: ietf-yang-types: uuid	None	A sequence of Unique IDs of QKD physical links which constitute the deployed path.

5.5 Notifications

The SDN orchestrator can receive notifications from applications, SDN controller, QKD nodes, QKD interfaces, QKD physical links and QKD service links in case of an event, alarm, and performance threshold management. These notifications expose the information based on changes in QKD network entities or on a regular basis. The notifications from applications, QKD nodes, QKD interfaces, QKD physical links and QKD service links to SDN orchestrator include the list of notifications received by an SDN controller from QKD network entities and can be extended for the purpose of QKD network operation and management policy.

Additionally, any performance parameters and values such as QBER, secure key generation rate and effective secure key rate, can be configured by SDN orchestrator. In case of crossing the threshold of each performance parameter, the notifications from the QKD network entities such as QKD interfaces, QKD physical links, and QKD service links can be created for performance management by the SDN orchestrator.

The additional notifications from QKD network entities to the SDN orchestrator are added as follows:

QKD nodes:

- sdqkdn_new: Includes all the information about the new QKD node installed in the QKD network.
- sdqkdn_operating: Identifies a QKD node that is working as expected.
- sdqkdn_down: Identifies a QKD node that is not working as expected.
- sdqkdn_failure: Includes communication failure between SDN controller and the QKD node.
- sdqkdn_out: Identifies a QKD node that is switched off and uninstalled in the QKD network.

Applications:

- sdqkdn_application_deployed: Includes application and service link information when the path is deployed.

QKD physical links:

- sdqkdn_phys_link_down: Identifies a QKD physical link that is not working as expected.
- sdqkdn_phys_link_perf_update: Includes any major changes in the QKD physical link performance values.
- sdqkdn_phys_link_overloaded: Identifies a QKD physical link when its performance reaches the threshold.

QKD service links:

- sdqkdn_svc_link_down: Identifies a QKD service link that is not working as expected.
- sdqkdn_svc_link_perf_update: Includes any major changes in the QKD service link performance values.
- sdqkdn_svc_link_overloaded: Identifies a QKD service link when its performance reaches the threshold.

The SDN controller of the QKD network can provide the following parameters and values to the SDN orchestrator as a generic structure for events and alarms:

Table 10: Notification for QKD service

Name	Type	Detail	Description
notification_type	string	None	Type of notification.
message	string	None	Description of notification.
severity	etsi-qkdn-types: SEVERITY-TYPES	None	Severity levels of error or failure.
timestamp	ietf-yang-types: date-and-time	None	Date and time of the notification.
app_id	ietf-yang-types: uuid	None	This value uniquely identifies a pair of external applications extracting keys from the specific QKD service link.
app_status	etsi-qkdn-types: APP-STATUS-TYPES	None	Status of the application.
svc_link_id	ietf-yang-types: uuid	None	Universally Unique ID of the QKD service link.
svc_link_status	etsi-qkdn-types: LINK-STATUS-TYPES	None	Status of each QKD service link.
phys_link_id	ietf-yang-types: uuid	None	Universally Unique ID of the QKD physical link.
phys_link_status	etsi-qkdn-types: LINK-STATUS-TYPES	None	Status of each QKD physical link.
qkdn_id	ietf-yang-types: uuid	None	Unique ID of the SD-QKD node.
qkdn_status	QKDN-STATUS-TYPES	None	Status of the SD-QKD node.
qkdi_id	uint32	None	Interface id. It is described as a locally unique number, which is globally unique when combined with the SD-QKD node ID.
qkdi_status	etsi-qkdn-types: IFACE-STATUS-TYPES	None	Status of each QKD interface (module).

6 Sequence diagrams and workflows

6.1 Introduction

This clause describes how the information through the SDN orchestration interface of the QKD network is related to the QKD service operation.

In the first stage as the QKD network is introduced to the network operator's communications networks, the existing SDN orchestrator receives information about the topology of the QKD network from the QKD network SDN controller (as described in clause 5.1) when the QKD network is newly deployed, or there are some changes in the QKD network installation.

When the QKD network starts to operate, the SDN orchestrator receives information about the status of the QKD network and key resource inventory from the QKD network SDN controller (as described in clause 5.2), and this information is provided from the QKD network SDN controller to the SDN orchestrator on-demand or at periodic intervals.

After the SDN orchestrator requests a new end-to-end QKD service configuration from the QKD network SDN controller (as described in clause 5.4 and as shown in clause 6.2 below), the SDN orchestrator receives the information about ongoing end-to-end QKD service status from the QKD network SDN controller (as described in clause 5.3), and this information is provided from the QKD network SDN controller to the SDN orchestrator on-demand or at periodic intervals.

During the QKD network operation, event or alarm notifications (as described in clause 5.5) are made on the condition that a predefined condition occurs or a performance counter value crosses a threshold.

6.2 QKD service request and end-to-end service provisioning across multi-domain networks

This clause provides a use case in which the SDN orchestrator and the SDN controller in a QKD network and the SDN controller in an OTN interact to perform the end-to-end QKD service provisioning across the QKD network and OTN. The workflow as a sequence diagram shows the interactions and the exchange of information among the SDN orchestrator, the SDN controller in a QKD network and the SDN controller in an OTN. In this particular scenario, the SDN controller in a QKD network and the SDN controller in an OTN do not directly communicate with each other, and the SDN orchestrator requests the SDN controller in a QKD network and the SDN controller in an OTN respectively in the middle of its execution for the end-to-end QKD service provisioning.

A network administrator starts to make an end-to-end QKD service request from the SDN orchestrator with the information about the secure application entities in OTN nodes and their QoS, etc. With this service setup request received, the SDN orchestrator finds the address of each node matched in the network topology between the OTN node that is to receive QKD-derived keys and the QKD node that is to supply QKD-derived keys.

After finding this address matching between the QKD node and OTN node, the SDN orchestrator makes a QKD service link creation request from the SDN controller in a QKD network through the SDN orchestration interface with the information about the endpoints of the QKD service link and some constraints about the path calculation for the QKD service link, as described in Table 8. Then, the SDN controller in a QKD network issues the QKD service link id to designate the QKD service link calculates a candidate paths list for the QKD service link with those calculation constraints from the SDN orchestrator fulfilled and responds to the SDN orchestrator with the calculated candidate paths list, as described in Table 8. The request of candidate paths list with varying constraints by the SDN orchestrator and the calculation of candidate paths list with the priority of each path by the SDN controller in a QKD network can be made repeatedly until the selection of a certain path among candidate path lists by the SDN orchestrator.

After the SDN orchestrator selects which path from the candidate paths list will be used for the specific QKD service link, it requests to deploy the path in the QKD network from the SDN controller in a QKD network as described in Table 9. After QKD service link creation and QKD-derived key transport via the path for the QKD service link, the SDN controller in a QKD network responds to the SDN orchestrator that the QKD network is ready to provide QKD service using the QKD service link.

Then the SDN orchestrator requests the end-to-end QKD service provisioning with the information about the application using the QKD service link id as described in Table 7. The SDN orchestrator also requests from the SDN controller in an OTN to receive QKD-derived keys from the QKD nodes in the QKD network for the secure application entities in OTN. How the SDN orchestrator and the SDN controller in an OTN interact to perform a part of end-to-end QKD service provisioning through the northbound interface of the SDN controller in an OTN is outside of the scope of the present document.

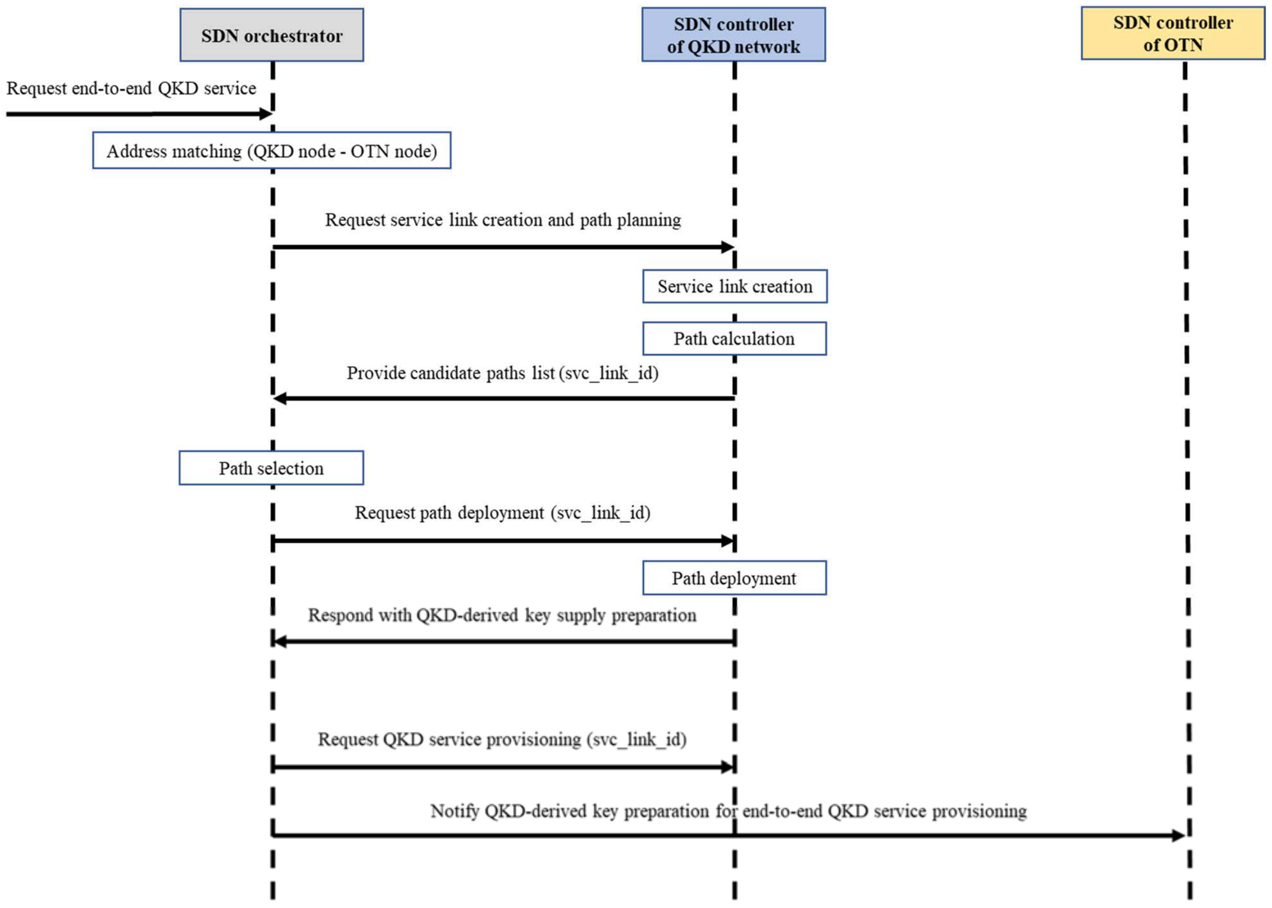

SDN orchestrator
SDN controller of QKD network
SDN controller of OTN
Request end-to-end QKD service
Address matching (QKD node - OTN node)
Request service link creation and path planning
Service link creation
Path calculation
Provide candidate paths list (svc_link_id)
Path selection
Request path deployment (svc_link_id)
Path deployment
Respond with QKD-derived key supply preparation
Request QKD service provisioning (svc_link_id)
Notify QKD-derived key preparation for end-to-end QKD service provisioning

Figure 2: Sequence diagram for end-to-end QKD service provisioning by an SDN orchestrator

The workflow description associated with end-to-end QKD service provisioning by an SDN orchestrator is as follows:

- Initially, a network administrator makes an end-to-end QKD service request from the SDN orchestrator with information about the secure application entities in OTN nodes and their QoS.
- The SDN orchestrator finds the appropriate SDN controller in a QKD network and the SDN controller in an OTN and matches the address of the OTN node under the SDN controller in an OTN to receive QKD-derived keys from the QKD node and the address of the QKD node under the SDN controller in a QKD network to provide QKD-derived keys for the secure application entities in the OTN node. The SDN orchestrator already has the address of all QKD nodes in the QKD network as it receives the QKD network topology and inventory information.
- The SDN orchestrator requests QKD service link creation and path calculation for QKD-derived key transport with path calculation constraints from the SDN controller in a QKD network through the SDN orchestration interface.
- The SDN controller in a QKD network issues QKD service link id, calculates a candidate paths list to meet those path calculation constraints from the SDN orchestrator, and responds with the calculated candidate paths list to the SDN orchestrator. The SDN orchestrator can repeatedly request a candidate paths calculation from the SDN controller in a QKD network until the SDN orchestrator selects a path for deployment in the QKD network from the candidate paths list from the SDN controller in a QKD network.
- After receiving information about the path for deployment from the SDN orchestrator, the SDN controller in a QKD network deploys the path in the QKD network. Then, QKD nodes generate QKD-derived keys and transport them along the deployed path through the QKD network until two QKD nodes in the QKD service link are ready to provide QKD-derived keys for the secure application entities in OTN nodes.

- When the SDN orchestrator receives a response that QKD-derived keys are ready for supply, the SDN orchestrator requests end-to-end QKD service provisioning from the SDN controller in a QKD network and notifies the SDN controller in an OTN of the preparation of QKD-derived keys in the QKD network and that OTN nodes can start to receive QKD-derived keys from the designated QKD nodes in the QKD network based on address matching information.

7 Security consideration

The SDN controller in a QKD network has a central view of the QKD network, and a subset of this information will be passed to the SDN orchestrator via the interface specified in the present document. Appropriate security measures should be considered in implementations of the orchestration interface between the SDN controller in a QKD network and the SDN orchestrator but outside the present document. However, QKD-derived keys for secure application entities are not handled by the SDN controller or the SDN orchestrator. Such QKD-derived keys can be isolated from both the control and orchestration interfaces. Careful design, installation and personalization procedures, etc., for the QKD network can help to mitigate attacks involving malicious requests from a compromised SDN controller or SDN orchestrator, as well as attacks on communication channels used in implementation of the related interfaces.

8 Protocol considerations

The orchestration interface of SDQNC can be implemented with application programming interfaces. Any application programming interface for network management consists of two parts. One part is the data model, which defines the language and content of the messages exchanged through the interface. The other part is the transport protocol, which defines all the rules that allow communication between the interacting entities.

YANG data modelling language IETF RFC 6020 [1] can be used to specify network management data models, which model both the configuration data and the operational state data. The YANG modules for the present document (see Annex A) use YANG version 1.1 in IETF RFC 7950 [2].

YANG data model can be transported over such protocols as NETCONF and RESTCONF. NETCONF in IETF RFC 6241 [3] is a network management protocol used to configure network devices. It uses basic operations to edit and query the network configuration on a device. RESTCONF in IETF RFC 8040 [4] is a REST-based protocol that runs over HTTP. It provides a subset of NETCONF functionality and is used to access YANG-defined data stored in a NETCONF defined data store.

In the RESTCONF, configuration data and operational state data can be retrieved with the HTTP GET method and configuration data can be modified with HTTP DELETE, PATCH, POST, and PUT methods. Data is encoded in either XML or JSON, and every resource is described using Uniform Resource Identifiers (URIs). The notification event defined by the YANG data model can be received by subscribing to the corresponding RESTCONF paths (HTTP URLs). The REST paradigm is convenient for the APIs implementation due to the need for stateless communication between client and server entities.

Annex A (normative): SDN orchestration interface YANG data models

A.1 General

YANG data models have been designed according to the present document and YANG modules have been produced using YANG 1.1.

A.2 YANG modules

There are 3 YANG modules corresponding to YANG data models according to the present document.

QKD network topology YANG module is available at:

https://forge.etsi.org/rep/qkd/gs018-orch-int-sdn/tree/v1.1.1/etsi-qkd-sdn-topology.yang

QKD network inventory YANG module is available at:

https://forge.etsi.org/rep/qkd/gs018-orch-int-sdn/tree/v1.1.1/etsi-qkd-sdn-inventory.yang

QKD network connectivity service YANG module is available at:

https://forge.etsi.org/rep/qkd/gs018-orch-int-sdn/tree/v1.1.1/etsi-qkd-sdn-connectivity.yang

Annex B (informative): Bibliography

- IETF RFC 6991 (July 2013): "Common YANG Data Types".
- IETF RFC 8345 (March 2018): "A YANG Data Model for Network Topologies".

Annex C (informative): Change History

Date	Version	Information about changes
October 2020	0.0.1	Early draft. Clauses 4.1 and 4.2.
December 2020	0.0.2	Clauses 5.1 and 5.2.
February 2021	0.0.4	Clauses 5.3, 5.4, 5.5, 6.1 and 6.2.
March 2021	0.0.5	Clauses 7 and 8.
June 2021	0.0.6	References, Bibliography, Corrections to the tables, Inclusion of YANG files.
July 2021	0.0.7	Figure 1 is clarified with added NOTE. IP address and port number of KME in SD-QKD node are added in clause 5.2. Server and client application identifiers are added in clause 5.3 and 5.4. Notifications are extended in clause 5.5. YANG modules are revised accordingly with new name and namespace proposed.
September 2021	0.0.8	The term virtual link is replaced by the term service link in the whole document. YANG modules are revised accordingly and are integrated with the updated QKD-015 YANG modules. The procedure of the use case is added in clause 4.1.
October 2021	0.0.9	Editorial changes and revisions.
November 2021	0.0.10	Figure 1 is modified.
November 2021	0.0.11	Clause 7 is revised. Editorial updates are made.

History

Document history		
V1.1.1	April 2022	Publication