IEC 63173-2

Edition 1.0 2022-05

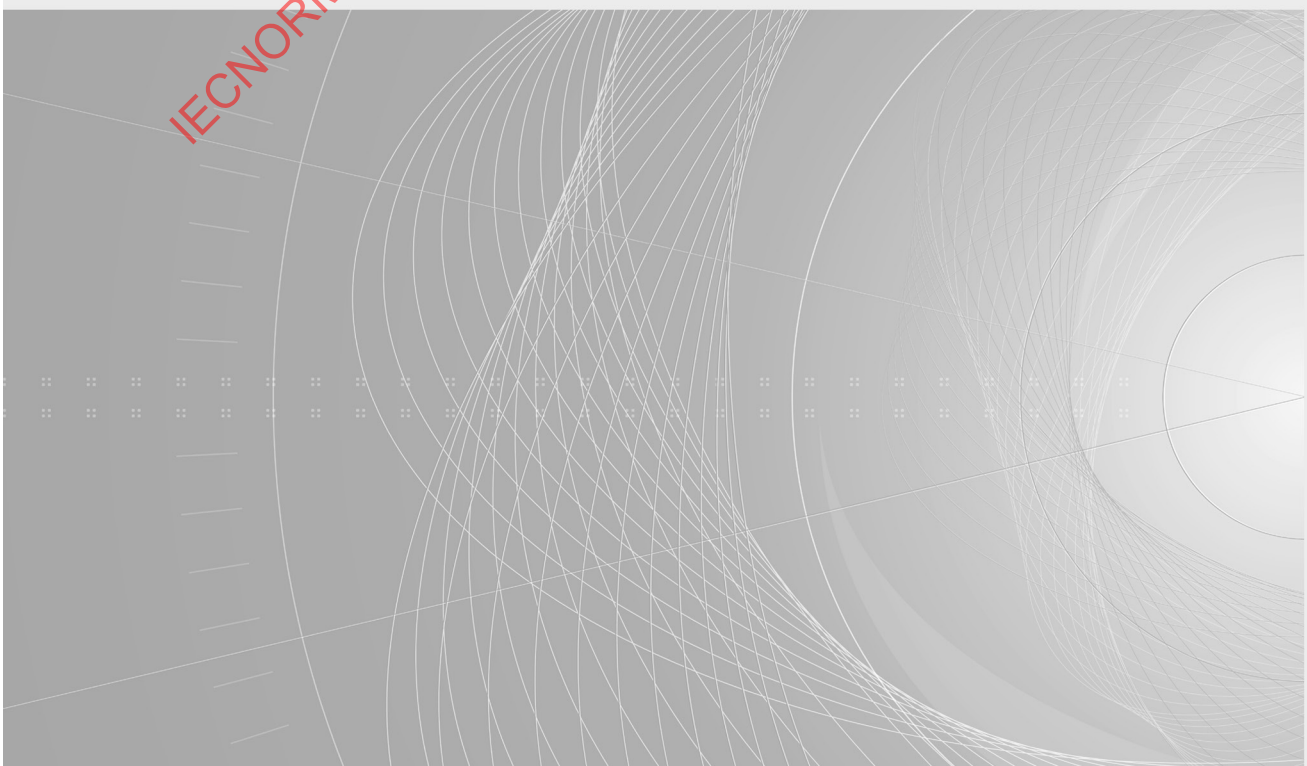
INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Maritime navigation and radiocommunication equipment and systems – Data interfaces – Part 2: Secure communication between ship and shore (SECOM)

Matériels et systèmes de navigation et de radiocommunication maritimes – Interfaces de données – Partie 2: Communications sécurisées entre le navire et la terre (SECOM)



IEC 63173-2:2022-05(en-fr)

IEC Secretariat
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

IEC 63173-2

Edition 1.0 2022-05

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Maritime navigation and radiocommunication equipment and systems – Data interfaces – Part 2: Secure communication between ship and shore (SECOM)

Matériels et systèmes de navigation et de radiocommunication maritimes – Interfaces de données – Partie 2: Communications sécurisées entre le navire et la terre (SECOM)

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 47.020.70

ISBN 978-2-8322-3802-8

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

MARITIME NAVIGATION AND RADIOCOMMUNICATION EQUIPMENT AND SYSTEMS – DATA INTERFACES –

Part 2: Secure communication between ship and shore (SECOM)

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

IEC 63173-2 has been prepared by IEC technical committee 80: Maritime navigation and radiocommunication equipment and systems. It is an International Standard.

The text of this International Standard is based on the following documents:

Draft	Report on voting
80/1030/FDIS	80/1039/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/standardsdev/publications.

A list of all parts in the IEC 63173 series, published under the general *Maritime navigation and radiocommunication equipment and systems – Data interfaces*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

E-navigation has been defined as the means of providing electronic information in a harmonized way, and maritime services have been specified by the International Maritime Organization (IMO). The maritime services are operational services for actors both ashore and onboard. To make the maritime services interoperable between different actors and systems from different manufactures standards, specifications and guidelines in several layers are required, for example technical services and data/product formats. Technical services comprises a set of technical solutions and communications means to provide a maritime service. IMO's e-navigation strategy implementation plan (SIP) requires that all maritime services are IHO S-100 conformant as a baseline. Further, IEC is expected to implement the details as outlined in the SIP.

Secure communication between ship and shore (SECOM) provides standards for secure data exchange with technical services. Further, it contains a technical service interface design that is in accordance with the service guidelines and templates defined by IALA and partly included in IHO S-100.

SECOM specifies service interfaces (APIs) for data exchange, data protection measures to enable secure communication and interfaces for service discoverability. SECOM is applicable for IHO S-100 based products but also other data (payload) formats are supported, i.e. SECOM is generally independent of which data type is exchanged.

The standardisation of a common service interface for data exchange will enable wider technical interoperability where the same service interface can be used for exchanging information regardless of its operational use.

Accordingly, the purpose of SECOM is to:

- facilitate standardized information exchange of, for example, IHO S-100 based products part of maritime services such as route plans, nautical chart updates and navigational warnings;
- facilitate interoperability between maritime IT systems;
- reduce the need to support many different (proprietary) service designs;
- utilize the benefits of service oriented architecture in maritime communication, for example to enable ship systems to interact with port systems on the first call to a specific port.

MARITIME NAVIGATION AND RADIOCOMMUNICATION EQUIPMENT AND SYSTEMS – DATA INTERFACES –

Part 2: Secure communication between ship and shore (SECOM)

1 Scope

The scope of SECOM includes interfaces (APIs) for data exchange (information services), information security measures to enable secure communication and interfaces for service discoverability. SECOM provides technical interoperability, where the same service interface is used for exchanging the information regardless of its operational use, up to the level of exchanging information securely online. Although designed for IHO S-100 based products, SECOM is technically payload agnostic and applicable also for other types of data.

Communication between SECOM information services for data exchange relies on IP based web services. The "last mile" links between a SECOM information service and the end-user application is not defined in this document, thus the communication technology between the vendor API and a ship/shore system can be non-IP based as well as IP based. The informative Annex D describes one such implementation of this. This allows different solutions between the service and shore/ship's system/applications.

SECOM does not define physical layer or link layer for transport of data between SECOM information services, but requires that the transport supports IP communication. SECOM is applicable for both public (governmental) and private (business) services. SECOM is applicable for ship-shore and shore-ship communication, and can be used for ship-ship communication.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IHO S-100:2018, *IHO Universal Hydrographic Data Model, ed. 4.0.0*

RFC 2315, *PKCS #7: Cryptographic Message Syntax*

RFC 2459, *Internet X.509 Public-key infrastructure and attribute certificate frameworks*

RFC 2818, *HTTP Over TLS (2000)*

RFC 2986, *PKCS #10: Certification Request Syntax Specification*

RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*

RFC 5246, *TLS version 1.2 (2008)*

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 6960, *X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP*

RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*

RFC 8446, *TLS version 1.3 (2018)*

3 Terms, definitions and abbreviated terms

For the purposes of this document the following terms, definitions and abbreviated terms apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org
- ISO Online browsing platform: available at http://www.iso.org/obp

3.1 Terms and definitions

3.1.1
access control
access authorization and access restriction

Note 1 to entry: Access control refers to all the steps that are taken to selectively authorize and restrict entry, contact, or use of assets. Access authorizations and restrictions are often established in accordance with business and security requirements.

[SOURCE: ISO/IEC 27000:2018, 3.1, modified – The definition has been rephrased, and a note to entry added.]

3.1.2
actor
role played by a user or any other system that interacts with the subject

[SOURCE: Unified Modeling Language:2017, 18.2.1.1]

3.1.3
authentication
process to confirm that a claimed characteristic of an entity is actually correct

Note 1 to entry: To authenticate is to verify that a characteristic or attribute that appears to be true is in fact true.

[SOURCE: ISO/IEC 27000:2018, 3.5, modified – The definition has been reformulated, and the note to entry added.]

3.1.4
authentication data
information used to verify the claimed identity of an entity, such as an individual, defined role, corporation or institution

[SOURCE: ISO 21188:2018, 3.4]

3.1.5
authorization
granting of rights, which includes the granting of access based on access rights

[SOURCE: ISO 7498-2:1989, 3.3.10]

3.1.6
certificate
public key and identity of an entity (*authentication data* (3.1.4)), together with some other information, rendered unforgeable by signing the certificate information with the private key of the certifying authority that issued that public key certificate

[SOURCE: ISO 21188:2018, 3.8]

3.1.7
data client
consumer of data

EXAMPLE A vessel.

3.1.8
data server
provider of data

EXAMPLE Navigational warning centre.

3.1.9
digital signature
data appended to, or cryptographic transformation of, a data unit that allows the recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient

[SOURCE: ISO 7498-2:1989, 3.3.26]

3.1.10
encryption key
cryptographic secret key used with symmetric cryptographic techniques

3.1.11
hash
string of bits which is the output of a function that calculates a value of the input bytes according to some algorithm, for example checksum calculation, compression, SHA-256

Note 1 to entry: The literature on this subject contains a variety of terms that have the same or similar meaning as hash-code. Modification detection code, manipulation detection code, digest, hash-result, hash-value and imprint are some examples.

Note 2 to entry: NIST SP 800-63B uses "message digest" for "hash".

[SOURCE: ISO/EC 10118-1:2016, 3.3, modified – The definition has been completed, and Note 2 to entry added.]

3.1.12
identity registry
registry that holds the identities and bindings to public keys

3.1.13
integrity
<information security> protection of the accuracy and completeness of information

[SOURCE: ISO/IEC 27000:2018, 3.36, modified – The definition has been rephrased.]

3.1.14
javascript object notation
JSON
lightweight, text-based, language-independent syntax for defining data interchange formats

[SOURCE: IEC 21778:2017]

3.1.15
private key
cryptographic key of an entity's asymmetric key pair which can only be used by that entity

[SOURCE: ISO/IEC 11770-3:2021, 3.32, modified – The definition has been reformulated, and the note to entry added.]

3.1.16
public key
cryptographic key of an entity's asymmetric key pair which can be made public

[SOURCE: ISO/IEC 11770-3:2021, 3.33, modified – The definition has been reformulated and the note to entry deleted.]

3.1.17
public key infrastructure
PKI
structure of hardware, software, people, processes and policies that employs digital signature technology to facilitate a verifiable association between the public component of an asymmetric public key pair with a specific subscriber that possesses the corresponding private key

Note 1 to entry: The public key may be provided for digital signature verification, authentication of the subject in communication dialogues, and/or for message encryption key exchange or negotiation

[SOURCE: ISO 21188:2018, 3.48]

3.1.18
representational state transfer
REST
software architectural style that defines a set of constraints to be used for creating Web services

Note 1 to entry: Web services that conform to the REST architectural style, called RESTful Web services (RWS), provide interoperability between computer systems on the Internet

[SOURCE: Roy Fielding – Representational State Transfer (REST)]

3.1.19
route plan exchange format
RTZ
format based on standardizing a route plan, which consists of waypoints where each waypoint contains information related to the leg from the previous waypoint

Note 1 to entry: The route exchange format is a file – RTZ – containing an XML coded version of the route plan.

[SOURCE: IEC 61174:2015]

3.1.20
scheme administrator
SA
body solely responsible for maintaining and coordinating the protection scheme, by controlling membership of the scheme using the identity registry to ensure that all participants operate according to defined procedures

Note 1 to entry: The SA maintains the top level digital root certificate used to operate the protection scheme and is the only body that can certify the identity of the other participants of the scheme. The SA is responsible for distributing the certificate directly to all registered users participating in the protection scheme.

3.1.21
service
mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description

[SOURCE: OASIS Reference Model for Service Oriented Architecture 1.0]

3.1.22
service consumer
user which uses service instances provided by service providers

Note 1 to entry: All users within the maritime domain can be service customers, for example ships and their crew, authorities, VTS stations, organizations (e.g. meteorological), commercial service providers, etc.

[SOURCE: IALA Service Documentation Guidelines G1128]

3.1.23
service instance
service implementation which can be deployed at different places by same or different service providers

3.1.24
service provider
user which provides instances of services according to a service specification and service instance description

Note 1 to entry: All users within the maritime domain can be service providers, for example authorities, VTS stations, organizations (e.g. meteorological), commercial service providers, etc.

[SOURCE: IALA Service Documentation Guidelines G1128]

3.1.25
service registry
registry (e.g. database) that holds public descriptions (metadata) of services

3.1.26
nonce
random or non-repeating value that is included in data exchanged by a protocol, usually for the purpose of guaranteeing the transmittal of live data rather than replayed data, thus detecting and protecting against replay attacks

[SOURCE: NIST, Computer security resource center]

3.2 Abbreviated terms

AES advanced encryption standard
API application program interface
CA certificate authority
CBC cipher block chaining
CRL certificate revocation list
CSR certificate signing request
DSA digital signature algorithm
DSS digital signature standard
ENC electronic navigational chart
IALA International Association of Marine Aids to Navigation and Lighthouse Authorities
IHO International Hydrographic Organization
IMO International Maritime Organization
ISO International Organization for Standardization
MRN maritime resource name
OCSP online certificate status protocol
OEM original equipment manufacturer
PEM privacy enhanced mail
PKCS public key cryptography standards
PKI public key infrastructure
SA scheme administrator
SSK self signed key
TLS transport layer security
UML unified modeling language
URI uniform resource identifier
URL uniform resource locator (web address)
VTS vessel traffic service
WKT well-known text

4 General description of SECOM

4.1 General

SECOM includes information services interfaces (APIs) for data exchange, information security measures by a SECOM PKI, communication channel security and data protection to enable secure communication. Further, SECOM includes interfaces for service discoverability as shown in Figure 1. The purpose with these included components is for SECOM to provide technical interoperability, where the same service interface is used for exchanging the information regardless of its operational use, up to the level of exchanging information securely online. Although designed for IHO S-100 based products, SECOM is technically payload agnostic and applicable also for other types of data.

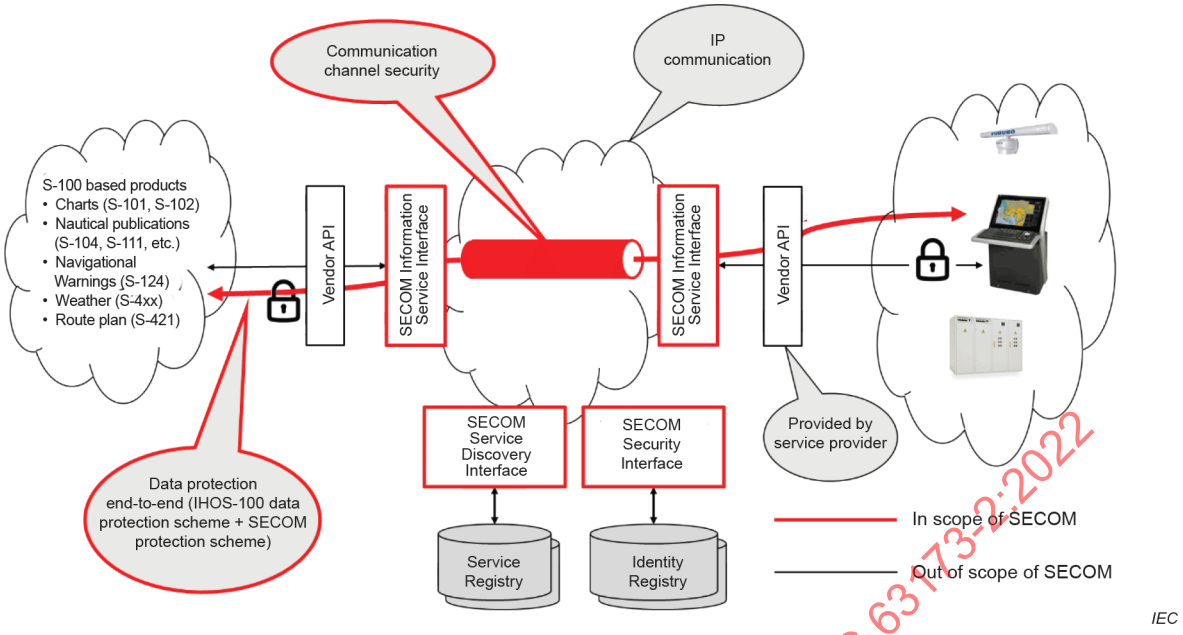


Figure 1 – Overview of SECOM

The SECOM information service interface includes the public side exposed on the Internet as depicted in Figure 1. The "last mile" links between a SECOM information service interface and the end-user application is not defined in this document and hence different solutions between the service instance and shore/ship's system are possible. The informative Annex D describes one such implementation.

SECOM information security contains communication channel security, a variant of PKI (public key infrastructure) and data protection scheme alternatives for the information exchange with full or partial compliance with IHO S-100. The data protection is provided between end-users. SECOM PKI includes the definition of a set of service interfaces for key management.

The service discovery interface includes operations to search for service instances from a service registry to meet some criteria, for example chart updates, navigational warnings, updated estimated time of arrival (ETA) information or route optimization services. The service discovery interface allows the user to choose a service instance to consume.

The information exchange between actors is bi-directional, which means that both the ship/shipping company as well as shore authorities and private service providers can initiate the information exchange and act as information provider. SECOM is thus applicable for both ship-to-shore and shore-to-ship communication. Guidance for implementation is available in Annex D.

4.2 Information service interface

The standardisation of service interfaces for information exchange enables a user that implemented the interface, for example for exchange of S-421 Route Plan (IEC 63173-1), to consume a set of operational services with different purpose, but all based on the exchange (push) of S-421 Route Plan. The standard service interface also contains interfaces for subscription, access request, pulling and dynamic request for the instance capability, status and description. See Annex B for the detailed need for each interface and operation.

Implementing a standardized interface creates significant advantages for a ship as opposed to being forced to implement a vast number of different interfaces in order to adhere to proprietary interfaces from various service providers and consumers. Allowing manufacturers of maritime equipment, such as ECDIS manufacturers, to focus on a standardised service interface will facilitate the implementation of information products onboard ships and generate less requirements for changes onboard when new services are introduced.

Service providers will have similar benefits in producing services based on exchange of standard information products, hereby gaining technical interoperability with all ships having implemented the standard service interface. Since most SECOM interfaces are optional, a service provider of a SECOM information service implementation can decide and pick what to support and hence, enable the possibility to tailor the service to its business operational needs. Each SECOM information service capability can be retrieved from the SECOM information service itself.

The SECOM information service interface is designed for IHO S-100 based products, but does also support arbitrary payloads using the SECOM data protection scheme for the exchange. The data can be sent either as open and signed or encrypted and signed.

4.3 Information security

4.3.1 Measures

SECOM contains several measures to meet the following quality attributes.

- Authentication is a process by which a system verifies the identity of a user (human or machine) who wishes to access it (i.e. to confirm, you are who you claim to be, like a passport). This corresponds to the written signature and official stamp on paper. Authentication is achieved in SECOM on two levels: 1) data authentication through the signature (SECOM data protection), and 2) service authentication through the communication channel security and X.509 (RFC 5280 and RFC 2459) certificates (SECOM communication channel security). Authentication is dependent on unique identities which is achieved by SECOM PKI.
- Integrity is achieved in SECOM by signing data where the checksum is part of the signature, described in SECOM data protection. The signing and verification of the signature is dependent on unique identities which is achieved by SECOM PKI.
- Confidentiality is achieved in SECOM on two levels: 1) data encryption according to SECOM data protection and IHO S-100, and 2) secure communication channel (transport security) for IP and HTTP through TLS and X.509 certificates, described in SECOM communication channel security.
- Access control uses authorization which is the process of determining a set of permissions that is granted to a specific trusted identity. It is achieved in SECOM by the information owner authorizing access to chosen identities from the common identity registry, described in SECOM PKI. Only public interfaces are defined in SECOM (e.g. request access).
- Identification is achieved in SECOM by the common registry for unique identities, bindings to asymmetric key pair and standardized API to SECOM PKI.
- Non-repudiation, i.e. a service intended to protect against the originator's false denial of having created the content of a message and of having sent a message, is achieved in SECOM by signing the data with the private key.

4.3.2 SECOM PKI

SECOM PKI (public key infrastructure) describes the common interface for key management and the expected functionality to provide the binding between identity and key used for signing data and authentication in service interaction.

A SECOM PKI can be provided by any internationally recognized PKI provider complying with the SECOM PKI requirements. A SECOM PKI provides an interface and an identity registry governed by its scheme administrator. There can be several instances of SECOM PKI provided by multiple scheme administrators. An actor can participate/be registered in several instances of SECOM PKI. SECOM compatible equipment are expected to support multiple instances of SECOM PKI.

SECOM does not specify which instance of identity registry to use, and SECOM does not define or constrain number of identity registries.

4.3.3 Communication channel security

When consuming service instances according to SECOM, the Internet transport is protected with TLS and valid certificates from trusted party as stated in SECOM communication channel security. The protection of the channel is a complement to the protection of the data itself and is necessary to be included to secure also other service requests, such as subscription request, access request and notifications. The SECOM communication channel security describes the usage of certificate obtained from a trusted identity registry and thereby enables authentication on the service interaction itself as depicted in Figure 2.

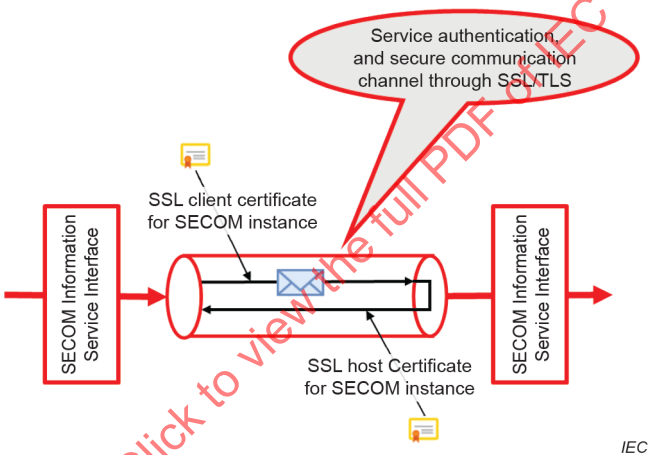


Figure 2 – Secure communication channel

The SECOM communication channel security relies on a SECOM public key infrastructure (SECOM PKI) or public-private key management using X.509 certificates to exchange the keys.

The SECOM communication channel security scheme does not comprise the "last mile" links between the SECOM information service interface and the end-user application. In Annex D, there are informative examples and guidance of how such protection could be achieved.

4.3.4 Data protection

Data protection includes both signing of data for authentication and integrity (digital signature), and optionally encryption of data for confidentiality. Data protection is aligned with the IHO data protection scheme (IHO S-100 part 15).

The digital signature contains both the checksum used to check integrity of received data, and claimed identity for data authentication. The verification of the signature includes access to common public key infrastructure (PKI) where the public key for claimed identity can be downloaded. Whenever data is exchanged, the digital signature shall always be attached and verified as close to the end user as possible. This gives the basic data end-to-end protection, see Figure 1.

The optional encryption of data uses a common protection scheme and offers two alternatives for encryption key management: 1) the existing encryption key management with permits for ENC data, such as IHO, can be used with SECOM information service interface and 2) SECOM also includes its own encryption key management that is specially designed for more dynamic data exchange ship to shore and shore to ship, such as route plan exchange. In the SECOM data protection scheme, the sender of classified data creates a temporary encryption key which is transferred securely before the encrypted data is transferred. The temporary encryption key is encrypted with the receiver's public key, the same key as used for authentication, thus can only be decrypted by the actor having the private key in its possession. The receiver's public key if not already present, can be retrieved from the receiver's SECOM interface PublicKey, see 5.7.16, or from the SECOM PKI interface GetPublicKey, see 8.6.3. This requires a strong key pair long enough for encrypting and decrypting the encryption key.

The data protection described above only protects the data part in the message. To secure the complete message, an additional signature is added, an envelope signature that includes all information of the envelope contained in the uploadObject, including the data and its dataSignature. The parts of the message that are protected by the different signatures are illustrated in Figure 3.

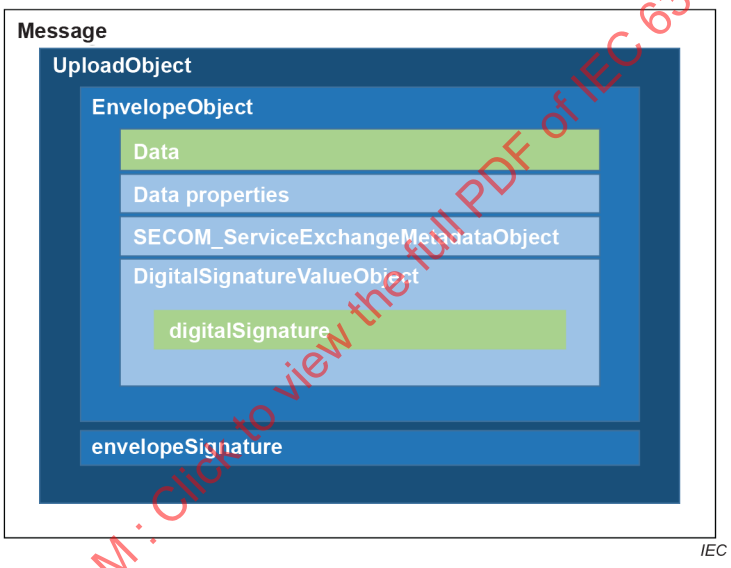


Figure 3 – Illustration of what parts of the message are protected by the two signatures

Another rationale for introducing an envelope signature is to facilitate complete message integrity check at the receiving SECOM information service side thus prohibiting forwarding of corrupt messages during the last mile to the receiver, see Figure 4.

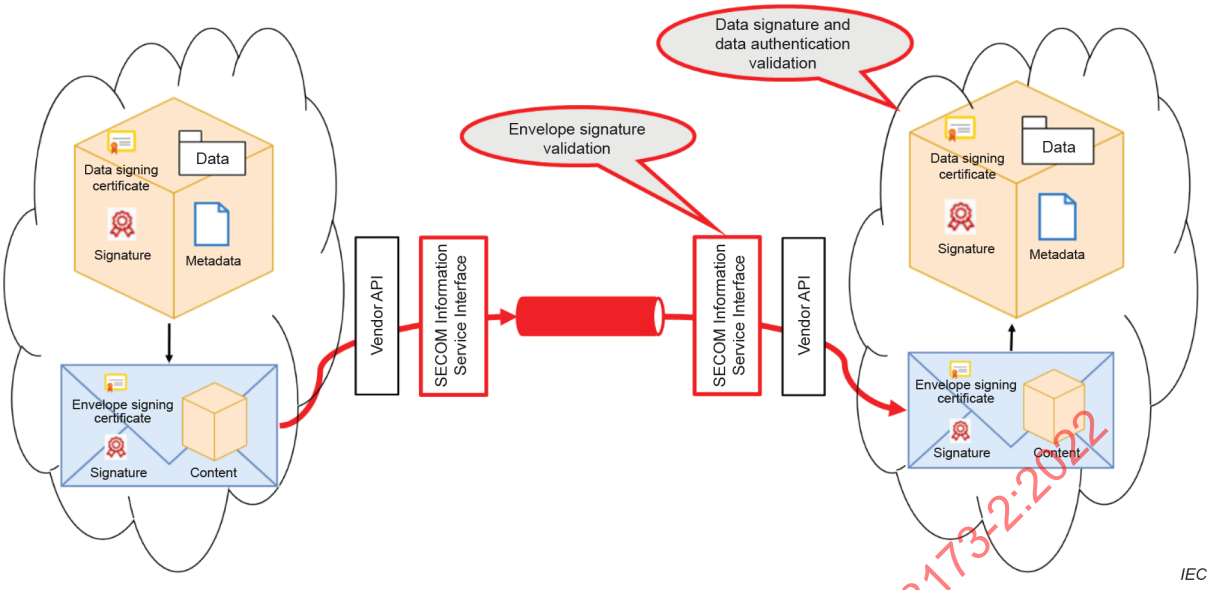


Figure 4 – Envelope and data validation

The SECOM data protection scheme relies on a SECOM public key infrastructure (SECOM PKI) using X.509 certificates to exchange the keys. The private key is generated by the actor and stored internally while the public key is securely uploaded to a SECOM PKI and attached to the identity.

The data protection scheme and the thumbprint of the root certificate indicates the identity registry used in the specific data exchange. The trusted root certificates identified with their thumbprints are expected to be stored in a local trust store.

4.3.5 Certificate revocation status

A certificate revocation list (CRL) is a list of revoked certificates that is downloaded from the certificate authority (CA). Online certificate status protocol (OCSP) is a protocol for checking revocation of a single certificate interactively using an online service called an OCSP responder (see 8.5).

4.4 Service discoverability

To support dynamic use of services, a SECOM service discovery interface has been defined. The intended operational usage is to make it possible for both officers onboard ships and operators ashore to search for service instances to interact with. Some examples from an onboard perspective are a ship that wants to find a provider of route optimization services or a VTS along the intended route to share the route plan with or providers of navigational warnings or electronic navigational charts (ENC). From a shore-based perspective, it is possible to search for registered ships targeted for information requests such as required reporting information, intended route or estimated arrival time. The SECOM service discovery contains common interfaces for discovery of service instances.

A standardized interface for finding service instances reduces the need to support several different proprietary and company specific interfaces. Further, it facilitates increased stability of the implementation, i.e. no new implementation and certification is necessary if a new service registry is used.

SECOM does not specify which service registry to use, and SECOM does not define or constrain the number of service registries, hence there can be several providers of service registries.

SECOM defines only the interface for discovery of service instance, not the registration of service instances in the service registry. This needs to be described by the provider of the chosen service registry.

4.5 Structure of this document

The following clauses are included in SECOM:

- Clause 5 SECOM information service interface;
- Clause 6 SECOM communication channel security;
- Clause 7 SECOM data protection;
- Clause 8 SECOM PKI;
- Clause 9 SECOM service discovery service interface;
- Clause 10 SECOM error cases;
- Clause 11 Test methods and expected results.

5 SECOM information service interface

5.1 General

This Clause 5 describes the SECOM information service interface for exchanging IHO S-100 based products and other payloads. The implementation of the SECOM information service interface supports digital signature and optional data encryption. The SECOM information service interface requires the use of common certificates from the SECOM PKI for authentication between service instances, as described in SECOM communication channel security.

The definition below contains a compilation of several service interfaces that, combined together, constitutes the SECOM information service interface.

In the definition, the expression "consumer" refers to the data client, and the expression "provider" refers to the data server.

- Four of the service interfaces are used for the actual exchange of data; Upload, Upload Link, Get and Get By Link. The Upload service interface is used by providers of information to push data to a consumer with maximum size of 350 kB encoded as Base64 (see RFC 2045). This covers the primary need for exchange of data. The Get interface is used when a consumer pulls data from the provider. For larger amounts of data where the upload (push) cannot be used, the Upload Link interface is used to send a link to the data, and then the data is retrieved with the Get By Link interface.
- The Get Summary interface is used to receive a short summary of available data from where the user can select one or several items to be retrieved by using the Get interface. The Subscription interface is used when a consumer wants to create a subscription to data from the provider. The provider then sends data to the consumer using the Upload service interface. The consumer uses the Remove Subscription interface to delete the subscription. The provider can also "nominate" a consumer by creating a subscription for the consumer using proprietary methods within the provider's system. The provider then uses the Subscription Notification to inform the consumer that there is an active subscription. The provider can also remove a subscription using proprietary methods, and the provider then uses the Subscription Notification to inform the consumer.
- If a consumer does not have access to data, for example retrieval of specific data is not authorized, the consumer uses the Access interface from the provider. The provider then responds with Access Notification to the consumer with the decision, accept or deny.

- Both the provider and consumer use the Capability interface when requesting information regarding accessible interfaces and available types of data. This facilitates for example functions to ask for which version of the S-100 based Product specification a provider or consumer accepts, and functions to only present accessible interfaces for a certain service instance, thus avoiding incorrect use of the service instance.
- The Ping interface is used to check the technical status of the service instance.
- The EncryptionKey interface is used for secure exchange of the symmetric encryption key.
- The PublicKey interface is used for exchanging public certificates used for data encryption and digital signatures. Through this interface, a consumer can both request a certificate (pull) and upload a certificate (push).

5.2 How to read descriptions of service interface definition

The service interface is described according to the service definition model in IHO S-100:2018, part 14 "Online Data Exchange".

The SECOM information service interface constitutes a set of service interfaces, where each service interface is first described as a specification (technology agnostic) and then a technical design of the service interface in REST technology.

Figure 5 describes how to read the service interface as specified in the UML diagram, showing the interface name with the operation(s) defined, input and output. The input and output is defined in the data exchange model. Each interface provides (exposes) at least one service interface (operation) and may require (consume) other service interface(s).

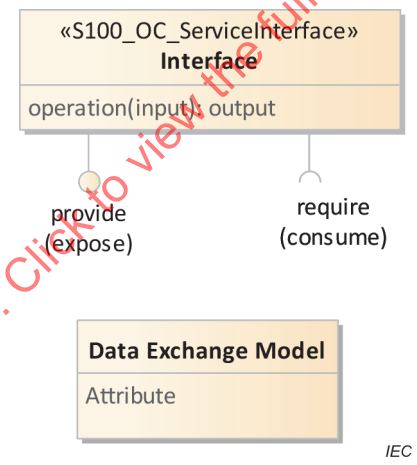


Figure 5 – Service definition model for the service interface definitions

The service specification of the service interface definition describes information for input and output in tables. The multiplicity ("Mult") column describes if data for the attribute is required or not, where 0..1 and 0..* indicates the value may be sent, and 1..* and 1 indicates that value shall be sent (i.e. is required to send). The processing column describes if received data is mandatory or optional to process. If mandatory, the data shall be processed according to the description in dynamic behavior for the service. The processing may include sending asynchronous response. See Table 1.

Table 1 – Read instructions for tables in service interface definitions

<information object>				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
Name of attribute	This describes for the implementer of the service call (i.e. initial sender of request) if the attribute is required to be sent in the service call or if the attribute may be omitted. Describes for the implementer of the service interface (i.e. responder of initial request) if the attribute is required or not. If an attribute is required by the multiplicity but the attribute is not present, the service call is considered "Bad request" and the service call shall be ignored. Alternatives of multiplicity as defined in UML. 0..1 0..* 1..* 1	This describes for the implementer of the service call if processing of the attribute by the receiver of sent data (i.e. implementer of the service interface) is mandatory or optional. This describes for the implementer of the service interface if processing of the attribute is mandatory or optional. Alternatives of processing: Mandatory Optional	This describes the logical (service design independent) encoding of the attribute	The definition and description of the attribute

Each service interface also describes a REST service interface design. A table describes each operation in the interface designed according to REST technology. For service interface designed according to REST, the type of data is JSON. See Annex A for a formal and detailed definition of the whole SECOM information service interface as REST service.

5.3 Service technology and service transportation protocol

The technology (architectural style) chosen is REST (REpresentational State Transfer) upon HTTP/1.1 (RFC 7231).

REST is an architectural style, and an approach to communications that is often used in the development of Web services. The use of REST in SECOM is preferred over other more heavyweight protocols such as SOAP (simple object access protocol) because REST does not leverage as much bandwidth, which makes it a better fit for use in communication between vessels and shore based representation of the same.

REST, which typically runs over HTTP (hypertext transfer protocol), has several architectural constraints.

- Decoupling – Decouples consumers from producers which suits SECOM decentralized architecture well.
- Stateless existence – Also a good prerequisite for a decentralized architecture design.
- Able to leverage a cache – Probably less important in SECOM since most of the interaction is between machines, although for services with man-machine interfaces this is of importance.
- Leverages a layered system – SECOM is dependent on good scaling capabilities which REST supports.
- Leverages a uniform interface – Again since SECOM defines the available services centrally in one or several service registry(s), this constraint supports implementations being decoupled from the services they provide.

The definition of each operation defines additional error messages that the operation specifically shall respond with as a complement to the HTTP response codes defined by HTTP/1.1 (RFC 7231).

5.4 Service interface versioning

The version on the service interface definition follows the version of this document.

The version on the design as REST service interface is reflected in the path for the service instance. The version is defined in the formal REST service definition as OpenAPI specification (Swagger) in Annex A.

Table 2 describes the structure of the versioning of the SECOM service interface.

Table 2 – SECOM Service interface versioning

REST method	URL	base URL with version	SECOM REST operation
POST	https://iec.ch	/v1	/object

Example
POST https://iec.ch/v1/object{…}

NOTE The versioning described here is the version of the SECOM information service interface defined in this document. Version of the service instance itself or other versioning by the manufacturer can be included in the URL.

5.5 Pagination

The Get and Get Summary interfaces may return more than one result. In order to limit the response size, these interfaces support pagination.

When requests to these service interfaces are made, the optional "page" and "pageSize" parameters may be specified by the client. If provided, these allow the client to request how many items are to be returned and the starting point of these items. For example, if a particular request would produce a result consisting of 30 items, setting page = 1 and pageSize = 20 would return the first twenty items; and page = 2, pageSize = 20 would return the remaining ten.

Regardless of whether page or pageSize is specified, the server shall always return the total number of items in the totalItems attribute, and the maximum number of items per page in the maxItemsPerPage attribute.

Errors shall be returned by the server if either an invalid page is asked for or if too many items are requested at a time.

If the page parameter is not present, the server shall send back the first page of the result.

If the pageSize parameter is not present, the server shall send back the maximum number of items per page that it supports.

If the page parameter is set to 0, the server shall not return any items, but shall return maxItemsPerPage. This value shall be constant for any query on the server (so that the client only needs to request this once).

5.6 Common information objects and data types

5.6.1 General

The information objects shared among several interfaces are shown in Table 3 to Table 14.

5.6.2 Basic data types

Table 3 – Basic data types

Name	Description
Integer	A signed integer number, the representation of an integer is encapsulation and usage dependent. EXAMPLE 29, -65547
PositiveInteger	An unsigned integer number greater than 0.
Boolean	A value representing binary logic. The value can be either true or false.
CharacterString	A CharacterString is an arbitrary-length sequence of characters including accents and special characters from repertoire of one of the adopted character sets
Date	A date gives values for year, month and day according to the Gregorian Calendar. Character encoding of a date is a string which shall follow the calendar date format (complete representation, basic format) for date. EXAMPLE 19980918 (YYYYMMDD)
Time	A time is given by an hour, minute and second in the 24-hour clock system. Character encoding of a time shall be a complete representation of the basic format. Complete representation means that hours, minutes and seconds shall be used. Basic format means that separating characters are omitted. Time is preferably expressed as Universal Time Coordinated (UTC). EXAMPLE 183059Z Time may be expressed as a Local Time with a given offset to UTC. EXAMPLE 183059+0100 Time may be expressed as a Local Time without a specified offset to UTC. EXAMPLE 183059 The complete representation of the time of 27 min and 46 s past 15 h locally in Geneva (in winter one hour ahead of UTC), and in New York (in winter five hours behind UTC), together with the indication of the difference between the time scale of local time and UTC, are used as examples. Geneva: 152746+0100 New York: 152746-0500 The service hours for a service that is available all year in an area where Daylight Saving Hour affects the offset to UTC could be expressed as Local Time without specified offset. Opening: 074500 Closing: 161500
DateTime	A DateTime is a combination of a date and a time type. Character encoding of a DateTime shall follow as the above. EXAMPLE: 19850412T101530
byte[]	Array of bytes

5.6.3 SECOM_ExchangeMetadataObject

The objects shown in Table 4 and Table 5 contain metadata attributes with information of protection, compression and public keys used for the digital signature of the data. The order of adding attributes as part of the generation and verification of the envelope signature is achieved by traversing the attributes in the order first to last, i.e. dataProtection, protectionScheme and so on, see 7.3.4.

Table 4 – SECOM_ExchangeMetadataObject

SECOM_ServiceExchangeMetadataObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
dataProtection	1	Mandatory	Boolean	(S-100) Indicates if the data is encrypted. 0 indicates unencrypted data 1 indicates encrypted data
protectionScheme	1	Mandatory	CharacterString	(S-100) Specification or method used for data protection For example S-63, SECOM
digitalSignatureReference	1	Mandatory	CharacterString	(S-100) Specifies the algorithm used to compute digitalSignatureValue For example "dsa"
digitalSignatureValue	1	Mandatory	DigitalSignatureValueObject	Signed Public Key thumbprint plus the digital signature Table 5
compressionFlag	1	Mandatory	Boolean	(S-100) Indicates if data is compressed. 0 indicates uncompressed 1 indicates compressed

Table 5 – DigitalSignatureValueObject

DigitalSignatureValueObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
publicRootCertificate Thumbprint	0..1	Optional	CharacterString	Claimed Thumbprint for Signed Root Key (X.509 Certificate)
publicCertificate	1	Mandatory	CharacterString	Public Key (X.509 Certificate), PEM encoded Base64 string
digitalSignature	1	Mandatory	CharacterString	(S100) The digital signature in HEX format as one row, no trailing return/new line

5.6.4 Transfer of public key

5.6.4.1 General

In all cases when the payload includes claimed public keys used, for example, for signatures and classified data, the keys need to be converted in a manner to fit inside a JSON format and to secure interoperability. Certificates in PEM format contain line feeds and BEGIN/END header/footer. Line feed encoding characters differ among different operational systems (OS) where for example Unix uses only \n for line feed while Windows uses carriage return + line feed characters (\r\n). In order to secure interoperability between different OS machines, a minified public key is introduced to be used inside the different JSON object payloads. More examples on how to embed PEM encoded keys in payloads can be found in for example IHO S-100:2018, Part 15-7, Data encryption and licensing.

5.6.4.2 Producer

During the build of a payload JSON object and before adding the public key to the same object, the original key is minified into one single line string by:

1) Remove all line feed characters
2) Remove header -----BEGIN CERTIFICATE-----

3) Remove footer -----END CERTIFICATE-----

Figure 6 shows an example of the minimisation above using C# .NET. The specific *Environment.NewLine* (the corresponding routine in Java is the *System.lineSeparator()*) is used to ensure that the correct line feed character is used depending on what machine the system is installed on.

```
public string GetMinifiedPemFromCert(X509Certificate2 cert)
{
    var sb = new StringBuilder();
    using (var sw = new StringWriter(sb))
    {
        var writer = new PemWriter(sw);
        writer.WriteObject(DotNetUtilities.FromX509Certificate(cert));
    }
    //1.
    sb.Replace(Environment.NewLine, "");
    //2.
    sb.Replace("-----BEGIN CERTIFICATE-----", "");
    //3.
    sb.Replace("-----END CERTIFICATE-----", "");
    return sb.ToString();
}
```

IEC

Figure 6 – Example in C# of conversion from PEM format to minified public key

Figure 7 shows an example of a public key before and after the conversion.

IEC

Figure 7 – Example of a public key in PEM format converted to a single line string

5.6.4.3 Consumer

For the receiver, the conversion is the opposite. When consuming the transferred payload, the public key is converted back to its original format.

1) Add header -----BEGIN CERTIFICATE-----
2) Add OS specific line feed character
3) Split the public key string from the payload into an array of max 64 characters per row
4) For each element in the array
   a) add element as new row
   b) add OS specific line feed character

5) Add footer -----END CERTIFICATE-----

6) Add OS specific line feed character

Figure 8 shows an example of a conversion using C# .NET where the line feeds are added using .NET.

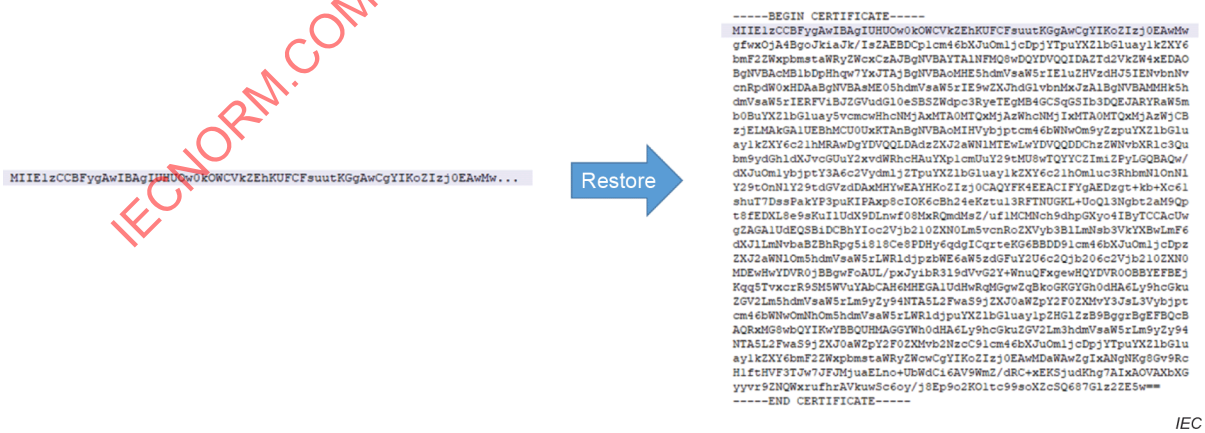
```
public X509Certificate GetCertFromPem(string publicKeyMinified)
{
    //1.
    string publicKey = "-----BEGIN CERTIFICATE-----";
    //2.
    publicKey += Environment.NewLine;
    //3.
    IEnumerable<string> keyArray = SplitIntoArray(publicKeyMinified, 64);
    //4.
    foreach(var row in keyArray)
    {
        //4a.
        publicKey += row;
        //4b.
        publicKey += Environment.NewLine;
    }
    //5.
    publicKey += "-----END CERTIFICATE-----";
    //6.
    publicKey += Environment.NewLine;

    return new X509Certificate(Encoding.UTF8.GetBytes(publicKey));
}
```

IEC

Figure 8 – Example in C# of conversion from minified public key to PEM format

Figure 9 shows an example of the result when restoring a public key PEM format from a minified string.



IEC

Figure 9 – Example of a minified public key string restored to the original PEM format

5.6.5 PaginationObject

The "Get" and "Get Summary" service interfaces may return more than one result. In order to limit the response size, these interfaces support pagination, for more details, see 5.5. Table 6 shows the attributes included in the PaginationObject.

Table 6 – PaginationObject

PaginationObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
totalItems	1	Mandatory	PositiveInteger	The total number of items that satisfy the query. This is always returned.
maxItemsPerPage	1	Mandatory	PositiveInteger	The maximum number of items the service shall return per page.

5.6.6 ContainerTypeEnum

Container types used in SECOM interface definitions. Table 7 shows the values of the enumeration.

Table 7 – ContainerTypeEnum

ContainerTypeEnum	
Value	**Definition**
0	S100_DataSet
1	S100_ExchangeSet
2	NONE

5.6.7 SECOM_DataProductType

Table 8 contains the supported product types used in SECOM information interfaces 5.7.5, 5.7.6, 5.7.8, 5.7.10 and 5.7.13.

Table 8 – SECOM_DataProductType

SECOM_DataProductType	
Name	**Description**
OTHER	Other data types not covered in this table
S57	S-57 Electronic Navigational Chart (ENC)
S101	S-101 Electronic Navigational Chart (ENC)
S102	S-102 Bathymetric Surface
S104	S-104 Water Level Information for Surface Navigation
S111	S-111 Surface Currents
S122	S-122 Marine Protected Areas (MPAs)
S123	S-123 Marine Radio Services
S124	S-124 Navigational Warnings
S125	S-125 Marine Navigational Services
S126	S-126 Marine Physical Environment
S127	S-127 Marine Traffic Management
S128	S-128 Catalogue of Nautical Products
S129	S-129 Under Keel Clearance Management (UKCM)
S131	S-131 Marine Harbour Infrastructure
S210	S-210 Inter-VTS Exchange Format

SECOM_DataProductType	
Name	**Description**
S211	S-211 Port Call Message Format
S212	S-212 VTS Digital Information Service
S401	S-401 Inland ENC
S402	S-402 Bathymetric Contour Overlay for Inland ENC
S411	S-411 Sea Ice Information
S412	S-412 Weather Overlay
S413	S-413 Marine Weather Conditions
S414	S-414 Marine Weather Observations
S421	S-421 Route Plan
RTZ	Route Plan
EPC	Electronic Port Clearance

5.6.8 SECOM_ResponseCodeEnum

The enumeration in Table 9 is added to the synchronous HTTP response object if a validation error occurs at the SECOM instance level for information to the service consumer. This is used in push interfaces such as Upload, UploadLink and Acknowledgement to provide a more specific error message.

Table 9 – SECOM_ResponseCodeEnum

SECOM_ResponseCodeEnum	
Value	**Definition**
0	Missing required data for the service
1	Failed signature verification
2	Invalid certificate
3	Schema validation error

5.6.9 AckRequest Enum

A flag to indicate that acknowledgement is expected to be returned when the data is delivered to the end user, and/or when the content of the data is processed (opened) by the end user. The value of this enumeration acts as a condition for whether an acknowledgment should be sent (value = 1,2,3) or not (value = 0). It is mandatory to implement the above functionality. Table 10 shows the enumeration values.

Table 10 – AckRequest Enum

AckRequestEnum	
Value	**Definition**
0 (default)	No ACK requested
1	Delivered ACK requested
2	Opened ACK requested
3	Delivered + Opened ACK requested

5.6.10 Common HTTP response codes

In Table 11, HTTP response codes common to all interfaces optional to implement are listed. For mandatory interfaces such as Ping and Capability, code 501 is not applicable. For interfaces using the SECOM_ResponseCodeEnum in Table 9, the corresponding column value is set to null.

Table 11 – Common HTTP codes

HTTP Code	Message
401	Unauthorized
405	Method not allowed
500	Internal server error
501	Not implemented

5.6.11 Well-known text – WKT

Query parameter "geometry" type used in interface Get, Get Summary and Subscription.

WKT is a human readable representation for spatial objects like points, lines, or enclosed areas on a map. SECOM uses a subset of the objects in WKT CRS 2 (see ISO 19162) with EPSG projection 4326 – WGS 84.

The objects used are listed in Table 12.

Table 12 – Supported WKT geometric objects

Supported WKT geometric objects		
Object	**Defintion**	**Example**
Point	A zero-dimensional geometry that represents a single location in coordinate space on a map. The (x,y) coordinate is represented in order (longitude, latitude) and expressed as real value, degree/degree decimal format.	POINT (30 10)
LineString	A curve with linear interpolation between points. Each consecutive pair of points defines a line segment.	LINESTRING (30 10, 10 30, 40 40)
Polygon	A planar surface, i.e. a two-dimensionsal geometry object, defined by 1 exterior boundary and 0 or more interior boundaries. Each interior boundary defines a hole in the polygon.	POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))
MultiPoint	An array of point objects	MULTIPOINT ((10 40), (40 30), (20 20), (30 10))
MultiLineString	An array of LineString objects	MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))
MultiPolygon	An array of polygon objects	MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))
GeometryCollection	A collection of 1 or more geometry objects	GEOMETRYCOLLECTION (POINT (40 10), LINESTRING (10 10, 20 20, 10 40), POLYGON ((40 40, 20 45, 45 30, 40 40)))

5.6.12 Universally Unique Identifier – UUID

A UUID (see RFC 4122) is a 128-bit (16 bytes) number used to identify some entity in an information system. When generated according to standard methods, UUIDs are unique with a probability of duplication close enough to zero. In SECOM, the UUID type is used as data references and as transaction identifiers created by the information owner in the OEM system.

The 128-bit string is represented as 32 hexadecimal digits displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters, see Figure 10.

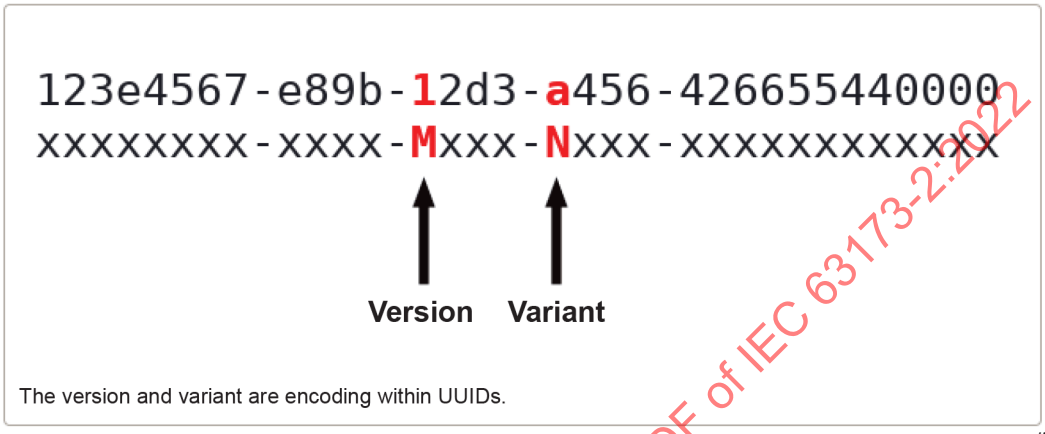


Figure 10 – UUID version and variant

The four-bit M (13th digit) and the first 1 to 3 most significant bit N (17th digit) fields code the format of the UUID itself. In the example in Figure 10, M = 1 i.e. version 1 and N = a i.e. variant 1 since the only possible hexadecimal values are 8 (1000), 9 (1001), a (1010) or b (1011). There are five versions of UUIDs and four variants where variant 1 (big-endian) is the most common. See Table 13 and Table 14.

Table 13 – UUID variants

UUID Variants			
Name	**Binary**	**Hex Digit**	**Description**
0	0xxx	0-7	Reserved, NCS backwards compatibility
1	10xx	8-b	RFC 4122/DCE 1.1 UUIDs
2	110x	c-d	Reserved, Microsoft Corporation backward compatibility
reserved	111x	e-f	Reserved for future definition

Table 14 – UUID versions

UUID Versions		
Name	**Name descr**	**Description**
0	nil	A special case where all bits are set to zero
1	date-time and MAC address	Generated from a time and a node ID
2	date-time and MAC address, security version	Generated from an identifier (usually a group or user id), time, and a node id. RFC 4122 reserves version 2 for "DCE (Distributed Computing Environment) security" UUIDs but without providing any details. For this reason, many UUID implementations omit version-2.
3	name-based	Generated by MD5 (128 bits) hashing of a namespace identifier and name.
4	random	Generated using a random or pseudo-random number.
5	name-based	Generated by SHA-1 (160 bits) hashing of a namespace identifier and name.

The version and variant used when generating the UUID is up to the information owner and hence is outside of SECOM as long as the resulting formatted string complies with the RFC 4122 standard format and can be decoded by the information consumer.

5.6.13 UN/LOCODE

The "United Nations Code for Trade and Transport Locations" is commonly more known as "UN/LOCODE". It is managed and maintained by the UNECE. All details are available from the link below

https://unece.org/trade/uncefact/unlocode

Code of location according to UN/Locode codelist restricted to regular expression [a-zA-Z]{2}[a-zA-Z2-9]{3}. For example SEGOT, USAA7.

5.7 Service interface definitions

5.7.1 General

Table 15 gives an overview of the service interfaces that constitutes the SECOM information service interface. Service interfaces Capability and Ping shall be supported while the remaining interfaces are optional to implement.

Table 15 – Service interfaces overview

Service Interface	Exchange Pattern	Definition	Use case, see Annex B
Upload	ONE_WAY	Interface to send (push) information to consumer	B.2.1, B.2.2, B.2.3, B.2.4, B.2.7
UploadLink	ONE_WAY	Interface to send (push) a link to information for a consumer to get	B.2.6, B.2.8
Acknowledgement	ONE_WAY	Interface to send acknowledgement on uploaded information.	B.2.1
Get	REQUEST_RESPONSE	Interface to ask for (pull) information from provider	B.2.2
Get Summary	REQUEST_RESPONSE	Interface to ask for (pull) an information list from provider	B.2.2
Get By Link	ONE-WAY	Interface to retrieve information from uploaded link	B.2.6, B.2.8

Service Interface	Exchange Pattern	Definition	Use case, see Annex B
Access	REQUEST_CALLBACK	Interface to ask for access to information	B.2.1, B.2.4
Access Notification	ONE_WAY	Interface for notification from access request	B.2.1, B.2.4
Subscription	PUBLISH_SUBSCRIBE	Interface to create subscription of information	B.2.1, B.2.4, B.2.7
Remove Subscription	ONE_WAY	Interface to remove subscription	B.2.1, B.2.4, B.2.7
Subscription Notification	ONE_WAY	Interface for notification from subscription events	B.2.1, B.2.4, B.2.7
Capability	REQUEST_RESPONSE	Interface to ask for the interface capabilities. Mandatory to implement.	B.2.1
Ping	REQUEST_RESPONSE	Interface to check status on the service instance. Mandatory to implement.	B.2.4
Encryption Key	ONE_WAY, REQUEST_CALLBACK	Interface to securely send symmetric key for data encryption	B.2.3 B.2.6
PublicKey	ONE_WAY, REQUEST_RESPONSE	Interface to request (pull) and send (push) a public certificate	B.2.3

5.7.2 Service interface – Upload

5.7.2.1 Specification

The purpose with this interface is to upload (push) information that shall not be larger than a maximum size of 350 kB (Base64 encoded) to an information consumer. An information consumer shall implement this interface in order to receive information while an information provider may implement it.

When uploading (sending) information, acknowledgement can be requested. See interface Acknowledgement for the details of sending acknowledgement.

Uploading (sending) of information can either be within an active subscription of information or uploaded (sent) once as a single message as indicated by the attribute fromSubscription.

The data can be either XML, compressed XML or encrypted binary, typically compressed, as described in the exchange metadata object. The data can contain either one message, such as an S-421 Route Plan (IEC 63173-1) dataset, or a set of messages (files) compressed in a ZIP container and described by an ExchangeSet as described in IHO S-100. The data is encoded as one row Base64 before transferred.

Attached to the data there is a set of metadata describing the protection of the data. The metadata refers to the content in the data attribute, independent of its type. For exchange of an IHO S-100 DataSet, this means that the dataset might be in XML (compressionFlag = false) or in ZIP (compressionFlag = true), or optionally compressed and encrypted (compressionFlag = true and dataProtection = true) and the data is signed (hash calculated and signed). For exchange of an IHO S-100 ExchangeSet (exchange catalogue), the same procedure is used. The IHO S-100 ExchangeCatalogue contains a set of files, the folder is compressed, and optionally encrypted and the data is signed (hash calculated and signed).

Figure 11 shows the interface in UML.

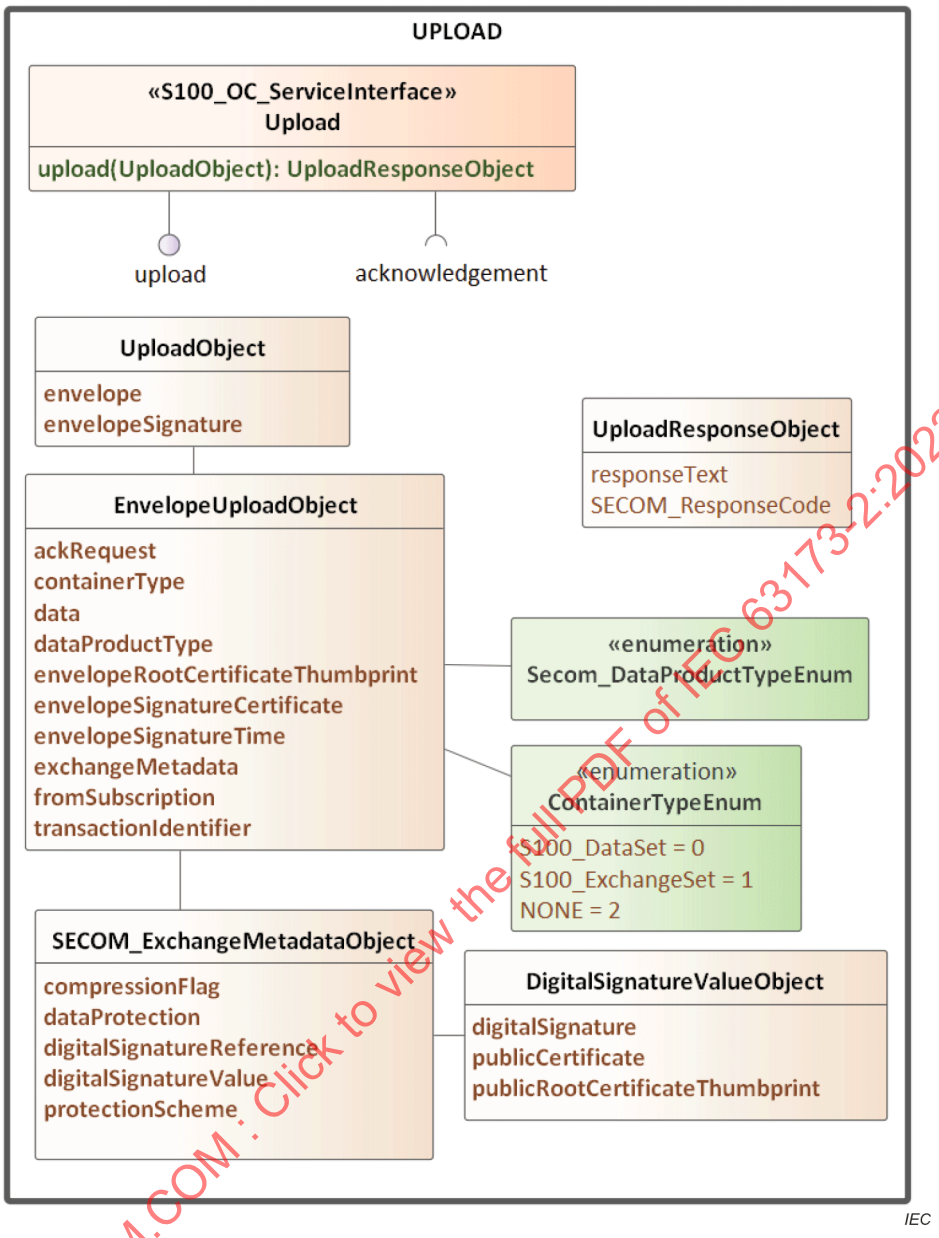


Figure 11 – Upload interface UML diagram

5.7.2.2 Data exchange model

Table 16 and Table 17 describe the data exchanged in the interface.

Table 16 – Information input for Upload interface

UploadObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
envelope	1	Mandatory	EnvelopeUploadObject	The complete EnvelopeObject being uploaded to receiver including data and message properties
envelopeSignature	1	Mandatory	CharacterString	The signature of the EnvelopeObject in HEX format without whitespace or linebreaks
EnvelopeUploadObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
data	1	Mandatory	Base64	The payload XML dataProductType, base64 encoded (e.g. inside a S100_ExchangeSet, S100_DataSet), ZIP or binary. The data can be open, protected and/or compressed.
containerType	1	Mandatory	ContainerTypeEnum	Container type of message in the data object Table 7
dataProductType	1	Mandatory	SECOM_DataProductType.Name	The name column of the SECOM_DataProductType in Table 8
exchangeMetadata	1	Mandatory	SECOM_ExchangeMetadataObject	The exchange metadata contains information regarding protection scheme, compression, signature and claimed identity according to Table 4
fromSubscription	1	Optional	Boolean	Flag to indicate whether the data has been uploaded within an active subscription or not.
ackRequest	1	Mandatory	AckRequestEnum	Flag to indicate that acknowledgement is expected to be returned when the data is delivered to the end user, and/or when the content of the data is processed (opened) by the end user.
transactionIdentifier	1	Mandatory	UUID	Transaction identifier to be used e.g. in acknowledgement
envelopeSignatureCertificate	1	Mandatory	CharacterString	The public certificate of the sender. Used to verify the envelopeObject signature
envelopeRootCertificateThumbprint	1	Optional	CharacterString	Claimed Thumbprint for Signed Root Key (X.509 Certificate)
envelopeSignatureTime	1	Optional	DateTime	Time stamp when the envelope is signed

Table 17 – Information output for Upload interface

UploadResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
SECOM_ResponseCode	0..1	Mandatory	SECOM_ResponseCodeEnum	Additional error code for internal errors Table 9
message	1	Optional	CharacterString	Success or error response message

5.7.2.3 REST design

5.7.2.3.1 General

The service interface and its data exchange model is defined with REST technology.

See Annex A for formal and detailed definition of the service interface as OpenAPI format.

5.7.2.3.2 Operation POST/object

The interface shall be used for uploading (pushing) data to a consumer. The operation expects one single data object and its metadata. The details are described in Table 18. For detailed description of data exchange model, see 5.7.2.2.

Table 18 – REST implementation of Upload

REST Operation			
POST URL/v1/object {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
UploadObject	application/json	1	The data (payload) package with its metadata
Return (out)	**REST Encoding**	**Mult**	**Definition**
UploadResponseObject	application/json	1	Confirmation of upload or error message, including description of incorrect or missing values in payload for the specific service.

5.7.2.3.3 Service response

The REST service instance shall respond with HTTP codes and message according to Table 19.

Table 19 describes internal error codes and messages generated by the service implementer. Error codes and messages generated by the deployment environment shall follow RFC 7231 HTTP1/1 and is out of the scope for SECOM.

See also Table 11 for codes common to all interfaces. For this interface with an extra attribute, SECOM_ResponseCodeEnum, this attribute is set to null for the common response codes.

For tests related to these error codes, see 10.14.

Table 19 – HTTP Response codes and message in response object

HTTP Code	SECOM_ResponseCodeEnum	Message examples
200	null	Message successfully uploaded
400	0	Missing required data for the service
400	1	Failed signature verification
400	2	Invalid certificate
400	3	Schema validation error
403	null	Not authorized to upload
413	null	Request entity too large

5.7.2.4 Dynamic behaviour

Figure 12 describes the dynamic use of the Upload interface and the relation to the Acknowledgement interface.

The Upload interface is used when data is pushed (sent) to another actor. The size of the data is technically constrained by the chosen REST technology to messages less than 350 kB. In the sequence diagram, the left side actor, SECOM service consumer, produces the data to be pushed to the right side actor. The SECOM service consumer invokes the Upload interface (consumes the Upload service interface) and sends data, data signature and complementary metadata to the receiving actor. The complementary metadata contains attributes for the acknowledgement request (optional), transaction identifier, subscription flag and the signature for the complete transferred data package.

The receiver checks the size of data received (normally by the web server) and will respond with HTTP response 413 if the size is too large for the current configuration. A service consumer shall always be prepared to handle such error response. The receiver of data checks the integrity as part of the verify signature step of the data package and responds accordingly with either HTTP response 200 if OK, or with 400 and SECOM_ResponseCodeEnum = 1 if not OK, see Table 19.

The data is then forwarded to a user application, either directly or through vendor specific communication channels. If delivery acknowledgement has been requested, "delivery ACK" is sent asynchronously to the acknowledgement endpoint appointed in the metadata when the data is correctly forwarded to the end-user.

The end-user application verifies access rights and verifies the signatures as received, both as integrity check and authentication of data received.

If opened acknowledgement has been requested, "opened ACK" shall be sent asynchronously to the acknowledgement endpoint appointed in the metadata when the data is correctly opened or processed by the end-user. SECOM does not specify a time limit between receiving a message and sending an "opened ACK".

Further information on message exchange patterns in given in Annex C.

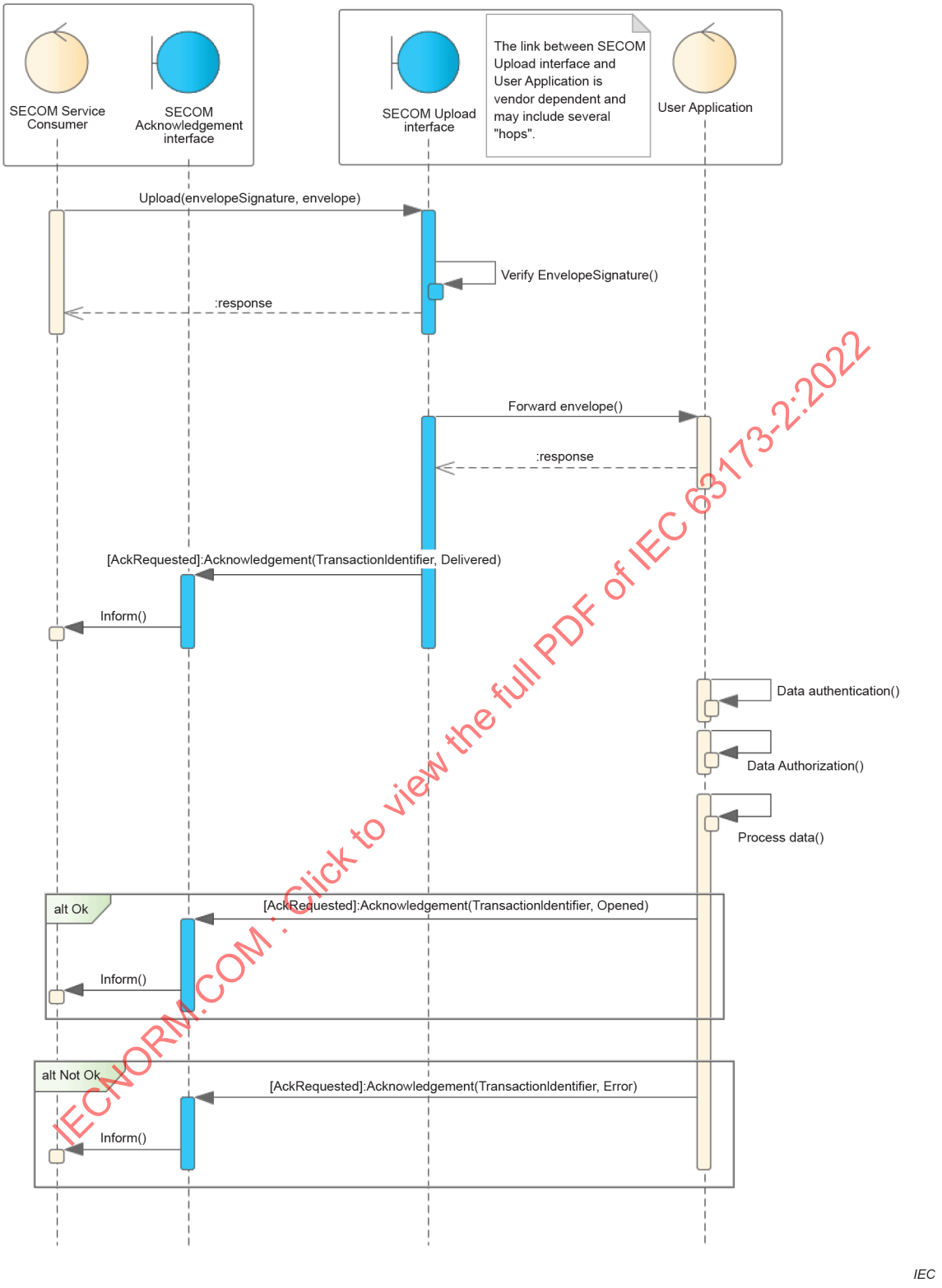


Figure 12 – Sequence diagram for upload signed unclassified data with acknowledgement

NOTE The acknowledgements can be used for supervision and diagnostic purposes.

5.7.3 Service interface – Upload Link

5.7.3.1 Specification

The purpose with this interface is to upload (push) a link to information to a consumer. Hence, a consumer implements this interface in order to receive a link to the information that can be retrieved.

This interface is used when large amounts of data are to be exchanged. The provider of information uploads a link to a consumer, and the consumer then uses the Get by Link interface to pull the data from the provider.

The process of uploading data in a two-step procedure (Upload Link followed by Get By Link) adds a complexity to the signature verification. The transactionIdentifier relates to the actual data but it is still the underlying data that is signed similar to the signing in the Upload interface. Hence, the verification of the data signature can only be done after the data has been downloaded using Get By Link. Figure 13 describes the interface in UML.

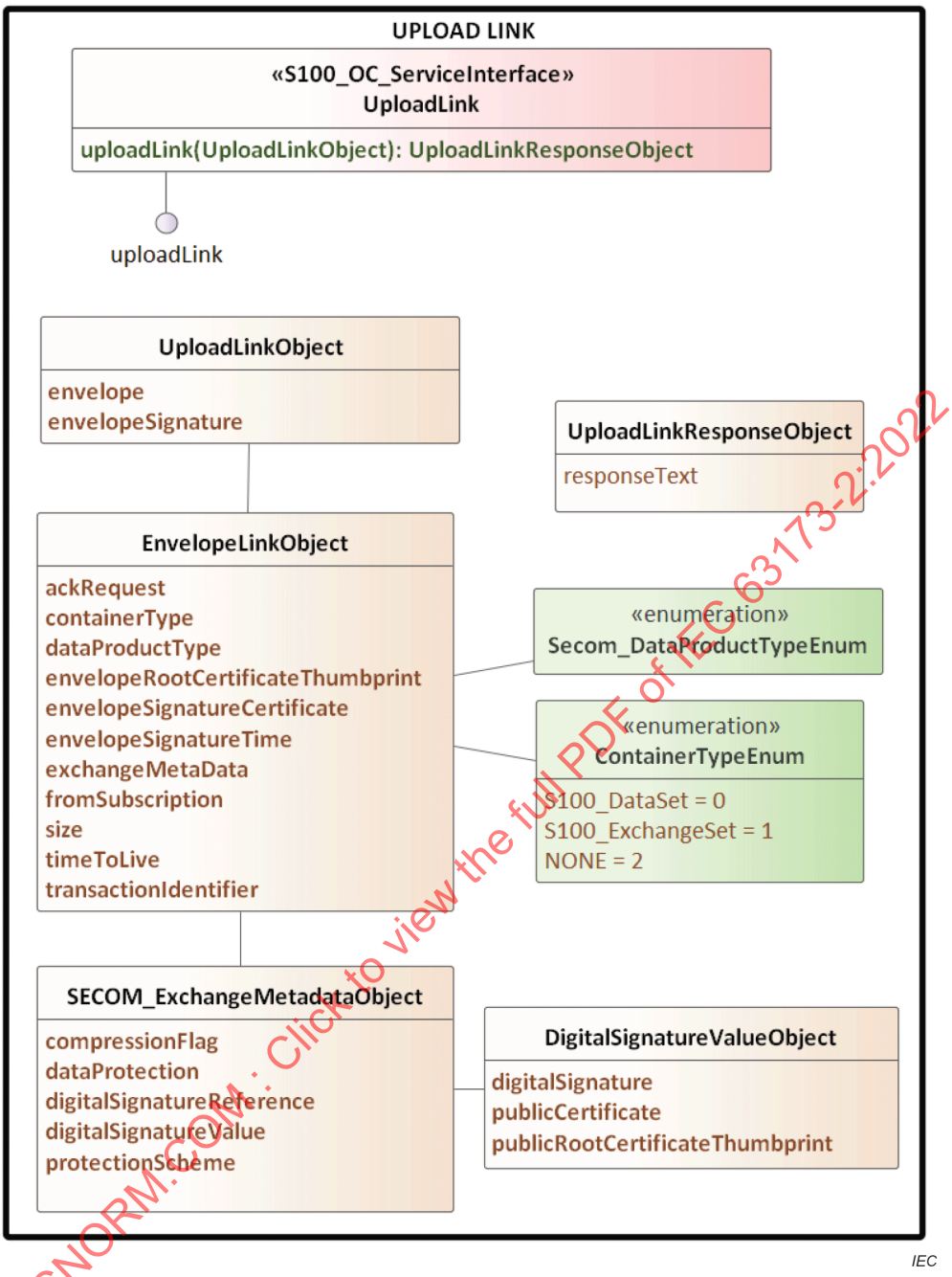


Figure 13 – Update link interface UML diagram

5.7.3.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 20 and Table 21.

Table 20 – Information input for Upload Link interface

UploadLinkObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
envelope	1	Mandatory	EnvelopeLinkObject	The complete EnvelopeLinkObject being uploaded to receiver including message properties
envelopeSignature	1	Mandatory	CharacterString	The signature of the EnvelopeLinkObject in HEX format without whitespace or linebreaks
EnvelopeLinkObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
containerType	1	Mandatory	ContainerTypeEnum	Container type of message in the data object Table 7
dataProductType	1	Mandatory	SECOM_DataProductType.Name	The name column of SECOM_DataProductType in Table 8
exchangeMetadata	1	Mandatory	SECOM_ExchangeMetadataObject	Service exchange metadata with signature, ID etc.
fromSubscription	1	Optional	Boolean	Flag to indicate whether the data has been uploaded within an active subscription or not.
ackRequest	1	Mandatory	AckRequestEnum	Flag to indicate that acknowledgement is expected to be returned when the data is delivered to the end user, and/or when the content of the data is processed (opened) by the end user.
transactionIdentifier	1	Mandatory	UUID	Transaction identifier to be used in acknowledgement and when retrieving the message using Get By Link
envelopeSignatureCertificate	1	Mandatory	CharacterString	The public certificate of the sender, used to verify the envelopeLinkObject signature
envelopeRootCertificateThumbprint	1	Optional	CharacterString	Claimed Thumbprint for Signed Root Key (X.509 Certificate)
size	1	Optional	PositiveInteger	Size of the data in kBytes to be downloaded.
timeToLive	1	Mandatory	DateTime	DateTime when data will be deleted on server. The data need to be fetched before this time.
envelopeSignatureTime	1	Optional	DateTime	Time stamp when the envelope is signed

Table 21 – Information output for Upload Link interface

UploadLinkResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
SECOM_ResponseCode	0..1	Mandatory	SECOM_ResponseCodeEnum	Additional error code for internal errors, see Table 9
message	1	Optional	CharacterString	Success or error response message

5.7.3.3 REST Design

5.7.3.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.3.3.2 Operation – POST /object/link

The REST operation POST /object/link. The interface shall be used for uploading (pushing) a link to data to a consumer. The details are described in Table 22.

Table 22 – REST implementation of Upload Link

REST Operation			
POST URL/v1/object/link {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
UploadLinkObject	application/json	1	The message link with its metadata
Return (out)	**REST Encoding**	**Mult**	**Definition**
UploadLinkResponseObject	application/json	1	Confirmation of upload or error message

5.7.3.3.3 Service response

The service instance shall respond with HTTP codes and message according to Table 23.

See also Table 11 for codes common to all interfaces. For this interface with an extra attribute, SECOM_ResponseCodeEnum, this attribute is set to null for the common response codes.

For tests related to these error codes, see 10.14.

Table 23 – HTTP Response codes and message in response object

HTTP Code	SECOM_ResponseCodeEnum	Message
200	null	Link successfully uploaded
400	0	Missing required data for the service
400	1	Failed signature verification
400	2	Invalid certificate
403	null	Not authorized to upload link

5.7.3.4 Dynamic behavior

Figure 14 describes the dynamic behavior of the service interface.

The Upload Link service interface is used when data is larger than technically possible to POST in a REST service call such as Upload. The left side actor in the sequence diagram shall send a larger data package to the user (right side in Figure 14).

The data is prepared and compressed to enable reference to a single file. The compressed file is given a unique transaction identifier.

The transaction identifier, together with metadata containing time data if available, size of data and signature, is uploaded (sent) to the user (information consumer).

The receiver of the link (transaction identifier) verifies the envelope signature and checks the metadata, for example size and time to live, and decides if data shall be retrieved immediately or wait until a cheaper connection is established. When ready, the receiving actor invokes the service Get By Link to retrieve the data and forwards it to the end-user application. After received in the end-user application, the data signature can be verified.

The same procedure described in Upload service interface regarding acknowledgement is implemented here as well. The receiving actor shall always send the acknowledgement upon request. SECOM does not specify a time limit between receiving a message and sending an "opened ACK".

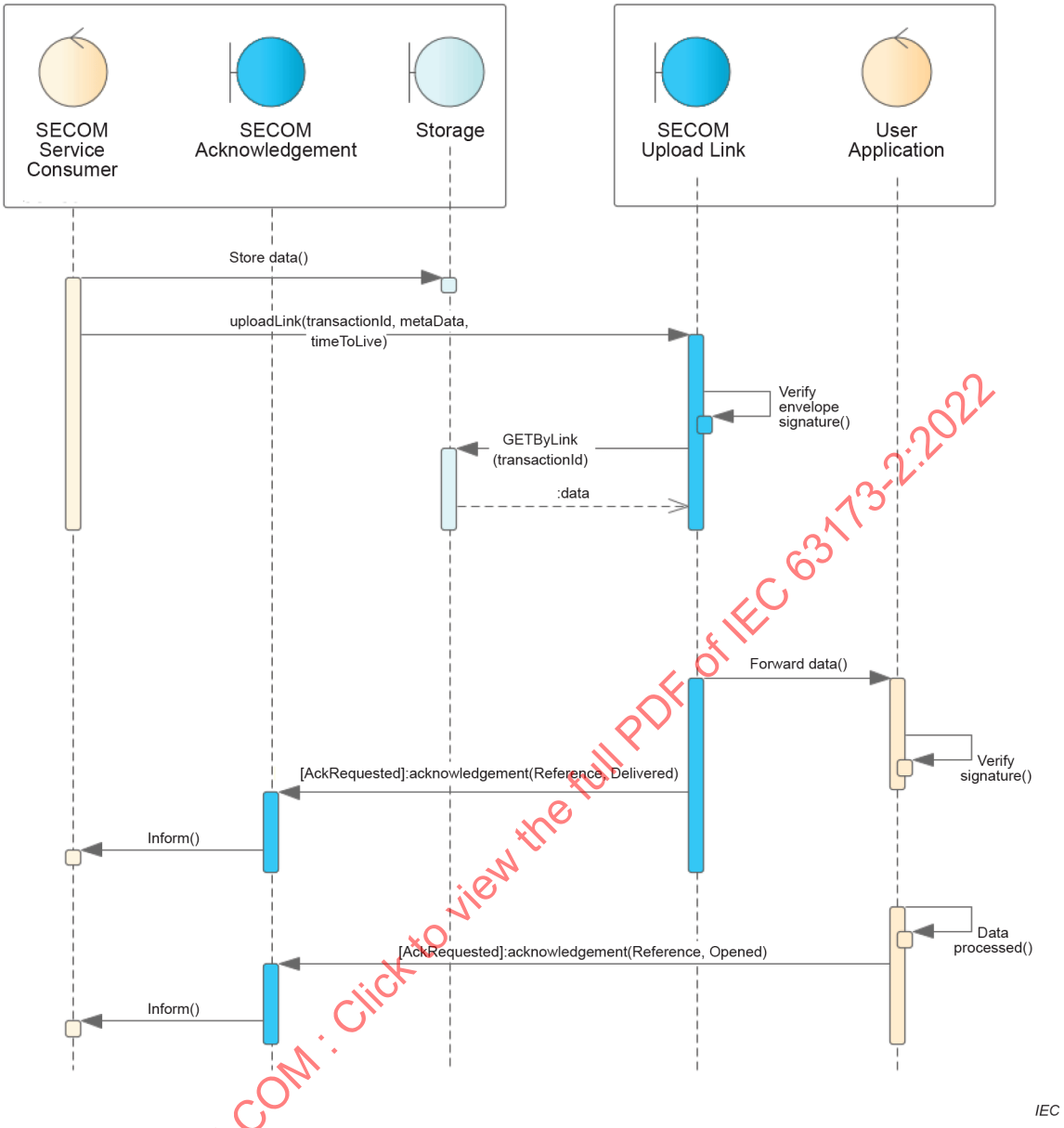


Figure 14 – Sequence diagram for Upload link to large data

5.7.4 Service interface – Acknowledgement

5.7.4.1 Specification

During upload of information, an acknowledgement can be requested which is expected to be asynchronously received when the uploaded message has been delivered to the end system (technical acknowledgement), and an acknowledgement when the message has been opened and/or processed by the end user (operational acknowledgement). The acknowledgement contains a reference to the object delivered and has no time limit.

Figure 15 describes the Acknowledgement interface in UML diagram.

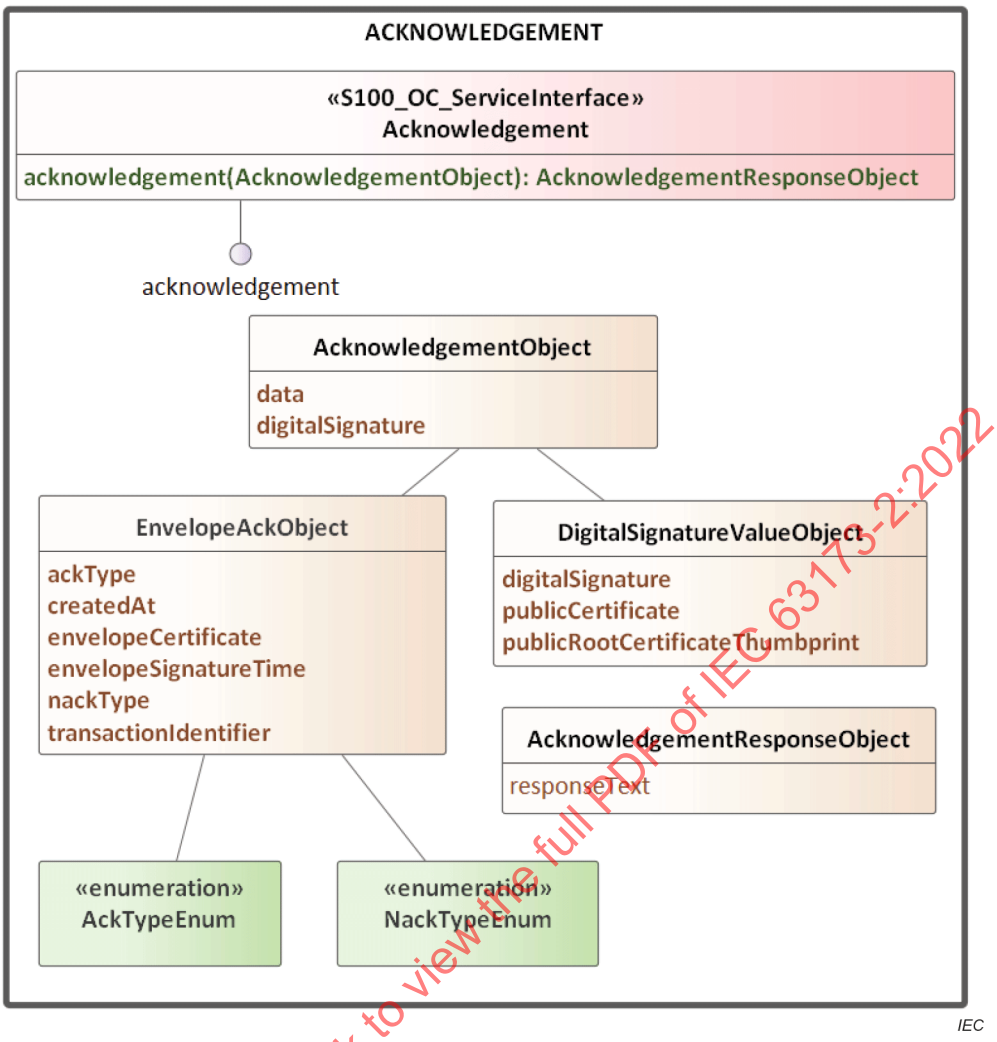


Figure 15 – Acknowledgement interface UML diagram

5.7.4.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 24 to Table 27.

Table 24 – Information input for Acknowledgement interface

AcknowledgementObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
envelope	1	Mandatory	EnvelopeAckObject	The acknowledgment payload to be signed
digitalSignature	1	Optional	CharacterString	The signature ot the EnvelopeAckObject in HEX format without whitespace or linebreaks
EnvelopeAckObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
createdAt	1	Mandatory	DateTime	Creation time for the acknowledgement
envelopeCertificate	1	Mandatory	CharacterString	The public certificate of the sender used to verify the envelopeObject signature
envelopeRootCertificateThumbprint	1	Optional	CharacterString	Claimed Thumbprint for Signed Root Key (X.509 Certificate)
transactionIdentifier	1	Mandatory	UUID	Reference identifier given in upload
ackType	1	Mandatory	AckTypeEnum	Type of acknowledgement according to Table 27
nackType	0..1	Optional	NackTypeEnum	Information details if error according to Table 25
envelopeSignatureTime	1	Optional	DateTime	Time stamp when the envelope is signed

Table 25 – Enumerations for not acknowledged

NackTypeEnum	
Value	**Definition**
0	XML Schema validation error
1	Unknown data type or version
2	Failed data signature verification
3	Failed decryption
4	Failed decompression

Table 26 – Information output for Acknowledgement interface

AcknowledgementResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
message	1	Optional	String	Success or error response message
SECOM_ResponseCode	0..1	Mandatory	SECOM_ResponseCodeEnum	Additional error code for internal errors see Table 9

Table 27 – Enumerations for Acknowledgement interface

AckTypeEnum	
Value	**Definition**
1	Delivered ACK Technical acknowledgement such as delivered to end system
2	Opened ACK Operational acknowledgement such as when opened/read by end user.
3	Error

5.7.4.3 REST Design

5.7.4.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.4.3.2 Operation – POST /acknowledgement

The details for operation acknowledgement in the interface are described in Table 28.

Table 28 – REST implementation of acknowledgement

REST Operation			
POST URL/v1/acknowledgement {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
AcknowledgementObject	application/json	1	Object with reference to information and time when delivered
Return (out)	**REST Encoding**	**Mult**	**Definition**
AcknowledgementResponseObject	application/json	1	Confirmation or error message according to Table 26

5.7.4.3.3 Service response

The service instance shall respond with HTTP codes and message according to Table 29.

See also Table 11 for codes common to all interfaces. For this interface with an extra attribute, SECOM_ResponseCodeEnum, this attribute is set to null for the common response codes.

For tests related to these error codes, see 10.12.

Table 29 – HTTP Response codes and response message

HTTP Code	SECOM_ResponseCodeEnum	Message
200	null	Successfully received ACK for <reference>
400	null	Bad request
400	0	Missing required data for the service
400	1	Failed signature verification
400	2	Invalid certificate
403	null	Not authorized to upload ACK

5.7.4.4 Dynamic behavior

Figure 16 describes the dynamic behavior of the Acknowledgement interface. See also the dynamic behavior of Upload interface and Upload Link interface for the operational context of the Acknowledgement interface.

The acknowledgement interface is used in conjunction with Upload, Upload Link and Get. The sender (uploader) of data can request acknowledgement from the receiver enabling the sender to follow (trace) the sent data. This is especially important when uploading (sending) data to a ship that can be offline at the time of sending, but receives the data next time it is online. Also, the sender (responder) of data after a request to Get can request acknowledgement from the receiver.

NOTE When the "opened ACK" is requested in conjunction with Upload Link, the acknowledgement is sent after the data has been received using the Get By Link request.

There are two different acknowledgements defined in SECOM. The first is the "deliver ACK" that is sent by the receiver when data has been accepted and forwarded to the end-user. The second is the "opened ACK" that is sent by the end user when received data is correctly opened and processed.

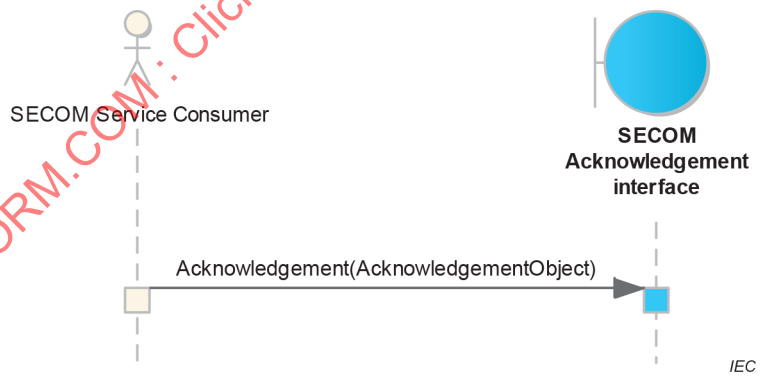


Figure 16 – Sequence diagram for Acknowledgement interface

5.7.5 Service interface – Get

5.7.5.1 Specification

The Get interface is used for pulling information from a service provider. The owner of the information (provider) is responsible for the authorization procedure before returning information.

The consumer can ask for information by its reference, geometry, time or arbitrary query for status on the information product for example. If no filtering parameters are given, all authorized information shall be sent. The information owner decides what information the consumer is authorized to, based on the identity in the TLS client certificate, i.e. the identity the service instance belongs to.

This interface may return many information objects and supports pagination as described in 5.5. If no pagination parameters are given, the result is all messages returned for the first page. Figure 17 shows the Get interface in UML.

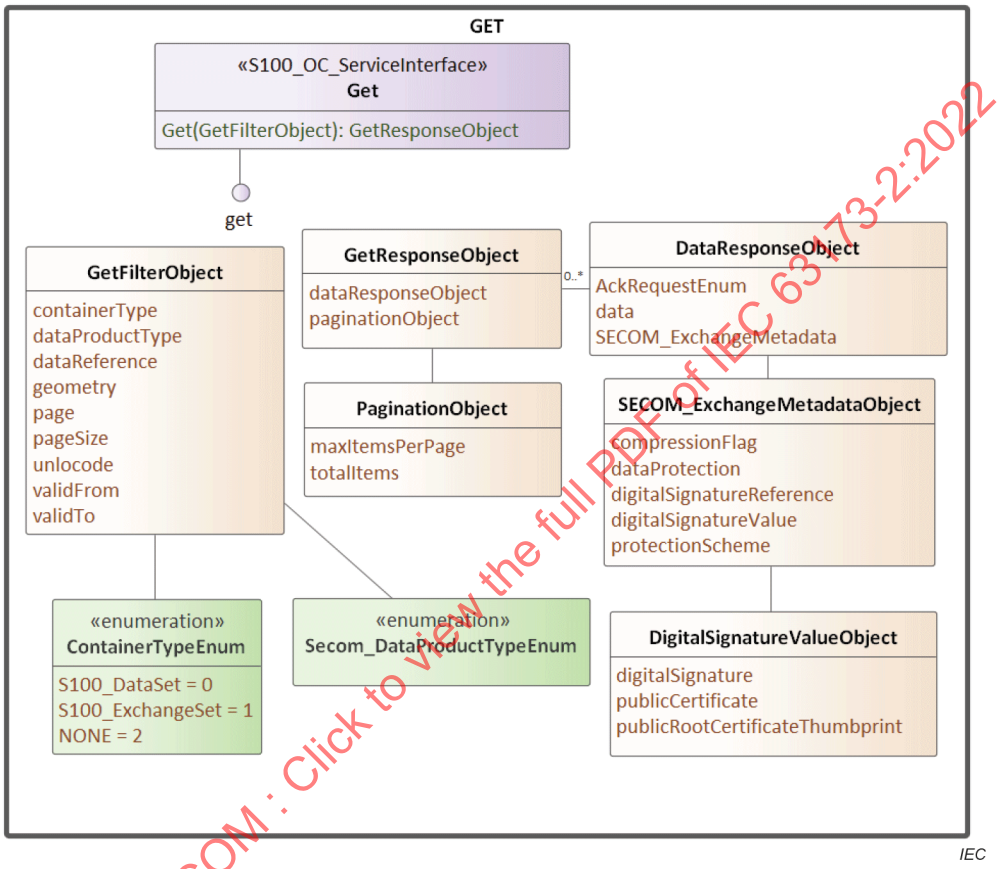


Figure 17 – Get interface UML diagram

5.7.5.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 30 and Table 31.

Table 30 – Information input for Get interface

GetFilterObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
dataReference	0..1	Mandatory	UUID	Reference to information object, e.g. from Get Summary
containerType	0..1	Mandatory	ContainerTypeEnum	Container type requested, see Table 7
dataProductType	0..1	Mandatory	SECOM_DataProductType.Name	The name column of SECOM_DataProductType in Table 8
productVersion	0..1	Optional	CharacterString	S-100 based Product type version, e.g. 1.0.0
geometry	0..1	Optional	CharacterString	Geometry condition for geo-located information objects, see Table 12
unlocode	0..1	Optional	CharacterString	Code of defined object see 5.6.13
validFrom	0..1	Optional	DateTime	Valid from time
validTo	0..1	Optional	DateTime	Valid until time
page	0..1	Mandatory	PositiveInteger	Requested pagination page
pageSize	0..1	Mandatory	PositiveInteger	Requested pagination page size

Table 31 – Information output for Get interface

GetResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
dataResponseObject	0..*	Mandatory	DataResponseObject	List containing the data
pagination	1	Mandatory	PaginationObject	Pagination information
DataResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
data	1	Mandatory	Base64	Base64 encoded byte array. The data according to Product Specification embedded in JSON. The data may be encrypted and signed.
exchangeMetadata	1	Mandatory	SECOM_ExchangeMetadataObject	Service exchange metadata with signature, ID etc.
ackRequest	1	Mandatory	AckRequestEnum	Flag to indicate that acknowledgement is expected to be returned when the data is delivered to the end user, and/or when the content of the data is processed (opened) by the end user.

5.7.5.3 REST Design

5.7.5.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.5.3.2 Operation – GET /object

This operation receives a Get request for information. If authorized, the data is sent back in the response. It is up to the service provider to apply relevant authorization procedure and access control to information.

Table 32 describes the logical parameters provided in the interface.

Table 32 – REST implementation of Get

REST Operation			
GET URL/v1/object?parameters: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
dataReference	String	0..1	Information retrieved by using reference given in Get Summary or Upload Link (UUID)
containerType	DataTypeEnum	0..1	Container type requested, see Table 7
dataProductType	SECOM_DataProductType.Name	0..1	Product type requested, e.g. S-122
productVersion	String	0..1	Product type version requested, e.g 1.0.0
geometry	String	0..1	Geometry as search parameter, see Table 12
unlocode	String	0..1	Code of defined object see 5.6.13
validFrom	DateTime	0..1	Time related to validity period start for information object
validTo	DateTime	0..1	Time related to validity period end for information object
page	PositiveInteger	0..1	Requested pagination page
pageSize	PositiveInteger	0..1	Requested pagination page size
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
No body defined	n/a	n/a	n/a
Return (out)	**REST Encoding**	**Mult**	**Definition**
GetResponseObject	application/json	1	Set of messages package with metadata or an error message

5.7.5.3.3 Service response

Table 33 describes the service instance response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.15.

Table 33 – HTTP Response code and message of Get

HTTP Code	Message
200	GetMessageResponseObject
400	Bad request
403	"Not authorized to requested information"
404	Information not found

5.7.5.4 Dynamic behavior

Figure 18 and Figure 19 describe the dynamic behavior of the Get interface.

The Get interface is used to pull data from an actor's published information. The service request contains filtering parameters that allows the information owner to search and prepare data to be returned. Once the authentication of the requester has been verified, the preparation of data includes an authorization check against internal access control list and packaging of the data with data signatures. If access is accepted, the requested data is returned, if not, an error messages is given.

If the information owner decides to publish protected data, the consumer needs a possibility to request the encryption key if not already available, to be able to decrypt the protected data. The consumer requests the encryption key by data reference ID and its public certificate used for symmetric key derivation. The information owner encrypts the random key used when encrypting the data with a symmetric key derived from the consumer's public key and its own private key. The information owner sends the protected random key via the consumer's SECOM encryption key interface.

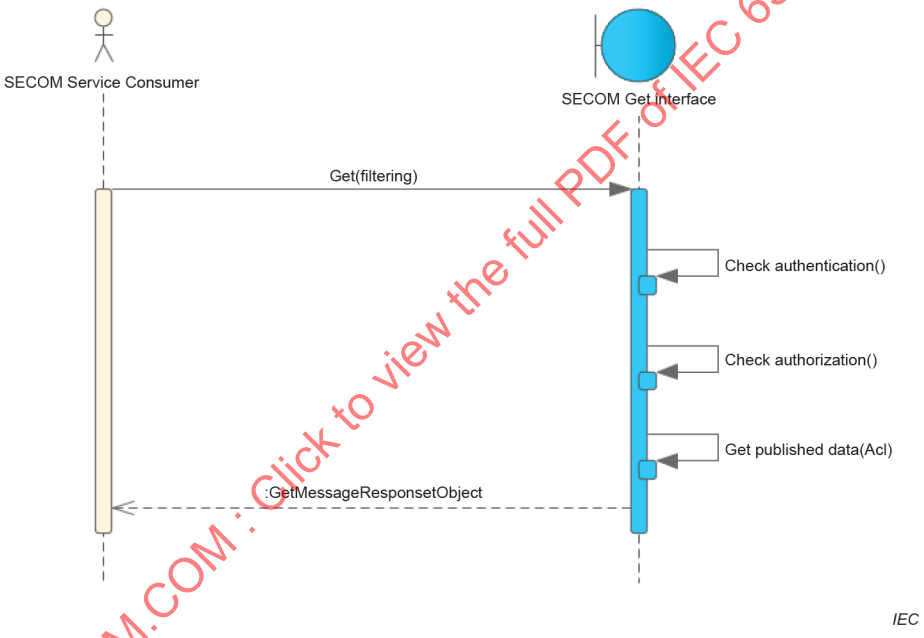


Figure 18 – Sequence diagram for Get interface

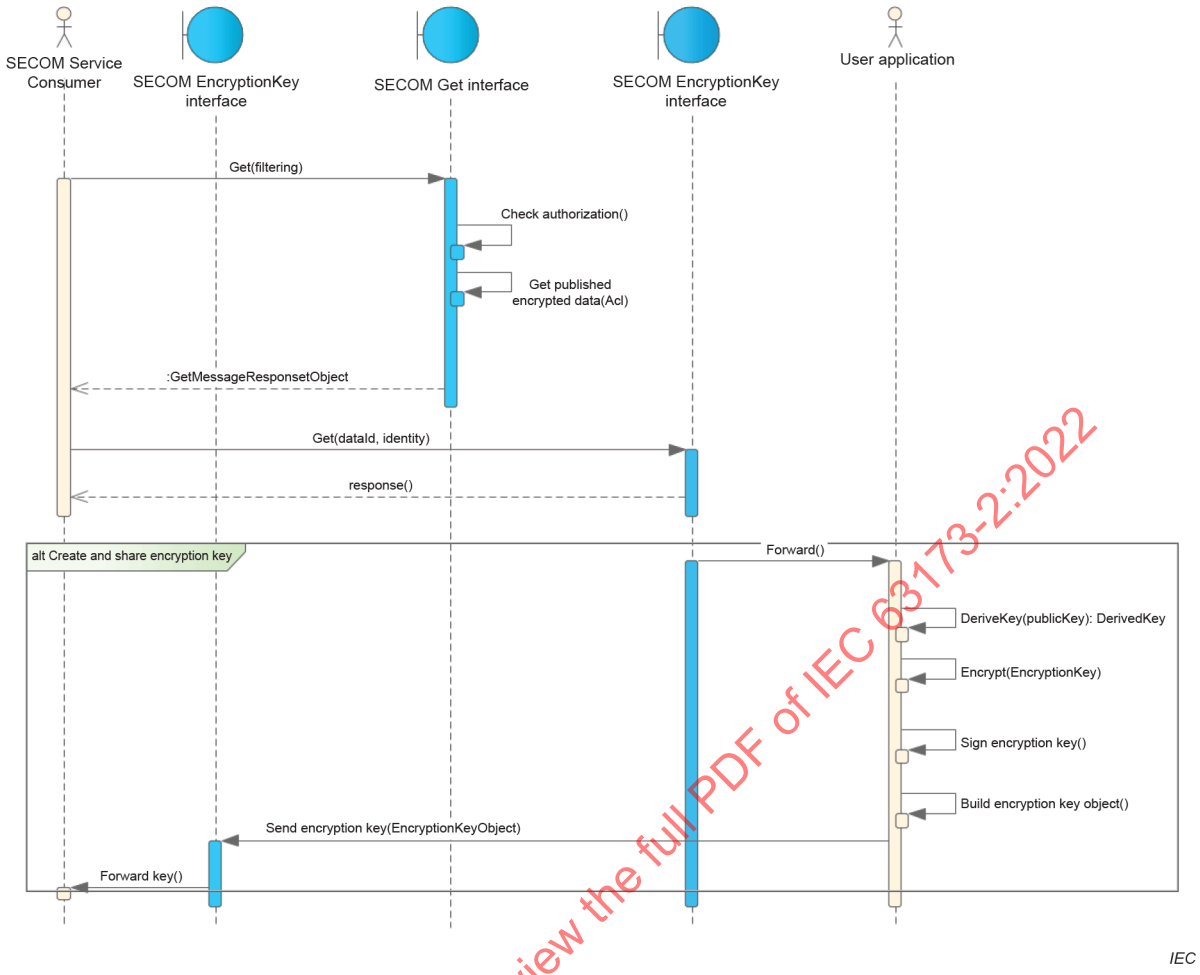


Figure 19 – Sequence diagram for Get interface and classified data

5.7.6 Service interface – Get Summary

5.7.6.1 Specification

A list of information shall be returned from this interface. The summary contains identity, status, size and a short description of each information object. The actual information object shall be retrieved using the Get interface. The consumer can ask for information by geometry, location and time. If no filtering parameters are given, available summary information shall be sent.

NOTE Information known by the information provider can be sensitive. The information provider judges what is available and thus to be included in the response to Get Summary.

This interface may return many information objects and supports pagination described in 5.5.

Figure 20 describes the Get Summary interface in UML diagram.

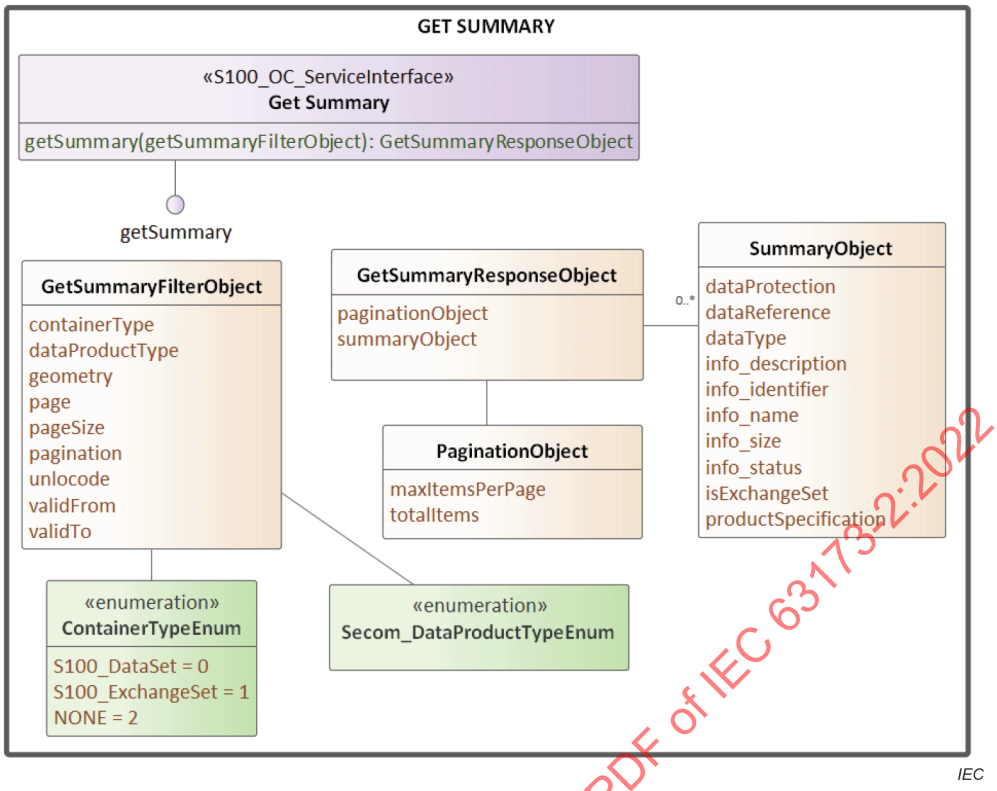


Figure 20 – Get Summary interface UML diagram

5.7.6.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 34 and Table 35.

Table 34 – Information input for Get Summary interface

GetSummaryFilterObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
containerType	0..1	Mandatory	ContainerTypeEnum	Container type requested, Table 7
dataProductType	0..1	Mandatory	SECOM_DataProductType.Name	The name column of SECOM_DataProductType in Table 8
productVersion	0..1	Optional	CharacterString	S-100 based Product type version, e.g. 1.0.0
geometry	0..1	Optional	CharacterString	Geometry condition for geo-located information objects, see Table 12
unlocode	0..1	Optional	CharacterString	Code of defined object, see 5.6.13
validFrom	0..1	Optional	DateTime	Valid from time
validTo	0..1	Optional	DateTime	Valid until time
page	0..1	Mandatory	PositiveInteger	Requested pagination page
pageSize	0..1	Mandatory	PositiveInteger	Requested pagination page size

Table 35 – Information output for Get Summary interface

GetSummaryResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
summaryObject	0..*	Mandatory	SummaryObject	Description of the information object
pagination	1	Mandatory	PaginationObject	Pagination information
SummaryObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
dataReference	1	Mandatory	UUID	Reference to data
dataProtection	1	Optional	Boolean	Flag indicating if data is encrypted or not
dataCompression	1	Optional	Boolean	Flag indicating if data is compressed or not
containerType	1	Optional	ContainerTypeEnum	Container type, see Table 7
dataProductType	1	Optional	SECOM_DataProductType.Name	The name column of SECOM_DataProductType in Table 8
info_identifier	0..1	Optional	CharacterString	Identifier of the information object, e.g. <S100XC:identifier>, gml:id
info_name	0..1	Optional	CharacterString	Name of the information object, e.g. <S100XC:exchangeCatalogueName>, <S100:datasetFileIdentifier>
info_status	0..1	Optional	CharacterString	Status of the information object, e.g. active, inactive
info_description	0..1	Optional	CharacterString	Description of the information object, e.g. <S100XC:description>, <S100:datasetTitle>
info_lastModifiedDate	0..1	Optional	DateTime	Date for last modified, e.g. <S100XC:date>, <S100:datasetReferenceDate>
info_productVersion	0..1	Optional	CharacterString	S-100 based Product type version, e.g. 1.0.0
info_size	0..1	Optional	PositiveInteger	Size of the data in kB expressed as unsigned long or unsigned int64 with max value = 2^64

5.7.6.3 REST Design

5.7.6.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.6.3.2 Operation – GET /object/summary

This operation receives a Get Summary request. The response contains a summary of available information. The actual information can be retrieved through Subscription request or Get request.

Table 36 describes the REST service interface.

Table 36 – REST implementation of Get Summary

REST Operation			
GET URL/v1/object/summary?parameters: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Defintion**
geometry	String	0..1	Geometry as search parameter, see Table 12
unlocode	String	0..1	Code of defined object, see 5.6.13
validFrom	DateTime	0..1	Time related to validity period start for information object
validTo	DateTime	0..1	Time related to validity period end for information object
page	PositiveInteger	0..1	Page number requested
pageSize	PositiveInteger	0..1	Page size requested
REST Body (in)	**REST Encoding**	**Mult**	**Defintion**
No body defined	n/a	n/a	n/a
Return (out)	**REST Encoding**	**Mult**	**Defintion**
GetSummaryResponseObject	application/json	1	List of information objects available, identified by identity, status and short description or an error message

5.7.6.3.3 Service response

Table 37 describes the service instance response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.15.

Table 37 – HTTP Response codes and messages of Get Summary

HTTP Code	Message
200	GetSummaryResponseObject
400	Bad request
403	Not authorized to requested information
404	Information not found

5.7.6.4 Dynamic behavior

Figure 21 describe the dynamic behavior of the Get Summary interface.

The Get Summary interface is used to pull metadata from an actor. The received metadata response might be used for selecting which actual data to be retrieved by a new request using the Get interface in 5.7.5. The service request contains filtering parameters that allows the information owner to search and prepare metadata to be returned. Once the authentication of the requester has been verified, the preparation of metadata includes internal judgement of information availability and packaging the metadata. If metadata is available, the requested metadata is returned, if not, an error messages is returned.

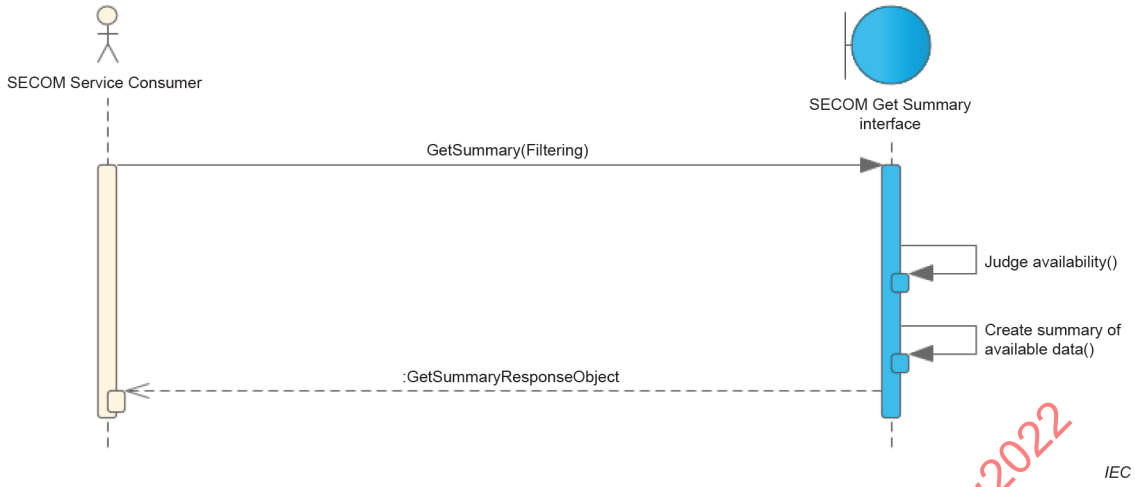


Figure 21 – Sequence diagram for Get Summary interface

5.7.7 Service interface – Get By Link

5.7.7.1 Specification

The Get By Link interface is used for pulling information from a data storage handled by the information owner. The link to the data storage can be exchanged with Upload Link interface. The owner of the information (provider) is responsible for relevant authentication and authorization procedure before returning information.

Figure 22 describes the Get By Link interface in UML.

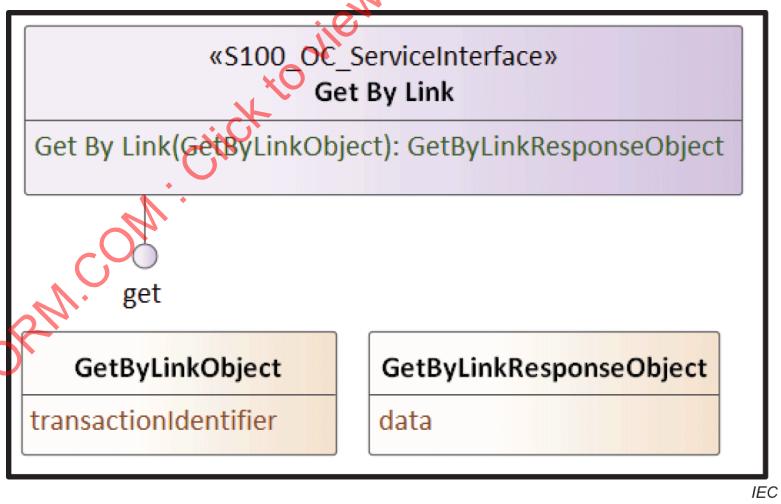


Figure 22 – Get By Link interface in UML

5.7.7.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 38 and Table 39.

Table 38 – Information input for Get By Link interface

GetByLinkObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
transactionIdentifier	1	Mandatory	UUID	Identifier uploaded in Upload Link

Table 39 – Information output for Get By Link interface

GetByLinkResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
data	1	Mandatory	Base64	Data as Base64 encoded binary format

5.7.7.3 REST Design

5.7.7.3.1 General

The service interface is defined with REST technology.

5.7.7.3.2 Operation – GET /object/link

This operation receives a Get By Link request for information. If authorized, the data is sent back in the response. It is up to the service provider to apply relevant authorization procedure and access control to information.

Table 40 describes the logical parameters provided in the interface.

Table 40 – REST implementation of Get By Link

REST Operation			
GET URL/v1/object/link?parameter: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
transactionIdentifier	String	1	Reference to data transaction from Upload Link, UUID
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
No body defined	n/a	n/a	n/a
Return (out)	**REST Encoding**	**Mult**	**Definition**
data	application/octet-stream	1	Data binary

5.7.7.3.3 Service response

Table 41 describes the service instance response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.11.

Table 41 – HTTP Response code and message of Get By Link

HTTP code	SECOM_ResponseCodeEnum	Message
200	null	Success
400	null	Bad request
400	2	Invalid certificate
403	null	Not authorized to requested information
404	null	Information with <transactionIdentifier > not found

5.7.7.4 Dynamic behavior

Figure 23 describes the dynamic behavior of the Get By Link interface.

The Get By Link interface is used to retrieve the data object provided by the information owner after invoking the receiver's Upload Link interface. The receiver of the link (transactionIdentifier) checks the metadata, for example size and time to live, and decides if data shall be retrieved immediately or wait until a cheaper connection is established. When ready, the receiving actor invokes the operation Get By Link to retrieve the data. Finally, the data signature verification is achieved using the received data matched to the digitalSignatureValue object from the previously received Upload Link request object.

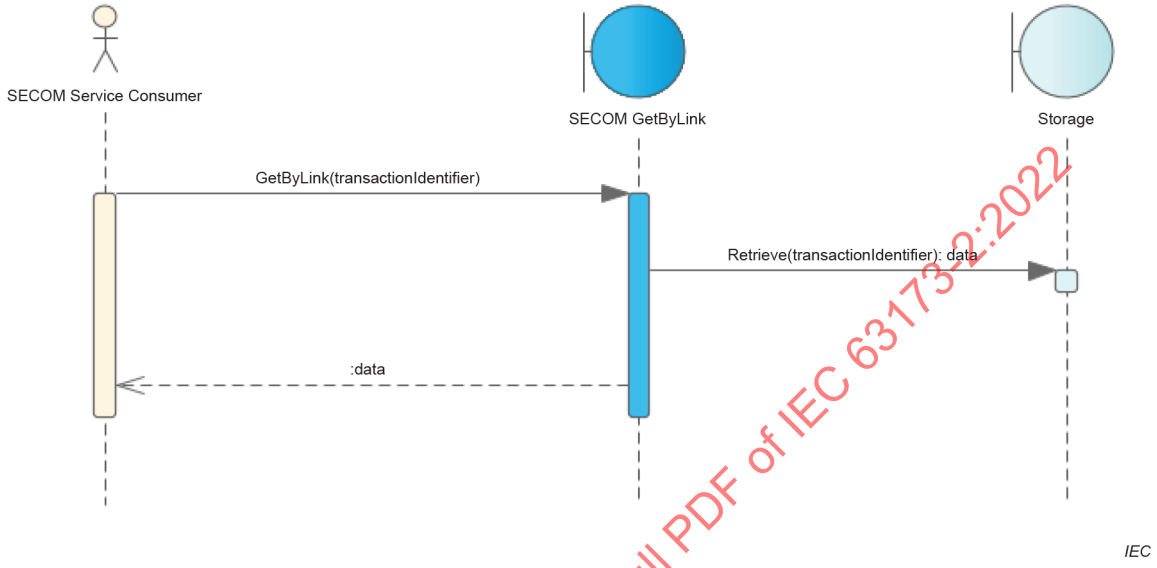


Figure 23 – Sequence diagram for Get By Link interface

5.7.8 Service interface – Access

5.7.8.1 Specification

Access to information can be requested through the Access interface. The result is sent asynchronously through the Access Notification interface.

Figure 24 describes the Access interface in UML diagram.

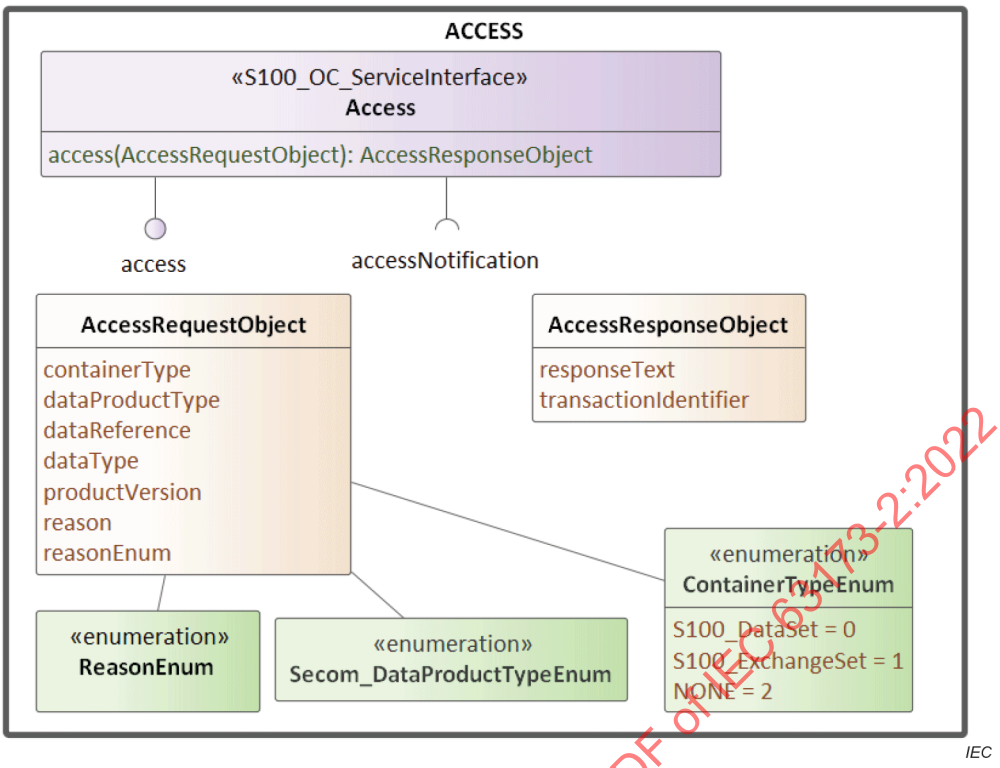


Figure 24 – Access interface UML diagram

5.7.8.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 42, Table 43 and Table 44.

Table 42 – Information input for Access interface

AccessRequestObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
reason	1	Mandatory	CharacterString	Human readable reason for requesting access
reasonEnum	1	Mandatory	ReasonEnum	Machine readable reason for requesting access
containerType	0..1	Mandatory	ContainerTypeEnum	Container type requested, see Table 7
dataProductType	0..1	Mandatory	SECOM_DataProduct Type.Name	The name column of SECOM_DataProductType in Table 8
dataReference	0..1	Mandatory	UUID	Reference to the data requested access to
productVersion	0..1	Optional	CharacterString	S-100 based Product type version, e.g. 1.0.0

Table 43 – Information output for Access interface

AccessResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
message	1	Optional	CharacterString	Success or error response message

Table 44 – Enumerations for Access interface

ReasonEnum	
Value	**Definition**
0	Required by authority
1	Required by service provider

5.7.8.3 REST Design

5.7.8.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.8.3.2 Operation – POST /access

This operation receives a request for access by a consumer. The result is sent asynchronously.

Table 45 describes the logical parameters in the interface.

Table 45 – Parameter binding for the operation

REST Operation			
POST URL/v1/access {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
AccessRequestObject	application/json	1	Description of reason for requesting access to information
Return (out)	**REST Encoding**	**Mult**	**Definition**
AccessResponsObject	application/json	1	Confirmation or error message. Result from request is sent asynchronous through Access Notification interface.

5.7.8.3.3 Service response

Table 46 describes the service instance response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.9.

Table 46 – HTTP Response codes

HTTP Code	Message
200	Success
400	Bad request
403	Not authorized to requested information

5.7.8.4 Dynamic behavior

Figure 25 describes the dynamic behavior of Request Access and Access Notification interfaces.

The Access interface is used to request access to one or several information object(s). The service consumer gets an asynchronous response with the decision through the consumer's Access Notification interface.

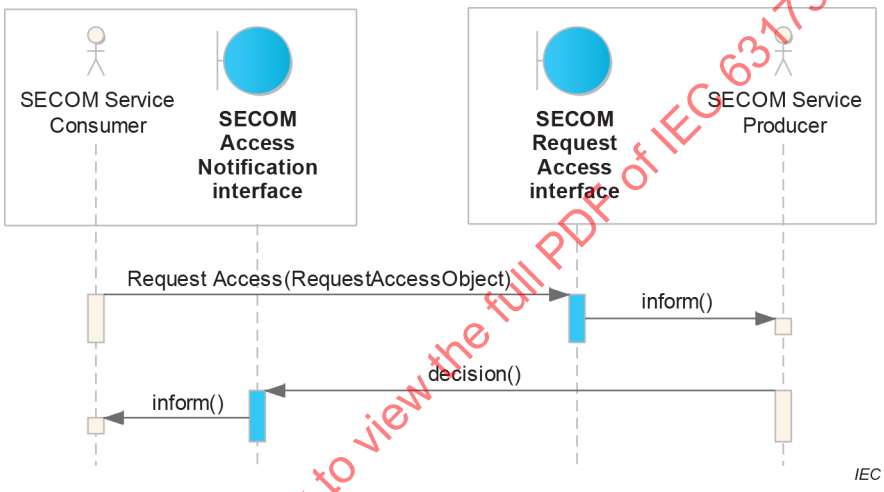


Figure 25 – Sequence diagram for Request Access and Access Notification interface

5.7.9 Service interface – Access Notification

5.7.9.1 Specification

Result from Access request shall be sent asynchronously through the Access Notification interface.

Figure 26 describes the Access Notification in UML diagram.

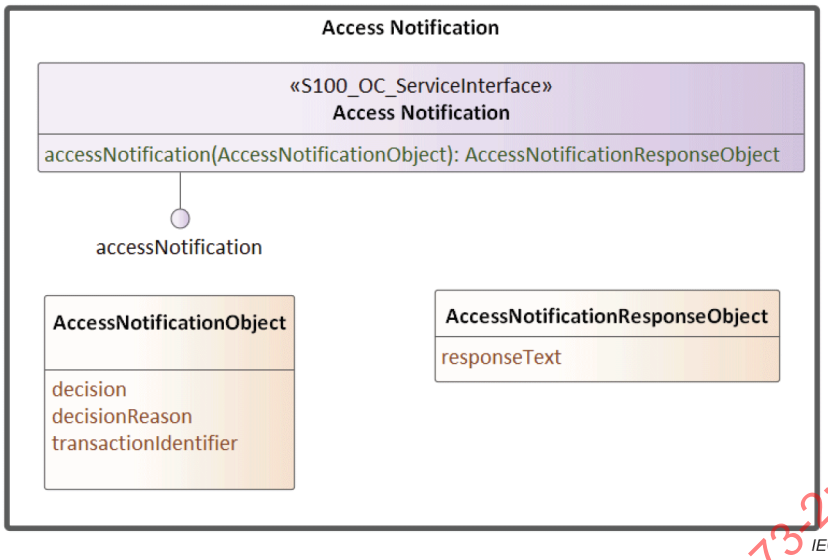


Figure 26 – Access Notification interface UML diagram

5.7.9.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 47 and Table 48.

Table 47 – Information input for Access Notification interface

AccessNotificationObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
decision	1	Mandatory	Boolean	Access request decision, yes or no
decisionReason	1	Optional	CharacterString	Human readable reason for decision
transactionIdentifier	1	Mandatory	UUID	Transaction identifier corresponding to Request Access

Table 48 – Information output for Access Notification interface

AccessNotificationResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
message	1	Optional	CharacterString	Success response message

5.7.9.3 REST Design

5.7.9.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.9.3.2 Operation – POST /access/notification

This operation receives result from access request.

Table 49 describes the logical parameters in the interface.

Table 49 – Parameter binding for the operation

REST Operation			
POST URL/v1/access/notification {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
AccessNotificationObject	application/json	1	Result from the request for access; True or False and the reason.
Return (out)	**REST Encoding**	**Mult**	**Definition**
AccessNotificationResponseObject	application/json	1	Confirmation or error message

5.7.9.3.3 Service response

Table 50 describes the service response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.9.

Table 50 – HTTP response codes

HTTP Code	Message
200	Success
400	Bad request

5.7.9.4 Dynamic behavior

This interface is an asynchronous response to service interface Request Access. See Figure 25.

5.7.10 Service interface – Subscription

5.7.10.1 Specification

The purpose of the interface is to request subscription on information, either specific information according to parameters, or the information accessible upon decision by the information provider. Each subscription request reflects one parameter query set.

Figure 27 describes the Subscribe interface in UML diagram.

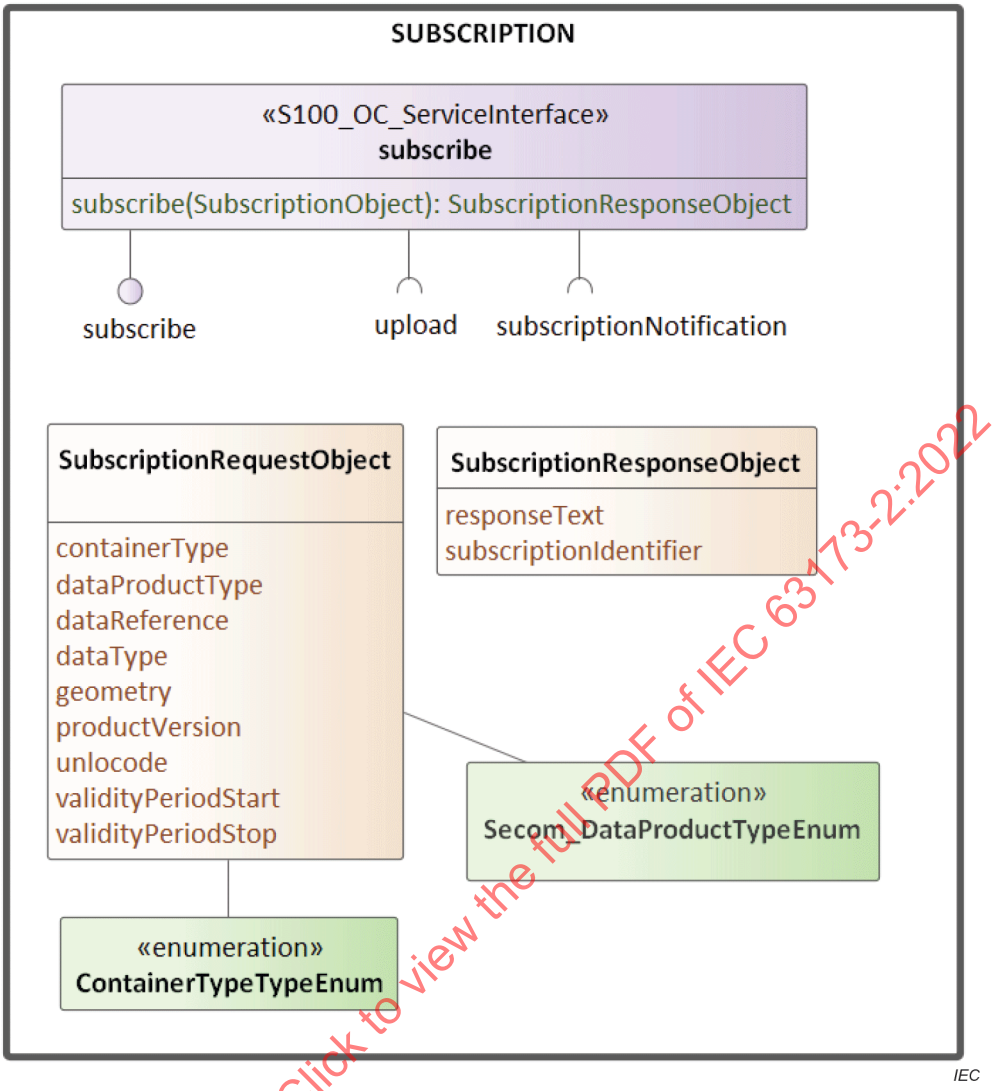


Figure 27 – Subscribe interface UML diagram

5.7.10.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 51 and Table 52.

Table 51 – Information input for Subscription interface

SubscriptionRequestObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
containerType	0..1	Mandatory	ContainerTypeEnum	Container type requested, see Table 7
dataProductType	0..1	Mandatory	SECOM_DataProductType.Name	The name column of SECOM_DataProductType in Table 8
dataReference	0..1	Mandatory	UUID	Reference to data
productVersion	0..1	Optional	CharacterString	S-100 based Product type version, e.g. 1.0.0
geometry	0..1	Optional	CharacterString	Geometry as criteria, see Table 12
unlocode	0..1	Optional	CharacterString	Code of defined object, see 5.6.13
subscriptionPeriodStart	0..1	Optional	DateTime	Start time of subscription
subscriptionPeriodEnd	0..1	Optional	DateTime	End time of subscription

Table 52 – Information output for Subscription interface

SubscriptionResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
message	1	Optional	CharacterString	Response message
subscriptionIdentifier	0..1	Optional	CharacterString	Subscription identifier, if successful (UUID)

5.7.10.3 REST Design

5.7.10.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.10.3.2 Operation – POST /subscription

This operation receives a request for subscription. If the requester is authorized, a subscription is created and the latest message is sent.

Table 53 describes the logical parameters in the interface.

Table 53 – REST implementation of Subscription

REST Operation			
POST URL/v1/subscription {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
SubscriptionObject	application/json	1	Subscription request including filtering parameters
Return (out)	**REST Encoding**	**Mult**	**Definition**
SubscriptionResponseObject	application/json	1	Confirmation or error message Subscription identifier in return, if ok.

5.7.10.3.3 Service response

Table 54 describes the service response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.16.

Table 54 – HTTP response codes and messages of Subscription

HTTP Code	Message
200	Subscription successfully created
400	Bad request
403	Not authorized to requested information

5.7.10.4 Dynamic behavior

Figure 28, Figure 29 and Figure 30 describe the dynamic behavior of the service interface.

The Subscription interface is used to request subscription to information objects. Subscription could both be requested by the consumer (Figure 30) and nominated by the producer (Figure 29). Also removing a subscription can be initiated by the consumer as well as the producer. The requested or nominated subscription creation and removal is notified using the Subscription Notification interface seen in Figure 29 and Figure 30.

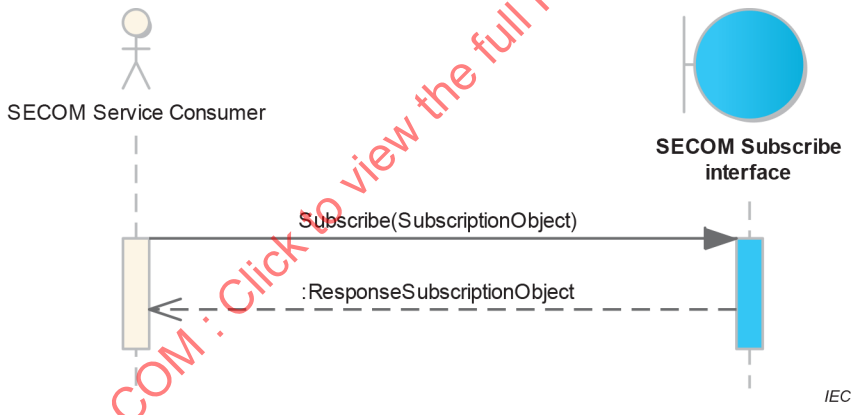


Figure 28 – Sequence diagram for Subscribe interface

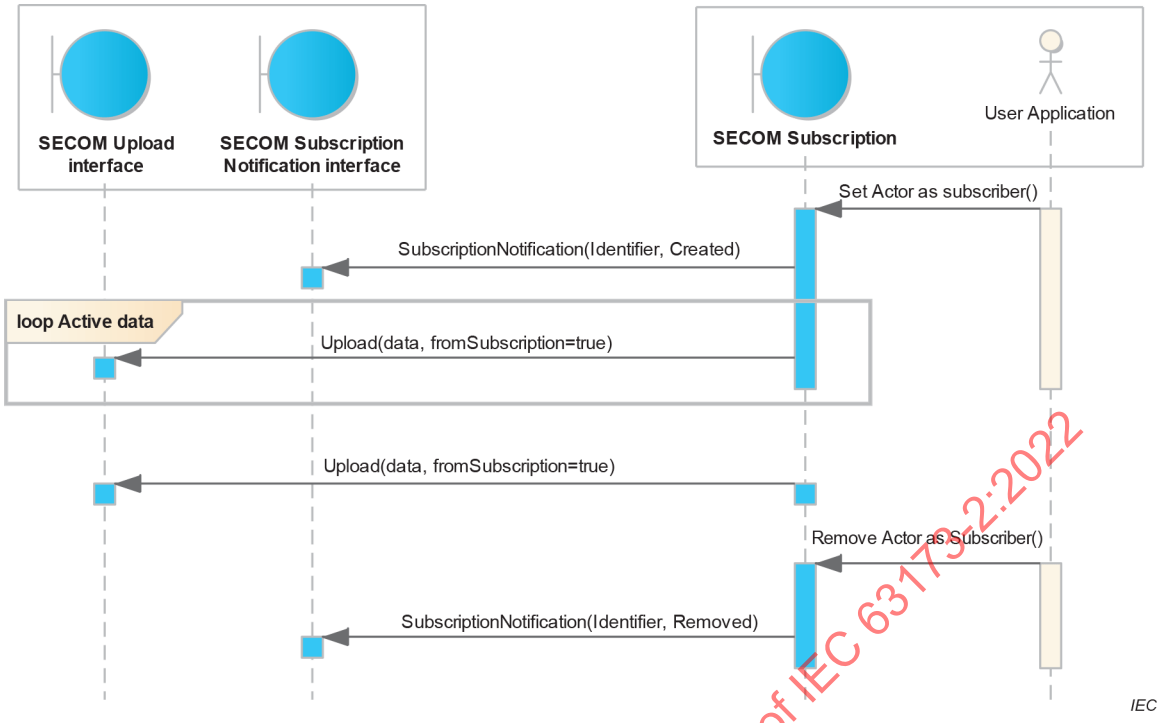


Figure 29 – Operational sequence diagram for Subscription interfaces

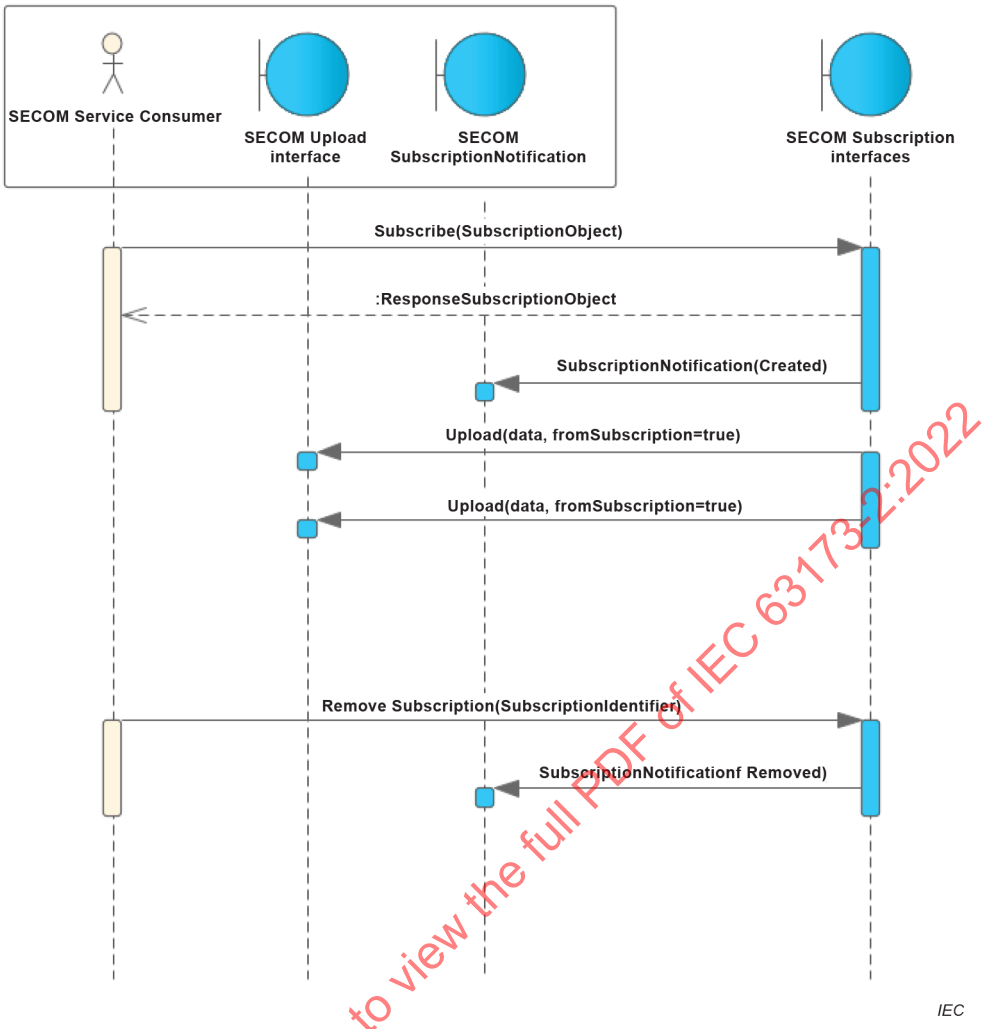


Figure 30 – Sequence diagram for Subscription interfaces with external subscription request

5.7.11 Service interface – Remove Subscription

5.7.11.1 Specification

Subscription(s) can be removed either internally by information owner, or externally by the consumer. This interface shall be used by the consumer to request removal of subscription.

Figure 31 describes the Remove Subscription interface in UML.

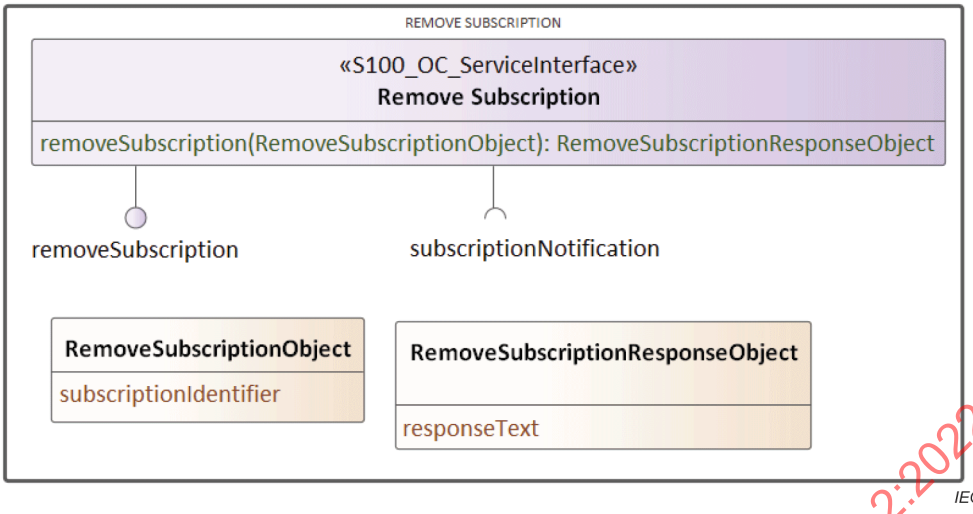


Figure 31 – Remove Subscription interface UML diagram

5.7.11.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 55 and Table 56.

Table 55 – Information input for Remove Subscription interface

RemoveSubscriptionObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
subscriptionIdentifier	1	Mandatory	UUID	Subscription identifier to the subscription to be removed. Received in subscription request or subscription notification.

Table 56 – Information output for Remove Subscription interface

RemoveSubscriptionResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
message	1	Optional	CharacterString	Response message

5.7.11.3 REST Design

5.7.11.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.11.3.2 Operation – DELETE /subscription

This operation receives a request to remove subscription.

Table 57 describes the logical parameters in the interface.

Table 57 – REST implementation of Remove Subscription

REST Operation			
DELETE URL/v1/subscription {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Description**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Description**
RemoveSubscriptionObject	application/json	1	Specific identity of the information object to remove subscription for. If no ID entity provided, all subscriptions for the caller is removed
Return (out)	**REST Encoding**	**Mult**	**Description**
RemoveSubscriptionResponseObject	application/json	1	Confirmation or error message

5.7.11.3.3 Service response

Table 58 describes the service response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.16.

Table 58 – HTTP Response codes and messages of Remove Subscription

HTTP Code	Message
200	Subscription <identifier> removed
400	Bad request
403	Not authorized to remove subscription
404	Subscriber identifier not found

5.7.11.4 Dynamic behavior

Figure 32 describes the dynamic behavior of the service interface.

The Remove Subscription interface is used to remove existing subscription(s). See also 5.7.10.4 for an operational description.

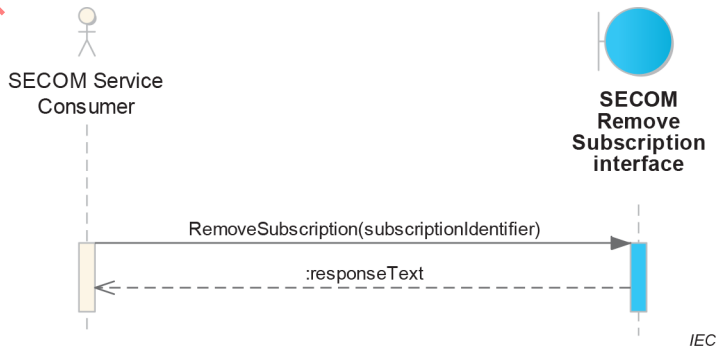


Figure 32 – Sequence diagram for Remove Subscription interface

5.7.12 Service interface – Subscription Notification

5.7.12.1 Specification

The interface receives notifications when a subscription is created or removed by the information provider.

Figure 33 describes the Subscription Notification in UML.

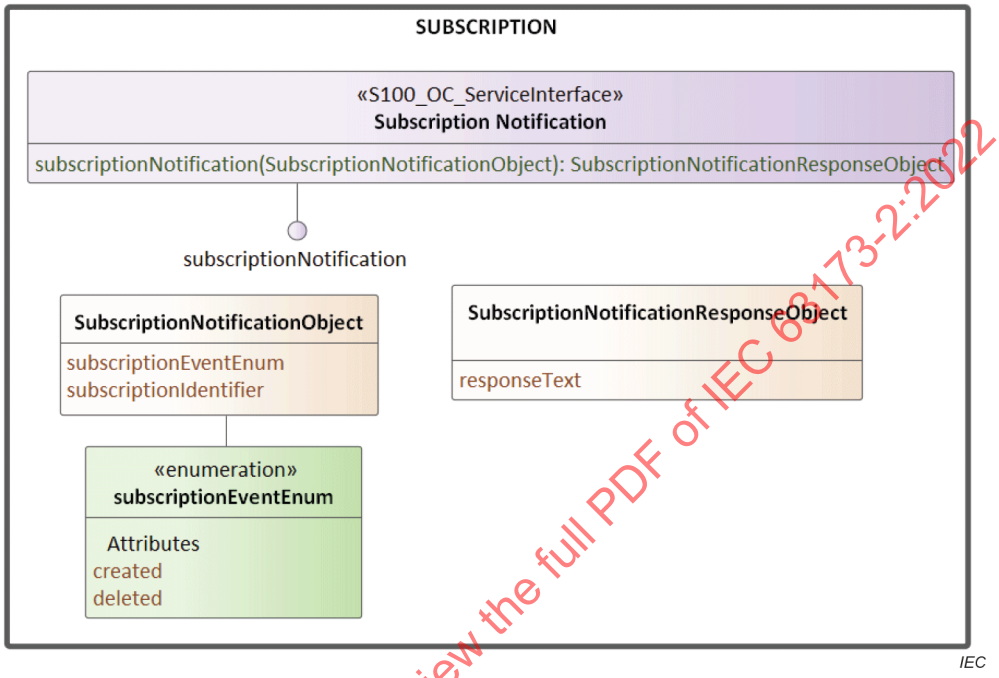


Figure 33 – Subscription Notification interface UML diagram

5.7.12.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 59, Table 60 and Table 61.

Table 59 – Information input for Subscription Notification interface

SubscriptionNotificationObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
subscriptionIdentifier	1	Mandatory	UUID	Identifier of the subscription
eventEnum	1	Mandatory	SubscriptionEventEnum	Subscription event enum according to Table 61

Table 60 – Information output for Subscription Notification interface

SubscriptionNotificationResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
message	1	Optional	CharacterString	Success or error response message

Table 61 – Enumerations for Subscription Notification interface

SubscriptionEventEnum	
Value	**Definition**
1	Subscription created
2	Subscription removed

5.7.12.3 REST Design

5.7.12.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.12.3.2 Operation – POST /subscription/notification

This operation receives notification when subscription is created, either internally by information provider, or externally on request by consumer.

Table 62 describes the logical parameters in the interface.

Table 62 – Information exchange for Subscription Notification

REST Operation			
POST URL/v1/subscription/notification {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Description**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Description**
SubscriptionNotificationObject	application/json	1	Contains the identity of the information object in focus and type of event; Create or Delete.
Return (out)	**REST Encoding**	**Mult**	**Description**
SubscriptionNotificationResponseObject	application/json	1	Confirmation or error message

5.7.12.3.3 Service response

The service shall respond with HTTP codes and message according to Table 63. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.16.

Table 63 – HTTP response codes for Subscription Notification

HTTP Code	Message
200	OK
400	Bad request

5.7.12.4 Dynamic behavior

Figure 34 describes the dynamic behavior of the service interface.

The Subscription Notification interface is used to get notifications on created and deleted subscriptions. See also 5.7.10.4 for an operational description.

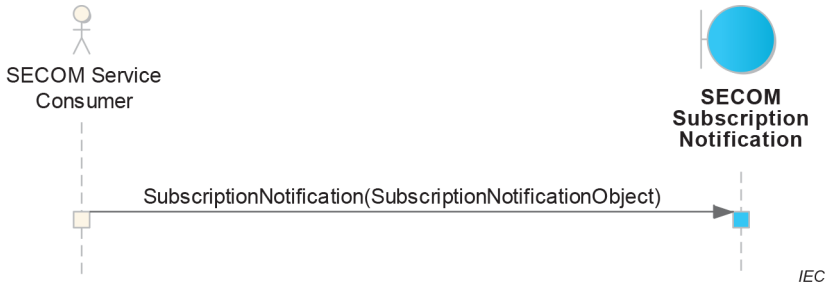


Figure 34 – Sequence diagram for Subscription Notification interface

5.7.13 Service interface – Capability

5.7.13.1 Specification

The purpose of this interface is to provide a dynamic method for asking a service instance at runtime what interfaces are accessible for respective valid payload, formats and versions. If an interface is not implemented, HTTP 501 response should be returned.

Table 64 shows an example of a capability for a specific SECOM information service. This instance is capable of receiving product types S-124 capsulated in a S100_DataSet and RTZ, publishing and sending RTZ as response from Get requests, authorizing access and enabling subscription to RTZ and S-124, supporting encrypted RTZ and open S-124.

Table 64 – Capability example

containerType	S100_DataSet	NONE
dataProductType	S124	RTZ
productSchemaUrl	http://iho.int/s124.xsd	http://cirm.org/rtz/index.html
upload	true	true
uploadLink	true	true
get	false	true
getByLink	false	true
getSummary	false	true
subscription	true	true
access	true	true
encryptionKey	false	true

A SECOM instance implementation shall reflect its capabilities by always validating the product type in Table 8 of incoming requests. If the container type is a single S100_DataSet or a S100_ExchangeSet, the data content or product type is checked and appropriate actions are taken.

Figure 35 describe the capability interface in UML.

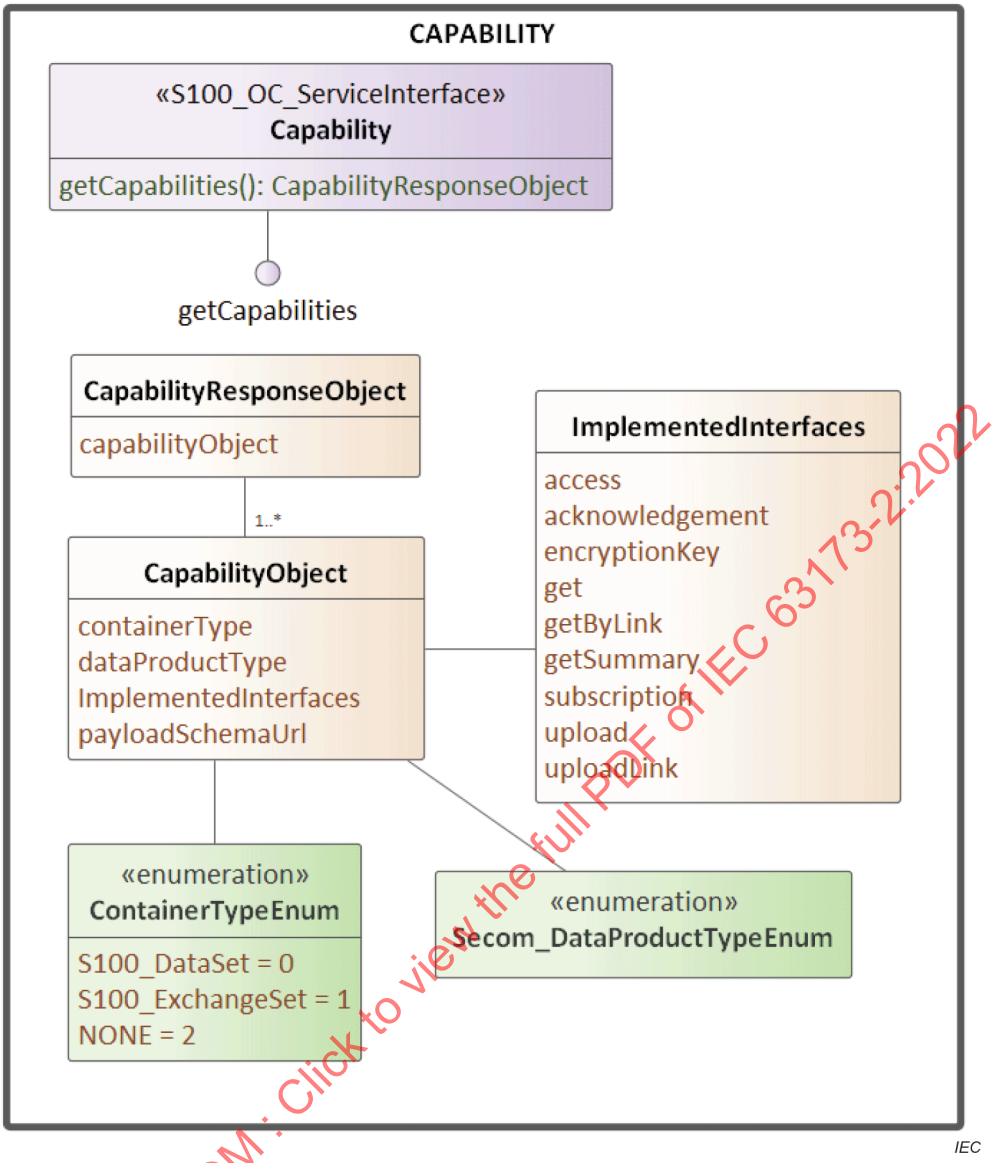


Figure 35 – Capability interface UML diagram

5.7.13.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 65.

Table 65 – Information output for Capability interface

CapabilityResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
capability	1 ..*	Optional	CapabilityObject	List of capability objects
CapabilityObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
containerType	1	Optional	ContainerTypeEnum	The container type of the data attribute, see Table 7
dataProductType	1	Optional	SECOM_DataProductType.Name	The name column of SECOM_DataProductType in Table 8
productSchemaUrl	1	Optional	URL	The schemaUrl attribute of the S100 Product
implementedInterfaces	1	Optional	ImplementedInterfaces	List of service interfaces implemented with the payload (product) in focus
serviceVersion	1	Optional	CharacterString	Version of service instance, for example New Editions denoted as n.0.0 Revisions denoted as n.n.0 Clarifications denoted as n.n.n
ImplementedInterfaces				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
upload	1	Optional	Boolean	True if service interface is implemented
uploadLink	1	Optional	Boolean	True if service interface is implemented
get	1	Optional	Boolean	True if service interface is implemented
getByLink	1	Optional	Boolean	True if service interface is implemented
getSummary	1	Optional	Boolean	True if service interface is implemented
subscription	1	Optional	Boolean	True if service interface is implemented
access	1	Optional	Boolean	True if service interface is implemented
encryptionKey	1	Optional	Boolean	True if service interface is implemented

5.7.13.3 REST Design

5.7.13.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.13.3.2 Operation – GET /capability

This operation receives request for service interface capabilities. The result contains the interfaces provided by the service instance, and the information product type handled.

Table 66 describes the logical parameters in the interface.

Table 66 – REST implementation of Capability

REST Operation			
GET URL/v1/capability {body}: return			
REST Parameter (in)	**Type**	**Mult**	**Definition**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**Type**	**Mult**	**Definition**
No body defined	n/a	n/a	n/a
Return (out)	**Type**	**Mult**	**Definition**
CapabilityResponseObject	application/json	1..*	Description of service capabilities; Which interfaces that are valid for the specific service instance, the accepted payload format and version, and specific requirements in payload etc.

5.7.13.3.3 Service response

Table 67 describes the service response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.17.

Table 67 – HTTP response codes and messages of Capability

HTTP Code	Message
200	OK

5.7.13.4 Dynamic behavior

Figure 36 describes the dynamic behavior of the service interface.

The Capability interface is used to retrieve information of a SECOM instance's supported interfaces and payload.

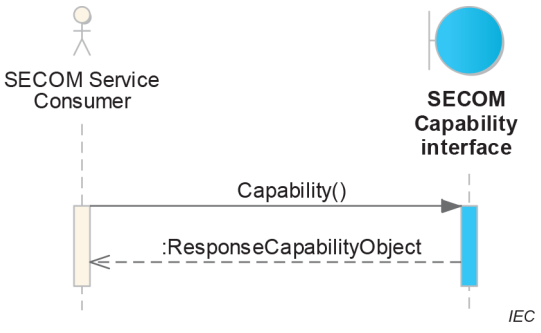


Figure 36 – Sequence diagram for Capability interface

5.7.14 Service interface – Ping

5.7.14.1 Specification

The purpose of the interface is to provide a dynamic method to ask for the technical status of the specific service instance.

Figure 37 describes the Ping interface in UML.

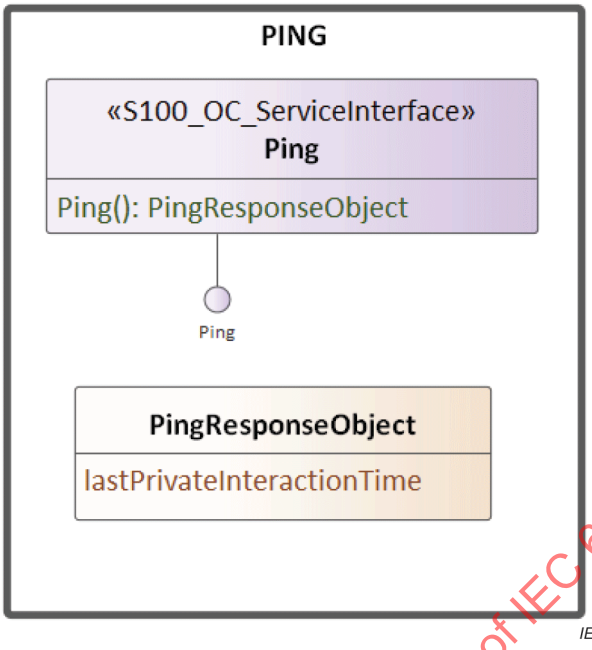


Figure 37 – Ping interface UML diagram

5.7.14.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 68.

Table 68 – Information output for Ping interface

PingResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
lastPrivateInteractionTime	0..1	Optional	DateTime	Describes the time for last interaction with end user application

5.7.14.3 REST Design

5.7.14.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.14.3.2 Operation – GET /ping

This operation receives a request for status on the service instance.

Table 69 describes the logical parameters in the interface.

Table 69 – REST implementation of Ping

REST Operation			
GET URL/v1/ping {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Description**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Description**
No body defined	n/a	n/a	n/a
Return (out)	**REST Encoding**	**Mult**	**Description**
PingResponseObject	application/json	1	Status of the service instance.

5.7.14.3.3 Service response

Table 70 describes the service response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.18.

Table 70 – HTTP response codes of Ping

HTTP Code	Message
200	ResponsePingObject

5.7.14.4 Dynamic behavior

Figure 38 describes the dynamic behavior of the ping service interface.

The Ping interface is used for retrieve information of the current status (version and last interaction time) of a service instance.

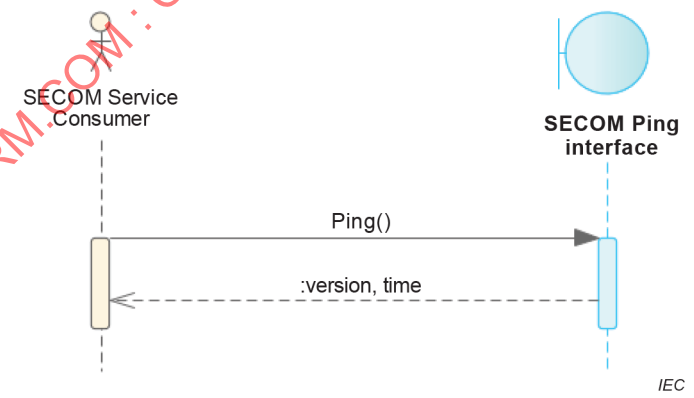


Figure 38 – Check status on service

5.7.15 Service interface – EncryptionKey

5.7.15.1 Specification

The purpose of the interface is to exchange a temporary secret key.

Figure 39 describes the Encryption Key interface in UML.

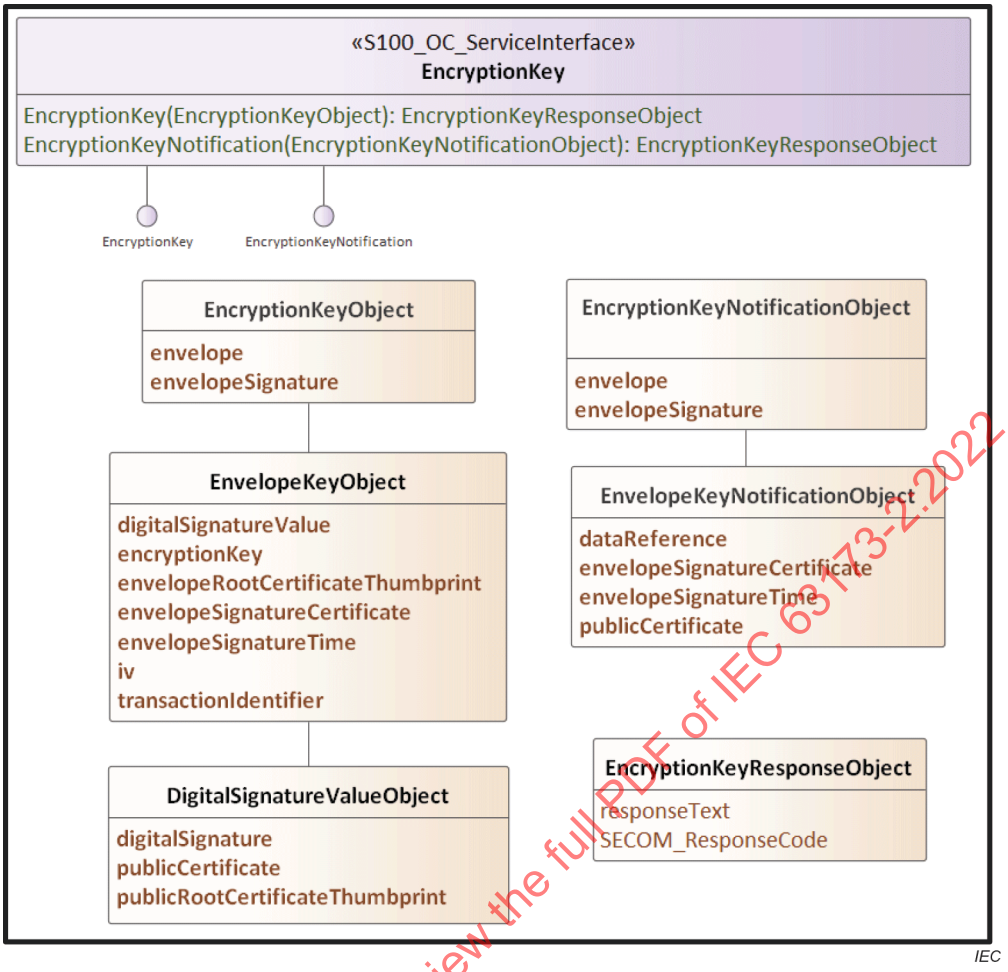


Figure 39 – Encryption Key interface UML diagram

5.7.15.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 71, Table 72 and Table 73.

Table 71 – Information input for Encryption Key interface

EncryptionKeyObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
envelope	1	Mandatory	EnvelopeKeyObject	The complete EnvelopeKeyObject being uploaded to receiver including data and message properties
envelopeSignature	1	Mandatory	CharacterString	The signature of the EnvelopeKeyObject in HEX format without whitespace or linebreaks
EnvelopeKeyObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
encryptionKey	1	Mandatory	Base64	The protected symmetric encryption key
iv	1	Mandatory	Base64	Inititalisation vector
transactionIdentifier	1	Mandatory	UUID	Identifier to the transaction with the encrypted data
digitalSignatureValue	1	Mandatory	DigitalSignatureValue Object	See Table 5
envelopeSignatureCertificate	1	Mandatory	CharacterString	The public certificate of the sender, used to verify the envelopeLinkObject signature
envelopeRootCertificateThumbprint	1	Optional	CharacterString	Claimed Thumbprint for Signed Root Key (X.509 Certificate)
envelopeSignatureTime	1	Optional	DateTime	Time when encryptionKey envelope is signed

Table 72 – Information input for Encryption Key Notification interface

EncryptionKeyNotificationObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
envelope	1	Mandatory	EnvelopeKeyObject	The complete EnvelopeKeyNotificationObject being uploaded to receiver including data and message properties
envelopeSignature	1	Mandatory	CharacterString	The signature ot the EnvelopeKeyNotificationObject in HEX format without whitespace or linebreaks
EnvelopeKeyNotificationObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
dataReference	1	Mandatory	UUID	Information identifier retrieved by using reference from response in Get Summary request (UUID)
publicCertificate	1	Mandatory	CharacterString	Public certificate of data consumer for deriving shared symmetric key
envelopeSignatureCertificate	1	Optional	CharacterString	The public certificate of the sender, used to verify the EnvelopeKeyNotificationObject signature Key (X.509 Certificate)
envelopeSignatureTime	1	Optional	DateTime	Time when envelope is signed

Table 73 – Information output for Encryption Key interface

EncryptionKeyResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
SECOM_ResponseCode	0..1	Mandatory	SECOM_ResponseCodeEnum	Additional error code for internal errors
message	1	Optional	CharacterString	Success or error response message

5.7.15.3 REST Design

5.7.15.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.15.3.2 Operation – POST /encryptionkey

This operation is used to upload (push) an encrypted secret key to a consumer.

Table 74 describes the logical parameters in the interface.

Table 74 – REST implementation of EncryptionKey upload

REST Operation			
POST URL/v1/encryptionkey {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Description**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Description**
EncryptionKeyObject	application/json	1	The protected encryption key with its metadata
Return (out)	**REST Encoding**	**Mult**	**Description**
EncryptionKeyResponseObject	application/json	1	Response from service

5.7.15.3.3 Service response

Table 75 describes the service response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.6.

Table 75 – HTTP response codes of EncryptionKey upload

HTTP Code	Message
200	EncryptionKeyResponseObject
400	Bad request

5.7.15.3.4 Operation – POST /encryptionkey/notify

This operation enables a consumer to request an encrypted secret key from a producer by providing a reference to the encrypted data and a public certificate for symmetric key derivation used to protect the temporary encryption key during transfer. The response is sent asynchronously using the consumer's encryption key operation described in 5.7.15.3.2. Table 76 describes the REST implementation.

Table 76 – REST implementation of EncryptionKey notification

REST Operation			
POST URL/v1/encryptionkey/notify {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
EncryptionKeyRequestObject	application/json	1	The encryption key request object
Return (out)	**REST Encoding**	**Mult**	**Definition**
EncryptionKeyResponseObject	application/json	1	HTTP code with message

5.7.15.3.5 Service response

Table 77 describes the service response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.6.

Table 77 – HTTP response codes of EncryptionKey notification

HTTP Code	Message
202	Notification received
403	Not authorized

5.7.15.4 Dynamic behavior

Figure 40 and Figure 41 describes the dynamic behavior of the EncryptionKey service interface.

The information owner decides to protect the information and generates a random key and initialization vector which are used to encrypt the data. In order for the information consumer to decrypt the data, the random key is protected using a derived symmetric key using the information consumer's public certificate together with the information owner's private key. The encrypted random key is signed and the random key, initialization vector, signature and the information owner's public key are put in the payload object and uploaded to the information consumer's EncryptionKey interface. See Figure 40.

The information owner decides to publish protected information and internally stores the generated random key and initialization vector. The service consumer retrieves the protected encryption key by notifying the information owner through his SECOM EncryptionKey notification interface. The information owner protects the encryption key with a derived symmetric key, derived using the publicKey from the notification request body together with its own private key. The encryption key is encrypted with the derived key and the encrypted encryption key is signed and sent to the service consumer's SECOM EncryptionKey interface. Finally the consumer gets the encryption key from its SECOM instance and can decrypt the retrieved protected information. See Figure 41.

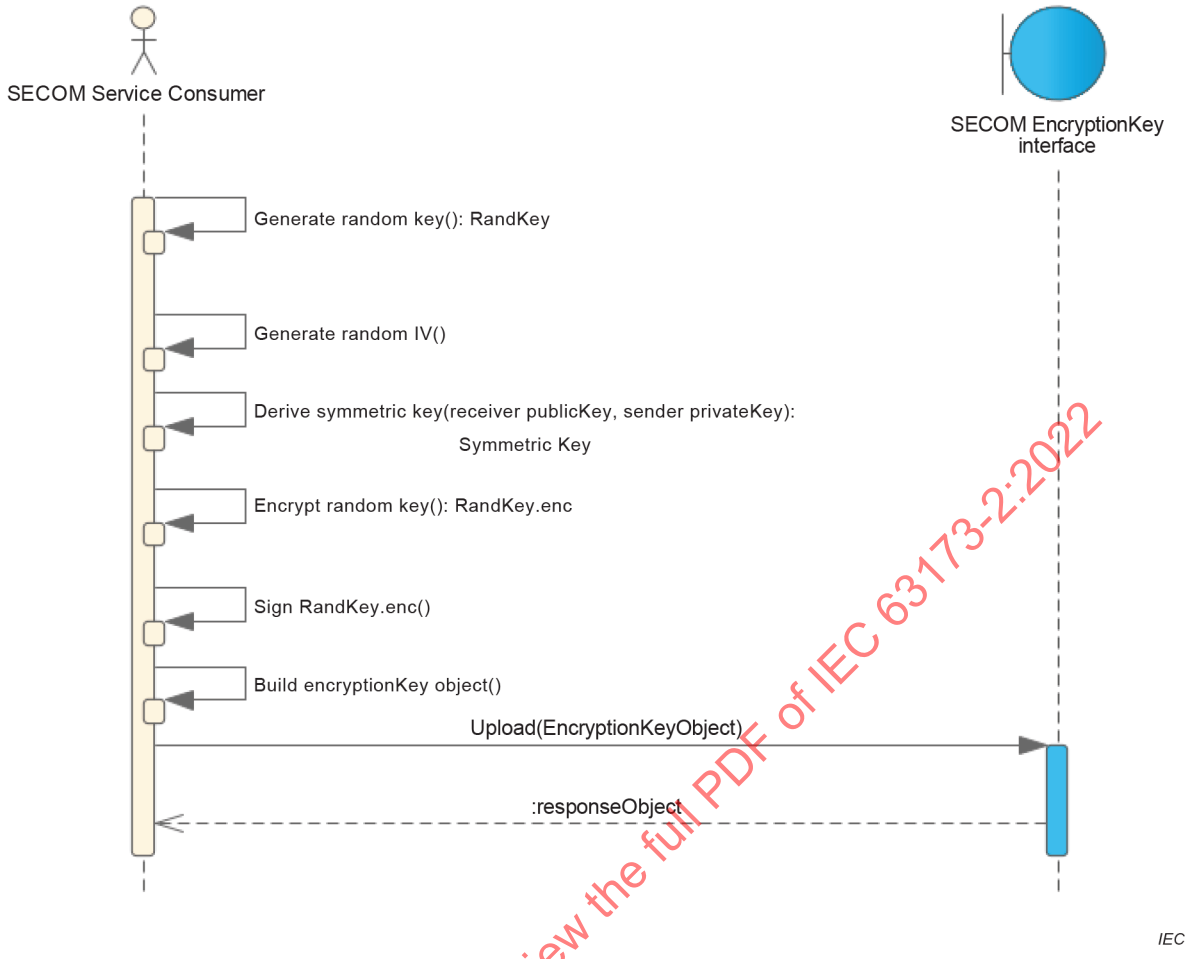


Figure 40 – Operational sequence diagram for EncryptionKey upload interface

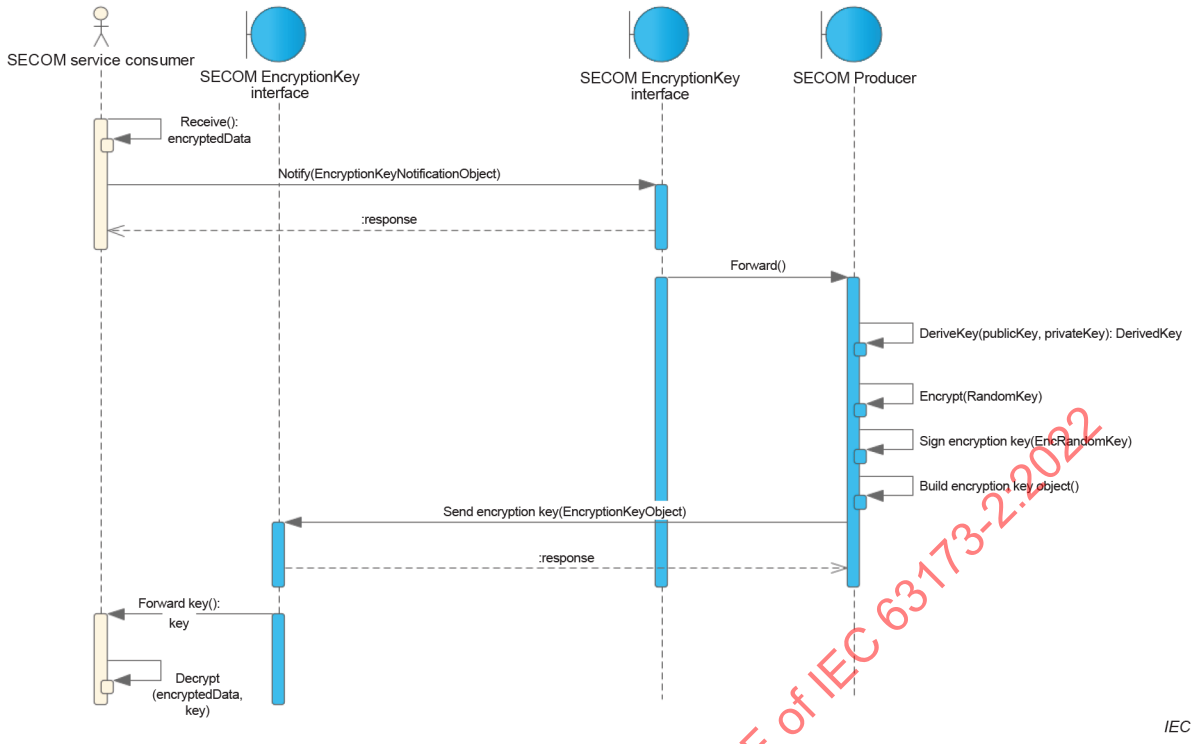


Figure 41 – Operational sequence diagram for EncryptionKey notification interface

5.7.16 Service interface – PublicKey

5.7.16.1 Specification

The purpose of this interface is to request (pull) a public key and to send (push) a public key.

The purpose of the boolean attribute "dataProtection" set to true is to retrieve the public certificate of the data recipient doing the decryption. The returned certificate is later used to derive a symmetric Diffie-Hellman key to protect the transfer of an encryption key. The "dataProtection" set to false indicates a request for the public certificate used for the verification of the digital signature.

The purpose of the string attribute "certificateThumbprint" is to retrieve the public certificate related to this identifier.

Figure 42 shows the PublicKey UML diagram.

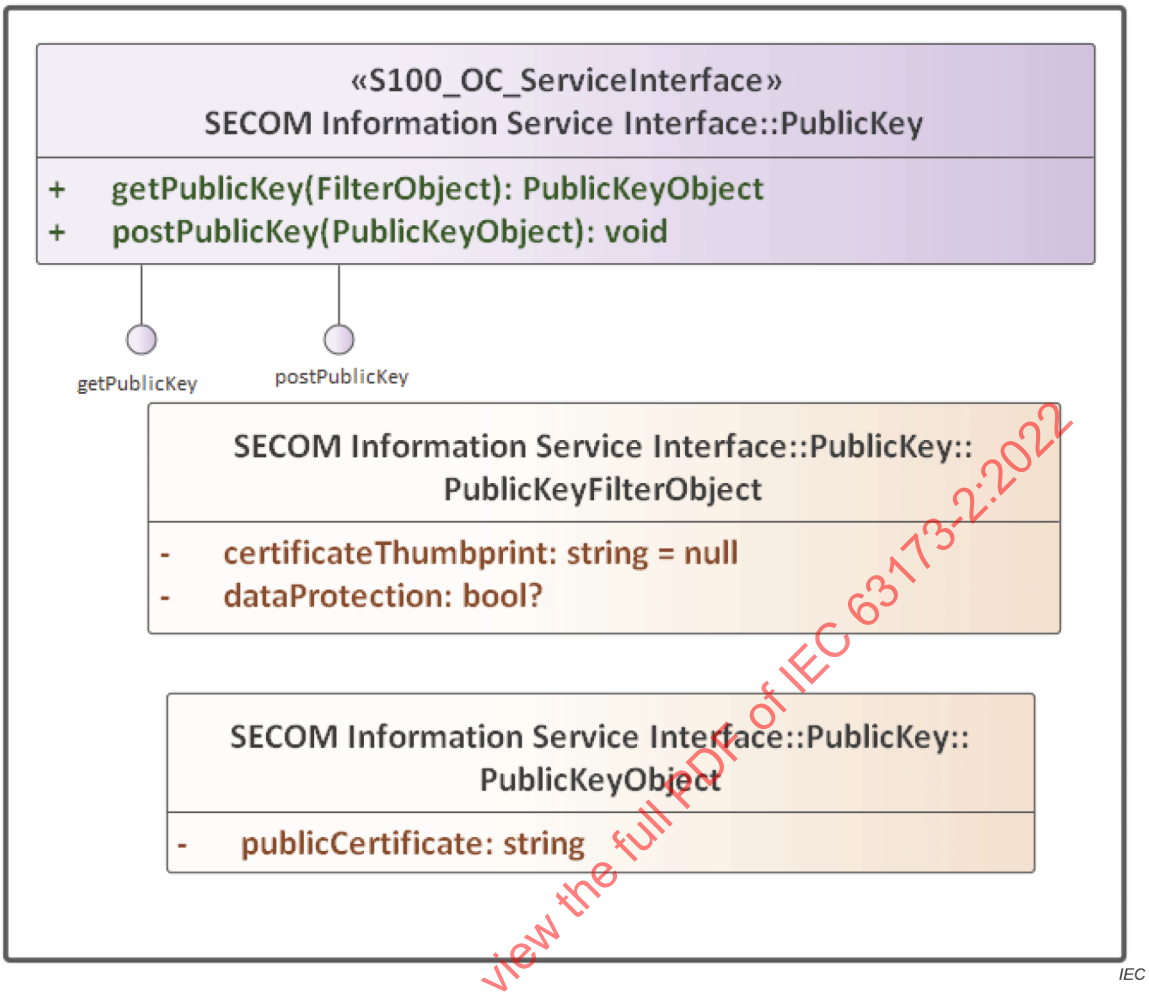


Figure 42 – PublicKey interface UML diagram

5.7.16.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 78 and Table 79.

Table 78 – Information input for PublicKey interface

PublicKeyFilterObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
dataProtection	0..1	Mandatory	Boolean	Flag indicating public certificate needed for encryption key exchange
certificateThumbprint	0..1	Mandatory	CharacterString	Claimed Thumbprint for signed key (X.509 Certificate)

Table 79 – Information output for PublicKey interface GET and information input for PublicKey interface POST

PublicKeyObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
publicCertificate	1	Mandatory	Base64	Public certificate x.509 in PEM format, Base64 encoded byte array.

5.7.16.3 REST Design

5.7.16.3.1 General

The service interface and its data exchange model is defined with REST technology.

5.7.16.3.2 Operation Get /publicKey

This operation receives a get request for a public key. If authorized, the key is sent back in the response. It is up to the service provider to apply relevant authorization procedure and access control to information.

Table 80 describes the logical parameters provided in this operation.

Table 80 – REST implementation of PublicKey (GET)

REST Operation			
GET URL/v1/publicKey?parameters: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
dataProtection	Boolean	0..1	Flag indicating that the requested key is for symmetric key derivation in exchange of a random encryption key
certificateThumbprint	String	0..1	Claimed Thumbprint for signed key (X.509 Certificate)
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
No body defined	n/a	n/a	n/a
Return (out)	**REST Encoding**	**Mult**	**Definition**
PublicKeyObject	application/json	1	Public certificate x.509 in PEM format, Base64 encoded byte array.

5.7.16.3.3 Service response

Table 81 describes the service instance response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.7.

Table 81 – HTTP response code and message of PublicKey (GET)

HTTP Code	Message
200	PublicKeyObject
400	Bad request
403	Not authorized to requested information
404	Not found

5.7.16.3.4 Operation – POST /publicKey

This operation uploads (pushes) a public key.

Table 82 describes the logical parameters provided in this operation.

Table 82 – REST implementation of PublicKey (POST)

REST Operation			
POST URL/v1/publickey {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Description**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**REST Encoding**	**Mult**	**Description**
PublicKeyObject	application/json	1	Public certificate x.509 in PEM format, Base64 encoded byte array.
Return (out)	**REST Encoding**	**Mult**	**Description**
ResponseObject	application/json	1	Response from service

5.7.16.3.5 Service response

Table 83 describes the service instance response. See also Table 11 for codes common to all interfaces.

For tests related to these error codes, see 10.7.

Table 83 – HTTP response code and message of PublicKey (POST)

HTTP Code	Message
200	OK
400	Bad request

5.7.16.4 Dynamic behaviour

Figure 43 describes the dynamic behaviour of the PublicKey service interface. A consumer can request a public X.509 certificate by filter combinations from Table 78 or upload (push) a public X.509 certificate.

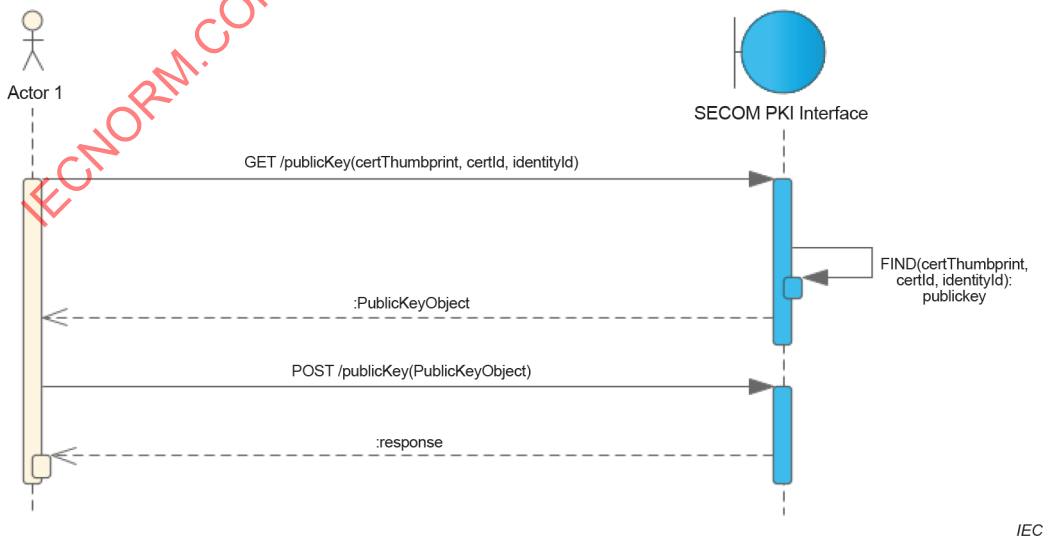


Figure 43 – Operational sequence diagram for PublicKey interface

6 SECOM communication channel security

6.1 General

The rationale to define the communication channel security is to protect the transfer of data and to facilitate authentication on the transfer when using SECOM information services. Both parties in the communication need to follow the same communication scheme to achieve interoperability when exchanging information with web services.

The SECOM information service interface is focused on the exchange of data, and this data shall always be signed. The SECOM information service interface also contains supporting service interfaces where no operational data is exchanged but still authentication is needed to verify who actually did the request, here called "service authentication". When a consumer for instance requests to subscribe to information, or asks for access to information, it is essential to authenticate the consumer before executing the request. For this purpose, a client certificate issued from SECOM PKI (see Clause 8) shall be exchanged enabling the service instance provider to authenticate the service instance consumer.

In this context, the service instance provider is the actor exposing an instance of SECOM information service interface, i.e. acting as server waiting for request. The service instance consumer is another service instance or application making a request to a SECOM information service instance, i.e. acting as client and initiating the service request. When a ship sends (uploads) information, such as a S-421 Route Plan (IEC 63173-1), to a VTS, the VTS is acting as service instance provider (server) and the ship is acting as the service instance consumer (client). Vice versa, when the VTS is sending (uploading) information, such as recommended change of route (S-421), the ship is acting as service instance provider (server) and the VTS is acting as the service instance consumer (client). Hence, as the SECOM information service interface is designed, most actors need to have the capability to act as both service instance provider (server) and service instance consumer (client).

Clause 6 describes technology and procedures for the protection of IP and HTTP based communication channels for web services, such as service instances based on SECOM information services.

6.2 Secure transfer

6.2.1 Secure communication channel

Encryption of channel by the use of Transport Layer Security, TLS version 1.2 (RFC 5246) and TLS version 1.3 (RFC 8446) shall be supported.

HTTP over TLS shall be used according to RFC 2818.

HTTP/1.1 shall be used according to RFC 7231.

This document describes business-to-business communication with mutual authentication, where both client and server authenticate each other using certificate-based TLS mutual authentication (TLS client-side X.509 authentication). See Figure 44.

Service authentication shall be based on X.509 certificates (RFC 5280 and RFC 2459).

Certificates shall be possible to be revoked, hence the authentication procedure shall contain a check against the certificate revocation list (CRL) or OCSP.

6.2.2 Authentication procedure

The authentication procedure in a service request is outlined in Figure 44 and describes how the client certificate shall be verified that it is issued from a SECOM PKI. The server (host) certificate shall be verified that it is issued by a trusted certificate authority, which does not need to be SECOM PKI.

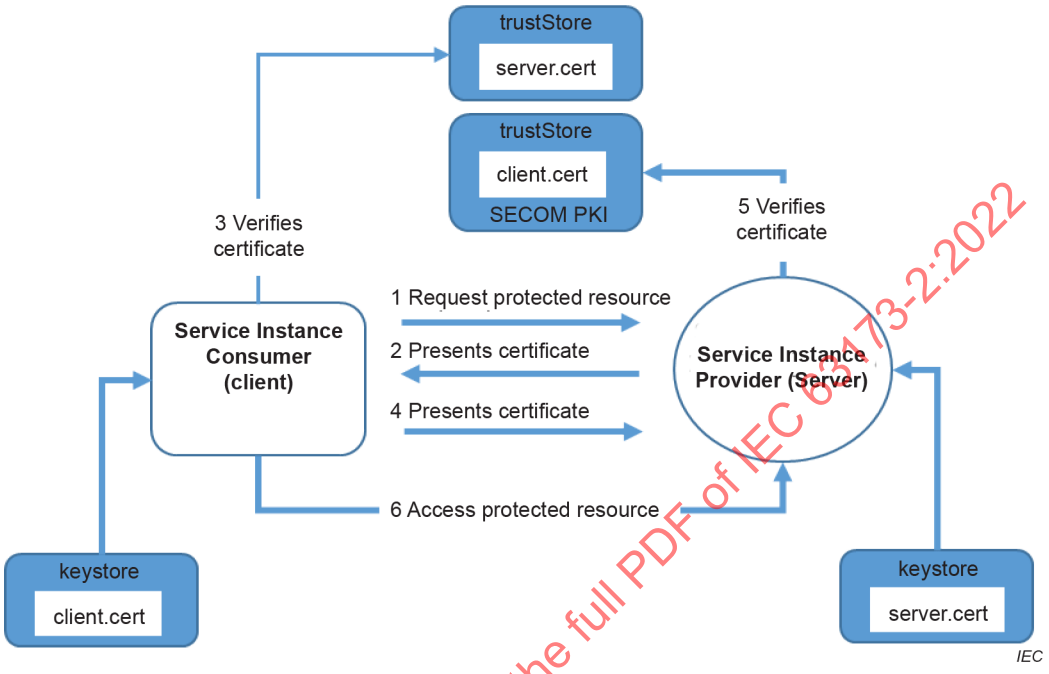


Figure 44 – Principle for service authentication

The procedure to verify the server's certificate shall be:

1) the client shall verify that the server's certificate is valid;
2) the client shall verify that the server's certificate is issued by trusted CA, normally issued by one of the trusted certificate authorities on the market but can also be issued by SECOM PKI;
3) the client shall verify that the server's domain name corresponds to the domain name in the certificate.

The procedure to verify the client's certificate shall be:

4) the server shall verify that the client's certificate is valid;
5) the server shall verify that the client's certificate is issued by SECOM PKI.

7 SECOM data protection

7.1 General

The SECOM data protection includes digital signature on the data exchanged that facilitate end-to-end data authentication. The digital signature includes a checksum and claimed identity. The checksum facilitates the data integrity check. The checksum shall be encrypted with the claimed identity (private key for the sender), which facilitates data authentication by the receiver using the public key of the sender.

The SECOM data protection also describes optional encryption of data.

For information regarding private and public keys, see Clause 8.

7.2 Data compression and packaging

The use of compression may be applied on data. The individual IHO S-100 based Product Specifications provides the details for use of compression.

If compression of classified data is applied, the data shall be compressed before encrypted.

The compression shall use ZIP algorithm and method DEFLATE. The encryption and digital signature feature of ZIP are not used.

SECOM data compression and packaging is aligned with data compression and packaging in IHO S-100. See IHO S-100:2018, Part 15-5, Data compression and packaging.

NOTE IHO S-100:2018, ed 4.0.0, is the latest released version at the time of writing of this document.

7.3 Data authentication and signing

7.3.1 General

SECOM is based on the same technology for data signing as described in IHO S-100:2018, Part 15-8, Data authentication, and uses a standard algorithm and key exchange mechanism widely available and used.

The digital signature uses asymmetric public key algorithms within SECOM PKI scheme to unbreakingly bind data with the identity of the issuer.

The scheme relies on asymmetric encryption of a checksum of the data. By verifying the signature against the issuer's public key, and also verifying the issuer's public key against a top level identity, the user is assured of the signer's identity. A detailed technical description of digital signatures is beyond the scope of this document and the reader is referred to the NIST Digital Signature Standard (DSS–FIPS Publication 186-3) for more detailed explanation.

The difference between the SECOM data authentication scheme and the data authentication scheme described in IHO S-100:2018, Part 15-8, is that SECOM is designed for duplex information exchange, hence an actor such as a ship is both sender and receiver of information, both server and client in the scheme. This means that the number of data servers will increase a lot in comparison to the assumption in IHO S-100 which is designed primarily for a shore server providing information to ships (clients). It also means that each client needs to get access to a lot more public keys for data authentication. But the principle remains the same.

The payload and the signature for authentication and integrity purpose is always exchanged which is also described in the SECOM information service interface.

7.3.2 Data formats and standards for digital signatures, keys and certificates

This description is aligned technically with the corresponding description in IHO S-100:2018 (ed. 4.0.0, Part 15-8.3).

The following categories of content are required for data authentication.

1) Key pairs, private and public keys (See Clause 8).
2) Certificate signing requests and digitally signed public keys. When a public key is itself digitally signed, it is referred to as a "certificate". When the public key is signed by its corresponding private key it is referred to as a "self-signed" certificate. These are laid out as X.509 (RFC 5280 and RFC 2459) records and can be either DER (ISO/IEC 9594-8) or PEM (RFC 2045) encoded to be sent to the SECOM PKI for signing. When embedded within XML files, keys shall be PEM encoded so that the plain text can be inserted as an XML element.
3) The digital format of the signed public keys ("certificates") (See Clause 8).

PEM format defines a textual encoding of the multiple large numbers required by the Digital Signature Algorithm (DSA) (along with the DSA parameters required by the DSA algorithm). PEM encoding allows the embedding of public keys within JSON and XML objects.

More information on creation of keys and signing request is found in Clause 8.

Digital signatures in SECOM (and S-100) are implementations of the Digital Signature Standard (DSS). The DSS uses the Secure Hash Algorithm (SHA-256, see ISO/IEC 10118-1) to create a message digest (hash) of the file content that is 256 bits long. The message digest is then input to the DSA to generate the digital signature for the message using an asymmetric encryption algorithm and the "private key" of the signer's key pair.

In the DSA algorithm, a signature is a sequence of two integers. By convention, these are referred to as R and S (an "R,S pair"). The format of digital signatures when embedded in XML files is as follows:

```
<digitalSignature>

302C021433796C6647CC1C55A67DC72FA7C6E157A6594B2B02145D3768B44F3A6ABA1

1A77178B738AD3B6A0DE344

</digitalSignature>
```

The R,S pair are represented by a hexadecimal encoding (digits 0-9, letters A-F). The encoding of the two R,S large integers is an abstract syntax notation one (ASN.1) sequence 2. These are produced natively by the openssl implementation and can be generated and verified without the need to unpack the individual R and S integers. This encoding conveniently wraps the two integers into a single Hexadecimal encoded string. For example, the ASN.1 schema for the above example is:

```
SEQUENCE (2 elem)

  INTEGER (158 bit) 357903468461694385184092679318935791752802642315
  INTEGER (158 bit) 351274375691773793245737972991313380578601275707
```

7.3.3 Creation of digital signature

This description is aligned technically with the corresponding description in IHO S-100:2018 (ed 4.0.0, Part 15-8.6).

As stated in 4.1, the SECOM data protection scope is between end users. Since the data transfer might occur over non-secure communication channels, for example the "last mile", the original data needs to be signed by the information owner. The signature is created by:

1) the information owner creates a digital signature for the data using the DSA algorithm and their private key as described in 7.3.2;
2) the R,S pair making up the digital signature is stored as an ASN.1 sequence of 2 elements in the digitalSignature field of a DigitalSignatureValueObject.

In the resulting json, the signature looks as follows:

```
"digitalSignature":"302C021433796C6647CC1C55A67DC72FA7C6E157A6594B2B02145D3768B44F3A6ABA11A77178B738AD3B6A0DE344"
```

The signature shall always be exchanged together with the data content.

7.3.4 Creation of envelope signature

For interfaces designed for pushing data, i.e. with REST method POST, an extra layered signature is added containing not only the data but also the metadata, see Table 86 for the ones concerned. The main reason for adding another layer of signature is to ensure data integrity end to end for the complete payload where the data and metadata are packaged in an envelope object. To ensure interoperability between different environments, programming languages and server operating systems, the signing is processed using a custom serializing procedure described herein.

The basic idea is to parse attributes in an envelope and concatenate their values to a comma-separated value string (CSV). As a separator, a dot has been chosen. The reason for a separating dot is mainly to show where properties start/end in the long combined string. The size of the final signature is not affected by these added characters.

The order of each added attribute (see Table 16, Table 20, Table 24, Table 71 and Table 72) is achieved by adding the attribute names in the order the attributes appear in each payload table. For envelope objects including other objects as attributes, the same order is applied for each object sub-structure. The next step is to transverse the ordered object and its sub-objects and add the attribute values following the order. The CSV now contains all data needed for the signature.

The conversion from each attribute datatype to its string representation needs to be environment and location agnostic. Therefore, a set of string conversion rules is defined that need to be performed in order for the signature creation and verification to work for both sender and receiver actor. In Table 84, the different rules for each datatype conversion to string is listed, and Table 85 shows the interfaces the conversion applies to.

Table 84 – Conversion rules

Datatype	Rule	Example
DateTime	1. Convert datetime object to integer Unix timestamp in seconds 2. Parse integer as string	Input: 2020-07-01T15:00:00 Output: "1593608400"
bool	Convert bool value to lower case string	Input: True Output: "true"
Guid	Convert guid object to lowercase string	Input:[0f8fad5b-d9cb-469f-a165-70867728950e] Output: "0f8fad5b-d9cb-469f-a165-70867728950e"
enum	1. Convert enum to its integer representation 2. Parse integer as string	Input: Third: 3 Output: "3"
byte[]	Convert byte array to Base64String	Input: {10, 20, 30, 40, 0, 0, 0, 0} Output: "ChQeKDI8RIA="
integer	Parse integer as string	Input: 17 Output: "17"
null	Represent null value as empty string	Input: null Output: ""

EXAMPLE Signing an acknowledgement envelope is performed as follows.

1) Set start value of result; CSV = ""

2) Add properties according to the ordering in Table 24 producing an array: [createdAt, envelopeCertificate, transactionIdentifier, ackType, nackType, envelopeSignatureTime]

3) Start adding values from the traversed array

   a) Convert DateTime createdAt to a Unix timestamp string; CSV += "1593608400."

   b) envelopeCertificate is already a string; CSV += "MIIE1zCCBFygAwIBAgIU."

c) Convert Guid transactionIdentifier to lowercase string; CSV += "10f032cc-2c31-4441-b6e5-bc0511983cd0."

d) Convert AckEnum ackType to integer and then string; CSV += "2."

e) Convert NackEnum nackType value null to empty string; CSV += "."

f) Convert DateTime envelopeSignatureTime to a Unix timestamp string; CSV += "1593608405"

4) Remove last dot delimiter

5) CSV now contains; "1593608400.MIIE1zCCBFygAwIBAgIU.10f032cc-2c31-4441-b6e5-bc0511983cd0.2,,1593608405"

6) Finally convert the generated string to a byte array using UTF8 encoding; CSV[] = ConvertToArray(CSV)

7) Use CSV[] as input to the DSA algorithm

Table 85 – Interfaces with envelope signature

Section	Interface	Type
5.7.2	Service interface – Upload	EnvelopeUploadObject
5.7.3	Service interface – Upload Link	EnvelopeLinkObject
5.7.4	Service interface – Acknowledgement	EnvelopeAckObject
5.7.15	Service interface – EncryptionKey	EnvelopeKeyObject

7.3.5 Verification of digital signature

The purpose of the digital signature is to verify data integrity and authenticate the data. Digital signature verification is an algorithm which operates on three independent pieces of data:

1) some content which requires validation (the format of this content is arbitrary);
2) a Public Key, suitably encoded. In the DSA algorithm adopted this Public Key is composed of a set of DSA parameters together with a Public Key;
3) a signature. In the DSA algorithm a signature is composed of two numbers, by convention these are referred to as R and S (an R,S pair).

A signature verification process identifies whether the R,S pair authenticate the content against the given Public Key. This can only result in a true or false result.

DSA digital signature verification achieves two results:

- Authentication: The implementing system verifies the data server public key ("content") and the signature in the data server certificate ("signature") against the SA public key ("Public Key") to confirm that the supplier's public key in the certificate is valid and that the data server is a bona fide member of the designated data protection scheme;
- Integrity Check: The implementing system verifies the data file signature ("signature") and the data server public key in the data server certificate ("Public Key") against the data file ("content"). This verifies the content of the data file.

If this validation check is successful, then it proves that the data file has not been corrupted in any way and that the identity of the data server within the dataset signatures is validated by the scheme administrator's (SA's) identity as defined in the SA root certificate.

The signature is calculated on the original data, so before calculating the signature all compression, encryption and Base64 encoding needs to be reversed in the following order.

1) Binary data encoded as Base64 is reverted back to bytes.
2) If the binary data is compressed, it needs to be reverted.
3) If the data is encrypted, the data is decrypted using the appropriate key (assuming the receiver has received the key and has been able to decrypt it).

4) If the previous steps are successful, the data should now be an array of bytes and a SHA-256 hash can be calculated.
5) The hash calculated in step 4) is then used together with the public key in the digitalSignatureValue as input to the DSS verification.

NOTE If the data in the message is encrypted, the SECOM instance will not be able to decrypt the data part in the message, since this requires access to the private key of the end receiver.

7.3.6 Verification of envelope signature

When receiving a signed SECOM message, the receiving SECOM instance can verify that the message remains as the sender sent it (data integrity) by validating the envelope signature. This is done by recalculating the signature hash from the message data and verifying it toward the sender's certificate that is included in the message. This however only verifies that the certificate in the message is the one used to sign it. It is also important to verify that the certificate used is issued by a trusted CA to prove the claimed identity of the sender.

The verification process is similar to the signing process.

1) All properties are serialized and concatenated as text using the same principle as in the signing process in 7.3.4.
2) The text is converted to an array of bytes using UTF8 encoding.
3) A hash is calculated for the byte array created in step 2), using SHA-256.
4) The hash calculated in step 3) is verified according to DSA using the public key from the sender certificate.

This verification can be done by anyone receiving the message, which means that if the message is sent via a vendor API, the receiving SECOM instance can perform this verification before passing the message on to its final recipient.

7.3.7 Example of commands for data authentication

Table 86 shows examples of commands for signature creation and verification.

Table 86 – Command examples

Description	Example commands

Sign data	`openssl dgst -sha256 -sign privateKey.pem data.txt > data.sig`
	
	`xxd -u -ps data.sig > data.sig.hex`

Verify signature	`xxd -r -u -ps data.sig.hex > data.sig`
	
	`openssl dgst -sha256 -verify publicKey.pem -keyform pem`
	
	`-signature data.sig data.txt`

7.4 Data encryption

7.4.1 General

The use of data encryption is optional and, if available, is typically specified in each S-Product Specification. If it is described as optional in an S-Product Specification, the data server decides as the information owner if the data shall be encrypted.

SECOM supports use of IHO S-100:2018, Part 15-6, Data encryption algorithm for both shore to ship, and ship to shore.

7.4.2 Encryption algorithm

The algorithm for encryption used in SECOM shall be the same as described in IHO S-100:2018, Part 15-6, Data encryption, and is summarised below.

When data is encrypted, the algorithm shall be the advanced encryption standard (AES) in cipher block chaining (CBC) mode of operation. It is always assumed that the complete data (payload) shall be encrypted.

The AES block cipher algorithm is a symmetric-key algorithm. This means that the same key is used for encryption and decryption. The algorithm defines how one block of plain text is converted to one block of cipher text and vice versa. The block size of the AES is always 16 bytes (128 bit). The key length can be chosen from 128 bit, 192 bit or 256 bit. The corresponding variants are named AES-128, AES-192, or AES-256.

The AES algorithm can only encrypt one block of plain text. For larger messages, a block cipher mode of operation shall be used. This protection scheme chooses the cipher block chaining (CBC) mode for encryption of more than one block of data. In this mode of operation, it is required that the length of the plain text shall be an exact multiple of the block size. Padding is required.

The padding method that shall be used is described in PKCS#7 (RFC 2315). It adds N bytes to the message until its length is a multiple of 16 bytes. The value of each byte is N. Note that if the original plain text already has a multiple of 16 as length, a full block of 16 bytes each having the value of 16 shall be added.

7.5 Creation and transfer of encryption key

7.5.1 General

SECOM supports both encryption key management as described in IHO S-100:2018, 15-7 "Data encryption and licensing", and as described here. The provider of information decides which encryption key scheme (data protection scheme) to use.

The scheme described in IHO S-100:2018, 15-7, is focused on information from shore to ship, where the data server has submitted the symmetric keys used for data encryption wrapped in permits and linked to a hardware ID for the data client.

If S-100:2018, 15-7 Data encryption and licensing, is not applied, an alternative SECOM encryption key management scheme may be used. This scheme is suitable where an actor acts as both data server and data client, such as a ship sharing its route plan to several other actors, and other actors may send updated or optimized route plans back to the ship. In the SECOM encryption key management scheme, the symmetric key used for data encryption is generated as a temporary key and sent to the data client through a dedicated and protected service interface. The symmetric key shall only be used in limited time (session).

For details on encryption key management according to IHO S-100:2018, see reference 15-7 Data encryption and licensing.

7.5.2 SECOM encryption key management

The secret key shall be a temporary symmetrical key that is generated for the specific exchange of data. The encryption key shall be transferred to the client using the service interface defined in 5.7.15. The order of transfer of data and encryption key shall be insignificant and encryption key may be transferred before or after the encrypted data.

The secret key shall be valid and used only for a limited time. Figure 45 and Figure 46 shows the principal interaction between server and client for exchange of classified data with the SECOM key management scheme. The procedure used is a Diffie-Hellman key exchange method. The receiver's public certificate is retrieved either from the receiver's SECOM PublicKey interface in 5.7.16 (Figure 45) or from the SECOM PKI GetPublicKey interface in 8.6.3 (Figure 46).

The data server is responsible for deciding when to create a new encryption key.

If the data server has decided to encrypt its data, and the data client responds with an updated data edition, the data client shall also encrypt the responded data.

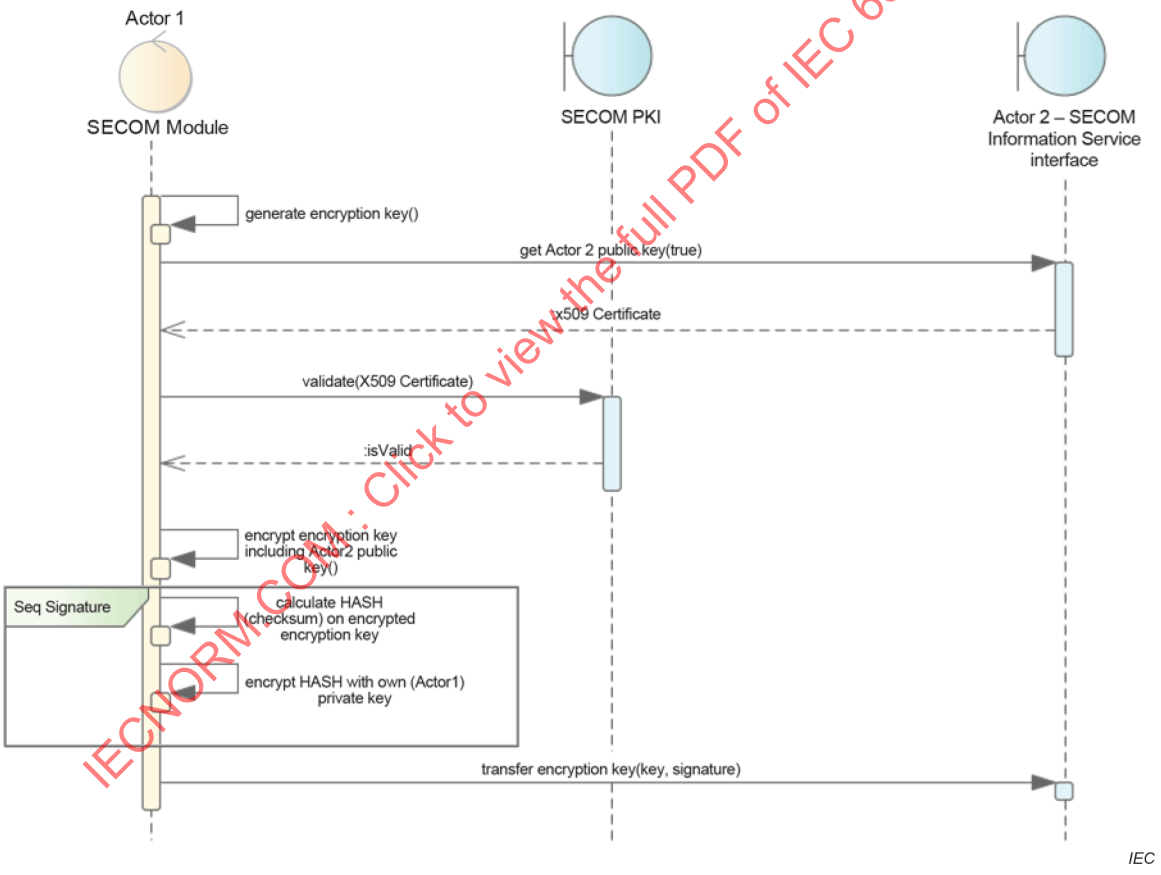


Figure 45 – Sequence for SECOM encryption key management

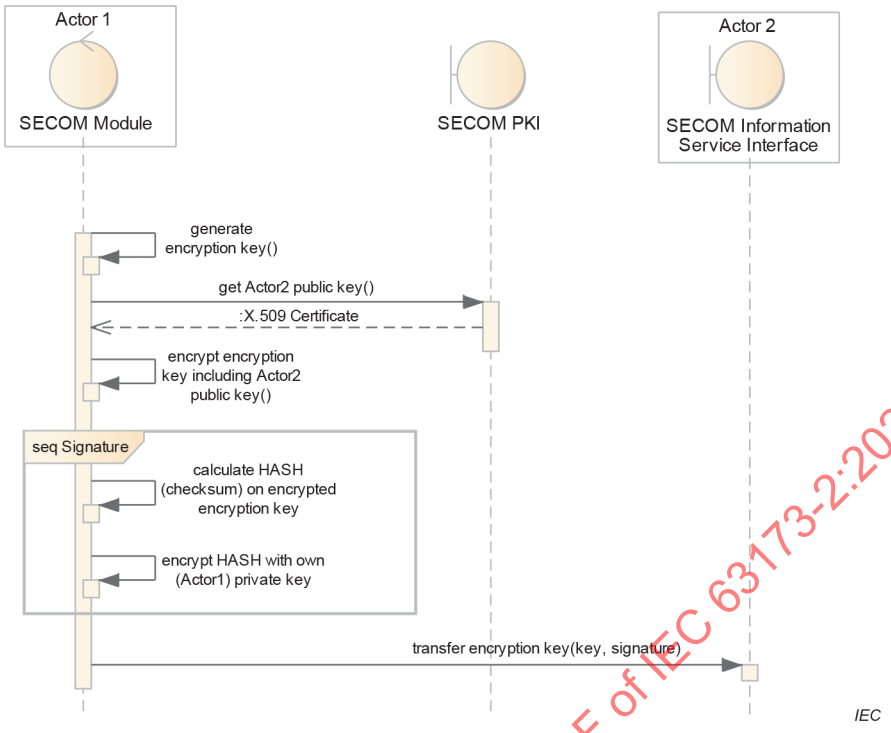


Figure 46 – Alternative sequence for SECOM encryption key management

7.5.3 Generate encryption key

The encryption key shall be generated as a random number that corresponds to the encryption algorithm and key length selected according to 7.4.2.

A random initialization vector (IV) shall be added to be used in the encryption and decryption procedure. The IV shall be generated as a random number and be used onetime only, i.e. it is important to never reuse IVs with the same encryption key. For AES, the size of the IV is always 16 bytes, i.e. equal to the block size described in 7.4.2. The rational for using an IV is to comply with the current encryption AES standard and hence to ensure compatibility with existing commercial frameworks such as .NET and Java where the underlying methods always require an IV.

Before the transfer of the encryption key, it shall be protected (encrypted). The protection of the encryption key shall include use of receiver's public asymmetric key (from SECOM PKI).

Since both receiver and sender use the elliptic curve cryptography (ECC) algorithm in asymmetric keys, pair the keys and encrypt the encryption key with the ECC pair (own private paired with the receiver's public key).

7.5.4 Sign the protected encryption key

Before the transfer, the protected encryption key shall be signed with the sender's private asymmetric key (from SECOM PKI). The signing procedure shall be same procedure as for signing any data as described in 7.3 according to digital signature standard (DSS).

7.5.5 Transfer of the encryption key

The protected encrypted key and the signature shall then be transferred to the receiver using the SECOM information service interface (5.7.15).

The receiver authenticates the signature using the sender's public asymmetric key previously stored, retrieved from the PublicKey interface, see 5.7.16, or retrieved from the SECOM PKI interface GetPublicKey, see 8.6.3. If authentication passes, the receiver shall use its private asymmetric key (from SECOM PKI) together with the sender's public asymmetric key to derive the shared symmetric key using a Diffie-Hellman algorithm (see 7.5.2) and use that derived key to decrypt the encryption key.

7.5.6 Example

An example with openssl is shown in Table 87.

Table 87 – Example of commands

Derive shared key from receiver's Public Key and sender Private Key	`openssl pkeyutl -derive -inkey Actor1PrivateKey.pem -peerkey Actor2PublicKey.pem -out sharedSecret.tmp`
Calculate hash of shared key	`openssl dgst -sha256 -out sharedSecret_hashed.tmp sharedSecret.tmp`

Parse the raw valued derived key	`$temp = (Get-Content -Path .\sharedSecret_hashed.tmp -Raw)`
	
	`$sharedKey = $temp.Substring($temp.IndexOf("=") + 2, 64)`

Encrypt secret key with derived shared key and initialization vector	`openssl enc -aes-256-cbc -nosalt -p -K $sharedKey -iv $iv -e -in encryptionKey.bin -out encryptionKey.enc`

Create digital signature for the encrypted secret key	`openssl dgst -sha256 -sign actor1PrivateKey.pem encryptionKey.enc > encryptionKey.enc.sign`
	
	`xxd -u -ps encryptionKey.enc.sign > encryptionKey.enc.sign.hex`

The signed and encrypted secret key together with the initialization vector shall then be uploaded to the SECOM information service interface of the receiver.

8 SECOM PKI

8.1 General

Clause 8 describes the functionality expected from SECOM PKI and the public interfaces exposed to users for key management.

The main functionality is to provide an interface to access the storage in an identity registry for public keys and identities for authentication purposes. The public keys are signed and wrapped in X.509 certificates (RFC 5280 and RFC 2459) when exchanged, see 8.4. It is possible to check certificates using a standard API for the purpose, OCSP (online certificate status protocol) and CRL (certificate revocation list), see 8.5. It is possible to retrieve public keys for a certain identity from SECOM PKI, and to retrieve identity for a certain public key, see 8.6.3.

A SECOM PKI is governed by a scheme administrator (SA). There can be several instances of SECOM PKI provided by multiple SAs, and an actor can be registered in several instances of SECOM PKI. SECOM compatible equipment are expected to support multiple instances of SECOM PKI.

The private key is generated by the actor and is never exchanged on the Internet. The corresponding public key is uploaded securely to the SECOM PKI for signing and storage, see 8.4. The public key is uploaded to SECOM PKI through a certificate signing request (CSR) as described in 8.2.5 and 8.4. If the signing request is accepted by SECOM PKI, it is signed and the resulting certificate is stored in SECOM PKI and also returned.

SECOM does not define nor constrain the number of identity registries, hence there can be several providers of identity registries. The data protection scheme used is described in the attached exchange metadata object.

SECOM does not specify which identity registry to use. See Annex D.

8.2 Scheme

8.2.1 General

The scheme is aligned with IHO S-100:2018, Part 15.8, Data authentication. The description and terminology of participants in the scheme is aligned with IHO S-100.

There are several types of users of the scheme:

- the scheme administrator (SA), of which there is only one within a SECOM PKI instance;
- the data server (DS), of which there are many;
- the data client (DC), of which there are many;
- the original equipment manufacturer (OEM), of which there are many.

A more detailed explanation of these terms is given in 8.2.2 to 8.2.4.

8.2.2 Scheme administrator

The scheme administrator (SA) is solely responsible for maintaining and coordinating the scheme.

The SA is responsible for controlling membership of the scheme and ensuring that all participants operate according to defined procedures. The SA maintains the top level digital root certificate used to operate the scheme and is the only body that can certify the identity of the other participants of the scheme.

8.2.3 Data servers

Data servers (DS) are responsible for the encryption and digital signing of the datasets in compliance with the procedures and processes defined in the scheme.

8.2.4 Data clients

Data clients (DC) are the end users of datasets and will receive protected information from the data servers to access and use the datasets and services. The data client's software application (OEM system) is responsible for authenticating the digital signatures and decrypting the dataset information in compliance with the procedures defined in the scheme.

8.2.5 Procedure

The following is a list of the steps taken by each participant in the scheme before digital signing of data files.

1) Scheme creation and setup (in the creation of a SECOM PKI)
   - The scheme administrator (SA) creates their own public/private key pair and self-signs it.
   - The SA puts their self-signed public key (also known as their "root certificate") in the public domain.
   - The SA public key ("root certificate" with its thumbprint) is embedded where required in OEM systems ("truststore").
2) Data server setup (where anyone can act as data server, e.g. ship, shore)
   - The data server creates a public and private key pair.
   - The data server signs their public key (with their private key) creating a self signed key (SSK).
   - The data server's SSK is sent to the SA for validation when applying to get the keys signed by a SECOM data protection scheme (also sometimes called a "certificate signing request"). The SA may have its own administrative or technical requirements for acceptance of the data server. Such administrative or technical requirements are outside the scope of SECOM.
3) Data server identity verification
   - If accepted, the SA verifies the data server's SSK and identity.
   - The SA signs the data server's SSK with its own private key to produce an SA signed data server certificate. The data server certificate is stored and bound to the identity of the data server in the identity registry.
   - The data server certificate is then returned to the data server.
   - The data server verifies that the data server certificate authenticates against the SA public key.
4) The data server can then produce digital signatures of data.

8.3 Generation of public and private key

The asymmetric key pair shall be generated according to one of the following algorithm and key-length pairs; RSA ≥ 2 048, DSA ≥ 1 024, EC ≥ 224 and EdDSA ≥ 256.

For IHO S-100:2018, the asymmetric key pair is generated as digital signature algorithm (DSA) keys with 1 024 bits length.

The asymmetric key pair shall be generated by each participant in the scheme.

These are all PEM encoded DSA keys together with their DSA key parameters. See Table 88 for an example of basic commands. The actual commands needed are dependent on the PKI provider's implementation.

Table 88 – Creation of public and private key pairs – Example of basic commands

Task	Command
SA-1 create DSA parameters	`openssl dsaparam 1024 -out dsaparam.txt`
SA-2 create SA root key and self signed root certificate	`openssl req -x509 -sha256 -nodes -days 365 -newkey dsa:dsaparam.txt -keyout iho.key -out iho.crt`
SA-3 sign a verified certificate signing request	`openssl x509 -req -in CSR.csr -sha256 -CA iho.crt -CAkey iho.key -CAcreateserial -out signedicds.crt`

8.4 Certificate signing request

The public key shall be PEM X509v3 format encoded and packaged in PKCS#10 (RFC 2986) certificate signing request.

The certificate signing request is sent securely to SECOM PKI through the defined service interface in 8.6.2.

The public key is signed by the SECOM PKI CA certificate.

8.5 Certificate revocation

8.5.1 General

The administration of certificates occurs outside the SECOM interface and it is the SA that is responsible for update, creation, signing and revocation processes.

There are two main techniques to check revocation status on a certificate: CRL and OCSP.

8.5.2 CRL – Certificate revocation list

CRL in SECOM PKI shall be based upon a standard CRL that is published upon set intervals. Between the set intervals, a delta CRL may be published that always refers back to the last published CRL. This results in a CRL handling that lessens the overall need for large data-packets to be forwarded to ships under way.

The need for end users to subscribe to the CRL is the trade-off from OCSP that should be weighed against the ability to check credentials locally without an OCSP response. The issue of transferring a large single file over low bandwidth transmission could be mitigated by the use of an intermediate delta CRL that handles the CRL.

8.5.3 OCSP – Online certificate status protocol

OCSP main advantage over CRL is the ability to centralize the revocation list and validation of certificates to a single point. However, a centralized certificate verification function increases the need for communication and transmission. If a latency is introduced into verifying a certificate added to the latency of a payload-transfer, the increased time for delivery could be high.

The largest gain in using OCSP is the centralization of revocation and that a revoked certificate is not needed to be added and updated to several local CRL-lists. This also decreases the need of endpoint systems to pull, update and retain a CRL. The decrease in endpoint use is spread across several systems need for transmission and calls/requests for information towards centralized IT-clusters.

8.6 SECOM PKI service interface

8.6.1 General

The security service interface includes the operations shown in Table 89. While the CRL and the OCSP interfaces are open, the remaining interfaces CSR, GetPublicKey and Revoke require authentication and PKI provider specific authorisation.

Table 89 – PKI interface overview

PKI Interface	Exchange Pattern	Definition	Provider of information	Consumer of information
CSR	REQUEST_RESPONSE	Interface to send public keys in a certificate signing request and get them signed by the SECOM PKI provider.	Mandatory	Mandatory
GetPublicKey	REQUEST_RESPONSE	Interface to request a signed Public Key from the Scheme Administrator (SA).	Mandatory	Mandatory
CRL	REQUEST_RESPONSE	Interface to retrieve a Certificate Revocation List (CRL), complete or delta.	Mandatory	Mandatory
OCSP	REQUEST_RESPONSE	Interface to check revocation status of certificates with the Online Certificate Status Protocol (OCSP).	Mandatory	Mandatory
Revoke	ONE_WAY	Interface to revoke a certificate	Mandatory	Mandatory

8.6.2 Service interface – CSR

8.6.2.1 Specification

The purpose of this interface is to send public keys in a certificate signing request and get them signed by the SECOM PKI. The signing request shall be according to PKCS #10 described in RFC 2986.

Figure 47 shows the interface in UML; Table 90 and Table 91 describe the data exchanged in the interface.

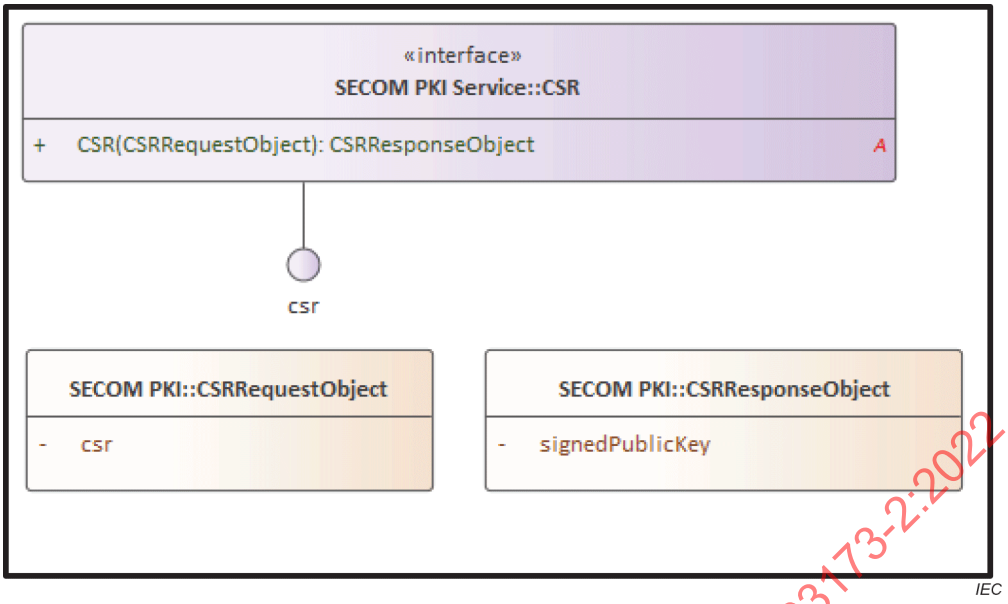


Figure 47 – CSR interface UML diagram

8.6.2.2 Data exchange model

Table 90 – Information input for CSR interface

SignRequestObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
csr	1	Mandatory	string	A PEM encoded PKCS#10 CSR

Table 91 – Information output for CSR interface

SignRequestResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
signedPublicKey	1	Mandatory	string	PEM encoded PKCS#10

8.6.2.3 REST design

8.6.2.3.1 General

The service interface and its data exchange model is defined with REST technology.

8.6.2.3.2 Operation – POST /csr

This operation is used to request signing of public key by the SECOM PKI. The signed public key is sent back in the response.

Table 92 describes the logical parameters in the interface.

Table 92 – REST implementation of CSR

REST Operation			
POST URL/v1/csr{body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
Not applicable (no parameters accepted)			
REST Body (in)	**REST Encoding**	**Mult**	**Definition**
SignRequestObject	text/plain	1	Sign request as string, a PEM encoded PKCS#10 CSR (Certificate Signing Request)
Return (out)	**REST Encoding**	**Mult**	**Definition**
SignRequestResponseObject	application/pem-certificate-chain	1	Confirmation or error message

8.6.2.3.3 Service response

The service shall respond with HTTP codes and message according to Table 93.

Table 93 – HTTP response codes and message in response object

HTTP Code	Message
200	OK
201	Created
401	Unauthorized
403	Not authorized
404	Not found

8.6.2.4 Dynamic behavior

Figure 48 shows the dynamic behavior of a certificate signing request. Actor 1 creates or updates self-signed certificates to its end application which issues a request to the SECOM PKI. The PKI returns a signed public-private key pair.

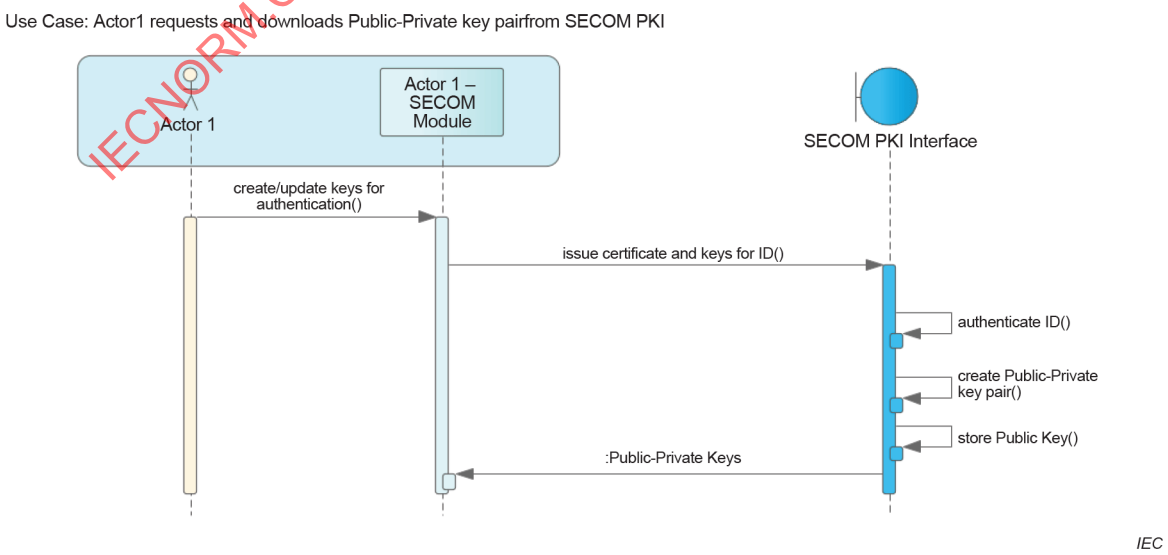


Figure 48 – Operational sequence diagram for CSR

8.6.3 Service interface – GetPublicKey

8.6.3.1 Specification

The purpose with this interface is to facilitate fetching of a public key from a PKI provided by a thumbprint. In order to optimize bandwidth, SECOM interfaces exchange certificate thumbprints instead of the complete certificate(s). The service interface consumer needs a way to retrieve the corresponding public certificate using the exchanged certificate thumbprint. Figure 49 shows the interface in UML.

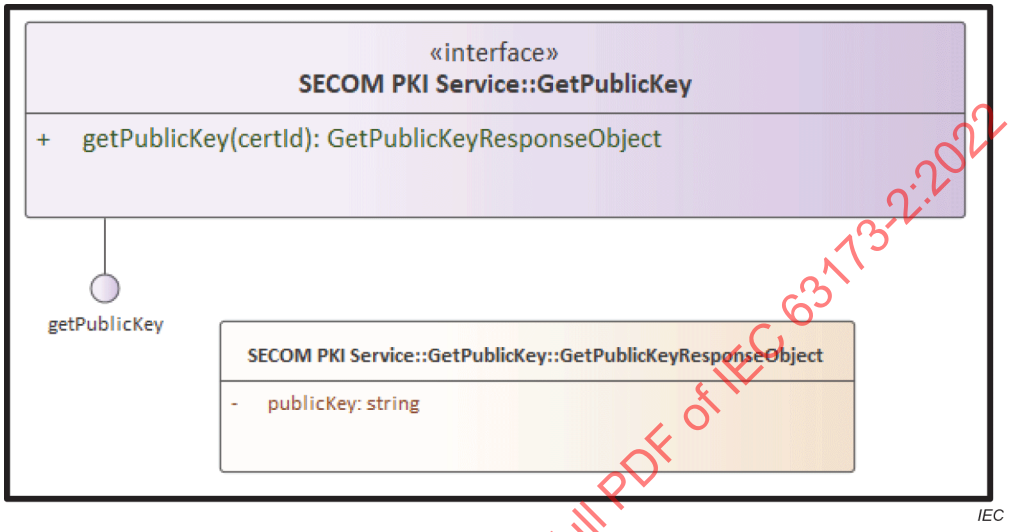


Figure 49 – GetPublicKey interface UML diagram

8.6.3.2 Data exchange model

Information objects in the data exchange are shown in Table 94 and Table 95.

Table 94 – Information input for GetPublicKey interface

GetPublicKeyObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
certThumbprint	0..1	Mandatory	string	Thumbprint of claimed public key, Base64 encoded
certId	0..1	Mandatory	string	The serial number of the certificate given in decimal
identityId	0..1	Mandatory	string	Identifier of the identity related to the certificate, e.g. a MRN urn:mrn:org:ca:env:identity

Table 95 – Information output for GetPublicKey interface

GetPublicKeyResponseObject				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
publicKey	0..1	Mandatory	string	Public Key (X.509 Certificate)

8.6.3.3 REST design

8.6.3.3.1 General

The service interface and its data exchange model is defined with REST technology.

8.6.3.3.2 Operation – GET /publicKey

This operation is used to fetch the full content of a public key from the SECOM PKI. The public key is sent back in the response.

Table 96 describes the logical parameters in the interface.

Table 96 – REST implementation of GetPublicKey interface

REST Operation			
GET URL/v1/publicKey/parameter: return			
REST Parameter (in)	**Encoding**	**Mult**	**Definition**
certThumbprint	string	0..1	Thumbprint of claimed public key, Base64 encoded
certId	string	0..1	The serial number of the certificate given in decimal
identityId	string	0..1	Identifier of the identity related to the certificate, e.g. a MRN urn:mrn:org:ca:env:identity
REST Body (in)	**Encoding**	**Mult**	**Definition**
n/a	n/a	n/a	n/a
Return (out)	**Encoding**	**Mult**	**Definition**
GetPublicKeyResponseObject	application/x-pem-file	1	Binary PEM encoded X.509 Certificate

8.6.3.3.3 Service response

The service shall respond with HTTP codes and message according to Table 97.

Table 97 – HTTP Response codes and message in response object

HTTP Code	Message
200	OK
401	Unauthorized
403	Not authorized
404	Not found

8.6.3.4 Dynamic behavior

Figure 50 shows the dynamic behavior of get request to retrieve a X.509 public certificate.

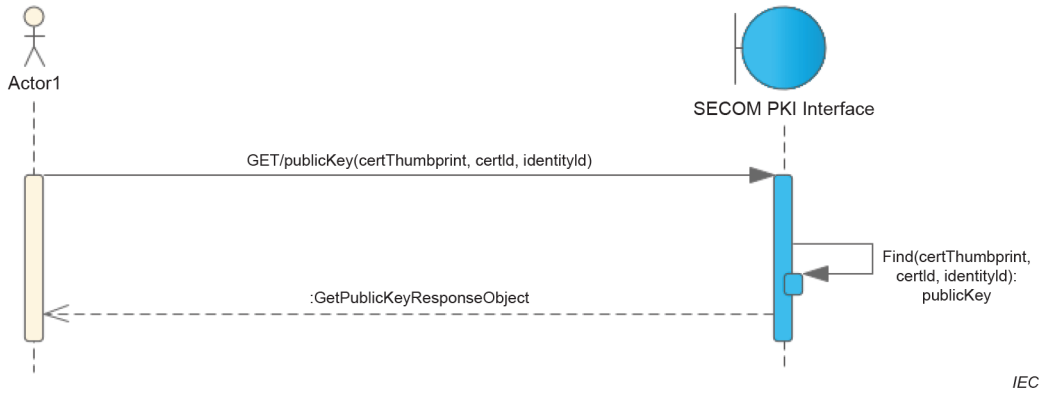


Figure 50 – Operational sequence diagram for GetPublicKey

8.6.4 Service interface – CRL

8.6.4.1 Specification

The purpose with this interface is to retrieve a certificate revocation list. Figure 51 shows the interface in UML.

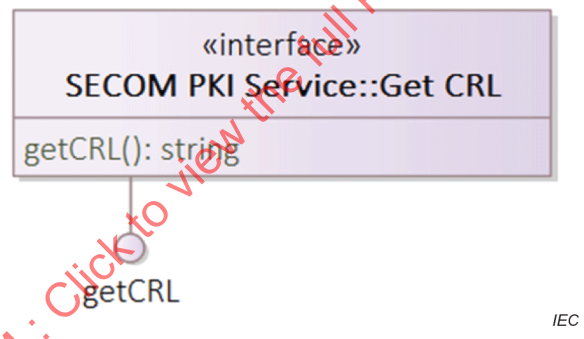


Figure 51 – GetCRL interface UML diagram

8.6.4.2 Data exchange model

The exchanged data in the service interface is described in detail in RFC 5280.

8.6.4.3 REST design

8.6.4.3.1 General

CRL shall be based on RFC 5280.

8.6.4.3.2 Operation – GET /crl

The REST operation crl is described in RFC 5280. Table 98 shows the REST implementation of the CRL interface.

Table 98 – REST implementation of CRL

REST Operation			
GET URL/v1/crl/parameter: return			
REST Parameter (in)	**Encoding**	**Mult**	**Definition**
caAlias	string	1	Certificate Authority alias e.g. a MRN urn:mrn:org:ca:env:identity
REST Body (in)	**Encoding**	**Mult**	**Definition**
n/a	n/a	n/a	n/a
Return (out)	**Encoding**	**Mult**	**Definition**
CRL	application/x-pem-file	1	Binary PEM encoded X.509 CRL

8.6.4.3.3 Service response

The service shall respond with HTTP codes and message according to Table 99.

Table 99 – HTTP response codes and message in response object

HTTP Code	Message
200	OK
401	Unauthorized
403	Not authorized
404	Not found

8.6.4.4 Dynamic behaviour

Figure 52 shows the dynamic behaviour for retrieving a certificate revocation list.

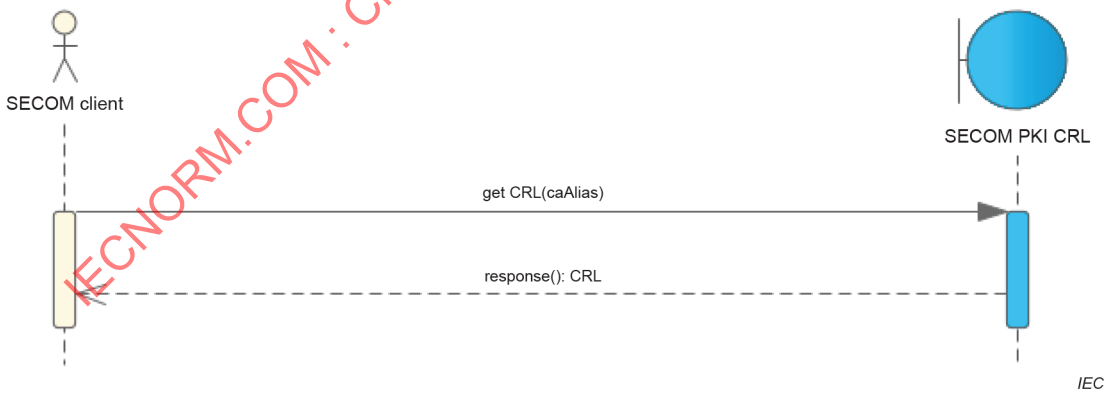


Figure 52 – Operational sequence diagram for CRL

8.6.5 Service interface – OCSP

8.6.5.1 Specification

The purpose with this interface is to check revocation status of certificates with the OCSP protocol. Figure 53 shows the interface in UML.

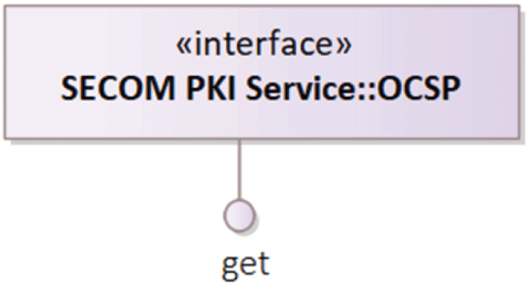


Figure 53 – GetOCSP interface UML diagram

8.6.5.2 Data exchange model

The exchanged data in the service interface is described in detail in RFC 6960.

8.6.5.3 REST design

8.6.5.3.1 General

OCSP shall be based on RFC 6960.

8.6.5.3.2 Operation – GET /ocsp

The REST operations for OCSP is described in RFC 6960. Table 100 shows the REST implementation of the OCSP interface.

Table 100 – REST implementation of OCSP

REST Operation			
GET URL/v1/ocsp/parameter: return			
REST Parameter (in)	**Encoding**	**Mult**	**Definition**
caAlias	string	1	Certificate Authority alias e.g. a MRN urn:mrn:org:ca:env:identity
REST Body (in)	**Encoding**	**Mult**	**Definition**
n/a	n/a	n/a	n/a
Return (out)	**Encoding**	**Mult**	**Definition**
ocsp-response	application/ocsp-response	1	RFC 6960

8.6.5.3.3 Service response

The service shall respond with HTTP codes and message according to Table 101.

Table 101 – HTTP response codes and message in response object

HTTP Code	Message
200	OK
401	Unauthorized
403	Not authorized
404	Not found

8.6.5.3.4 Operation – POST /ocsp

The REST operations for OCSP is described in RFC 6960. Table 102 shows the REST implementation of the OCSP interface.

Table 102 – REST implementation of OCSP

REST Operation			
POST URL/v1/ocsp/{body}: return			
REST Parameter (in)	**Encoding**	**Mult**	**Definition**
n/a	n/a	n/a	n/a
REST Body (in)	**Encoding**	**Mult**	**Definition**
ocsp-request	application/ocsp-request	1	RFC 6960
Return (out)	**Encoding**	**Mult**	**Definition**
ocsp-response	application/ocsp-response	1	RFC 6960

8.6.5.3.5 Service response

The service shall respond with HTTP codes and message according to Table 103.

Table 103 – HTTP response codes and message in response object

HTTP Code	Message
200	OK
201	Created
401	Unauthorized
403	Not authorized
404	Not found

8.6.5.4 Dynamic behaviour

Figure 54 shows the dynamic behaviour of an online check of a certificate's validity.

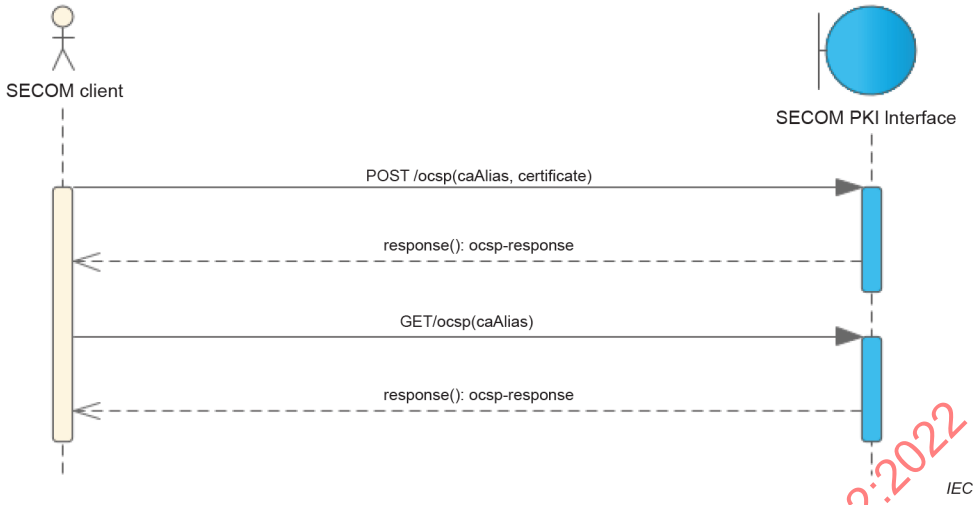


Figure 54 – Operational sequence diagram for OCSP

8.6.6 Service interface – Revoke

8.6.6.1 Specification

The purpose with this interface is to revoke a certificate. Figure 55 shows the interface in UML.

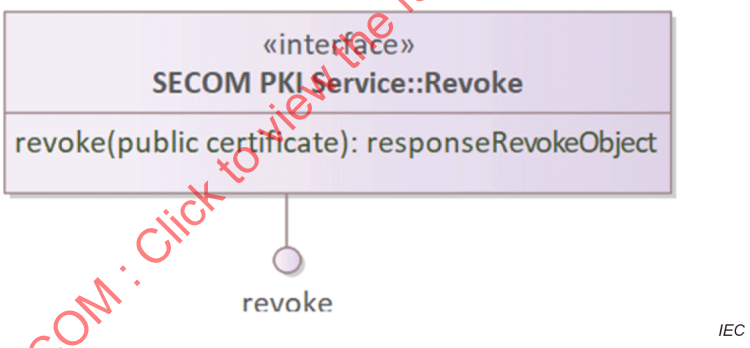


Figure 55 – PostRevoke interface UML diagram

8.6.6.2 Data exchange model

The exchanged data in the service interface is described in detail in Table 104, Table 105 and Table 106.

Table 104 – Information input for Revoke interface

CertificateRevocation				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
RevocationReason	0..1	Mandatory	RevocationReason Enum	The reason the certificate has been revoked
RevokedAt	0..1	Mandatory	DateTime	The date the certificate revocation should be activated

Table 105 – Enumerations for Revoke interface

RevocationReasonEnum		
Index	**Value**	**Description**
1	unspecified	Enum UnspecifiedEnum for unspecified
2	keycompromise	Enum KeycompromiseEnum for keycompromise
3	cacompromise	Enum CacompromiseEnum for cacompromise
4	affiliationchanged	Enum AffiliationchangedEnum for affiliationchanged
5	superseded	Enum SupersededEnum for superseded
6	cessationofoperation	Enum CessationofoperationEnum for cessationofoperation
7	certificatehold	Enum CertificateholdEnum for certificatehold
8	privilegewithdrawn	Enum PrivilegewithdrawnEnum for privilegewithdrawn
9	aacompromise	Enum AacompromiseEnum for aacompromise

Table 106 – Information output for Revoke interface

RevocationResponse				
Attribute	**Mult**	**Processing**	**Type**	**Definition**
Message	0..1	Optional	string	Response message

8.6.6.3 REST design

8.6.6.3.1 General

Revoke shall be based on RFC 6960.

8.6.6.3.2 Operation – POST/revoke

Table 107 shows the REST implementation of the Revoke interface.

Table 107 – REST implementation of Revoke

REST Operation			
POST URL/v1/revoke/parameter {body}: return			
REST Parameter (in)	**Encoding**	**Mult**	**Definition**
certId	application/json	1	The serial number of the certificate given in decimal
REST Body (in)	**Encoding**	**Mult**	**Definition**
CertificateRevocation	application/json	1	Revocation reason and activation time
Return (out)	**Encoding**	**Mult**	**Definition**
RevocationResponse	application/json	1	Response message

8.6.6.3.3 Service response

The service shall respond with HTTP codes and message according to Table 108.

Table 108 – HTTP response codes and message in response object

HTTP Code	Message
200	OK
201	Created
401	Unauthorized
403	Not authorized
404	Not found

8.6.6.4 Dynamic behaviour

Figure 56 shows the dynamic behaviour of a revoke request.

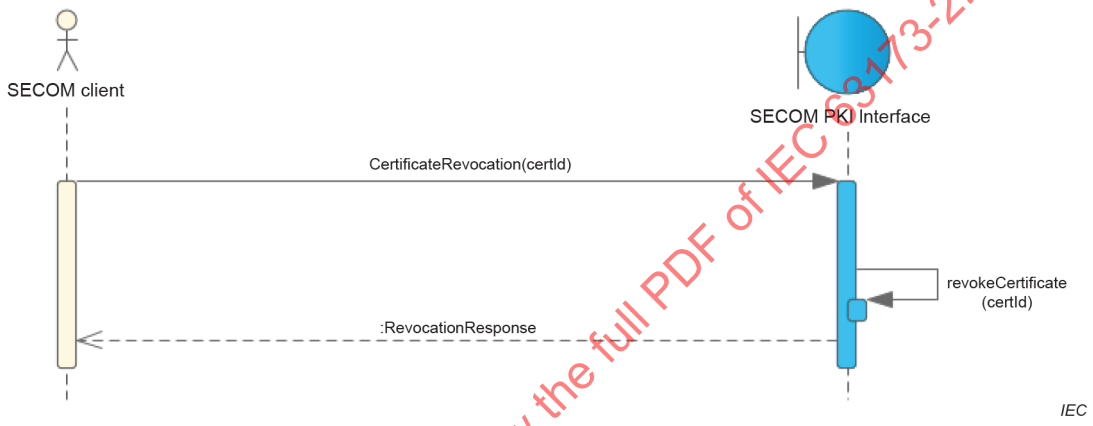


Figure 56 – Operational sequence diagram for Revoke

9 SECOM service discovery service interface

9.1 General

The rationale to standardize the service discoverability is to gain interoperability for searching for the service to consume. To encourage a service oriented approach and to get the desired flexibility and scalability, it is foreseen to be important to define a common service interface to find other services to consume. Services to consume need to be registered in a service registry, thus making the service endpoint searchable. The SECOM service discovery interface is placed in front of a service registry. See Annex B.

SECOM does not define the number of service registries, thus SECOM supports several providers of service registries, each implemented with the defined SECOM service discovery interface.

SECOM does not specify which service registry to use.

9.2 Service interface – Search service

9.2.1 Specification

The purpose of this interface is to search for service instances to consume. Figure 57 shows the information model UML diagram.

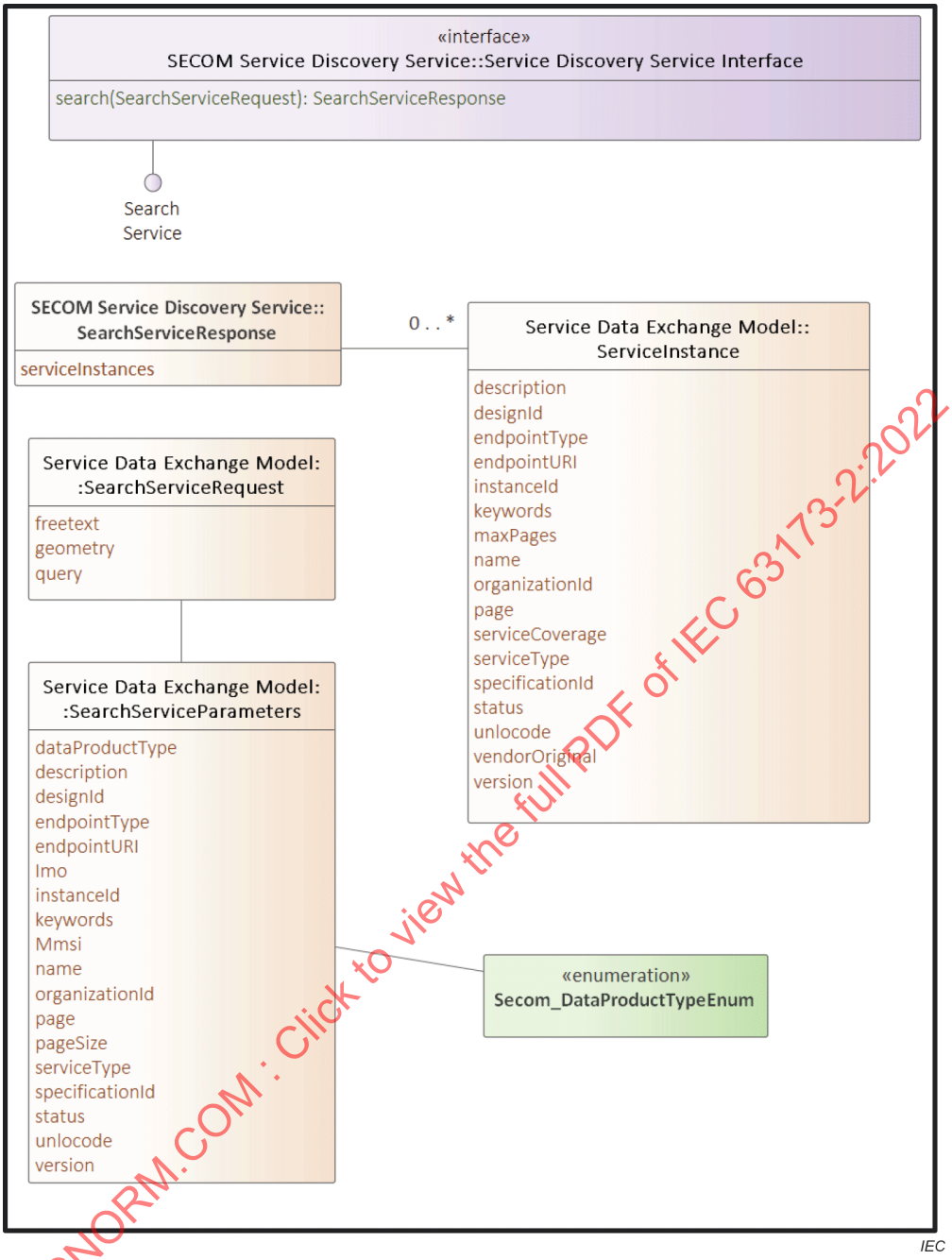


Figure 57 – Search service UML information diagram

9.2.2 Data exchange model

The operation shall be used for searching for service instance(s) to consume based on provided parameters. This interface caters for searching a service registry by combining query parameters separated by AND/OR together with possible geometry search criteria and free text; an example is provided in Clause D.7.

Table 109 describes the logical parameters in the interface, Table 110 describes the available search parameters for the query and Table 111 describes the output response.

Table 109 – Information input for search service interface

SearchFilterObject				
Attribute	**Type**	**Mult**	**Processing**	**Definition**
query	SearchParameters	0..1	Mandatory	Search query based on parameters in SearchParameters, Table 110
geometry	string	0..1	Mandatory	WKT geometry, see Table 12 for example a route plan as linestring
freetext	string	0..1	Mandatory	Free text search

Table 110 – Information input for search parameter object

SearchParameters				
Attribute	**Type**	**Mult**	**Processing**	**Definition**
name	string	0..1	Mandatory	From S-100 Service Model
status	string	0..1	Mandatory	From S-100 Service Model
version	string	0..1	Mandatory	From S-100 Service Model
keywords	string	0..1	Mandatory	From S-100 Service Model
description	string	0..1	Mandatory	From S-100 Service Model
dataProductType	SECOM_DataProductType	0..1	Mandatory	Name column from Table 8
specificationId	MRN	0..1	Mandatory	From IALA Service Model G1143
designId	MRN	0..1	Mandatory	From IALA Service Model G1143
instanceId	MRN	0..1	Mandatory	From IALA Service Model G1143
Mmsi	string	0..1	Mandatory	From IALA Service Model G1128
Imo	string	0..1	Mandatory	From IALA Service Model G1128
serviceType	string	0..1	Mandatory	From IALA Service Model G1128
unlocode	string	0..1	Mandatory	Code of defined object, see 5.6.13
endpointUri	URI	0..1	Mandatory	From IALA Service Model G1128
page	PositiveInteger	0..1	Mandatory	Requested pagination page
pageSize	PositiveInteger	0..1	Mandatory	Requested pagination page size

Table 111 – Information output for search service interface

ResponseSearchObject				
Attribute	**Type**	**Mult**	**Processing**	**Defintion**
searchServiceResult	SearchObjectResult	0..*	Optional	List of type SearchObjectResult
SearchObjectResult				
Attribute	**Type**	**Mult**	**Processing**	**Defintion**
instanceId	string	1	Optional	MRN Service instance identity (refer IALA G1143)
version	string	1	Optional	S100_OC_ServiceMetaData, version of the service instance
name	string	1	Optional	S100_OC_ServiceMetaData, name on the service instance
status	string	1	Optional	S100_OC_StatusType, status on the service
description	string	1	Optional	S100_OC_ServiceMetaData , description of the service instance
dataProductType	SECOM_DataProductType	0..1	Mandatory	Name column from Table 8
organizationId	string	1	Optional	MRN, organization identity of the registrator (refer IALA G1143)
endpointUri	string	1	Optional	URI, endpoint to the service instance
endpointType	string	1	Optional	S100_OC_ServiceTechnology i.e. REST
keywords	string	0..1	Optional	S100_OC_ServiceMetaData, a list of keywords associated with the service Searchable keywords for discoverability.
unlocode	string	0..1	Optional	Code of defined object, see 5.6.13
instanceAsXml	string	0..1	Optional	Original XML based on service registry used. (refer IALA G1128)

9.2.3 REST design

9.2.3.1 General

The service interface and its data exchange model is defined with REST technology.

9.2.3.2 Operation – POST /searchService

This operation searches for service. If no parameters given, a complete list of services is given in result.

Table 112 describes the logical parameters in the interface.

Table 112 – REST implementation for Search Service

REST operation			
POST URL/v1/searchService {body}: return			
REST Parameter (in)	**REST Encoding**	**Mult**	**Definition**
No parameters defined	n/a	n/a	n/a
REST Body (in)	**Encoding**	**Mult**	**Definition**
SearchServiceObject	application/json	1	Search object
Return (out)	**Encoding**	**Mult**	**Definition**
ResponseSearchObject	application/json	0..*	list of services

9.2.3.3 Service response

Table 113 describes the service response.

Table 113 – HTTP response codes

HTTP Code	Message
200	Array of FindServiceResponseObject
400	Bad request
404	Information not found

10 SECOM error cases

10.1 Error cases

In this Clause 10, possible error cases are outlined. Although SECOM is designed for S-100 based products, SECOM is technically payload agnostic and applicable also for other types of data. As a consequence, error cases are not focused on invalid test data, since this is covered in end-user application and thus not in scope for this document.

Instead, error cases are described referring to SECOM test objectives as outlined below.

- Message integrity
- Data integrity
- Transport confidentiality
- Data protection
- Service identity
- Client identity
- Client authorization
- Bandwidth optimization
- Large message transfer
- Closed loop communication
- Service discoverability
- Information push
- Information pull
- Subscribe to data

- Service information
- Service condition

10.2 General

In Table 11, HTTP response codes common to all interfaces are listed. For interfaces using the SECOM_ResponseCodeEnum in Table 9, the corresponding column value is set to null. Below are some examples of errors resulting in such common HTTP response code.

- For unauthenticated requests from a non SECOM authenticated requester, HTTP response code 401 "Unauthorized" is the expected response.
- SECOM interface REST method (POST/ GET) not supported for an implemented interface returns HTTP response code 405 "Method not allowed".
- Erroneous requests regarding wrong endpoint, wrong request format and/or erroneous supplied mandatory parameters results in 500 "Internal server error".
- If an interface in a SECOM service instance is not implemented, HTTP response code 501 "Not implemented" is the expected response.

10.3 Message integrity

The intention of message integrity is to ensure that the complete message is unchanged between SECOM services, i.e. not including last mile from vendor API to end user application. Envelope signing is the proposed solution to cater for this in SECOM. The envelope shall be signed in the sending instance and verified in the receiving instance according to 7.3.4 and 7.3.6.

Errors might be attributable to parsing attributes in the message, i.e incorrect datatype conversion and/or wrong attribute ordering, usage of incorrect private key, incorrect/revoked public certificate and possible differences in key-length for the DSA algorithm, see 7.3.3.

In case of such errors, HTTP response code 400 in combination with SECOM_ResponseCodeEnum 1 – "Failed signature verification" or 2 – "Invalid certificate" are expected.

10.4 Data integrity

Data integrity ensures the data transmitted is accurate and consistent end to end. This implies that verification of payload signature shall be performed in the end user application. However, if the sending end user application signs the data with a SECOM PKI key, it is possible to check data integrity already in the receiving SECOM service instance, thus eliminating further transfer of incorrectly signed data to the end user application.

The payload shall be signed in the sending end user application, then verified in the receiving service instance and/or verified in the receiving end user application, according to 7.3.3 and 7.3.5.

Errors related to failure of verification of data signatures are related to the following HTTP response codes for the interfaces Upload, Upload Link and Acknowledgement:

- HTTP response code 400 "Bad request" together with SECOM_ResponseCode = 1 "Failed signature verification";
- HTTP response code 400 "Bad request" together with SECOM_ResponseCode = 2 "Invalid certificate".

10.5 Transport confidentiality

Transport confidentiality ensures communication channel protection between SECOM service instances. Channel encryption in SECOM is realized using TLS according to 6.2. Errors related to the deployment environment shall follow RFC 7231 HTTP1/1.

HTTP response code 401 "Unauthorized" is expected if the presented service instance certificate in the TLS session fails authentication. This can be sent from either the service acting as a client and the server, since SECOM uses mutual authentication.

10.6 Data protection

Data protection ensures that information is confidential except for the intended recipient. In the SECOM implementation, AES is used for this purpose as described in 7.4.2. SECOM service interface EncryptionKey is used for exchange of the temporary secret key for decryption of encrypted data. The SECOM PublicKey interface facilitates request of a public key for data encryption in end user application.

In the EncryptionKey request operation, HTTP response code 403 "Not authorized" is expected if the requesting service instance identity is unauthorized according to the provided service instance certificate.

Error cases for the EncryptionKey upload operation are outlined below.

- HTTP response code 400 "Bad request" is expected if provided attributes in EncryptionKeyObject are incorrect and/or malformed.
- HTTP response code 400 "Error" with SECOM_ResponseCode = 0 "Missing required data for the service" is expected if mandatory provided attributes in EncryptionKeyObject are missing.
- HTTP response code 400 "Error" with SECOM_ResponseCode = 1 "Failed signature verification" is expected if the envelopeSignature cannot be verified.
- HTTP response code 400 "Error" with SECOM_ResponseCode = 2 "Invalid certificate" is expected if the provided envelopeSignatureCertificate is invalid.

Errors related to the PublicKey interface are listed below:

- HTTP response code 400 "Bad request" is expected if provided attributes in PublicKeyFilterObject are missing, incorrect and/or malformed.
- HTTP response code 403 "Not authorized to requested information" is expected if the requesting service instance identity based on provided service instance certificate is unauthorized to requested publicCertificate as indicated in PublicKeyFilterObject.
- HTTP response code 404 "Not found" is expected if requested publicCertificate based on provided certificateThumbprint in PublicKeyFilterObject cannot be found.

10.7 Service identity

SECOM service identification is based on the corresponding identity found in the SECOM service instance certificate from the SECOM PKI. SECOM interfaces Upload, Upload Link, Acknowledgement, Access, Get, Get Summary and Get By Link relies on SECOM service instance identification. Errors related to the Service instance identity are implicitly handled in validation of the provided service certificate.

10.8 Client identity

Client identification is realized in SECOM by usage of end user client certificate authentication (X.509 PKI). Note that identification is only possible if SECOM PKI is used for end user certificates. Client identification is performed in SECOM interfaces Acknowledgement, Access, Get, Get Summary and Get By Link. Errors related to the client identity are handled in validation of the provided client certificate.

10.9 Client authorization

Facilitation of client access to information in SECOM is based on corresponding service instance authorization, since it cannot be assumed that the client, i.e. end user application, is registered in SECOM PKI. For authorization purposes, SECOM implements interfaces Access and Access Notification.

The following error cases are possible for the Access interface:

- HTTP response code 400 "Bad request" is expected if provided attributes in AccessRequestObject are missing, incorrect and/or malformed.
- HTTP response code 403 "Not authorized to requested information" is expected if the requesting service instance identity lacks access to desired information, i.e the specified combination of attributes containerType, dataProductType, dataReference and/or productVersion.

The following error cases are possible for the Access Notification interface:

- HTTP response code 400 "Bad request" is expected if provided attributes in AccessNotificationObject are missing, incorrect and/or malformed.

10.10 Bandwidth optimization

Minimize size of data package sent to reduce required bandwidth is facilitated in SECOM by use of data compression. Errors attributable to compression relates to interfaces Upload and Get and is informed to the receiving end user through the Acknowledgement interface.

In the Acknowledgment interface, the end user application is expected to return attribute nackType with nackTypeEnum = 4 "Failed decompression" when experiencing decompression failures.

10.11 Large message transfer

Large message transfer, i.e. message sizes exceeding 350 kB, is implemented in SECOM by providing a link to data which is facilitated by interfaces Upload Link and Get By Link.

Errors in the Upload Link interface are attributable to the following.

- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = null is expected if provided attributes in UploadLinkObject are incorrect and/or malformed.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = 0 "Missing required data for the service" is expected if provided mandatory attributes in UploadLinkObject are missing.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = null is expected if provided attributes in AcknowledgementObject are incorrect and/or malformed.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = null is expected if provided attributes in AcknowledgementObject are incorrect and/or malformed.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = 1 "Failed signature verification" is expected if envelopeSignature attribute fails to verify according description in 7.3.6.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = 2 "Certificate failure" is expected if envelopeSignatureCertificate is found not to be valid.
- HTTP response code 403 "Not authorized to upload link" with SECOM_ResponseCodeEnum = null is expected if client authorization fails, which is based on provided client certificate in the corresponding upload message.

Errors in the Get By Link interface are attributable to the following:

- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = null is expected if provided attributes in GetByLinkObject are incorrect and/or malformed.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = 2 "Certificate failure" is expected if provided client certificate cannot be authenticated in a SECOM PKI compliant identity register.
- HTTP response code 403 "Not authorized to requested information" with SECOM_ResponseCodeEnum = null is expected if client authorization fails, which is based on provided client certificate.
- HTTP response code 404 "Information with <transactionIdentifier> not found" is expected upon requesting information with the wrong transactionIdentifier.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = 4 "Link expired" is expected when a Get By Link request is made after the time specified in timeToLive parameter.

10.12 Closed loop communication

Notification of message received, read etc. to ensure consistent dialogue between end-user applications is realized in SECOM by using the Acknowledgement interface which can be used in combination with upload and download of information. Acknowledgement can be requested upon delivery to the vendor API and/or when the message is opened in the end-user application.

The following error cases are possible for the Acknowledgement interface irrespective of requested ackType.

- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = null is expected if provided attributes in AcknowledgementObject are incorrect and/or malformed.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = 0 "Missing required data for the service" is expected if provided mandatory attributes in AcknowledgementObject are missing.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = 1 "Failed signature verification" is expected if digitalSignature attribute fails to verify according to description in 7.3.5.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCodeEnum = 2 "Invalid certificate" is expected if envelopeCertificate is found not to be valid.
- HTTP response code 403 "Not authorized to upload ACK" with SECOM_ResponseCodeEnum = null is expected if client authorization fails, which is based on provided client certificate in the corresponding upload message, see 10.9.

Additionally, in case of a requested ackType = "Opened", the following error cases are also possible as end-user application response.

- NackTypeEnum = 0 "XML Schema validation error" is returned if previously uploaded/downloaded data fails schema validation in the end user application according to stated containerType and dataProductType provided with the uploaded message.
- NackTypeEnum = 1 "Unknown data type or version" is returned when previously uploaded/downloaded message contains an unknown data type or version, i.e. not in the enumerated list of allowed dataProductType.
- NackTypeEnum = 2 "Failed data signature verification" is returned when the digital signature for the payload cannot be verified as described in 7.3.5.
- NackTypeEnum = 3 "Failed decryption" is returned when the payload cannot be decrypted using the received encryption key from the EncryptionKey interface.
- NackTypeEnum = 4 "Failed decompression" is returned when the payload cannot be decompressed using a compliant tool for unzipping the received compressed data.

10.13 Service discoverability

Search for services by means of service metadata in order to locate relevant services for consumption is implemented in the SECOM Search Service interface. Expected errors are related to search results when querying the service registry and/ or incorrect attribute format specified in SearchFilterObject.

HTTP response code 400 "Bad request" is expected if provided attributes in SearchFilterObject are incorrect.

Non-existent search results generates HTTP response code 404 "Information not found".

10.14 Information push

Information is shared by uploading data to a service which implements service interfaces to push information, i.e. Upload interface and Upload Link.

Possible errors emanating from the Upload interface are specified below:

- HTTP response code 400 "Bad request" together with SECOM_ResponseCode = 0 "Missing required data for the service" is expected if provided mandatory attributes in UploadObject are missing.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCode = 1 "Failed signature verification" is expected if envelopeSignature attribute fails to verify according description in 7.3.6.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCode = 2 "Invalid certificate" is expected if envelopeSignatureCertificate is found not to be valid.
- HTTP response code 400 "Bad request" together with SECOM_ResponseCode = 3 "Schema validation error" is expected if the provided payload is found not to be valid according to related schema. This validation is only possible for non-encrypted payload.
- HTTP response code 403 "Forbidden" together with SECOM_ResponseCode = null is expected if the requester is not authorized to Upload information to the service instance.
- HTTP response code 413 "Request entity too large" together with SECOM_ResponseCode = null is expected if the uploaded message size exceeds 350 kB.

Errors in the Upload Link interface are outlined in 10.11.

10.15 Information pull

Functions for retrieval of information by downloading data from a service are realized in SECOM service interfaces, Get, Get Summary and Get by link.

In the Get interface, the following error cases possible.

- HTTP response code 400 "Bad request" is expected if provided attributes in GetFilterObject are incorrect and/or malformed.
- HTTP response code 403 "Not authorized to requested information" is expected if the requester, i.e. corresponding identity for SECOM service instance, is not authorized to resulting information according to attributes provided in the GetFilterObject.
- HTTP response code 404 "Information with <dataReference> not found" is expected when non-existent information is requested according to attributes in the GetFilterObject.

Error cases in Get Summary interface are listed below.

- HTTP response code 400 "Bad request" is expected if provided attributes in GetSummaryFilterObject are incorrect and/or malformed.
- HTTP response code 403 "Not authorized to requested information" is expected if the requester, i.e. corresponding identity for SECOM service instance, is not authorized to resulting information according to attributes provided in the GetSummaryFilterObject.
- HTTP response code 404 "Information not found" is expected when non-existent information is requested according to attributes in the GetSummaryFilterObject.

Possible errors in the Get By Link are described under 10.11.

10.16 Subscribe to data

Subscribe to information to receive subsequent updates is implemented in SECOM service interfaces for Subscription request, remove and notify.

Errors in a subscription request are related to whether the requester is authorized to required data or not, in which case HTTP response code 403 "Not authorized to requested information" is returned. HTTP response code 400 "Bad request" is expected in the case information for a supplied attribute in SubscriptionRequestObject is missing and/or malformed.

Following the subscription request, a subscription notification is sent to the requester. HTTP response code 400 "Bad Request" is expected in the case a notification is sent for a non-existing subscriptionIdentifier, supplied parameters are malformed and/or missing.

Errors in the removal subscription interface used by the consumer to request removal of subscription are attributable to either of the following reasons.

- Request for a removal of a subscription with a non-existent subscriptionIdentifier results in HTTP response code 404 "Subscriber identifier not found". The same response code is expected if the requester does not exist as a subscriber.
- The requester is not authorized to remove the referenced subscription results in 403 "Not authorized to remove subscription".
- HTTP response code 400 "Bad Request" is expected in the case a removal is sent for a non-existing subscriptionIdentifier, supplied parameter is malformed and/or missing.

10.17 Service information

In order to facilitate information regarding service accepted payloads, payload versions, formats and endpoints, SECOM implements a service capability interface. Possible errors are the following.

- Consequential errors in the requesting service emanating from response CapabilityResponseObject attributes missing or not compliant with definitions referenced in Table 65.

10.18 Service condition

This is to cater for a contextual service status to check service operation. In SECOM, the Ping interface is used for checking the last interaction time with the related end-user application and SECOM service status in a ping response. Errors might be attributable to wrong endpoint registered in the service registry. Other errors are indicated by usage of common HTTP response codes in 5.6.10.

11 Test methods and expected results

11.1 General

A simulator or test arrangements with the following characteristics is required:

- capable of importing and exporting SECOM compliant data;
- capable of examining content of SECOM compliant data;
- capable of editing the imported and exported SECOM compliant data if necessary.

Test material is organized in folders and is provided in machine readable form at http://cirm.org/secom.

A simulator or test arrangements for SECOM PKI with the following characteristics is required:

- at least one scheme administrator which includes a private/public key pair and contains self-signed public key;
- at least one data server which generates public and private key pair and produces digital signature of the data;
- at least one data client which generates its public and private key pair and produces digital signature of the data;
- manufacturer document specifying the encryption algorithm for the data and means to encrypt and to decrypt the message if provided;
- manufacturer document specifying the encryption algorithm for the digital signature and means to generate and to verify the signature;
- instance is required to have a valid trusted SSL certificate.

Test methods are separated into four different EUT's:

- ship/shore equipment;
- SECOM Information service equipment;
- SECOM PKI equipment;
- service discovery equipment.

See descriptions in Annex F.

11.2 Communication channel security test

This test is mandatory for all other tests in this Clause 11.

Confirm by the inspection of documented evidence that the EUT provides use of Transport Layer Security, TLS version 1.2 (RFC-5246) and TLS version 1.3 (RFC-8446).

Confirm by the inspection of documented evidence that the EUT provides HTTP over TLS as specified in RFC-2818.

Confirm by the inspection of documented evidence that the EUT provides HTTP/1.1 as specified in RFC-7231.

Confirm by analytical evaluation that the EUT supports service authentication based on X.509 certificates and supports the revoke which checks against the Certificate Revocation List (CRL) or OCSP.

11.3 Data protection test

11.3.1 Data Compression and packaging

Confirm by analytical evaluation that the compression uses ZIP algorithm and DEFLATE methods.

11.3.2 Data authentication and signature

Confirm by analytical evaluation that the EUT creates a digital signature using the DSA algorithm and its private key and that the signature is stored as an ASN.1 sequence of 2 elements in the digitalSignature of DigitalSignatureValueObject if provided.

11.3.3 Encryption

Confirm by analytical evaluation or by the inspection of the documented evidence that the EUT supports the IHO S-100:2018, Part 15-6, Data encryption algorithm.

When each message is encrypted, the message shall be verified based on the proper encryption algorithm before it is processed. This test applies to all encrypted SECOM messages.

Confirm by analytical evaluation that the message is encrypted or decrypted with AES algorithm as specified in 7.4.2.

11.3.4 Digital signature test

When a message includes a digital signature in an Envelope or ExhangeMetadata object, the following test shall be applied.

Confirm by analytical evaluation that the digital signature is generated or verified correctly as specified in 7.3.3, 7.3.4, 7.3.5 and 7.3.6.

11.4 SECOM ship/shore test

11.4.1 General

This test describes message exchange between two imaginary end applications divided into data push (upload) and pull (download). For each test method, there is a corresponding set of test data available. The test data is organized into separate zip archives reflecting whether the data exchange is unprotected (open.zip), protected (protected.zip) or size optimized (compressed.zip). The auth.zip archive includes the client/server, intermediate and root certificates used. Inside each archive, the payload json file, for example uploadObject.json, getSummaryResponse.json etc., can be found and used when executing the tests. See Table 114.

Table 114 – Test data reference

Upload test		
File	**Description**	**Defintion**
upload\open\UploadObject.json	Upload request object	Table 16
upload\open\ EnvelopeUploadObject.txt	Upload envelope conversion for signature creation	7.3.4
upload\open\Ack_technical.json	Acknowledgement request, delivered ack	Table 24
upload\open\Ack_operational.json	Acknowledgement request, opened ack	Table 24
upload\open\EnvelopeAckObject.txt	Acknowledgement envelope conversion for signature creation	7.3.4
upload\compressed\UploadObject.json	Upload request object compressed	Table 16
upload\compressed\ SignatureEnvelopeUploadObject.txt	Upload envelope conversion for signature creation	7.3.4
upload\protected\ UploadObjectRequest.json	Upload request object protected	Table 16
upload\protected\ SignatureEnvelopeUploadObject.txt	Upload envelope conversion for signature creation	7.3.4
upload\protected\ EncryptionKeyRequest.json	EncryptionKey request object	Table 72
upload\protected\ SignatureEnvelopeKeyObject.txt	EncryptionKey envelope conversion for signature creation	7.3.4
upload\protected\ Ack_technicalRequest.json	Acknowledgement request, delivered ack	Table 24
upload\protected\ Ack_operationalRequest.json	Acknowledgement request, opened ack	Table 24
upload\protected\ SignatureEnvelopeAckObject.txt	Acknowledgement envelope conversion for signature creation	7.3.4
upload\link\open\UploadLinkObjRequest.json	UploadLink request object	Table 20
upload\link\open\SignatureEnvelopeLinkObject.txt	UploadLink envelope conversion for signature creation	7.3.4
upload\link\open\Ack_technicalRequest.json	Acknowledgement request, delivered ack	Table 24
upload\link\open\Ack_operationalRequest.json	Acknowledgement request, opened ack	Table 24
upload\link\open\SignatureEnvelopeAckObject.txt	Acknowledgement envelope conversion for signature creation	7.3.4
upload\link\compressed\UploadLinkObjRequest.json	UploadLink request object compressed	Table 20
upload\link\compressed\SignatureEnvelopeLinkObject.txt	UploadLink envelope conversion for signature creation	7.3.4
upload\link\protected\ UploadObjectRequest.json	UploadLink request object protected	Table 20
upload\link\protected\ SignatureEnvelopeUploadObject.txt	UploadLink envelope conversion for signature creation	7.3.4
upload\link\protected\ EncryptionKeyRequest.json	EncryptionKey request object	Table 72
upload\link\protected\ SignatureEnvelopeKeyObject.txt	EncryptionKey envelope conversion for signature creation	7.3.4
upload\dataSet.s421.xml	Payload data used in tests including transfer of data	Table 8

Download test		
File	**Description**	**Defintion**
download\dataset.s421_a.xml	Published information dataset a	Table 8
download\dataset.s421_b.xml	Published information dataset b	Table 8
download\dataset.s421_c.xml	Published information dataset c	Table 8
download\open\GetSummaryQuery.txt	GetSummary request	Table 35
download\open\GetSummaryResponse.json	GetSummary response object	Table 36
download\open\GetQuery-2a41c736-9004-4622-a170-1e9b3764de82.txt	Get request query	Table 31
download\open\GetResponse-2a41c736-9004-4622-a170-1e9b3764de82.json	Get response object	Table 32
download\open\GetQuery-cb0c1b4d-1348-4d68-b037-ca2eb1569512.txt	Get request	Table 31
download\open\GetResponse-cb0c1b4d-1348-4d68-b037-ca2eb1569512.json	Get response object	Table 32
download\open\GetQuery-ea7c7ac3-4a18-499e-aefd-c587fa3f399d.txt	Get request	Table 31
download\open\GetResponse-ea7c7ac3-4a18-499e-aefd-c587fa3f399d.json	Get response object	Table 32
download\compressed\GetSummaryQuery.txt	GetSummary request	Table 35
download\compressed\GetSummaryResponse.json	GetSummary response object	Table 36
download\compressed\GetQuery-5d43d57d-b07d-4ff9-aece-d60ab622af14.txt	Get request	Table 31
download\compressed\GetResponse-5d43d57d-b07d-4ff9-aece-d60ab622af14.json	Get response object	Table 32
download\compressed\GetQuery-21b801f8-6b64-469e-a2ab-bb57a7b38602.txt	Get request	Table 31
download\compressed\GetResponse-21b801f8-6b64-469e-a2ab-bb57a7b38602.json	Get response object	Table 32
download\compressed\GetQuery-ecd5ad61-5e7b-4c08-bad3-f0ca93762c38.txt	Get request	Table 31
download\compressed\GetResponse-ecd5ad61-5e7b-4c08-bad3-f0ca93762c38.json	Get response object	Table 32
download\protected\GetSummaryQuery.txt	GetSummary request	Table 35
download\protected\GetSummaryResponse.json	GetSummary response object	Table 36
download\protected\EncryptionKeyRequest.json	Get EncryptionKey request object	Table 74
download\protected\SignatureEnvelopeKeyRequestObject.txt	EncryptionKey envelope conversion for signature creation	7.3.4
download\protected\EncryptionKeyRequest.json	EncryptionKey request object	Table 72
download\protected\SignatureEnvelopeKeyObject.txt	EncryptionKey envelope conversion for signature creation	7.3.4
download\protected\GetQuery.txt	Get request	Table 31
download\protected\GetResponse.json	Get response object	Table 32
download\openedAck\GetSummaryQuery.txt	GetSummary request	Table 35
download\openedAck\GetSummaryResponse.json	GetSummary response object	Table 36
download\openedAck\GetQuery,txt	Get request	Table 31
download\openedAck\GetResponse.json	Get response object	Table 32
download\openedAck\Ack_operationalRequest.json	Acknowledgement request, opened ack	Table 24
download\openedAck\Signature-EnvelopeAckObject.txt	Acknowledgement envelope conversion for signature creation	7.3.4

Common		
File	**Description**	**Defintion**
auth\environment.json	Example of environment setup for two SECOM information service instances	n/a
auth\Keystore_secomtest01.p12	Server certificate for SECOM instance #1	n/a
auth\Keystore_secomtest02.p12	Server certificate for SECOM instance #2	n/a
auth\SECOM_IdReg.crt	Intermediate certificate from PKI provider	n/a
auth\SECOM_Root.crt	Root certificate from PKI provider	n/a

11.4.2 Prerequisites SECOM ship/shore EUT

Ship/Shore equipment according to Clause F.3.

The following are prerequisites for a ship/shore EUT to be able to perform tests:

- means for validating X.509 certificate;
- means for verifying digital signatures according to 7.3.5 and 7.3.6;
- means for creating digital signatures according to 7.3.3 and 7.3.4;
- means for data protection according to 7.5;
- means for data compression according to 7.2.

11.4.3 Upload data

11.4.3.1 Actors

Ship/shore sender EUT – Ship/shore receiver EUT

11.4.3.2 Testdata

The related folders are:

- upload\open\
- upload\compressed\
- upload\protected\
- upload\link\open\
- upload\link\compressed\
- upload\link\protected\
- auth.zip

11.4.3.3 Method

Table 115 contains the test execution steps.

Table 115 – Upload test method steps

Step	Applicable for sender EUT	Applicable for receiver EUT
1	Use the simulator or test arrangements as the EUT to export an UploadObject message as a single message. Confirm by observation that the data attribute in Table 16 is encoded as Base64 and that the size does not exceed 350 kB. If provided, confirm by the analytical evaluation that data is compressed successfully. If provided, confirm by the analytical evaluation that the data is encrypted correctly with the encryption algorithm and that the temporary encryption key is shared according to 11.5.7. Confirm by observation that the Upload message is set with fromSubscription. If provided, confirm by analytical evaluation that the envelopeRootCertificateThumbprint is encoded according to X.509. Confirm by observation that the remaining attributes are encoded and set according to Table 16.	Use the simulator or test arrangements as the EUT to import an Upload message. Confirm by observation that the UploadObject message is valid. Confirm by observation that the envelope certificate is valid and confirm by observation that the envelopeSignature is processed and that the envelopeSignature is verified. If provided, confirm by analytical evaluation that the data is decrypted correctly with the encryption algorithm and that the temporary encryption key is shared according to 11.5.7. If provided, confirm by the analytical evaluation that the data is decompressed successfully. Confirm by observation that the publicCertificate attribute in the DigitalSignatureValueObject in Table 5 is valid and confirm by observation that the digital dataSignature is processed and verified correctly. Confirm by observation that the dataProductType can be validated against its schema (XSD). Confirm by observation that the decompressed, decrypted data can be processed and viewed successfully.
2	Import the acknowledgement message to the EUT and confirm by observation that the acknowledgement is processed correctly with a valid transaction identifier and that there is an indication with invalid acknowledgement.	If attribute ackRequest from Table 10 is equal to 2 or 3: Confirm by observation that Opened Acknowledgement message is exported with a valid transaction identifier. (for acknowledgement test, see 11.5.5)

11.4.4 Download data

11.4.4.1 Prerequisites

The download occurs in two steps: get a list of published messages (request to Get Summary) and select one of the received messages as input to interface Get request.

When downloading protected data, refer to 5.7.15.4 and specifically Figure 41.

11.4.4.2 Actors

Ship/Shore data consumer EUT – Ship/Shore data producer EUT.

11.4.4.3 Testdata

The related folders are:

- download\open\
- download\compressed\
- download\protected\
- download\openedAck\
- auth.zip

11.4.4.4 Method

Table 116 contains the test execution steps.

Table 116 – Download test method steps

Step	Applicable for data consumer EUT	Applicable for data producer EUT
0	Use the simulator or test arrangement to perform a (simulated) GetSummary request with no optional parameters.	Use the test data to prepare a set of published messages encoded as for example S100_DataSet S421 If provided, confirm by the analytical evaluation that data is compressed successfully. If provided, confirm by the analytical evaluation that the data is encrypted correctly with the encryption algorithm and that the temporary encryption key is shared according to 11.5.7.
1	Use the simulator or test arrangements as the EUT to import a GetSummaryResponseObject message.	Use the simulator or test arrangements as the EUT to export a GetSummaryResponseObject message from Table 36.
2	Use the simulator or test arrangements as the EUT to select one SummaryObject from the GetSummaryResponseObject and observe the value of the dataReference attribute.	
3	Use the simulator or test arrangement to perform a (simulated) Get request with selected attribute dataReference from step 2.	
4	Use the simulator or test arrangements as the EUT to import a GetResponseObject message. Confirm by observation that the GetResponseObject message is valid. Confirm by observation that the envelope certificate is valid and confirm by observation that the envelopeSignature is processed and that the envelopeSignature is verified. If provided, confirm by the analytical evaluation that the data is decrypted correctly with the encryption algorithm and that the temporary encryption key is shared according to 11.5.7. If provided, confirm by the analytical evaluation that the data is decompressed successfully. Confirm by observation that the publicCertificate attribute in the DigitalSignatureValueObject in Table 5 is valid and confirm by observation that the digital dataSignature is processed and verified correctly. Confirm by observation that the dataProductType can be validated against its schema (XSD). Confirm by observation that the decompressed, decrypted data can be processed and viewed successfully.	Use the simulator or test arrangements as the EUT to export a GetResponseObject message as a single message and confirm by observation that the attributes are encoded according to Table 32.
5	If attribute ackRequest from Table 10 is equal to 2 or 3, confirm by observation that Opened Acknowledgement message is exported with a valid transactionId. (for Acknowledgement test, see 11.5.5)	Import the acknowledgement message to the EUT and confirm by observation that the acknowledgement is processed correctly with a valid transaction identifier and that there is an indication with invalid acknowledgement.

11.5 SECOM Information Service test

11.5.1 General

This test describes message exchange between a SECOM consumer client and a SECOM information service instance EUT. Hence, the EUT in focus are the SECOM information service operation interfaces where all test methods terminate inside the EUT. For each interface test, there is a corresponding set of test data available. The test data is found in the interface.zip archive where the payload json file naming reflects the operation under test, see Table 117.

Table 117 – Test data reference

File	Test	Description	Defintion
interface\accessRequest.json	11.5.3.2	Access request object	Table 43
interface\accessResponse.json	11.5.3.2	Access response object	Table 44
interface\accessNotificationRequest.json	11.5.4.2	Access notification request object	Table 45
interface\accessNotificationResponse.json	11.5.4.2	Access notification response object	Table 49
interface\ Ack_technicalRequest.json	11.5.5.3	Acknowledgement request, delivered ack	Table 24
interface\ Ack_operationalRequest.json	11.5.5.3	Acknowledgement request, opened ack	Table 24
interface\capabilityResponse.json	11.5.6.2	Capability response object	Table 66
interface\EncryptionKeyRequest.json	11.5.7.3	EncryptionKey request object	Table 72
interface\Signature-EnvelopeKeyObject.txt	11.5.7.3	EncryptionKey envelope conversion for signature creation	7.3.4
interface\GetEncryptionKeyRequest.json	11.5.8.3	Get EncryptionKey request object	Table 72
interface\Signature-EnvelopeKeyRequestObject.txt	11.5.8.3	EncryptionKey envelope conversion for signature creation	7.3.4
interface\GetQuery.txt	11.5.9.3	Get request query	Table 31
interface\GetResponse.json	11.5.9.3	Get response object	Table 32
interface\GetSummaryQuery.txt	11.5.11.3	GetSummary request query	Table 35
interface\GetSummaryResponse.json	11.5.11.3	GetSummary response object	Table 36
interface\GetByLinkQuery.txt	11.5.10.3	GetByLink request query	Table 39
interface\GetByLinkResponse.bin	11.5.10.3	GetByLink response object	Table 40
interface\publicKeyRequest.txt	11.5.12.2	GetPublicKey query	Table 81
interface\publicKeyResponse.json	11.5.12.2	GetPublicKey response object	Table 82
interface\ping.json	11.5.14.2	Ping response object	Table 69
interface\subscriptionRequest.json	11.5.15.3	Subscription request object	Table 52
interface\subscriptionResponse.json	11.5.15.3	Subscription response object	Table 53
interface\subscriptionNotificationCreatedRequest	11.5.16.2	Subscription Notification request object for created	Table 60
interface\subscriptionNotificationCreatedResponse	11.5.16.2	Subscription Notification response object for created	Table 61
interface\subscriptionNotificationDeletedRequest	11.5.16.2	Subscription Notification request object for deleted	Table 60
interface\subscriptionNotificationDeletedResponse	11.5.16.2	Subscription Notification response object for deleted	Table 61
interface\subscriptionRemoveRequest	11.5.17.3	Remove Subscription request object	Table 56
interface\subscriptionRemoveResponse	11.5.17.3	Remove Subscription response object	Table 57
interface\uploadObject.json	11.5.18.2	Upload request object	Table 16
interface\uploadLinkObject.json	11.5.19.2	UploadLink request object	Table 20

File	Test	Description	Defintion
interface\dataSet.s421.xml		Payload data used in tests including transfer of data	Table 8
auth\environment.json	all	Example of environment setup for two SECOM information service instances	n/a
auth\Keystore_secomtest01.p12	all	Server certificate for SECOM instance #1	n/a
auth\Keystore_secomtest02.p12	all	Server certificate for SECOM instance #2	n/a
auth\SECOM_IdReg.crt	all	Intermediate certificate from Pki provider	n/a
auth\SECOM_Root.crt	all	Root certificate from Pki provider	n/a
swagger\SECOM_Information_Service_Interface.json	all	SECOM REST Api definition	Clause A.2

11.5.2 Prerequisites SECOM information service EUT

SECOM information service equipment according to Clause F.4. The following are prerequisites for a SECOM service instance to be able to perform tests.

- Instance shall be registered in an identity registry simulated or real.
- Instance shall be registered in a service registry simulated or real.
- Instance shall have access to an agreed SECOM compliant PKI simulated or real.
- Instance shall have a valid SECOM PKI compliant certificate simulated or real.
- Instance shall be implemented according to OpenAPI definition in Annex A.2.
- Instance communication applies to Clause 6.
- Instance request is authenticated according to 4.3.1.
- A web client for consuming the SECOM instance REST API.
- A web client HTTP request header that contains a valid TLS certificate.

The following tests are required before the information service test is performed.

- Confirm by observation that all REST messages are defined as OpenAPI format as in Clause A.2.
- Confirm by observation that all service interfaces includes the correct version as specified in Table 2.

11.5.3 Access

11.5.3.1 Test data

The related files are:

- interface\accessRequest.json
- interface\accessResponse.json

11.5.3.2 Method

Table 118 contains the test execution steps.

Table 118 – Access test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as client to send an Access message. Confirm by observation that all attributes in Table 42 are provided. Confirm by observation that the client can perform a request to "Access" interface.	Use the simulator or test arrangements as the EUT to receive an Access message and confirm by observation that the message is verified and processed correctly.
2	Import the response message to the client and confirm by observation that response was processed correctly in addition to the HTTP code and that there is an indication of the error message if provided.	Confirm by observation that the response message is exported with correct HTTP code.

11.5.4 Access notification

11.5.4.1 Test data

The related file is:

- interface\accessNotification.json

11.5.4.2 Method

Table 119 contains the test execution steps.

Table 119 – Access Notification test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as client to send an Access Notification message as a single message. Confirm by observation that all attributes in Table 47 are provided. Confirm by observation that the client can perform a request to "Access Notification" interface.	Use the simulator or test arrangements as the EUT to import an Access Notification message and confirm by observation that the message is verified and processed correctly.
2	Import the response message to the client and confirm by observation that response was processed correctly in addition to the HTTP code and that there is an indication of the error message if provided.	Confirm by observation that the response message is exported with correct HTTP code.

11.5.5 Acknowledgement

11.5.5.1 Prerequisites

One of test cases 11.5.10, 11.5.18 or 11.5.19 has been performed with the attribute ackRequest not set to 0 (no ack requested).

11.5.5.2 Test data

The related files are:

- interface\Ack_technicalRequest.json
- interface\Ack_operationalRequest.json

11.5.5.3 Method

Table 120 contains the test execution steps.

Table 120 – Acknowledgement test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as client to send an Acknowledgement message object as a single message. Confirm by observation that all attributes in Table 24 are provided. Confirm by observation that the client can perform a request to "Acknowledgement" interface.	Use the simulator or test arrangements as the EUT to import an Acknowledgement message and confirm by observation that the message is verified and processed correctly. Confirm by observation that the envelopeCertificate is valid. Confirm by observation that the digitalSignature is valid. Confirm by observation that createdAt is later than the transaction time of the related payload exchange. Confirm by observation that the transactionIdentifier correlates with the related payload exchange. Confirm by observation that the ackType is as expected. Confirm by observation that the envelopeSignatureTime is within an acceptable time limit regarding the related payload exchange.
2	Import the response message to the client and confirm by observation that response was processed correctly in addition to the HTTP code and that there is an indication of the error message if provided.	Confirm by observation that the AcknowledgementResponseObject from Table 26 response message is exported with correct HTTP code and if error, a correct SECOM_ResponseCode from Table 9 value is set.

11.5.6 Capability

11.5.6.1 Test data

The related file is:

- interface\capabilityResponse.json

11.5.6.2 Method

Table 121 contains the test execution steps.

Table 121 – Capability test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as client to send a Capability request message. Confirm by observation that the client can perform a request to data producer's "Capability" interface.	Use the simulator or test arrangements as the EUT to import a Capability get request message and confirm by observation that the message is verified and processed correctly. Confirm by observation that the response message is built according to Table 65
2	Import the CapabilityResponseObject response message to the client and confirm by observation that response was processed correctly and that there is an indication of the error message if provided.	Confirm by observation that the response message is exported with correct HTTP code

11.5.7 EncryptionKey

11.5.7.1 Prerequisites

Data producer has created a temporary encryption key and an initialization vector according to 7.5.3. Data producer has protected the information using the temporary encryption key and the initialization vector. Data producer has received the data consumer's public certificate. Data producer has sent/will send protected information to consumer.

11.5.7.2 Test data

The related files are:

- interface\EncryptionKeyRequest.json
- interface\Signature-EnvelopeKeyObject.txt

11.5.7.3 Method

Table 122 contains the test execution steps.

Table 122 – EncryptionKey test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Confirm by observation that the encryption of the temporary encryption key is according to 7.5.3 using the consumer's public certificate. Confirm by observation that all attributes in the EnvelopeKeyObject in Table 71 are encoded correctly. Confirm by observation that the digitalSignature is created according to 7.3.3. Confirm by observation that the envelopeSignature is created according to 7.3.4. Use the simulator or test arrangements as client to send an EncryptionKeyObject message as a single message. Confirm by observation that the client can perform a request to data producer's "EncryptionKey" interface.	Use the simulator or test arrangements as the EUT to import an EncryptionKeyObject message and confirm by observation that the message is verified and processed correctly. Confirm by observation that the received public certificate is valid. Confirm by observation that the received envelope certificate is valid. Confirm by observation that the received envelopeSignature is valid. Confirm by observation that the received digitalSignature is valid. Verify decryption of the encryption key according to 7.5.5.
2	Import the response message to the client and confirm by observation that response was processed correctly and that there is an indication of the error message if provided.	Confirm by observation that the response message is exported with correct HTTP code.

11.5.8 EncryptionKey Notification

11.5.8.1 Prerequisites

Data producer has published protected information. Data producer has stored the related temporary encryption key. Message retrieved according to test case in 11.5.11 and attribute dataProtection = true.

11.5.8.2 Test data

The related files are:

- interface\EncryptionKeyNotificationRequest.json
- interface\Signature-EnvelopeKeyNotificationObject.txt
- auth.zip

11.5.8.3 Method

Table 123 contains the test execution steps.

Table 123 – EncryptionKey notification test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as client to send a request EncryptionKeyNotificationObject message as a single message. Confirm by observation that all attributes in Table 72 are encoded correctly. Confirm by observation that the client can perform a request to data producer's "EncryptionKey" interface.	Use the simulator or test arrangements as the EUT to import an EncryptionKeyNotificationObject message and confirm by observation that the message is verified and processed correctly. Confirm by observation that the received public certificate is valid. Confirm by observation that the received envelopeSignature is valid. Confirm by observation that the dataReference relates to the previously stored temporary encryption key.
2	Import the response message to the client and confirm by observation that response was processed correctly and that there is an indication of the error message if provided.	Confirm by observation that the response message is exported with correct HTTP code.
3		Confirm by observation that a separate work process is activated, using the stored temporary key, initialization vector, consumer's public certificate and dataReference to build an EncryptionKeyObject. Proceed according to 11.5.7.

11.5.9 Get

11.5.9.1 Prerequisites

Data producer has published information accessible for consumer. Client has consumed data producer's SECOM information service endpoint "Get Summary" according to 11.5.11 and received a reference to the accessible data to be downloaded.

11.5.9.2 Test data

The related files are:

- interface\GetQuery.txt
- interface\GetResponse.json
- auth.zip

11.5.9.3 Method

Table 124 contains the test execution steps.

Table 124 – Get test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements to import a request URL with parameter values according to Table 30. Confirm by observation that all attributes included in the input query are encoded as expected. Confirm by observation that the attribute dataReference from "Get Summary" response is set. Confirm by observation that the client can perform a request to data producer's "Get" interface.	Use the simulator or test arrangements as the EUT to receive, interpret and validate a GET query and find the related published data. Confirm by observation that the response message is built according to Table 31.
2	Receive the response message and confirm by observation that the response was processed correctly and that there is an indication of the error message if provided. If data protection is provided, confirm by the observation that the data is decrypted correctly. If the encryption key needed for decryption is not available, get the encryption key according to 11.5.8. Confirm by observation that the encryption key related to the protected data has been uploaded. Confirm by observation that the data is decrypted successfully. If data is compressed, confirm by observation that data is uncompressed successfully. The decrypted and/or decompressed data can now be used to validate the data digital signature according to 7.3.5. If any errors, confirm by observation that acknowledgement is requested by the data producer and if so, send a "Not acknowledged ack" according to 11.5.5. If provided, confirm by observation that the pagination information is processed correctly.	Confirm by observation that the response message is exported with correct HTTP code. If provided, confirm by observation that EUT sends the first page when the page parameter is not present or maximum number of items per page when pageSize is not present or no items when the page parameter is set to 0.

11.5.10 Get By Link

11.5.10.1 Prerequisites

Data producer has published data files exceeding 350 kB. Data producer has uploaded a link using data consumer's SECOM service interface UploadLink.

11.5.10.2 Test data

The related files are:

- interface\GetByLinkQuery.txt
- interface\GetByLinkResponse.bin
- auth.zip

11.5.10.3 Method

Table 125 contains the test execution steps.

Table 125 – Get By Link test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements to import a query according to Table 38. Confirm by observation that the attribute transactionIdentifier is encoded as required. Confirm by observation that the client can perform a request to the data producer's "Get by Link" interface.	Use the simulator or test arrangements as the EUT to receive, interpret and validate a "Get by Link" query and find the related published data. Confirm by observation that the published data is added to the response object according to Table 39.
2	Import the response message to the client and confirm by observation that response was processed correctly and that there is an indication of the error message if provided. If data protection is provided, confirm by observation that the data is decrypted correctly. If the encryption key needed for decryption is not available, get the encryption key according to 11.5.8. Confirm by observation that the encryption key related to the protected data has been uploaded. Confirm by observation that the data is decrypted successfully. If data is compressed, confirm by observation that data is uncompressed successfully. The decrypted and/or decompressed data can now be used to validate the data digital signature according to 7.3.5. If any errors, confirm by observation that acknowledgement is requested by data producer and if so, send a "Not acknowledged ack" according to 11.5.5.	Confirm by observation that the response message is exported with correct HTTP code.

11.5.11 Get Summary

11.5.11.1 Prerequisites

Data producer has published information accessible for consumer

11.5.11.2 Test data

The related files are:

- interface\GetSummaryQuery.txt
- interface\GetSummaryResponse.json
- auth.zip

11.5.11.3 Method

Table 126 contains the test execution steps.

Table 126 – Get Summary test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as client to import a request URL with parameter values according to Table 34. Confirm by observation that all attributes included in the input query are encoded as expected. Confirm by observation that the client can perform a request to "Get Summary" interface.	Use the simulator or test arrangements as the EUT to receive, interpret and validate a "Get Summary" query and finds the related published meta-data. Confirm by observation that the response message is built according to Table 35.
2	Receives the response message and confirm by observation that the response was processed correctly and that there is an indication of the error message if provided. If provided, confirm by observation that the pagination information is processed correctly.	Confirm by observation that the response message is sent with correct HTTP code. If provided, confirm by observation that the EUT sends the first page when the page parameter is not present or maximum number of items per page when pageSize is not present or no items when the page parameter is set to 0.

11.5.12 Get Public Key

11.5.12.1 Test data

The related files are:

- interface\publicKeyRequest.txt
- interface\publicKeyResponse.json
- auth.zip

11.5.12.2 Method

Table 127 contains the test execution steps.

Table 127 – Get Public Key test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as the client to send a Get Public Key message and confirm by observation that the message is encoded as in Table 78.	Use the simulator or test arrangements as the EUT to receive a Get Public Key message and confirm by observation that the message is processed correctly. Confirm by observation that the response message is built according to Table 79.
2	Receives the response message and confirm by observation that the response was processed correctly and that there is an indication of the response message if provided.	Confirm by analytical evaluation that the response message is sent with correct HTTP code and the requested public key is included.

11.5.13 Upload Public Key

11.5.13.1 Test data

The related files are:

- interface\publicKeyRequest.json
- auth.zip

11.5.13.2 Method

Table 128 contains the test execution steps.

Table 128 – Upload Public Key test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as the client to upload a Public Key message and confirm by observation that the message is encoded as in Table 79.	Use the simulator or test arrangements as the EUT to receive a Public Key message and confirm by observation that the message is processed correctly.
2	Receive the response message and confirm by observation that the response was processed correctly and that there is an indication of the response message if provided.	Confirm by analytical evaluation that the response message is sent with correct HTTP code and the requested public key is included.

11.5.14 Ping

11.5.14.1 Test data

The related files are:

- interface\ping.json
- auth.zip

11.5.14.2 Method

Table 129 contains the test execution steps.

Table 129 – Ping test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as client to post a Ping request message.	Use the simulator or test arrangements as the SECOM Ping interface to receive and interpret a Ping request.
2	Import the Ping response message to the client and confirm by observation that response was processed correctly and that there is an indication of the error message if provided.	Confirm by observation that the response message is exported with correct HTTP code and proper attributes according to Table 68.

11.5.15 Subscription

11.5.15.1 Prerequisites

This test methods only covers the related operations for the subscription interface. For the test for actual data transfer with the subscription, see the test methods in each operation.

11.5.15.2 Test data

The related files are:

- interface\subscriptionRequest.json
- interface\subscriptionResponse.json
- auth.zip

11.5.15.3 Method

Table 130 contains the test execution steps.

Table 130 – Subscription test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as the client to send a Subscription message as a single message. Confirm by observation that the message is encoded as defined in Table 51.	Use the simulator or test arrangements as the EUT to receive a Subscription message and confirm by observation that the message is verified and processed correctly.
2	Receive the response message and confirm by observation that the response was processed correctly and that there is an indication of the error message if provided.	Confirm by observation that the response message is exported with correct HTTP code and proper attributes in ResponseSubsciptionObject. Confirm by observation that an active subscription list is available.

11.5.16 Subscription Notification

11.5.16.1 Test data

The related files are:

- interface\subscriptionNotificationCreatedRequest.json
- interface\subscriptionNotificationCreatedResponse.json
- interface\subscriptionNotificationDeletedRequest.json
- interface\subscriptionNotificationDeletedResponse.json
- auth.zip

11.5.16.2 Method

Table 131 contains the test execution steps.

Table 131 – Subscription Notification test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Confirm by observation that client sends a Subscription Notification message with proper subscriptionIdentifier and eventEnum value according to Table 59.	Receive the subscription Notification message to the EUT and confirm by observation that message was processed correctly.
2	Receive the response message and confirm by observation that the response was processed correctly and that there is an indication of the error message if provided.	Confirm by observation that the response message is sent with correct HTTP code and proper attributes in ResponseSubsciptionNotificationObject

11.5.17 Remove Subscription

11.5.17.1 Prerequistes

An active subscription exists

11.5.17.2 Test data

The related files are:

- interface\subscriptionRemoveRequest.json
- interface\subscriptionRemoveResponse.json
- auth.zip

11.5.17.3 Method

Table 132 contains the test execution steps.

Table 132 – Remove Subscription test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as client to send a Remove Subscription message with proper subscription identifier as a single message. Confirm by observation that the message is encoded as defined in Table 55.	Use the simulator or test arrangements as the EUT to receive a Remove Subscription message and confirm by observation that the message is verified and processed correctly.
2	Receive the response message and confirm by observation that the response was processed correctly and that there is an indication of the error message if provided.	Confirm by observation that the response message is sent with correct HTTP code and proper attributes in ResponseRemoveSubsciptionObject. Confirm by observation that the removed subscription was deleted from an active subscription list.

11.5.18 Upload

11.5.18.1 Test data

The related files are:

- interface\UploadObject.json
- auth.zip

11.5.18.2 Method

Table 133 contains the test execution steps.

Table 133 – Upload test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements as the client to import an UploadObject message as a single message. Confirm by observation that the data attribute in Table 16 is encoded as Base64 and that the size does not exceed 350 kB. If provided, confirm by analytical evaluation that the data is encrypted correctly with the encryption algorithm and that the temporary encryption key is shared according to 11.5.7. If provided, confirm by analytical evaluation that data is compressed successfully. Confirm by observation that the Upload message is set with fromSubscription. If provided, confirm by analytical evaluation that the envelopeRootCertificateThumbprint is encoded according to X.509. Confirm by observation that the remaining attributes are encoded and set according to Table 16. Confirm by observation that the HTTP request header contains a valid certificate. Confirm by observation that the client can perform a request to SECOM information service "Upload" interface.	Use the simulator or test arrangements as the EUT to receive an Upload message. Confirm by observation that the UploadObject message is valid. Confirm by observation that the dataProductType is as expected. If not, return status code 501. Confirm by observation that the client certificate is valid. Confirm by observation that the envelope certificate is valid and confirm by observation that the envelopeSignature is processed and that the envelopeSignature is verified.
2	Receive the response message and confirm by observation that the response was processed correctly and that there is an indication of the response message if provided.	Confirm by observation that the response message is sent with correct HTTP code and SECOM_ResponseCodeEnum as specified in Table 9.

11.5.19 Upload Link

11.5.19.1 Test data

- interface\UploadLinkObj.json
- auth.zip

11.5.19.2 Method

Table 134 contains the test execution steps.

Table 134 – Upload Link test method steps

Step	Applicable for SECOM client	Applicable for SECOM service EUT
1	Use the simulator or test arrangements with at least three large files of at least 2 MB and select a file as the client to send an Upload Link message as a single message. Confirm by observation that the data attribute in Table 20 is encoded as Base64. If provided, confirm by analytical evaluation that the data is encrypted correctly with the encryption algorithm and that the temporary encryption key is shared according to 11.5.7. If provided, confirm by analytical evaluation that data is compressed successfully. If provided, confirm by observation that the Upload message is set with fromSubscription. If provided, confirm by analytical evaluation that the envelopeRootCertificateThumbprint is encoded according to X.509. Confirm by observation that the remaining attributes are encoded and set according to Table 20. Confirm by observation that the client can perform a request to the "Upload Link" interface.	Use the simulator or test arrangements as the EUT to import an UploadLink message and confirm by observation that the message is verified and processed correctly. Confirm by observation that the UploadObjectLink message is valid. Confirm by observation that the dataProductType is as expected. If not, return status code 501. Confirm by observation that the client certificate is valid. Confirm by observation that the envelope certificate is valid and confirm by observation that the envelopeSignature is processed and that the envelopeSignature is verified.
2	Receive the response message and confirm by observation that the response was processed correctly and that there is an indication of the response message if provided.	Confirm by observation that the response message is exported with correct HTTP code and SECOM_ErrorCode as specified in Table 8.

11.6 SECOM PKI Service test

11.6.1 Prerequisites PKI EUT

SECOM PKI equipment according to Clause F.5.

The following are prerequisites for SECOM PKI instance to be able to perform tests.

- SECOM PKI instance shall have a valid trusted SSL certificate.
- SECOM PKI instance shall be implemented according to OpenAPI definition in Clause A.3.
- SECOM PKI instance containing a CRL in accordance with RFC 5280.
- At least one scheme administrator which includes a private/public key pair and contains self-signed public key.
- At least one data server which generates its public and private key pair and produces digital signature of the data, refer to 8.3.
- At least three data clients which generate their public and private key pair and produce digital signature of the data.
- Instances communication applies to Clause 6.
- Instance request is authenticated according to 4.3.1.
- A web client for consuming the SECOM instance REST API.
- A web client HTTP request header that contains a valid TLS certificate.

11.6.2 CRL

11.6.2.1 Test data

The related files are:

- pki\Request_CRL.txt
- pki\Response_CRL,txt
- pki\Response_CRL_PemDecoded.txt
- pki\Response_CRL_PemEncoded.txt

11.6.2.2 Method

Table 135 contains the test execution steps.

Table 135 – CRL test method steps

Step	Applicable for SECOM client	Applicable for SECOM PKI EUT
1	Use the simulator or test arrangements as the client to invoke a CRL request with CRL request message and confirm by observation that the message includes mandatory attributes in Table 98 as described in RFC 5280.	Use the simulator or test arrangements as the EUT to receive a CRL request message and confirm by observation that the message is processed correctly.
2	Receives the response message and confirm by observation that the response was processed correctly and that there is an indication of the response message if provided together with a PEM encoded CRL.	Confirm by analytical evaluation that the response message (CRL) is sent with correct HTTP code.

11.6.3 OCSP

11.6.3.1 Test data

The related folder is:

- pki\ocsp*

11.6.3.2 Method

Table 136 contains the test execution steps.

Table 136 – OCSP test method steps

Step	Applicable for SECOM client	Applicable for SECOM Pki EUT
1	Use the simulator or test arrangements as the client to invoke a OCSP request with OCSP request message and confirm by observation that the message includes mandatory attributes in as described in RFC 6960.	Use the simulator or test arrangements as the EUT to receive a OCSP request message and confirm by observation that the message is processed correctly.
2	Receive the response message and confirm by observation that the response was processed correctly and that there is an indication of the response message if provided.	Confirm by analytical evaluation that the response message (OCSP-response) is sent with correct HTTP code.

11.6.4 Revoke

11.6.4.1 Test data

The related files are:

- pki\Request_CertificateRevocation.txt
- pki\Response_CertificateRevocation.json

11.6.4.2 Method

Table 137 contains the test execution steps.

Table 137 – Revoke test method steps

Step	Applicable for SECOM client	Applicable for SECOM Pki EUT
1	Use the simulator or test arrangements as the client to invoke a Revoke request with a certificate revoke message and confirm by observation that the message includes mandatory attributes as described in Table 104.	Use the simulator or test arrangements as the EUT to receive a certificate revocation request message and confirm by observation that the message is processed correctly.
2	Use the simulator or test arrangements as the client to receive a certificate revocation response message and confirm by observation that the message is processed correctly.	Confirm by analytical evaluation that the response message is sent with correct HTTP code and corresponding message.

11.6.5 CSR

11.6.5.1 Test data

The related file is:

- pki\csr-example.txt

11.6.5.2 Method

Table 138 contains the test execution steps.

Table 138 – CSR test method steps

Step	Applicable for SECOM client	Applicable for SECOM Pki EUT
1	Use the simulator or test arrangements as the client to invoke a CSR request with a CSR message and confirm by observation that the message includes mandatory attributes in Table 90 according to PKCS #10 described in RFC 2986. Refer to 8.4.	Use the simulator or test arrangements as the EUT to receive a CSR request message and confirm by observation that the message is processed correctly.
2	Receive the response message and confirm by observation that response was processed correctly and that there is an indication of the response message if provided.	Confirm by analytical evaluation that the response message is sent with correct HTTP code and public key is signed accordingly.

11.6.6 GetPublicKey

11.6.6.1 Test data

- pki\RequestPublicKey.txt
- pki\ResponsePublicKeyPemDecoded.txt
- pki\ResponsePublicKeyPemEncoded.txt

11.6.6.2 Method

Table 139 contains the test execution steps.

Table 139 – GetPublicKey test method steps

Step	Applicable for SECOM client	Applicable for SECOM Pki EUT
1	Use the simulator or test arrangements as the client to invoke a GetPublicKey request with a GetPublicKey message and confirm by observation that the message includes the serial number of the claimed public key as defined in Table 94.	Use the simulator or test arrangements as the EUT to receive a GetPublicKey request message and confirm by observation that the message is processed correctly.
2	Receive the response message and confirm by observation that response was processed correctly and that there is an indication of the response message if provided. Verify the GetPublicKeyResponseObject by examining the received PEM decoded public certificate.	Confirm by analytical evaluation that the response message is sent with correct HTTP code and the requested public key in PEM format is included.

11.7 SECOM Service Discovery test

11.7.1 General

Service Discovery equipment according to Clause F.6. This interface caters for searching a Service Registry by combining query parameters separated by and/or together with possible geometry search criteria; an example is provided in Clause D.7.

11.7.2 Prerequisites Service Discovery EUT

The following are prerequisites for SECOM Service Discovery instance to be able to perform the tests.

- Confirm by observation that all REST messages are defined as OpenAPI format as in Annex A.
- Confirm by observation that all service interfaces include the correct version as specified in Table 2.
- EUT Instance shall be implemented according to OpenAPI definition in Clause A.4.
- SECOM client instance shall be registered in an identity registry simulated or real.
- SECOM client instance shall be registered in a service registry simulated or real.
- SECOM client instance shall have access to an agreed SECOM compliant PKI simulated or real.
- SECOM client instance shall have a valid SECOM PKI compliant certificate simulated or real.
- A web client for consuming the SECOM instance REST API.
- A web client HTTP request header that contains a valid TLS certificate.

11.7.3 Search service – By geometry

11.7.3.1 Test data

The related files are:

- serviceDiscovery\Request_SearchService_geometry.txt
- serviceDiscovery\Response_SearchService_geometry.txt

11.7.3.2 Method

Table 140 contains the test execution steps.

Table 140 – Search service by geometry test method steps

Step	Applicable for SECOM client	Applicable for SECOM Service discovery EUT
1	Use the simulator or test arrangements as the client to invoke a Search request with a SearchFilterObject message as a single message. Confirm by observation that the attributes in Table 109 are encoded as specified. Specify a geometry search according to WKT geometry, see Table 12.	Use the simulator or test arrangements as the EUT to receive a Search request with a search message and confirm by observation that the message is processed correctly.
2	Receive the response message and confirm by observation that response was processed correctly and that there is a response message according to ResponseSearchObject together with expected HTTP code and message. Expected response shall be all registered service instances in the service registry with coverage area according to the provided geometry search parameter.	Confirm by analytical evaluation that the response message is sent with correct HTTP code and correct list of service instance information accordingly.

11.7.4 Search service – Without specified search criteria

11.7.4.1 Test data

The related files are:

- serviceDiscovery\Request_SearchService_empty.txt
- serviceDiscovery\Response_SearchService_empty.txt

11.7.4.2 Method

Table 141 contains the test execution steps.

Table 141 – Search service empty query test method steps

Step	Applicable for SECOM client	Applicable for SECOM Service discovery EUT
1	Use the simulator or test arrangements as the client to invoke a Search request with a SearchFilterObject message as a single message. Confirm by observation that the attributes in Table 109 are encoded as specified. Ensure that no search criteria is provided in the SearchFilterObject.	Use the simulator or test arrangements as the EUT to receive a Search request with a search message and confirm by observation that the message is processed correctly.
2	Receive the response message and confirm by observation that response was processed correctly and that there is a response message according to ResponseSearchObject together with expected HTTP code and message. Expected response shall be all registered service instances in the service registry.	Confirm by analytical evaluation that the response message is sent with correct HTTP code and correct list of service instance information accordingly.

Annex A
(normative)

REST service interface definitions

A.1 Purpose

Annex A describes the formal definitions of the SECOM service interfaces in OpenAPI[1] format. The json files are available at http://cirm.org/secom.

A.2 SECOM information service REST interface definition

The OpenAPI file is available at swagger\SECOM_Information_Service_Interface.json

A.3 SECOM PKI service REST interface definition

The OpenAPI file is available at swagger\SECOM_PKI_Interface.json

A.4 SECOM discovery service REST interface definition

The OpenAPI file is available at swagger\SECOM_Service_Discovery_Interface.json

[1] The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs https://swagger.io/.

Annex B
(informative)

Operational use cases and profiles

B.1 Purpose

Annex B describes use cases or profiles that describe usage of the service interface in different operational contexts. The use cases described are based on a generic voyage, from Busan to Quebec via Stockholm, where typical service usage is derived. Each service usage/call is described in a use case. The use cases are:

– route plan exchange ship-shore and shore-ship (Enhanced monitoring);
– pilot routes;
– route optimization;
– search for services to consume;
– chart (ENC) updates;
– navigational warnings.

The service interface is focused on the exchange of information products and not designed towards a specific operational service. Different operational services will require different exchange patterns and also different sets of active interfaces such as subscription. An enhanced monitoring service for example will most likely not allow ships to subscribe to route plans from them, but the enhanced monitoring service will want to subscribe to route plans from the ship.

Based on a set of defined use cases for a certain actor, the profile for that actor can be defined. Hence, a ship service interface profile is the sum of all service interfaces used in the use cases for the ship.

B.2 Use cases and service interface profiles

B.2.1 UC-1 Ship shares route plan with service providing enhanced monitoring

B.2.1.1 Pre-requisites

A service provider of enhanced monitoring, for example a VTS, exposes a discoverable service interface to receive S-421 Route Plans (IEC 63173-1), receives acknowledgement and receives subscription notifications.

Searching and finding of the service in the service registry is according to Clause 9.

The ship exposes a discoverable service interface for enhanced monitoring (receive S-421 Route Plan).

B.2.1.2 Description

In this use case for route plan exchange, an exchange between a ship and a VTS has been selected but also other scenarios and actors such as ports, fleet operation centres and various service providers could have similar route plan exchanges as the VTS. The ship/shipping company is the information owner of the route plan and as such chooses which actors should be granted access to the route plan. Some of the information in the route plan can be sensitive from a business perspective, for example the arrival port and arrival time and thus there is a need to protect the route plan from unauthorized actors.

The VTS can use the route plan to monitor that the ship stays within the planned corridor as defined in the route plan or in the VTS system. The VTS can receive updates of the ship position through for example automatic identification system (AIS) and compare these ship positions against the route plan for the vessel. The shore centre is able to detect not only if the ship deviates from the planned route geometry but also if the planned schedule is not kept. The VTS can also use the route plan information to foresee possible dangerous situations. The VTS may contact the ship by different means.

1) The ship wants assistance and searches for an appropriate service in the service registry similar to the use case described in B.2.5.
2) The ship checks the supported SECOM interfaces of the discovered service by calling the service capability endpoint, see 5.7.13 .
3) The ship prepares a S-421 Route Plan together with metadata for an UploadObject, see Table 16.
4) The S-421 is signed and the signature is added to the SECOM_ServiceExchangeMetadata part of the UploadObject.
5) The ship nominates the enhanced monitoring service and adds the service to its list of subscribers in order to provide possible route plan updates enroute. The service is added to the ship's whitelist.
6) The ship sends (uploading, pushing) the S-421 Route Plan to the enhanced monitoring service.
7) The VTS verifies the signature and the certificate and hence authenticates the sender (the ship) and the route message's integrity.
8) The VTS checks the route against local conditions and either confirms the original route plan or makes a new proposed one. If the latter, the enhanced monitoring service makes a proposal and sends (uploads) the proposed S-421 Route Plan to the ship and requests an operational acknowledgement (see 5.7.4) when delivered to the ship and processed by an operator on the ship.
9) The ship receives the proposed S-421 Route Plan and sends an operational acknowledgment when the officer on watch opens the message.
10) When the voyage is ended, the enhanced monitoring service is removed from the list of subscribers and a subscription notification message is sent.

B.2.1.3 Service interfaces required

Table B.1 describes used interfaces in this use case.

Table B.1 – UC-1 Ship shares route plan with service providing enhanced monitoring

Service interface	Ship	Enhanced monitoring
Upload	YES; step 6)	YES; step 8)
Acknowledgement	YES; step 9)	
Subscription	YES; step 5)	
Subscription Notification	YES; step 5), 10)	
Remove Subscription	YES; step 10)	
Capability	YES; step 2)	
NOTE The steps refer to B.2.1.2.		

B.2.2 UC-2 Pilot routes

B.2.2.1 Pre-requisites

The service provider of a pilot route service exposes a discoverable service interface to receive a S-421 Route Plan and receive acknowledgement.

The ship exposes a discoverable service interface to receive the S-421 Route Plan proposal.

B.2.2.2 Description

Planning a route passing through previous unvisited areas could be tedious and difficult. The officer of the watch on board might need to consider unknown obstacles such as shallow waters, speed and size of the ship, local regulations etc. and in most cases a pilot is needed. When the pilot boards the ship, a pilot route candidate could be included into the route that is loaded into the ship's ECDIS system and if the pilot agrees, it can be used for navigation into or out from the port.

To facilitate the voyage planning into the port of Stockholm, the ship wants to use known and safe local routes used by pilots, i.e. pilot routes. A pilot route service is an onshore service that can provide pilot routes to ships when planning their voyages. The ship can get one or several pilot routes in return and can choose among the returned routes which to add to their planned route plan.

If a pilot route service supports a functionality to receive and interpret route plans (S-421), this input could be used for the service to calculate and return the closest pilot route(s).

1) The ship searches in trusted service registry for service instance that can provide the ship with pilot routes (reference routes in S-421) – rough search by area or unlocode to get the right service instance.
2) The ship calls the service instance Get Summary with detailed search parameters relevant for the service (GET object/summary?geometry=route as line and/or unlocode=X, etc.) – search for the relevant information object the service instance can provide.
3) The pilot route service Instance interprets the Get Summary request and returns the summary of the information it can provide.
4) The ship receives the summary of the information the service can provide.
5) The ship selects the information it wants to download and calls the service instance Get object?dataReference.
6) The pilot route service Instance returns the selected information including the signature.
7) The ship receives the pilot route and verifies the signature.

B.2.2.3 Service interfaces required

Table B.2 describes the used interfaces in this use case.

Table B.2 – Required service interfaces in UC-3

Service interface	Ship	Pilot route service
Upload		YES; step 6)
Get	YES; step 5)	
Get Summary	YES; step 2)	
NOTE The steps refer to B.2.2.2.		

B.2.3 UC-3 Route optimization

B.2.3.1 Pre-requisites

A route optimization service provider exposes a discoverable service interface to receive S-421 Route Plans (IEC 63173-1).

Searching and finding the service in the service registry is according to Clause 9 SECOM service discoverability and not included in this use case.

The ship exposes a discoverable service interface for route optimization (receive S-421 Route Plan).

The ship is aware of that the route optimization service provides encrypted Route Plans.

B.2.3.2 Description

Given that both a ship and a route optimization service provider support SECOM and S-421, the ship can send a planned route to the service provider and receive in return its route optimized for defined parameters such as weather, wind currents etc. The optimized route can be imported directly into the ECDIS without the need for converting formats or transport by, for example, USB-stick. In this use case, the route optimization provider wants to be certain that only the specific ship can access the optimized route and therefore encrypts the route plan.

1) The ship sends its route and its public certificate to the route optimization provider.

   NOTE If the ship does not send its public certificate, the route optimization provider can request the ship's public certificate used for encryption by invoking the ship's SECOM PublicKey interface.

2) The Route optimization provider optimizes the received route and decides to protect the optimized route.
   a) Route optimization provider creates a random symmetric encryption key and encrypts the optimized route.
   b) Before exchanging the random encryption key, route optimization provider needs to protect the encryption key using a derived symmetric key.
   c) Route optimization provider derives a symmetric key using the public certificate received, together with its own private key.
   d) Route optimization provider encrypts the random encryption key with the derived symmetric key.
   e) Route optimization provider sends the protected encryption key payload using the ship's SECOM EncryptionKey interface.
3) The Route optimization provider uploads the protected optimized route to the ship's SECOM Upload interface.
4) The ship derives the same symmetric key using the public key sent by the route optimization service together with its private key and uses that to decrypt the encryption key.
5) The ship decrypts the optimized route with the decrypted encryption key.

B.2.3.3 Service interfaces required

Table B.3 shows the interfaces used in this use case.

Table B.3 – Required service interfaces in UC-3

Service interface	Ship	Route optimization service
Upload	YES; step 1)	YES; step 3)
EncryptionKey		YES; step 2)e)
PublicKey	YES; step 1)	
NOTE The steps refer to B.2.3.2.		

B.2.4 UC-4 Enhanced monitoring service requests route plan from/for ship for monitoring

B.2.4.1 Pre-requisites

The ship has not shared a S-421 Route Plan with an enhanced monitoring service.

The ship exposes a discoverable service interface to receive a S-421 Route Plan proposal.

The service provider of the enhanced monitoring exposes a discoverable service interface to receive a S-421 Route Plan, receive acknowledgement and receive subscription notifications.

B.2.4.2 Description

This use case is similar to UC-1 but the VTS initiates the communication instead of the ship. The VTS operator identifies the ship through AIS and wants the ships route plan. The operator searches for a service using MMSI as the search parameter. See Annex E for further description of the access process and the use of a whitelist.

1) Enhanced monitoring service wants to check the SECOM Information service condition for the ship and invokes the ship's SECOM Ping interface and confirms that the service is in operation.
2) Enhanced monitoring requests subscription on S-421 Route Plan from the ship and implicitly requests access.
3) The ship checks access rights to its route plan.
   a) If access, the ship sends (uploads) the S-421 Route Plan.
   b) If subscription is created, the ship notifies subscription created.
   c) If no access, the ship responds with 403 "Not authorized".
   d) Enhanced monitoring sends Access request with the reason "Enhanced monitoring service for area X".
   e) Ship receives and displays the access request from the enhanced monitoring service and can accept or reject the request.
4) When the voyage is ended, the subscription is removed by the ship and a notification is sent to the enhanced monitoring service.

B.2.4.3 Service interfaces required

Table B.4 describes used interfaces in this use case.

Table B.4 – Required service interfaces in UC-4

Service interface	Ship	Enhanced Monitoring
Upload	YES; step 3)a)	
Request Access		YES; step 2)
Access Notification	YES; step 3)e)	
Subscription		YES; step 2)
Subscription Notification	YES; step 3)b), 4)	
Remove Subscription	YES; step 4)	
Ping	YES; step 1)	
NOTE The steps refer to B.2.4.2.		

B.2.5 UC-5 Discover service instance to consume

B.2.5.1 Pre-requisites

The service end point (URL) is known and accessible for one or more service registries.

B.2.5.2 Description

B.2.5.2.1 General

A service registry functionality enables service consumers, for example ships, to search for services, i.e. service endpoints to interact with. As ships are also registered, it is also possible to search for ships and thus the use case is applicable shore-ship and ship-shore. The search is performed according to pre-defined parameters given in Table 110 such as geometry, service types, freetext and IMO number. This makes it possible to search for already known services, available route optimization services or which services are available in a given area or along the planned route. Other examples of service search scenarios are for instance ship wants to find MSI Service Provider along its route.

B.2.5.2.2 Ship perspective

- Ship wants to find MSI service provider along its route.
- Ship wants to find Enhanced Monitoring Services (VTS's) along its route.
- Ship wants to find Port and start Port Call Synchronization.
- Ship wants to find Pilot Route Service and get valid pilot routes.
- Ship wants to find Under Keel Clearance Monitoring service and start the procedure to receive speed recommendations.

Ship searches for service provider by Name, Type, Keyword, UN/Locode, Geometry (the route or marked area), based on known values.

The ship is given a list of service end points (URLs) and other descriptive information about the services in return.

B.2.5.2.3 VTS perspective

- VTS wants to find service for ship to ask for its monitored route.
- VTS wants to find service for ship to send information.
- VTS wants to find service for ship to send local MSI.
- Maritime rescue co-ordination centre (MRCC) wants to find service for rescue unit to send search area and search pattern to.

B.2.5.2.4 Port perspective

- Port wants to find service for ship to ask for its monitored route.
- Port wants to find service for ship to send information.

VTS/MRCC/Port searches for service provider (ship) by Name, Type, Keyword, MMSI, IMO, based on known values.

The VTS/MRCC/Port is given a list of service end points (URLs) and other descriptive information about the services in return.

B.2.5.3 Service interfaces required

This use case only references the Service interface "Search service" described in 9.2; therefore, no table with required Service interfaces is included.

B.2.6 UC-6 Chart (ENC) updates

B.2.6.1 Pre-requisites

A service provider of ENC charts, for example an ENC distributor, exposes a discoverable service interface to distribute ENC (S-57/S-101) data.

Searching and finding the service in the service registry is according to Clause 9.

The ship exposes a discoverable service interface for ENC charts inside S100_ExchangeSet (receive S-57/S-101).

B.2.6.2 Description

SECOM provides one common interface between the VAR (value added reseller) and the distributor, and between the distributor ant the users that could be used in ENC distribution. The common interface facilitates interoperability and reduces development costs and efforts to support several different proprietary interfaces.

SECOM could also facilitate for VAR to provide other types e-navigation services.

A common service registry or interoperable registries allows users to seamlessly interact with different service providers and vice versa, ensuring that provided services can be used by a multitude of users.

1) The ship has an active subscription to a defined set of ENC charts or has purchased a new set of charts outside SECOM.
2) Updated/new ENC charts are ready for distribution.
3) ENC distributor sends updated charts packed in a S100_ExchangeSet to the ship by consuming the ship's SECOM UploadLink interface.
4) The ship SECOM instance receives the link to the updates and stores the link.
5) In the next port, the ship retrieves the updated ENC charts from ENC SECOM instance GetByLink interface.

B.2.6.3 Service interfaces required

Table B.5 describes used interfaces in this use case.

Table B.5 – Required service interfaces in UC-6

Service interface	Ship	ENC distributor
UploadLink		YES; step 3)
GetByLink	YES; step 4)	
NOTE The steps refer to B.2.6.2.		

B.2.7 UC-7 navigational warning service

B.2.7.1 Pre-requisites

A service provider of navigational warning, for example a costal administration, exposes a discoverable service interface to distribute NAVWARN (S-124) data.

The navigational warning service has a configurable subscription function that include predefined areas (such as whole service area and sub areas), subscription duration, and push frequency (e.g. once per day, every 8 h, every hour, right away).

Available NAVWARN datasets are query enabled within the service to expose location and issue date and time for filtering available datasets to relevant datasets for a vessel's voyage.

SECOM PKI is used to sign the NAVWARN datasets.

The ship exposes a discoverable service interface (Upload) for NAVWARN datasets. The navigational warning service pushes any NAVWARN dataset, according to subscription criteria, which is not already in the ship system.

B.2.7.2 Description

The purpose with the navigational warning service is to provide the service consumer, i.e. ship, with only those warnings that are relevant for a specific area that intersects with the route under navigation and at the time of scheduled route. Moreover, the warnings could be displayed directly in ECDIS and automatically deleted when they are expired and no longer valid. The service is not intended to relieve the service users from ordinary receipt of maritime safety information (MSI) as part of the global maritime distress and safety system (GMDSS), which every ship, while at sea, has to comply to.

The notices/navigational warnings do not have to be limited to specific transmission times but could be sent as soon as warnings are registered in national/regional management systems.

1) The ship searches the service registry according to geography preferences and discovers a published navigational warning service according to Clause 9.
2) The ship requests subscription by consuming the NAVWARN service SECOM Subscription interface providing a route waypoint list as a geometry parameter.
3) The NAVWARN service responds to the ship with a message and subscriptionIdentifier.
4) New NAVWARN datasets are ready for distribution.
5) The Navigational Warning Service sends new NAVWARN datasets to the ship by consuming the ship's SECOM Upload interface, supplying parameter fromSubscription = "true".
6) The ship SECOM instance receives the datasets and receives consecutive NAVWARN throughout the duration of the voyage.
7) Ending the subscription alternatives.
   a) If the ship wants to terminate the subscription of NAVWARN service, for example due to another route plan, it consumes the NAVWARN "Remove Subscription" interface and the NAVWARN service responds by sending a "Subscription Notification" of the approved removal.
   b) When the voyage ends (i.e. current datetime > last waypoint ATA) or when leaving the NAVWARN coverage area, the navigational warning service consumes the ships Subscription Notification interface with eventEnum = "Subscription removed" for the subscriptionIdentifier in step 3).

B.2.7.3 Service interfaces required

Table B.6 describes used interfaces in this use case.

Table B.6 – Required service interfaces in UC-7

Service interface	Ship	NAVWARN distributor
Upload	YES; step 5)	
Subscription		YES; step 2)
Remove Subscription		YES; step 7) a)
Subscription Notification	YES; step 7) a), 7) b)	
NOTE The steps refer to B.2.7.2.		

B.2.8 UC-8 Updates for detailed bathymetry and tidal and water level forecasts

B.2.8.1 Pre-requisites

A service provider of detailed bathymetry (S-102) and tidal and water level forecast (S-104), for example a chart distributor, exposes a discoverable service interface to distribute S-102 and S-104 data.

Searching and finding of the service in the service registry is according to Clause 9.

The ship exposes a discoverable service interface for detailed bathymetry and tidal and water level forecast inside S100_ExchangeSet (receive S-102/S-104).

B.2.8.2 Description

SECOM provides one common interface between the VAR (value added reseller) and the distributor and between the distributor and the users that could be used in detailed bathymetry and tidal and water level forecast distribution. The common interface facilitates interoperability and reduces development costs and efforts to support several different proprietary interfaces.

SECOM could also facilitate for VAR to provide other types e-navigation services.

A common service registry or interoperable registries allows users to seamlessly interact with different service providers and vice versa ensuring that provided services can be used by a multitude of users.

1) The ship has an active subscription to a defined set of detailed bathymetry and tidal and water level forecast or has purchased a new set of detailed bathymetry and tidal and water level forecast outside SECOM.
2) Updated/new detailed bathymetry and tidal and water level forecast are ready for distribution.
3) Distributor sends updated detailed bathymetry and tidal and water level forecast packed in a S100_ExchangeSet to the ship by consuming the ship's SECOM UploadLink interface.
4) The ship SECOM instance receives the link to the updates and stores the link.
5) In the next port, the ship retrieves the updated detailed bathymetry and tidal and water level forecast from SECOM instance using GetByLink interface.

B.2.8.3 Service interfaces required

Table B.7 describes used interfaces in this use case.

Table B.7 – Required service interfaces in UC-8

Service interface	Ship	ENC distributor
UploadLink	YES; step 5)	YES; step 3)
GetByLink	YES; step 4)	
NOTE The steps refer to B.2.8.2.		

Annex C
(informative)

Message exchange patterns

C.1 Purpose

This Annex C contains the used message exchange patterns.

C.2 Message exchange pattern

C.2.1 Generic message exchange patterns

C.2.1.1 General

HTTP-based communication has been used when describing the sequences, hence the default return (if no other information) is the HTTP code from the HTTP protocol. The sequences also do not describe the authentication and authorization procedures.

C.2.1.2 ONE_WAY

Fire-and-forget, sends data to consumer without expecting information back, other than a technical response as described in Figure C.1.

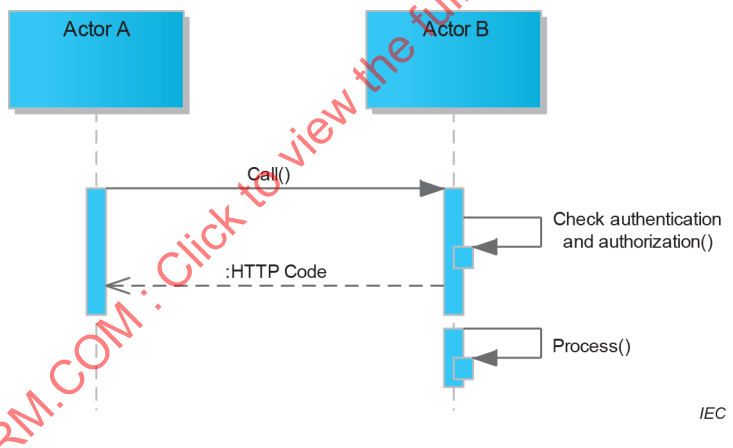


Figure C.1 – Message Exchange Pattern – ONE_WAY

C.2.1.3 REQUEST_CALLBACK

Sent messages are expected to result in message sent asynchronously back, such as optimized route, as described in Figure C.2.

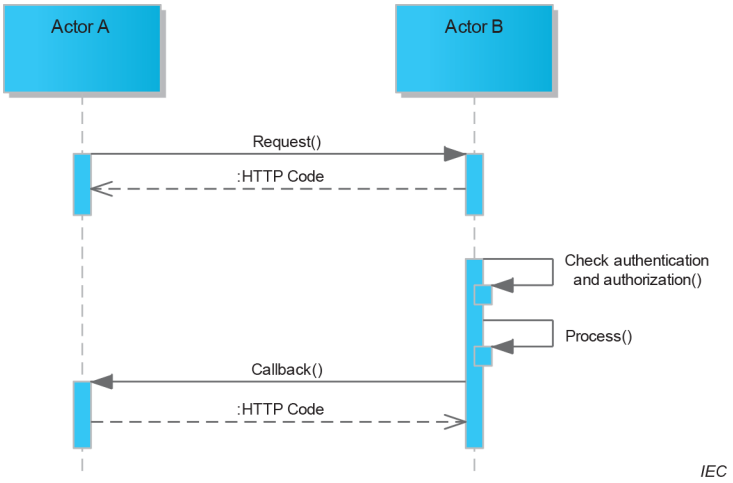


Figure C.2 – Message Exchange Pattern – REQUEST_CALLBACK

C.2.1.4 REQUEST_RESPONSE

Consumer requests information from a provider, and if authorized, information is sent back synchronously in the response, as described in Figure C.3.

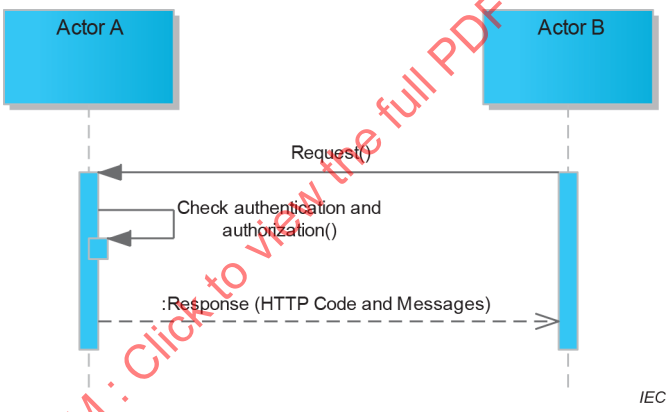


Figure C.3 – Message exchange pattern – REQUEST_RESPONSE

C.2.1.5 PUBLISH_SUBSCRIBE (Provider nominates)

The parent on the provider side nominates a consumer and sends messages until either provider or consumer ends the subscription.

All uploaded message can be combined with acknowledgement when the sent message has been forwarded to the parent, as described in Figure C.4.

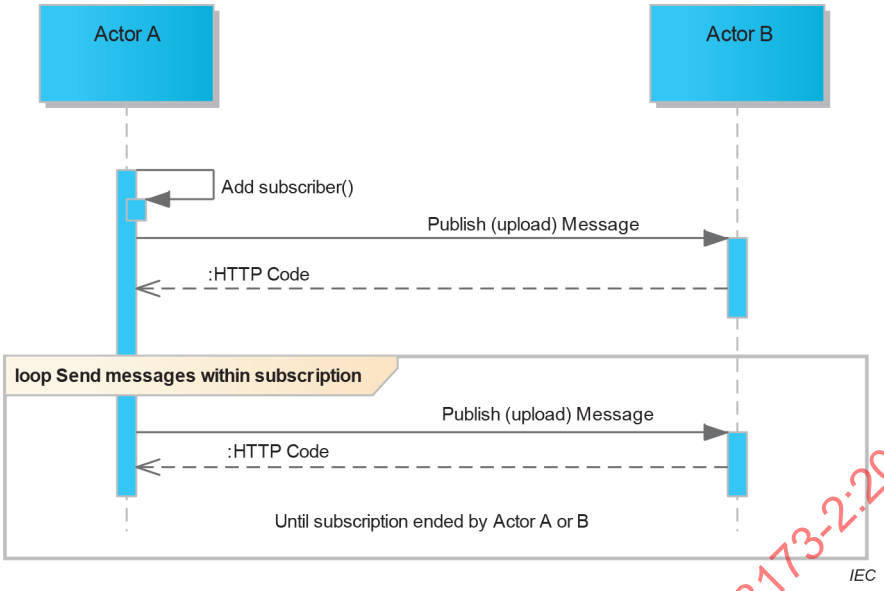


Figure C.4 – Message exchange pattern – PUBLISH_SUBSCRIBE (Provider nominates)

C.2.1.6 PUBLISH_SUBSCRIBE (Consumer requests)

Consumer requests subscription on a provider. The provider either rejects or accepts the subscription request. If accepted, message(s) in given subscription are sent, and then all updated messages are sent to the consumer until either provider or consumer ends the subscription. See Figure C.5.

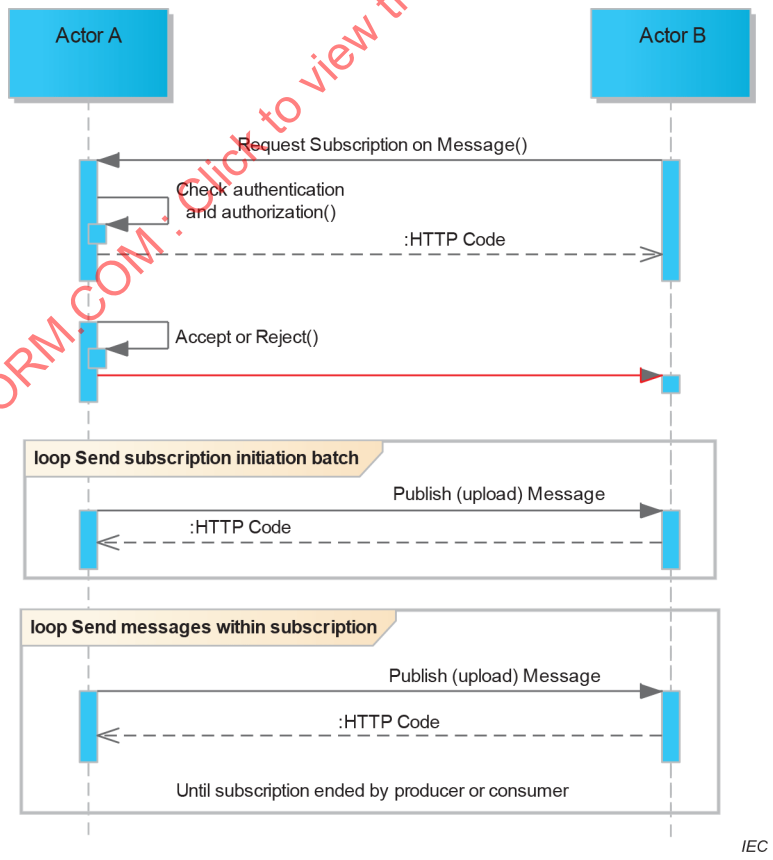


Figure C.5 – Message exchange pattern – PUBLISH_SUBSCRIBE (Consumer request)

C.2.2 Alternative and error sequences

C.2.2.1 Incorrect uploaded message

For Incorrect uploaded message, see Figure C.6.

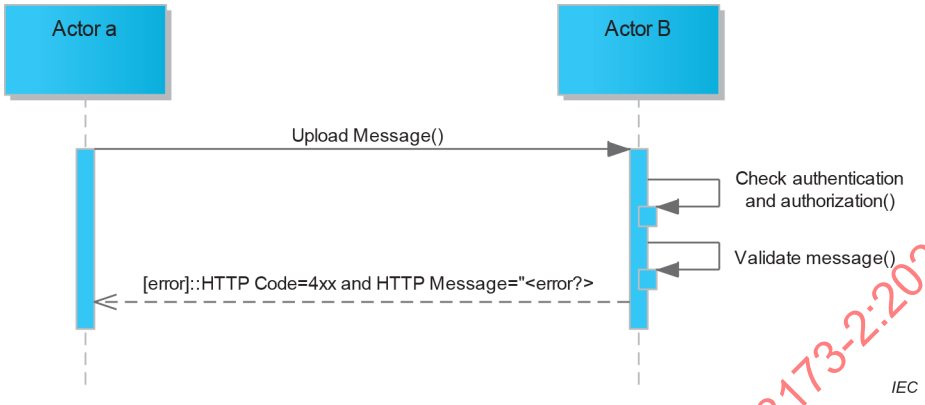


Figure C.6 – Error sequence; Incorrect uploaded message

C.2.2.2 Unauthorized uploaded message

For unauthorized upload message, see Figure C.7.

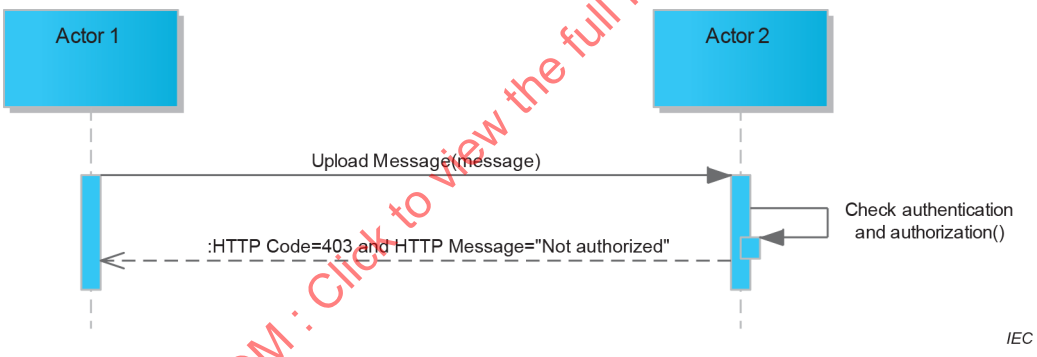


Figure C.7 – Error sequence; Unauthorized upload of message

C.2.2.3 Unauthorized subscription request

For unauthorized subscription request, see Figure C.8.

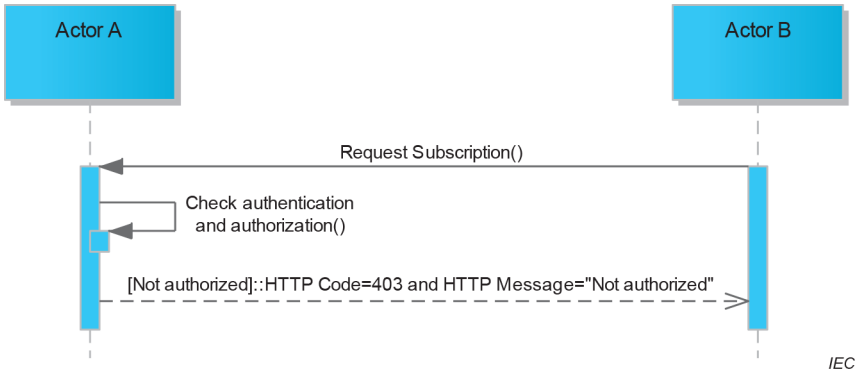


Figure C.8 – Error sequence; Unauthorized subscription request

Annex D
(informative)

Guidance on implementation

D.1 Purpose

This Annex D contains examples of SECOM deployment both onboard ship with 460-Gateway and shore side application.

Figure D.1 shows the elaborated scope for SECOM in the context of vendor specific private API towards, for example, a ship.

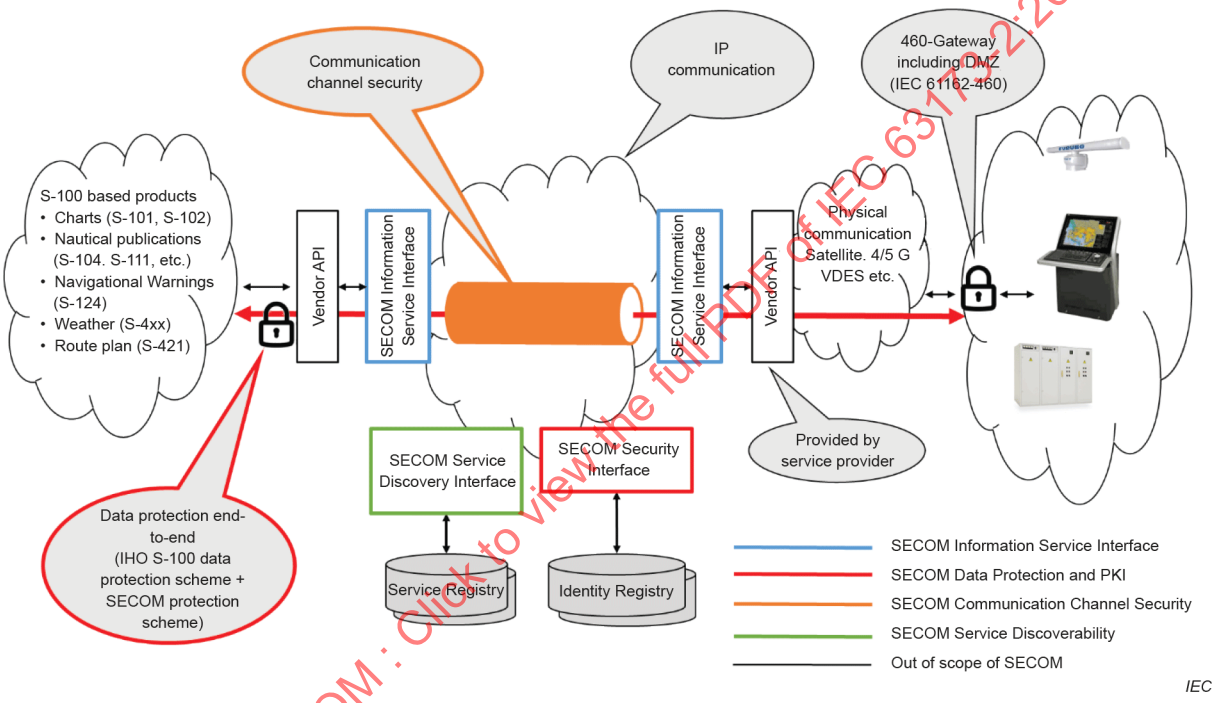


Figure D.1 – Overview of SECOM

Figure D.2 shows the different use of certificates for data authentication and communication channel protection. For security reasons, the recommendation is to use different certificates for signing of data and for encrypting the communication channel. The certificate used for signing data (data authentication) shall be a dedicated "user" certificate and the certificate used for channel protection shall be a service certificate for the actor.

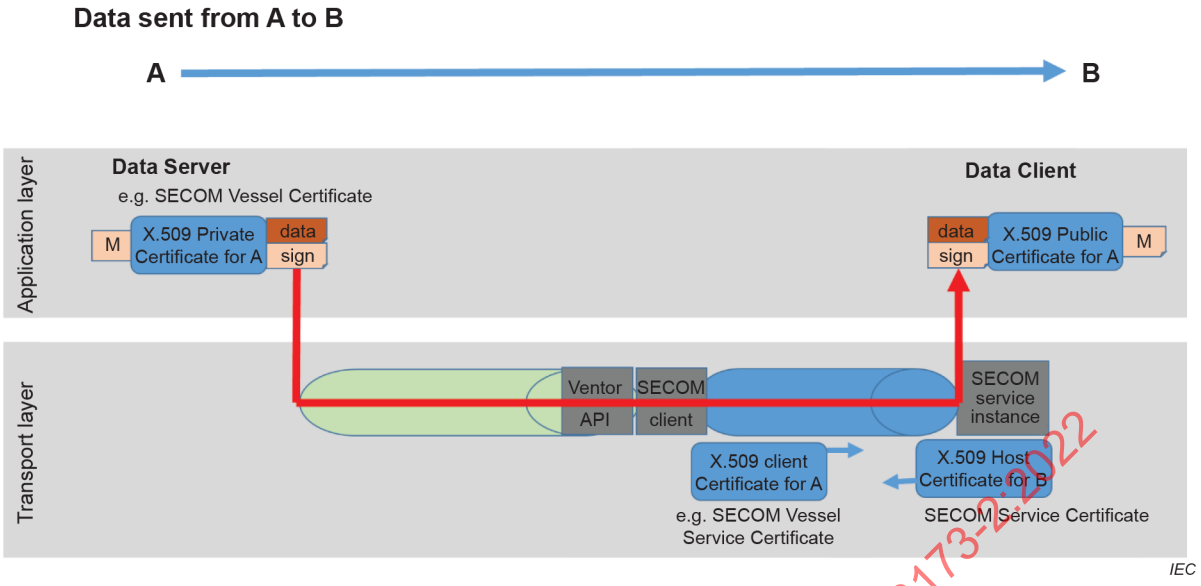


Figure D.2 – Overview of certificate usage

D.2 On ship

Figure D.3 shows a SECOM service with exposed and registered SECOM interfaces deployed outside the ship in a new DMZ.

Onboard ship:

- the external firewall on the 460-Gateway opens one outgoing port to the SECOM service, and opens ports for SECOM PKI and SECOM service registry (normally HTTPS 443);
- the firewall on the uncontrolled network on the ship opens corresponding outgoing ports from/to the SECOM service, SECOM PKI and SECOM service registry.

Outside ship:

- the exposed SECOM service for the ship is deployed outside the ship (shoreside) in a new DMZ;
- the URL to this SECOM service is registered in the SECOM service registry and can be consumed by other SECOM actors;
- the internal firewall opens one incoming port to the SECOM service;
- the external firewall opens one incoming port where the ship SECOM service is consumed, and a range of outgoing ports where the ship consumes other SECOM services.

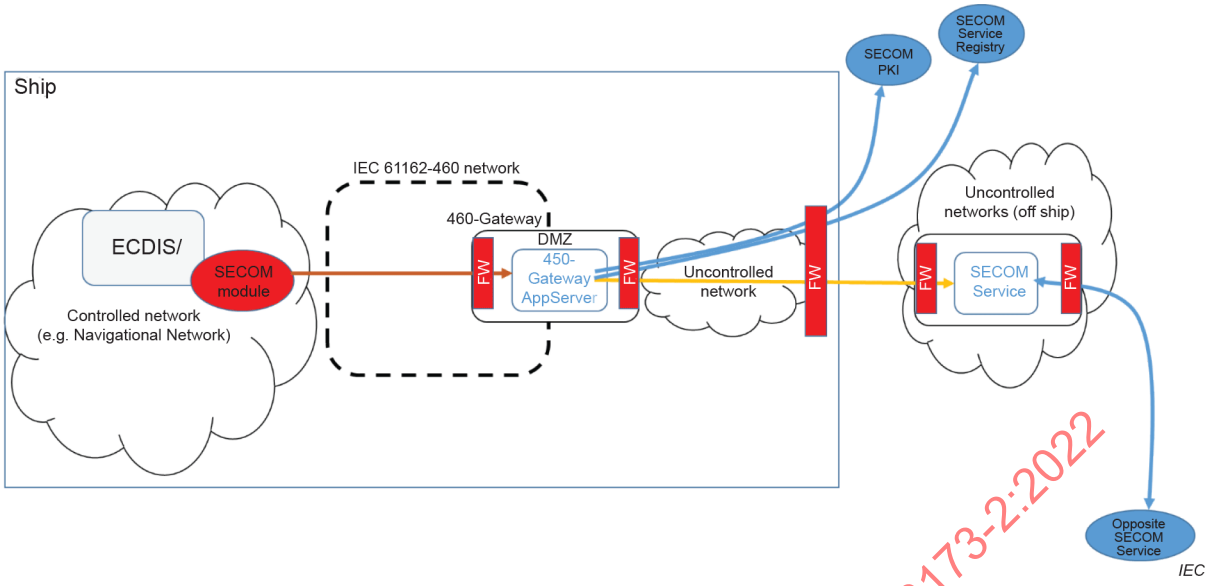


Figure D.3 – Deployment example for SECOM on ship

D.3 On shore

Figure D.4 shows where a DMZ is used between the operational application onshore and the exposed SECOM service.

- The exposed SECOM interface is deployed in the DMZ.
- The URL to this SECOM service is registered in the SECOM service registry and can be consumed by other SECOM actors.
- The internal firewall opens one incoming port to the SECOM service.
- The external firewall opens one incoming port to the SECOM service, and opens a range of outgoing ports to other SECOM services, and ports to SECOM PKI and SECOM service registry.
- The firewall on the uncontrolled network opens corresponding incoming and outgoing ports to and from SECOM service.

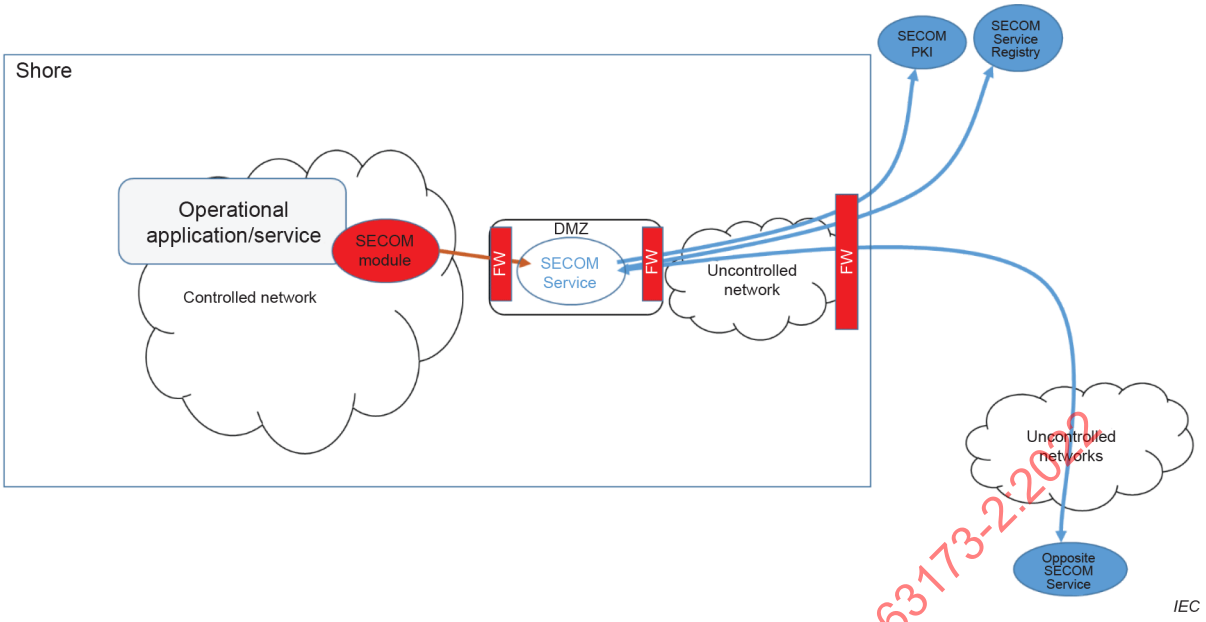


Figure D.4 – Deployment example for SECOM on shore

D.4 Service composition

There is no requirement for a monopoly of a single shore service provider or for a monopoly of a single provider of "vendor API + gateway". There is a freedom to any actor to implement services which could be useful for a ship. Further, there is no requirement that onboard at ship side there should be only a single "vendor API + gateway" to provide access to transfer of information. The reality is that there could be any amount of parallel implementation, for example S-124 services could be offered for separate geographical areas by different service providers, and for example there could be multiple implementations of identity registry and service registry either geographically separated or even geographically parallel. (See Figure D.5).

Service composition is about how to handle this multiplicity of SECOM compliant services. It is assumed that at the shore side, SECOM compliant services could be implemented ignoring the possible existence of other parallel SECOM compliant services at the shore side.

Onboard a ship, the situation might be such that there is a need to be a client for multiple SECOM compliant shore services when a single SECOM compliant shore service is unable to provide all services needed. Such a situation could be solved either by a single SECOM compliant "vendor API + gateway" or by multiple parallel SECOM compliant "vendor API + gateways" onboard the ship. In the case of a single SECOM compliant "vendor API + gateway" onboard the ship, the provider/manufacturer of the "vendor API + gateway" would implement within his transfer service a connection to all required separate implementations of SECOM compliant shore services or registries. In the case of multiple SECOM compliant "vendor API + gateways" onboard the ship, each separate provider/manufacturer of a "vendor API + gateway" would implement within his transfer service a connection to a subset of required separate implementations of SECOM compliant shore services or registries.

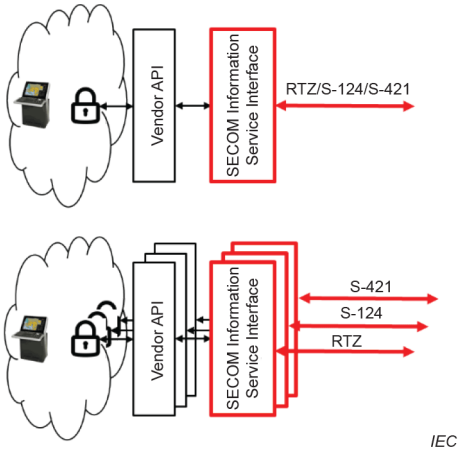


Figure D.5 – Service composition

A list of available service registries could be provided by different organisations such as IHO, IALA, CIRM or coastal states.

D.5 Private side security

Since the SECOM standard does not comprise standardization of the communication methods and interfaces on the private side, i.e. the communication link between vendor applications and its SECOM service instance(s), an example is described of a way to secure the communication. Figure D.2 shows a deployment example of a connectivity where the link between the ship network and its SECOM service needs to be secured.

In order to secure this open part of the communication, one could use the HMAC (hash-based message authentication code) authentication mechanism. HMAC is a mechanism for calculating a message authentication code using a hash function in combination with a shared secret key between the two parties involved in sending and receiving the data (frontend client and back-end HTTP service). The main use for HMAC is to verify the integrity, authenticity, and the identity of the message sender.

The server provides a public APP ID and a shared secret key (API key – shared only between server and client) to the client. The secret key exchange only occurs the first time the client registers with the server, i.e. the key is securely stored on each side. After the client and server has agreed on the API Key, for each new request the following process occurs.

1) The client creates a unique HMAC (hash) representing the request originated from it to the server. This is done by extending the request data with, for example, the public APP id, the request URI, the request content, the HTTP method, a time stamp and a nonce in order to produce a unique hash by using the API key.
2) The client then sends that hash to the server, along with all the information it was going to send already in the request.
3) When the server receives the request along with the hash, it tries to reconstruct the hash by using the received request data from the client along with the API Key. Once the hash is generated on the server, the server will be responsible for comparing the hash sent by the client with the regenerated one. If they match, the server considers this request authentic and can proceed to process it.

D.6 SECOM PKI

D.6.1 General

An informal description of SECOM PKI instance(s) is given exemplified as approached by the Maritime Connectivity Platform (MCP). The MCP build on the analysis, design choices, and experience with the testbed implementations during the EU projects EfficienSea2 and STM Validation Project and the SMART Navigation Project funded by the Republic of Korea (see https://maritimeconnectivity.net).

D.6.2 Structure and Functionality

D.6.2.1 General

The MCP specifies three core components and their interoperability: the Maritime Identity Registry (MIR), the Maritime Service Registry, and the Maritime Messaging Service. The MIR is responsible for identity management and providing security functionality to the other components. As shown in Figure D.6, the MIR consists of MIR governance and several MIR instances. In summary, MIR governance and instances together typically provide the following functionality.

1) Identity Management: The MIR enables that each maritime entity (such as a device, human, organization, service, or ship) can be registered as a participant of the MCP and be equipped with a unique identity. The identity is given in terms of a MRN (Maritime Resource Name). While MIR governance harmonizes the MRN namespace governed by the MCC and sets out criteria for the registration process, it is up to the MIR instances to implement and have certified concrete identity registries. The following terminology is used:
   a) MCP entity: an entity registered at some MIR instance;
   b) MCP namespace: the subspace of the MRN namespace that is governed by the MCC.
2) Public Key Infrastructure (PKI): The MIR enables that each MCP entity holds a cryptographic identity in terms of a public/private key pair and a certificate bound to their ID within the MCP. While the cryptographic identity of a MCP entity can change over time (due to updates of key material), the MIR ensures that each MCP entity holds only one valid cryptographic identity at any point in time bound to their ID within the MCP. MIR governance provides criteria as to the use and management of cryptographic identities but, similarly to above, it is up to the MIR instances to implement and have certified concrete PKIs.
3) Authentication and authorization for web services: The MIR enables that MCP entities benefit from login, single sign-on, and authorization for API access of web services, as well as secure integration of web services based on the widely used standards OAUTH 2.0 and OpenID Connect. To this end, MIR governance provides criteria as to interoperability and configurations while the MIR instances deliver concrete OAUTH 2.0/OpenID Connect platforms.

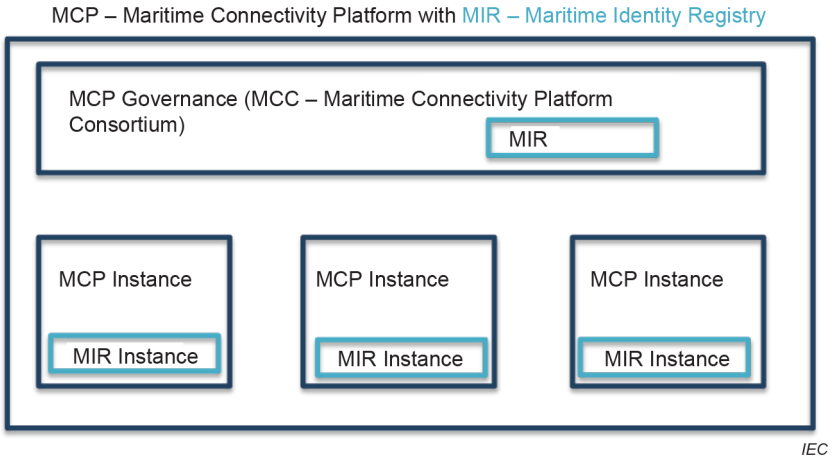


Figure D.6 – Structure of MIR within MCP

D.6.2.2 Governance and profiles

The focus of the MCP is currently changing from only providing reference implementations and a testbed to including governing several operational MCP instances as well as ensuring their interoperability. At the time of writing, there are two emerging operational instances: one is evolving from the STM project; the second is being deployed by the SMART project of the Republic of Korea. Hence, the MCP has to strike a balance between laying down criteria according to which the emerging deployments can be endorsed as MCP instances while remaining open to both, ongoing refinements of the first set of requirements (e.g. with respect to security) as well as new developments and technologies the MCP might wish to utilize (e.g. with respect to distributed PKI). This is why the MIR adopts the following approach of profiles.

The MCP will not develop a single set of criteria that every MIR instance has to comply with but rather allow several MIR profiles to coexist. Each MIR profile contains a set of requirements that define what MIR instances have to guarantee to be compliant with the profile. In addition, a profile will typically contain requirements that define what MIR governance is supposed to guarantee (e.g. to maintain operability and overall security). Each MCP instance can choose which of the current MIR profiles it aims to fulfil. While the MCC is not able to carry out assessments as to whether a MIR instance adheres to a profile itself (in particular with respect to security), it will endorse organizations that can provide this.

Two distinct MIR profiles can either be compatible in that one is a refinement of the other, or they can be non-compatible. To allow non-compatible profiles ensures that the MCP can evolve into different branches. This is to enable that an MCP instance or a cluster of MCP instances may adopt new developments without having to ensure downwards compatibility. As usual, downwards compatibility entails the risk of being forced to carry over security vulnerabilities or simply being bogged down by obsolete technology. Therefore, the approach of coexisting profiles is also meant to ensure that the MCP can evolve as a whole. The WG IDSec (Working Group Identity Security) will formulate requirements that will pin down how the profiles are managed and harmonized.

D.6.3 Identity management

D.6.3.1 The MCP namespace

As explained above, the MCP namespace is a subspace of the Maritime Resource Name (MRN) space, which is an official URN namespace. The syntax definitions below use the Augmented Backus-Naur Form as specified in RFC 5234.

The syntax for a MRN is as follows:

```
<MRN>::= "urn" ":" "mrn" ":" <OID> ":" <OSS>
            [ rq-components ]
            [ "#" f-component ]
<OID>::= (alphanum) 0*20(alphanum / "-") (alphanum)
<OSS>::= <OSNID> ":" <OSNS>
<OSNID>::= (alphanum) 0*32(alphanum / "-") (alphanum)
<OSNS>::= pchar *(pchar / "/")
```

The rules for alphanum and pchar are defined in RFC 3986.

The optional rq-components and f-component are specified in RFC 8141.

"mrn" specifies that the URN is within the MRN namespace. The Organization ID (OID) refers to an organization that is assigned a subspace of MRNs such as IMO, IALA, or the MCP. Syntactically, it is a string that is unique across the "mrn" scheme. The Organization Specific String (OSS) is specified and managed by the governing organization in a consistent way conform to the definitions of the MRN namespace. In particular, each organization structures the OSS into two parts: the Organization Specific Namespace ID (OSNID), and the Organization Specific Namespace String (OSNS). The OSNID identifies a particular type of resource (uniquely within the governing organization), while the OSNS identifies the particular resource (uniquely for its type within the governing organization). Altogether, this ensures that the resulting URN is globally unique.

For a MRN governed by the MCC, the OID reads "mcp", and the OSNID specifies one of the following types used within the MCP: device, organization, user, vessel, service, mir, mms, and msr. The latter three types are to be used for entities of the three MCP components MIR, Maritime Messaging Service, and Maritime Service Registry respectively. Moreover, the definition of the OSNS takes into account the distributed structure of the MCP. Identities can be provided and managed by several identity providers. In detail, the syntax of a MRN governed by the MCC (short: MCP MRN or MCP name) is as follows:

```
<MCP-MRN>::= "urn" ":" "mrn" ":" "mcp" ":" <MCP-TYPE> ":" <IPID> ":" <IPSS>
<MCP-TYPE>::= "device" | "org" | "user" | "vessel" | "service" |
              "mir" | "mms" | "msr"
<IPID>::= <CountryCode> | (alphanum) 0*20(alphanum / "-") (alphanum)
<IPSS>::= pchar *(pchar / "/")
```

"mcp" specifies that the governing organization is the MCC. The next element is MCP-TYPE. As explained above, this pins down one of the types currently used within the MCP. The Identity Provider ID (IPID) refers to a national authority or other kind of organization that acts as an identity provider within the MCP. If the identity provider is a national authority, then the IPID is a country code as defined by ISO 3166-1 alpha-2. Otherwise it will be a string of the same syntax as that for OIDs. The IPID is unique across the urn:mrn:mcp namespace. The Identity Provider Specific String (IPSS) can be defined and managed by the respective identity provider in a way that is consistent and conforms to the definitions of the MRN namespace and requirements laid down by the MCC. In particular, the identity provider ensures that the IPSS identifies a particular resource uniquely for its type within the domain of the identity provider. Altogether, this will ensure that the resulting URN is globally unique.

Examples:

- urn:mrn:mcp:user:dma:alice – valid MCP MRN for a user, where dma specifies the ID Provider, and the subsequent IPSS string is defined to give the username.
- urn:mrn:iala:aton:gb:sco:6789-1 – valid MRN for a marine aid to navigation (AtoN), where gb stands for United Kingdom, sco for Scotland, and the number is the Scottish asset identifier. In this example, this is not a MCP MRN.
- urn:mrn:mcp:device:mirX:aton:gb:sco:6789-1 – valid MCP MRN for the same AtoN, where mirX specifies the ID Provider, and the subsequent IPSS string is defined to first specify the type of the device, and then to follow the country-specific convention of the IALA scheme.
- urn:mrn:mcp:service:mirY:org1:instance:s124/busan – valid MCP MRN for a service instance, where mirY and org1 specifies the ID Provider and the organization respectively. Note the use of the character slash ("/") in the last element. It should be converted to percent-encoded ‘%2F’ by following RFC 8141 when the MRN is included as a part of a URL, for example, "http://example.com/urn:mrn:mcp:service:mirY:org1:instance:s124%2Fbusan".

The following requirements pin down how the MCP namespace can be managed decentrally.

- ID1 The MCC can delegate the assignment of part of the MCP namespace to other organizations that act as identity providers. More concretely, this means that the organization, say X, has to hold an IPID, say string "nameofx", and is then responsible for the namespace with the prefix "urn:mrn:mcp:<MCP-TYPE>:nameofx".
- ID1.1 The MCC ensures that each IPID refers to at most one identity provider.
- ID1.2 Each Identity Provider ensures to respect all syntax prescribed in the MRN specification. Moreover, each Identity Provider ensures that each IPSS of their name space refers to at most one entity of their domain.
- ID1.3 The MCC can give recommendations on how to structure the IPSS, for example to harmonize the syntax for particular types of entities. These recommendations will not be binding. However, the MCC reserves the right that a particular syntax can be binding with respect to conformance to certain profiles.

Note that ID1.1 and the second part of ID1.2 together ensure uniqueness: one MCP MRN is assigned to at most one entity. This is a general requirement for any URN. ID1.3 allows to harmonize the IP specific strings while not principally restricting the governance of an IP provider over its namespace.

EXAMPLE There are two ID providers, MIR X and MIR Y. Assume the MCC assigns the IPID "mirx" to MIR X, and "miry" to MIR Y respectively. The MCC ensures that the strings "mirx" and "miry" are not assigned to any other MIR. MIR X is responsible for the namespace "urn:mrn:mcp:<MCP-TYPE>:mirx:*", and MIR Y is responsible for the namespace "urn:mrn:mcp:<MCP-TYPE>:miry:*" respectively. They might decide to employ the same syntax for the IP specific string, and make this part of a profile they both adhere to. Other ID providers are not bound to use the same syntax. However, if they do not comply to it, they cannot be compliant to that profile.

Finally, the following is to ensure a good practice of transparency and interoperability:

- ID2 Every Identity Provider shall publish the syntax that describes their name space as well as provide a reference implementation that recognizes the strings of their namespace.

D.6.3.2 Further requirements for a strong notion of maritime identity

The vision of the MCP is to enable a strong concept of digital maritime identity. Hence, requirements that go beyond what is commonly required of URNs are established. Firstly, every MCP entity shall have a name within the MCP namespace. This gives a clear concept of MCP entity: those entities that are registered under an MCP MRN name. Secondly, one MCP entity shall not have several MCP MRNs. For example, this supports law enforcement: when a maritime entity gets discovered and blacklisted for "bad behaviour" (e.g. fake emergency signalling), then it cannot simply revert to another MCP identity and participate as usual.

- ID3 Every entity of the MCP shall hold exactly one MCP MRN (i.e. MRN governed by the MCP). This does not exclude that a MCP entity can hold other MRNs, but these shall be within namespaces governed by other organizations (e.g. IMO). Also, we will formulate exceptions concerning legacy MRNs within the MCP namespace.

Hence, the AtoN in the example above can be identified by its IALA MRN, or its MCP MRN respectively. However, Requirement ID3 rules out that the AtoN can be referred to by a second MCP MRN. The following requirements implement ID3 in a decentral manner.

- ID3.1 Each Identity Provider shall ensure that each entity they register holds at most one MCP MRN within their namespace.
- ID3.2 Each holder of a maritime entity shall ensure that this entity is registered with at most one MCP identity provider.

Note that practically it will not be possible to avoid that a "bad player" will seek to register their entity at several different Identity Providers and thereby obtain several MCP identities for it. However, ID3.1 ensures that they can obtain at most as many identities as there exist Identity Providers. And ID3.2 ensures that, when it is discovered that an entity holds several MCP MRNs of different providers, then it is clear that they have violated a rule (and action can be tied to this).

D.6.4 Public Key Infrastructure

D.6.4.1 General

In addition to a unique ID in the form of a MCP MRN, each MCP entity is provided with a cryptographic identity. This consists of a public/private key pair and a certificate for the public key bound to their ID. In the following, the concept of the PKI that enables this, as well as a first set of requirements for it, are described. Issues that need to be addressed and refined in the future are also identified.

In D.6.4.2.1, MCP core concepts of cryptographic identity and decentral PKI are explained. In D.6.4.3, requirements on cryptographic keys and mechanisms are specified. In D.6.4.4, the format of MCP certificates is described. Moreover, D.6.4.5 shows how a service can use an intermediary level of service certificates. For example, this is necessary if a service comes with cryptographic requirements that do not allow the direct use of the MCP ID credentials. Finally, in D.6.4.6, further aspects to be considered are identified.

D.6.4.2 Core concepts

D.6.4.2.1 Cryptographic identity

The cryptographic ID of a MCP entity consists of a public/private key pair and a certificate bound to their MRN. The certificate is issued by the identity provider responsible for the entity. The latter is clearly defined by the IPID string within the MRN of the entity.

Given an entity with MRN A (short: entity A), and its identity provider P, we use the following notation:

- pkA is the public key of A, and prA is the private key of A respectively;
- certP(A, pkA, V) is the certificate of A signed by its identity provider P. The certificate contains the MRN A, the public key of A, and the validity period V of the certificate (the precise format is provided in D.6.4.4).

The key pair is for use with a digital signature scheme. Hence, each MCP entity A can be verified by another party B to be the originator of a message or other data. As usual, this involves the following steps.

1) Entity A signs the message, say M, using its private key prA. The result is a cyphertext C.
2) Entity A makes available its certificate certP(A, pkA, V), and transmits the signed message M||C.
3) Entity B obtains the certificate, and receives the signed message.
4) Entity B validates the certificate. As a result, B trusts that pkA is the valid public key of the MCP entity with MRN A. (Necessary requirements on certificate validation will be specified.)
5) Entity B uses pkA to verify whether the ciphertext C is indeed the digital signature of M. If the verification is successful, then B has assurance that M indeed originates from A.

NOTE 1 Without 4), B only has assurance that M originates from the holder of the private key counterpart of pkA.

NOTE 2 B does not necessarily need to be an MCP entity.

At the time of writing, the MCC does not prescribe a policy on how to use ID credentials. They could be used as long-term credentials to obtain short-term credentials for use for a service, or they could be directly used as working credentials. Also see D.6.4.5 on a related issue.

D.6.4.2.2 Decentral PKI

One of the principles of the MCP is to make do without a global notion of trust: in the international context of the MCP, it is not possible to expect that all parties trust each other and each other's security management uniformly. Rather, the goal of the MCP is to provide the transparency that enables organizations to decide on whom to trust in which context, and to provide the technical framework to translate such decisions into executable policies. For the PKI, we put forward the following three principles.

1) A security breach within the realm of one identity provider's PKI instance shall not enable an attacker to impersonate an entity within the realm (i.e. namespace) of another identity provider.
2) A security breach within an organization or set of organizations predefined by the MCC to carry out some task shall not enable an attacker to impersonate any MCP entity, unless the identity provider of the entity coincides with (one of) the organization(s). In short, this means identity providers' PKI instances can always remain secure independently from any central or distributed management by the MCC.
3) It is possible for everyone to obtain assurance as to the security level of any identity provider's PKI instance.

The first two principles immediately imply that a classical hierarchical PKI with a root CA hosted by the MCC will not do. We illustrate this by giving examples of impersonation attacks. Assume a hierarchical PKI structure with MCC root CA as shown in Figure D.7. To verify a MCP certificate certP(A, pkA, V), a receiving party has to verify the signature of the identity provider P with the public key of P provided in an intermediary certificate certMCC(P, pkP, V) issued by the MCC root CA. Further, to verify the intermediary certificate, the receiving party has to verify the signature of the MCC with the public key provided in the MCC root certificate certMCC(MCC, pkMCC, V). The latter provides the trust anchor accepted by the receiving party.

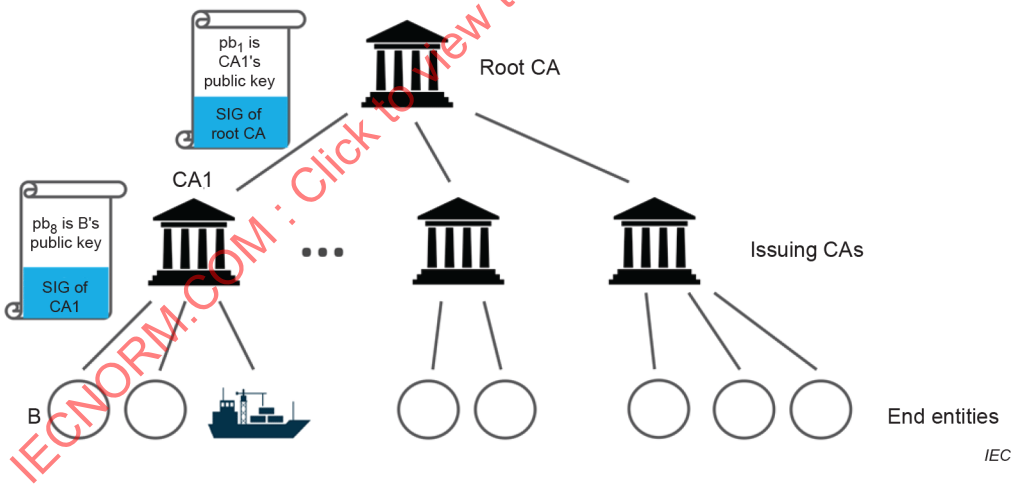


Figure D.7 – Hierarchical X.509 PKI Structure

Examples:

- Single point of attack: Assume the MCC root key prMCC is compromised. This will allow an attacker to impersonate any MCP entity A. Say P is the identity provider responsible for A. First, the attacker generates a key pair pkI(P), prI(P) and generates a fake certificate for P with their own key pkI(P): certMCC(P, pkI(P), V). This is possible since the attacker knows prMCC. Second, the attacker generates a key pair pkI(A), prI(A) and generates a fake certificate for A and their own key pkI(A): certI(P)(P, pkI(A), V). This is possible since they know prI(P). Altogether, the attacker can now present a valid certificate chain that establishes pkI(A) to be the public key of A while they know the private counterpart prI(A). Hence, they can impersonate A. Altogether, this violates principle 2) above.

- Weakest Link I: Say the attacker wishes to impersonate entity A of identity provider P. Note that, in classic X.509 certificate validation, it is only verified that there is a certificate chain up to a trusted root certificate. Say the attacker can easily obtain fake certificates signed by another identity provider P', perhaps, because the attacker is a state actor and P' is under their governance. Then, analogously to above, they only need to generate their own key pair pbI(A), prI(A) and generate a fake certificate for A and pbI(A) signed by P': certP'(A, pbI(A), V). Since there is no check whether P' is indeed the identity provider of A, this gives the attacker a valid certificate chain and corresponding private key, with which they can impersonate A. This violates principle 1) above.
- Weakest Link II: Assume P is an identity provider of low security level, for example with a vetting procedure that can easily be undermined. Assume an attacker aims to join the MCP under a false identity so that they are able to inject fake messages without the risk of being traced. The attacker will simply choose P as the identity provider from whom to obtain their false identity. Without principle 3) in place, a receiving party has no way to consider the low security level of P when processing the information within the message.

This motivates the following requirements.

- PKI1.1 (PKI Structure) There shall be no root CA at the top level of the MCC. Every identity provider that hosts a PKI instance shall provide their own root CA.
- PKI1.2. (Validation of IPID) When a receiving party verifies a MCP certificate, say certP(A, pkA, V), it shall verify that the certificate is indeed signed by the identity provider responsible for A. The identity provider responsible for A can be read by the receiving party from the IDIP string within the MRN A.

The following requirements ensure that information on root certificates and security levels are made publicly available.

- PKI1.3. Every identity provider shall publish their currently valid root certificate in a suitable fashion. For example, this can be made accessible via their web page, or they can commission a generally accepted authority or assurer to do so.
- PKI1.4. Every identity provider shall make their security level publicly accessible in a suitable fashion. For example, they can provide a link from their web page to the database of their certification body.
- PKI1.5. The MCC will provide a list of identity providers, links to obtain their root certificates, and security levels. However, the MCC will not provide any warranty for this information.

D.6.4.2.3 Security requirements and profiles

Security requirements to be defined will fall into the following categories:

1) requirements on vetting – this can be specified similarly to classes such as EV (extended validation);
2) requirements on certificate revocation;
3) requirements on the validity period of certificates;
4) requirements on security of keys and origin of signing – CA side (including requirements on HSMs);
5) requirements on security of keys and origin of signing – MCP entity side (including requirements on HSMs).

The requirements will be dependent on the currently emerging profiles:

- MCP entities generate their ID key pair themselves and in own responsibility and provide this to the responsible CA for certification.
- The CA (perhaps together with a manufacturer) provisions the initial ID key pair and certificate securely within HSMs (for/within endpoints) to be distributed to the MCP entities.

D.6.4.3 Cryptographic requirements

The cryptographic mechanism approved for ID digital signatures is the elliptic curve digital signature algorithm or ECDSA (FIPS 186-3) with the appropriate hash algorithm from the SHA-2 family (FIPS 180-3). The approved elliptic curve domain parameters are specified by reference to standardized curves. Table D.1 gives the combinations that are approved.

Table D.1 – Domain parameters

ECDSA key size (bits)	Hash algorithm	Elliptic curve domain parameters
384	SHA-384	P-384 [FIPS 186-3] (= secp384r1)
256	SHA-256	P-256 [FIPS 186-3] (= secp256r1)

D.6.4.4 Certificate format

The format of the MCP ID certificates is specified in D.6.4.4. The format is based on the X.509 standard. The standard information present in an X.509 certificate includes the following items:

- version – which X.509 version applies to the certificate (which indicates what data the certificate includes);
- serial number – a unique assigned serial number that distinguishes it from other certificates;
- algorithm information – the algorithm used to sign the certificate;
- issuer distinguished name – the name of the entity issuing the certificate (MCP);
- validity period of the certificate – start/end date and time;
- subject distinguished name – the name of the identity the certificate is issued to;
- subject public key information – the public key associated with the identity.

The subject distinguished name field consists of the items in Table D.2.

Table D.2 – Subject distinguished name field items

Field	User	Vessel	Device	Service	MMS	Organization
CN (CommonName)	Full name	Vessel name	Device name	Service Domain Name	MMS name	Organization Name
O (Organization)	Organization MRN					
OU (Organizational Unit)	"user"	"vessel"	"device"	"service"	"mms"	"organization"
E (Email)	User email					Organization email
C (Country)	Organization country code					
UID	Entity MRN					Organization MRN

The following gives an example of the Subject distinguished name field for a vessel with identity provider idp1:

C=DK, O=urn:mrn:mcp:org:idp1:dma, OU=vessel, CN=JENS SØRENSEN, UID=urn:mrn:mcp:vessel:idp1:dma:jens-soerensen

In addition to the information stored in the standard X.509 attributes listed above, the X509v3 extension SubjectAlternativeName (SAN) extension is used to store extra information. There already exists some predefined fields for the SAN extension, but they do not match the need in maritime related fields. Therefore, the "otherName" field is used, which allows for using an object identifier (OID) to define custom fields. The OIDs currently used are not registered at ITU, but are randomly generated using a tool provided by ITU (see http://www.itu.int/en/ITU-T/asn1/Pages/UUID/uuids.aspx). Table D.3 shows the fields defined, the OIDs of the fields and which kind of entities that use the fields.

Table D.3 – Fields and object identifiers

Field	OID	Used by
Flagstate	2.25.323100633285601570573910217875371967771	Vessels, Services
Callsign	2.25.208070283325144527098121348946972755227	Vessels, Services
IMO number	2.25.291283622413876360871493815653100799259	Vessels, Services
MMSI number	2.25.328433707816814908768060331477217690907	Vessels, Services
AIS shiptype	2.25.107857171638679641902842130101018412315	Vessels, Services
Port of register	2.25.285632790821948647314354670918887798603	Vessels, Services
Ship MRN	2.25.268095117363717005222833833642941669792	Services
MRN	2.25.271477598449775373676560215839310464283	Vessels, Users, Devices, Services, MMS
Permissions	2.25.174437629172304915481663724171734402331	Vessels, Users, Devices, Services, MMS
Subsidiary MRN	2.25.133833610339604538603087183843785923701	Vessels, Users, Devices, Services, MMS
Home MMS URL	2.25.171344478791913547554566856023141401757	Vessels, Users, Devices, Services, MMS
URL	2.25.245076023612240385163414144226581328607	MMS

Encoding of string values in certificates follows the specifications defined in RFC 5280, and where possible it is highly recommended to use UTF-8.

D.6.4.5 Service certificates

Several maritime services come with requirements concerning cryptography and/or certificate formats that might make it impossible to employ MCP ID credentials directly. For example, if an identity provider issues certificates for ECDSA with 384 bits key size, this will not meet the real-time requirements and low bandwidth conditions of Automatic Identification System (AIS) and VHF Data Exchange System (VDES). While the service has to then provide its own CA, the service CA can automatically issue its service certificates based on MCP ID credentials. We provide an example of how this can be done based on the concept of certificate signing requests (CSRs), also known as certification requests. The most common format for CSRs is defined by the PKCS#10 standard RFC 2986.

In the following, the steps carried out by an MCP entity to request a service certificate, and the steps performed by the service CA to issue the certificate respectively are shown as an example. The example follows the implementation of the Haptik CA from the project Haptik. This functionality will also be embedded in a web service and secured by the MCP OpenID Connect/OAuth 2.0 framework.

The MCP entity

1) generates a fresh key pair for use with the service,
2) builds a X.500 name for use in the service certificate,
3) builds a corresponding PKCS#10 CSR,
4) signs the CSR with their private MCP ID key, and
5) sends the CSR together with their MCP ID certificate to the service CA.

On receipt, the service CA

6) checks whether the CSR is valid,
7) builds a X.509v3 certificate according to the CSR and additional information provided by the CA such as issuer, serial number and validity period,
8) signs this with their CA private key, and
9) sends the new certificate to the requesting MCP party.

NOTE This pattern is also applicable when the MCP ID keys are mainly used as enrolment keys to obtain shorter lived "working keys".

D.6.4.6 Integration of other PKI systems

In the spirit of decentrality, the PKI remains open for PKI systems other than X.509, and also agile for update of certificate formats. Care will be taken to accommodate the necessary flexibility when defining usage of certificates. More to this point is provided by example of the P3KI approach.

D.6.5 Authentication and authorization for web services

For authentication and authorization in web services, the MCP offers users the ability to use OpenID Connect (OIDC). OpenID Connect is a token based authentication protocol that is built as a layer on top of the OAuth 2.0 authorization protocol.

A service provider can choose to use the MCP based OpenID Connect authentication for their web service. This means that when a user wants to use the service, they first login with their MCP credentials and receive an OIDC token from MCP that contains information about the user and what organization they belong to. The user can then use this token to authenticate themselves with the web service. The service provider can also choose to implement authorization based on the information in the token.

Table D.4 shows the claims that an MCP OpenID Connect token contains.

Table D.4 – MCP OpenID Connect token

Attribute	Description
preferred_username	The username of the user in the parent organization.
email	The email of the user.
given_name	First name of the user.
family_name	Last name of the user.
name	Full name of the user.
org	The Maritime Resource Name of the organization the user is a member of.
permissions	List of permissions for this user assigned by the organization the user is a member of.
mrn	The Maritime Resource Name of the user.
roles	List of roles that the user has in the role hierarchy.
acting_on_behalf_of	List of organizations that the user is allowed to act on behalf on excluding the user's own organization

D.6.6 Profile "Basic Requirements"

The profile "Basic Requirements" V1.0 consists of the following requirements:

1) Identity Management
   - MCP MRN syntax as specified in D.6.3.1.
   - ID1, ID1.1 to ID1.3: decentral management of MCP MRNs.
   - ID2: transparency of syntax.
   - ID3, ID3.1 to ID3.2: strong notion of MCP entity.
2) PKI
   - PKI1.1 – PKI1.5: decentral PKI concept.
   - The cryptographic requirements as specified in D.6.4.3.
   - The certificate format as specified in D.6.4.4.

D.7 SECOM service discovery

D.7.1 Example 1: geometry combined with serviceType search

Request

Search for services with provided geometry inside service coverage area and service type "Port Call Synchronization". See Figure D.8 and Figure D.9.

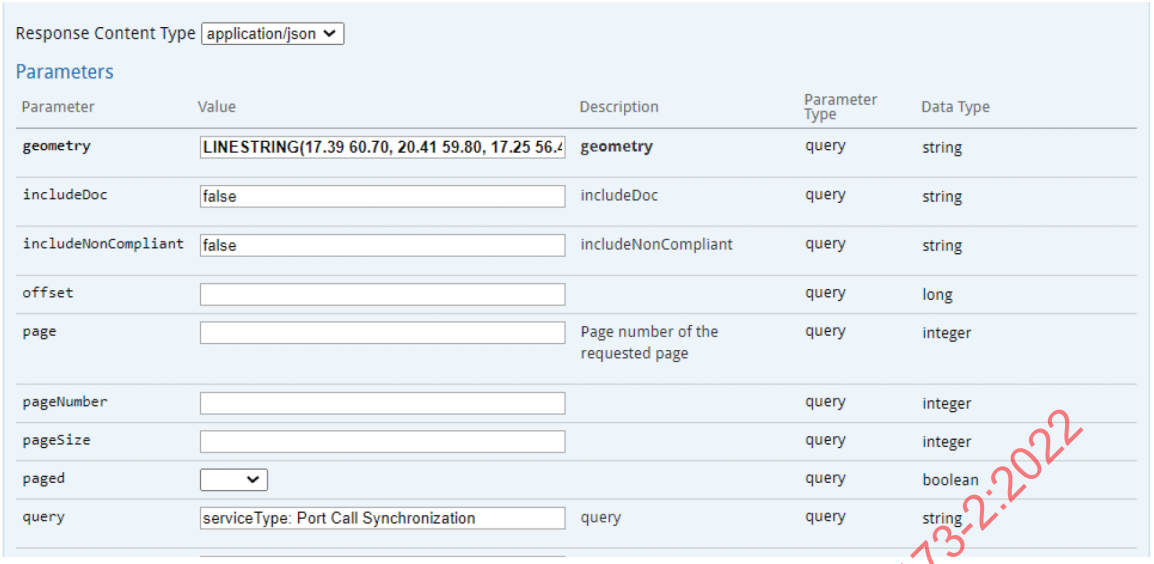


Figure D.8 – Request find service with geometry and query

Query =

https://serviceregistry.navelink.org/api/_searchGeometryWKT/serviceInstance?geometry=LINESTRING(17.39 60.70, 20.41 59.80, 17.25 56.43)&includeDoc=false&includeNonCompliant=false&query=serviceType%3A Port Call Synchronization

Response


```
[
  {
    "id": 136,
    "name": "Port of Gavle",
    "version": "1.0",
    "publishedAt": "2020-11-16T10:53Z",
    "lastUpdatedAt": "2020-11-18T13:50Z",
    "comment": "This service is used to report arrival times to Port of Gävle and get confirmation or recommended RTA...",
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [
            17.011284173205137,
            60.6578395558192
          ],
          [
            17.390312493517627,
            60.86243493611962
          ],
          [
            17.011284173205137,
            60.6578395558192
          ]
        ]
      ]
    },
    "geometryContentType": null,
    "keywords": "Voyageplan,Route,VIS,RTZ,ETA,SEGVX,SEKAS,Gavle",
    "status": "released",
    "unlocode": "SEGVX",
    "mmsi": "",
    "imo": "",
    "instanceAsXml": {
      "id": 147,
      "name": "SMA-MCP-PRODUCTION_Service_Instance-Port_of_Gavle_PCS.xml",
      "comment": "",
      "content": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\r\n<ServiceInstanceSchema:serviceInstance ...",
      "contentContentType": "text/xml"
    },
    "instanceAsDoc": null,
    "implementedSpecificationVersion": null,
    "docs": null,
    "compliant": true,
    "instanceId": "urn:mrn:mcp:service:navelink:sma:instance:vis:portofgavle",
    "organizationId": "urn:mrn:mcp:org:navelink:sma",
    "endpointUri": "https://vis.stm.sjofartsverket.se/portofgavle",
    "endpointType": null,
    "serviceType": "Port Call Synchronization",
    "designId": "urn:mrn:mcp:service:navelink:navelinkdesign:vis:rest:2.2",
    "specificationId": "urn:mrn:mcp:service:navelink:navelink:specification:vis:2.2"
  },
  {
    "id": 155,
    "name": "Facilitation of ship to shore reporting service",
    "version": "1.0.0",
    "publishedAt": "2020-11-19T13:48Z",
    "lastUpdatedAt": "2020-11-19T13:55Z",
    "comment": "The route plan of a ship must be reported to a coastal ..",
    "geometry": {
      "type": "Polygon",
      "coordinates": []
    }
  }
]
```


IEC

Figure D.9 – Response from service registry

D.7.2 Example 2: Search with AND/OR condition

Request

Search for services with specific IMO and MMSI OR services with name containing "Baltic".

Query = (imo: 9443255 AND mmsi: 276779000) OR name: Baltic

https://serviceregistry.navelink.org/api/_search/serviceInstance?includeDoc=false&includeNonCompliant=false&query=(imo: 9443255 AND mmsi: 276779000) OR name: Baltic

Response

See Figure D.10.

```
[
  {
    "id": 62,
    "name": "Baltic Queen",
    "version": "1.0.0",
    "publishedAt": "2020-11-09T15:29Z",
    "lastUpdatedAt": "2020-11-26T15:06Z",
    "comment": "Ship voyage information mainly for meeting point calculation."
  },
  {
    "id": 61,
    "name": "Baltic Princess",
    "version": "1.0.0",
    "publishedAt": "2020-11-09T15:29Z",
    "lastUpdatedAt": "2020-11-26T15:06Z",
    "comment": "Ship voyage information mainly for meeting point calculation."
  },
  {
    "id": 148,
    "name": "Baltic Bright",
    "version": "1.0.0",
    "publishedAt": "2020-11-17T12:56Z",
    "lastUpdatedAt": "2020-12-03T09:17Z",
    "comment": "Provides voyage plans for Baltic Bright in RTZ 1.1STM."
  },
  {
    "id": 143,
    "name": "Baltic Navigational Warning Service",
    "version": "0.1",
    "publishedAt": "2020-11-16T13:51Z",
    "lastUpdatedAt": "2020-12-03T13:49Z",
    "comment": "The service provides Navigational Warnings in the Baltic region and Swedish T&P info."
  }
]
```


IEC

Figure D.10 – Response from service registry

Annex E
(informative)

Use of white list

E.1 Purpose

A white list is an access control list of actors to whom a protected actor has granted access to information (for example from whom it is possible to receive S-421 based route plans). The protected actor manages its access control list. New entries for the access control list could originate from internal needs of the protected actor (for example, protected actor needs reference route to enter a port, protected actor needs weather optimization from external service provider) or from external needs (for example, a VTS centre would like to know the route plan of the protected actor i.e. vessel).

A white list is commonly used to enhance cyber security as it limits access to the information owner to other actors which have been granted the access before their attempts to access the information owner. The white list is also an effective tool against misbehaving by other actors which could be granted the access when properly behaving.

This Annex E contains a description and discussion around access principles, authorization and the internal use of white list to mitigate unwanted requests for information.

The white list management may include the methods below:

- the protected user as the owner of the information (for example, a ship in the case of a voyage plan) initiates the interaction and thereby spontaneously white-lists a service;
- other actors request access through an interface: if this is allowed, there can be "rules for behaviour", for example, how often an actor is allowed to send a request;
- the IMO or IALA e-navigation may evolve to provide "endorse services" through a trusted third party;
- other actors use a semi-administrative method (for example an automated email) to request white-listing from the protected actor. This may be supported by an entry in a service registry with an "administrative" email address for manual processing of such requests.

E.2 Authorization to access data

The next step after authentication of the actor is authorization, hence when the actor is known, the question is if the actor is allowed to read data from the protected actor, or if the actor is allowed to access the service of the protected actor.

The recommendation is to always consider authorization both to the service as well as authorization to data. The authorization depends on the design. The design decision can be that data is constrained to authorized actors only, or that the data is open for all actors; the importance is that this authorization is a conscious design decision.

The discussion and guidelines below is based on the assumption that data is exchanged through an information service that can perform authentication and authorization on the request and on its data. The fundamentals of the discussions are also to adhere to data authorization in general with the difference that, if there is no information service that handles the authorization, the data is forwarded until there is a component that can perform the authorization procedure.

Hence, the authorization procedure can be implemented in the information service representing the information owner, and/or authorization procedure can be implemented in the application of the information owner. If the information owner is a ship, the authorization may take place on shoreside service representing the ship and/or it may take place in the onboard systems.

- Authorization/access to upload (push) information to the protected actor:
  - the data server may be authorized based on predefined rules or list; or
  - the data server may get the response Not authorized to push data and may need to request access from the protected actor.
- Authorization/access to download/retrieve/get (pull) information from the protected actor:
  - the data client may be authorized based on predefined rules or list; or
  - the data client may get the response Not authorized to pull data and may need to request access from the protected actor.
- Authorization/access to be added in a subscription list of a service provider, hence new or updated data according to the subscription list of the service provider is uploaded (pushed) to the subscriber (i.e. protected actor)
  - If the subscriber (i.e. protected actor) is appointed (nominated) by the data server, the subscriber (data client) may also automatically be given access to data according to a subscription list.
  - If a data client (i.e. protected actor) requests to be a subscriber, the authorization procedure of the service provider is engaged:
    a) the data client may be authorized based on predefined rules or list of the service provider; or
    b) the data client may get the response Not authorized to data subscription and may need to request access from the service provider.

E.3 Access control list

From a ship point of view, access control is about who has rights to access data from or to the ship. Reasons to limit the access can be various, for example a ship does not wish that other actors see how they plan weather optimization, a ship wishes to limit who can ask for their data (motivation could be for example reduction of the management burden to accept or reject request), a ship wishes to block an actor (for example, if the actor is harmful, too aggressive, causes too much administrative burden, causes too much communication cost to be paid by the ship, etc.), a ship wishes to limit who can propose changes to the plans of the ship, etc.

The recommendation is to implement an access control list that contains the identifier to data and a list of actors with access rights to the data (white list). The use or no use of this facility to limit access will be under control of the ship.

The identifier to data can be to individual data objects (for example, S-421.RouteId) or a group of data based on filtering parameters (for example, S-421.routeStatus=Monitored or Terminated). It may also be to all data of a certain product (for example, all S-124 Navigational Warnings).

The actor is recommended to be identified with an identity from SECOM PKI, which ensures binding to keys and authenticated actor.

The access rights are typically specified as Read, Create, Update, Delete or None.

E.4 Authorization based on predefined rules or list

Both the data server and data client are recommended to implement a predefined white list (access control list) that can be

- manually updated by an operator or administrator,
- automatically updated based on rules, or
- a combination of both.

E.5 Manually updated list

The manually updated access control list is foreseen to be predefined by some administrator onboard or by a fleet operation centre. During sailing, there may be less time for manual update of the access control list.

E.6 Rule based handling on request to information (rule based authorization)

Rule based authorization requires more knowledge and trusted information about the requesting actor. Some examples of a rule are:

- allow read access to my S-421 Route Plan for all actors or services provided by (coastal state?) authority along my route;
- allow upload of S-421 recommended routes from all actors/services provided by (coastal state?) authority along my route.

This requires trusted information about the service provider, such as "provided by authority" or compliant with e-navigation service (for example MS1). This information may be possible to retrieve when searching for services. The information can be either in the format of a parameter in a registry that is "vetted" or a separate service (for example, Endorsement Service) where further information of the service provider can be retrieved.

E.7 Rule based request for information

An application has a rule to automatically request the active route plan from the ship. For example, a VTS identifies a ship via AIS but has not received its route plan.

E.8 Procedure when receiving "Not authorized"

Not authorized can refer to the service request itself or it can refer to the requested data object. When receiving this response, consuming the Access interface for requesting access can be performed.

Annex F
(informative)

Test and simulators

F.1 Purpose

The purpose is to give an overview of the test equipments required.

Three different types of EUTs are identified: Ship/Shore equipment, PKI equipment and Service Discovery equipment.

Overall, the test can be separated into two steps. The first step is to test the EUT using prepared test data without any live connections to other equipment and simulators. The second step is to connect the EUT to simulators where also the dynamic behavour can be tested.

F.2 Manual testing

The first initial step is manual tests of the EUT using prepared test data, open source tools and reference data for comparison as Figure F.1.

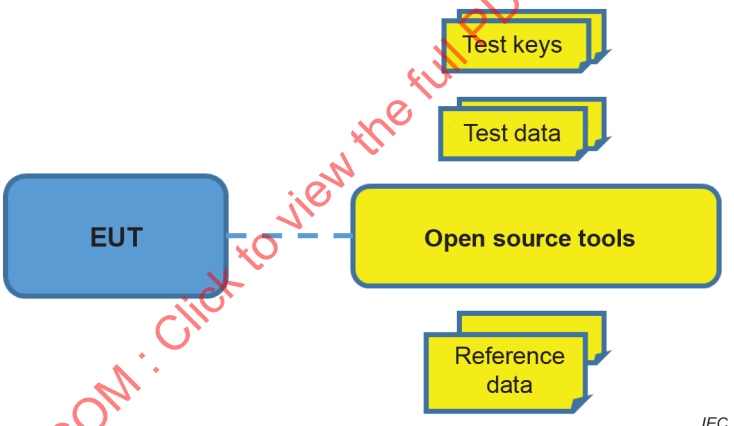


Figure F.1 – Manual testing

F.3 Ship and shore equipment

The main target for SECOM is the ship and shore equipment that shall exchange data in a secure way. See Figure F.2.

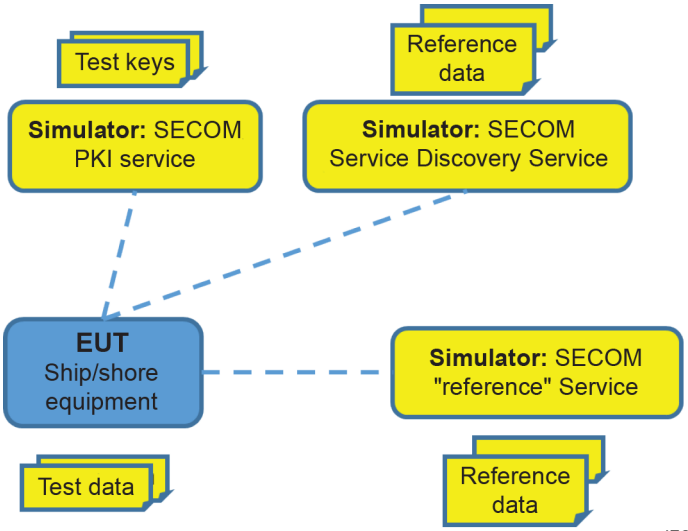


Figure F.2 – Overview of test equipment for ship and shore equipment

F.4 SECOM information service equipment

If the EUT is a SECOM Information Service, the simulator is a reference ship/shore REST-client equipment. See Figure F.3.

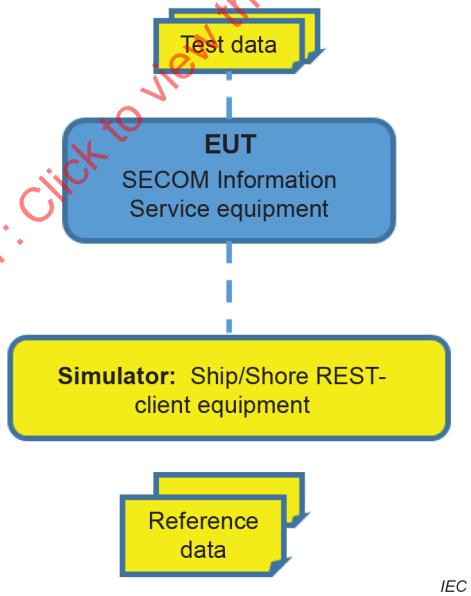


Figure F.3 – Overview of test equipment for SECOM information service equipment

F.5 SECOM PKI equipment

If the EUT is a SECOM PKI, the simulator is a reference ship/shore equipment. See Figure F.4.

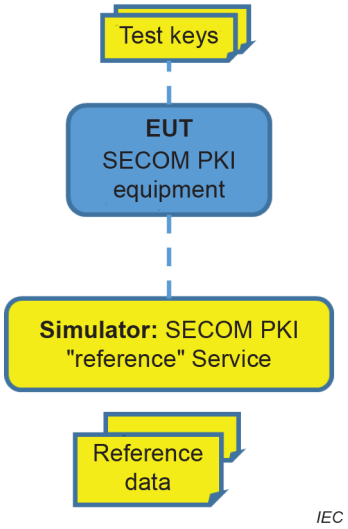


Figure F.4 – Overview of test equipment for SECOM PKI equipment

F.6 SECOM Service Discovery equipment

If the EUT is a SECOM service discovery, the simulator is a reference ship/shore equipment. See Figure F.5.

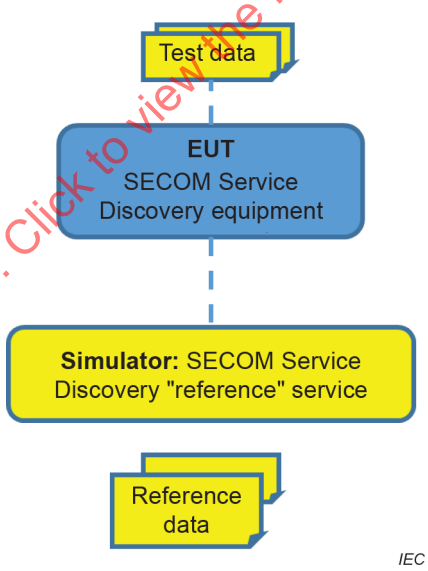


Figure F.5 – Overview of test equipment for SECOM service discovery equipment

Bibliography

IEC 61174:2015, *Maritime navigation and radiocommunication equipment and systems – Electronic chart display and information system (ECDIS) – Operational and performance requirements, methods of testing and required test results*

IEC 63173-1, *Maritime navigation and radiocommunication equipment and systems – Data interfaces – Part 1: S-421 route plan based on S-100*

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions – Part 1: Country code*

ISO 7498-2:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*

ISO 19162, *Geographic information – Well-known text representation of coordinate reference systems*

ISO 21188:2018, *Public key infrastructure for financial services – Practices and policy framework*

ISO/IEC 27000:2018, *Information technology – Security techniques – Information security management systems – Overview and vocabulary*

ISO 28005-2, *Ships and marine technology – Electronic port clearance (EPC) – Part 2: Core data elements*

ISO/IEC 9594-8, *Information technology – Open systems interconnection – Part 8: The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 10118-1:2016/AMD1:2021, *Information technology – Security techniques – Hash-functions – Part 1: General*

ISO/IEC 11770-3:2021, *Information technology – Key management – Part 3: Mechanisms using asymmetric techniques*

ISO/IEC 14888 (all parts), *Information technology – Security techniques – Digital signatures with appendix*

ISO/IEC 21778:2017, *Information technology – The JSON data interchange syntax*

IALA G1128:2018, *The Specification of e-Navigation Technical Services, ed. 1.1*

IALA G1143, *Unique Identifiers for Maritime Resources*

RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*

RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*

RFC 5234, *Augmented BNF for Syntax Specifications: ABNF*

RFC 8141, *Uniform Resource Names (URNs)*

RFC 8259, *The JavaScript Object Notation (JSON) Data Interchange Format*

Roy Fielding, *Architectural Styles and the Design of Network-based Software Architectures*

Unified Modeling Language (UML):2017, *OMG Unified Modeling Language v. 2.5.1 [viewed 2021-10-18]. Available at https://www.omg.org/spec/UML/*

OASIS, *Reference Model for Service Oriented Architecture 1.0*

NIST, *Computer security resource center* [viewed 2022-02-14]. Available at https://csrc.nist.gov/glossary/term/nonce

NIST DSS–FIPS Publication 180-3, *Secure Hash Standard*

NIST DSS–FIPS Publication 202, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*

NIST DSS–FIPS Publication 186-3, *Digital Signature Standard (DSS)*

NIST Special Publication (SP) 800-63B, *Digital Identity Guidelines: Authentication and Lifecycle Management*

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

MATÉRIELS ET SYSTÈMES DE NAVIGATION ET DE RADIOCOMMUNICATION MARITIMES – INTERFACES DE DONNÉES –

Partie 2: Communications sécurisées entre le navire et la terre (SECOM)

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets.

L'IEC 63173-2 a été établie par le comité d'études 80 de l'IEC: Matériels et systèmes de navigation et de radiocommunication maritimes. Il s'agit d'une Norme internationale.

Le texte de cette Norme internationale est issu des documents suivants:

Projet	Rapport de vote
80/1030/FDIS	80/1039/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à son approbation.

La langue employée pour l'élaboration de cette Norme internationale est l'anglais.

Le présent document a été rédigé selon les Directives ISO/IEC, Partie 2, il a été développé selon les Directives ISO/IEC, Partie 1 et les Directives ISO/IEC, Supplément IEC, disponibles sous www.iec.ch/members_experts/refdocs. Les principaux types de documents développés par l'IEC sont décrits plus en détail sous www.iec.ch/standardsdev/publications.

Une liste de toutes les parties de la série IEC 63173, publiées sous le titre général *Matériels et systèmes de navigation et de radiocommunication maritimes – Interfaces de données*, se trouve sur le site web de l'IEC.

Le comité a décidé que le contenu du présent document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous webstore.iec.ch dans les données relatives au document recherché. A cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

L'e-navigation a été définie comme le moyen de fournir des informations électroniques de manière harmonisée et les services maritimes ont été spécifiés par l'Organisation maritime internationale (OMI). Les services maritimes sont des services opérationnels pour les acteurs à terre et à bord. Pour rendre les services maritimes interopérables entre différents acteurs et systèmes de différents fabricants, des normes, des spécifications et des lignes directrices sur plusieurs niveaux sont exigées, par exemple pour les services techniques et les formats de données/produits. Les services techniques comprennent un ensemble de solutions techniques et de moyens de communication permettant de fournir un service maritime. Le Plan de mise en œuvre de la stratégie en matière d'e-navigation (SIP) de l'OMI exige que tous les services maritimes soient conformes à la S-100 de l'OHI comme base de référence. En outre, il est attendu de l'IEC qu'elle mette en œuvre les détails décrits dans le SIP.

Le standard de communications sécurisées (SECOM, SEcure COMmunications) entre le navire et la terre fournit des normes pour l'échange sécurisé de données avec les services techniques. En outre, il comporte une conception d'interface de service technique conforme aux lignes directrices et aux modèles de service définis par l'AISM et partiellement inclus dans la S-100 de l'OHI.

Le SECOM spécifie des interfaces de service (API) pour l'échange de données, les mesures de protection des données pour permettre des communications sécurisées et des interfaces pour la découvrabilité des services. Le SECOM est applicable aux produits basés sur la S-100 de l'OHI, mais aussi à d'autres formats de données (données utiles), c'est-à-dire qu'il est généralement indépendant du type de données échangées.

La normalisation d'une interface de service commune pour l'échange de données permet une interopérabilité technique plus étendue, la même interface de service pouvant être utilisée pour échanger des informations indépendamment de leur utilisation opérationnelle.

En conséquence, l'objectif du SECOM est le suivant:

- faciliter l'échange d'informations normalisées, par exemple pour les produits basés sur la S-100 de l'OHI qui font partie des services maritimes, tels que les plans de route, les mises à jour des cartes nautiques et les avertissements de navigation;
- faciliter l'interopérabilité entre les systèmes TI maritimes;
- réduire la nécessité de prendre en charge de nombreuses conceptions de services différentes (propriétaires);
- tirer parti des avantages de l'architecture orientée services dans les communications maritimes, par exemple pour permettre aux systèmes des navires d'interagir avec les systèmes portuaires lors du premier appel à un port spécifique.

MATÉRIELS ET SYSTÈMES DE NAVIGATION ET DE RADIOCOMMUNICATION MARITIMES – INTERFACES DE DONNÉES –

Partie 2: Communications sécurisées entre le navire et la terre (SECOM)

1 Domaine d'application

Le domaine d'application du SECOM comprend des interfaces (API) pour l'échange de données (services d'information), des mesures de sécurité de l'information pour permettre des communications sécurisées et des interfaces pour la découvrabilité des services. Le SECOM assure l'interopérabilité technique, où la même interface de service est utilisée pour l'échange d'informations indépendamment de son utilisation opérationnelle, jusqu'au niveau de l'échange d'informations en ligne sécurisé. Bien que conçu pour les produits basés sur la S-100 de l'OHI, le SECOM ne dépend pas techniquement des données utiles et est également applicable à d'autres types de données.

Les communications entre services d'information SECOM pour l'échange de données sont basées sur des services web sur IP. Les liens du "dernier kilomètre" entre un service d'information SECOM et l'application d'utilisateur ne sont pas définis dans le présent document et, par conséquent, la technologie de communication entre l'API du fournisseur et un système navire/terre peut être aussi bien basée sur IP que non basée sur IP. L'Annexe D informative décrit une mise en œuvre de celles-ci. Elle permet différentes solutions entre le service et les systèmes/applications à terre/du navire.

Le SECOM ne définit pas la couche physique ou la couche de liaison pour le transport des données entre services d'information SECOM, mais exige que le transport prenne en charge la communication IP. Le SECOM est applicable aux services publics (gouvernementaux) et privés (entreprises). Le SECOM est applicable aux communications navire-terre et terre-navire, et peut être utilisé pour les communications navire-navire.

2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

OHI, S-100:2018, *Modèle universel de données hydrographiques, éd. 4.0.0*

RFC 2315, *PKCS #7: Cryptographic Message Syntax* (disponible en anglais seulement)

RFC 2459, *Internet X.509 Public-key infrastructure and attribute certificate frameworks* (disponible en anglais seulement)

RFC 2818, *HTTP Over TLS (2000)* (disponible en anglais seulement)

RFC 2986, *PKCS #10: Certification Request Syntax Specification* (disponible en anglais seulement)

RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace* (disponible en anglais seulement)

RFC 5246, *TLS version 1.2 (2008)* (disponible en anglais seulement)

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* (disponible en anglais seulement)

RFC 6960, *X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP* (disponible en anglais seulement)

RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* (disponible en anglais seulement)

RFC 8446, *TLS version 1.3 (2018)* (disponible en anglais seulement)

3 Termes, définitions et termes abrégés

Pour les besoins du présent document, les termes, définitions et termes abrégés suivants s'appliquent.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

3.1 Termes et définitions

3.1.1
contrôle d'accès
autorisation d'accès et limitation d'accès

Note 1 à l'article: Le contrôle d'accès désigne l'ensemble des étapes qui sont prises pour autoriser et limiter de manière sélective l'entrée, le contact ou l'utilisation des biens. Les autorisations et les limitations d'accès sont souvent établies en fonction des exigences propres à la sécurité et à l'activité métier.

[SOURCE: ISO/IEC 27000:2018, 3.1, modifié – La définition a été reformulée et une note à l'article a été ajoutée]

3.1.2
acteur
rôle joué par un utilisateur ou tout autre système qui interagit avec le sujet

[SOURCE: Unified Modeling Language:2017, 18.2.1.1]

3.1.3
authentification
processus permettant de confirmer qu'une caractéristique revendiquée pour une entité est effectivement correcte

Note 1 à l'article: L'authentification consiste à vérifier qu'une caractéristique ou un attribut qui semble être vrai l'est en réalité.

[SOURCE: ISO/IEC 27000:2018, 3.5, modifié – La définition a été reformulée et la note à l'article a été ajoutée]

3.1.4
données d'authentification
informations utilisées pour vérifier l'identité déclarée d'une entité, telle qu'un individu, un rôle défini, une société ou une institution

[SOURCE: ISO 21188:2018, 3.4]

3.1.5
autorisation
attribution de droits, comprenant la permission d'accès sur la base de droits d'accès

[SOURCE: ISO 7498-2:1989, 3.3.10]

3.1.6
certificat
clé publique et identité d'une entité (*données d'authentification* [3.1.4]), ainsi que d'autres informations, rendues infalsifiables par la signature des informations du certificat avec la clé privée de l'autorité de certification qui a émis cette clé publique

[SOURCE: ISO 21188:2018, 3.8]

3.1.7
client de données
utilisateur de données

EXEMPLE Un navire.

3.1.8
serveur de données
fournisseur de données

EXEMPLE Centre d'avertissements de navigation.

3.1.9
signature numérique
données ajoutées à une unité de données, ou transformation cryptographique d'une unité de données, permettant à un destinataire de prouver la source et l'intégrité de l'unité de données et protégeant contre la contrefaçon (par le destinataire, par exemple)

[SOURCE: ISO 7498-2:1989, 3.3.26]

3.1.10
clé de chiffrement
clé secrète cryptographique utilisée avec les techniques cryptographiques symétriques

3.1.11
hachage
chaîne de bits issue d'une fonction qui calcule une valeur des octets d'entrée suivant un certain algorithme, par exemple le calcul d'une somme de contrôle, la compression, SHA-256

Note 1 à l'article: Les ouvrages de référence traitant à ce sujet contient divers termes ayant une signification identique ou similaire à celle de code de hachage. Le code de détection de modification, le code de détection de manipulation, l'empreinte de hachage, le résultat de hachage, la valeur de hachage et l'empreinte en sont quelques exemples.

Note 2 à l'article: La NIST SP 800-63B utilise l'"empreinte de hachage" pour le "hachage".

[SOURCE: ISO/IEC 10118-1:2016, 3.3, modifié – La définition a été complétée et la Note 2 à l'article a été ajoutée]

3.1.12
registre des identités
registre qui contient les identités et les liens avec les clés publiques

3.1.13
intégrité
<sécurité des informations> protection de l'exactitude et la complétude des informations

[SOURCE: ISO/IEC 27000:2018, 3.36, modifié – La définition a été reformulée]

3.1.14
javascript object notation
JSON
syntaxe légère, basée sur le texte et indépendante du langage de programmation, permettant de définir des formats d'échange de données

[SOURCE: IEC 21778:2017]

3.1.15
clé privée
clé cryptographique d'une paire de clés asymétriques d'une entité qui ne peut être utilisée que par cette entité

[SOURCE: ISO/IEC 11770-3:2021, 3.32, modifié – La définition a été reformulée et la note à l'article a été ajoutée]

3.1.16
clé publique
clé cryptographique d'une paire de clés asymétriques d'une entité qui peut être rendue publique

[SOURCE: ISO/IEC 11770-3:2021, 3.33, modifié – La définition a été reformulée et la note à l'article a été supprimée]

3.1.17
infrastructure de clé publique
PKI
structure matérielle, logicielle, de personnes, de procédures et de politiques qui utilise une technologie de signature numérique pour faciliter une association vérifiable entre la composante publique d'une paire de clés publiques asymétriques et un abonné spécifique qui possède la clé privée correspondante

Note 1 à l'article: La clé publique peut être fournie pour la vérification de la signature numérique, l'authentification du sujet dans les dialogues de communication, et/ou pour l'échange ou la négociation de clés de chiffrement de messages.

Note 2 à l'article: L'abréviation "PKI" est dérivée du terme anglais développé correspondant "public key infrastructure".

[SOURCE: ISO 21188:2018, 3.48]

3.1.18
transfert d'état représentationnel
REST
style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web

Note 1 à l'article: Les services web conformes au style d'architecture REST, aussi appelés services web RESTful, établissent une interopérabilité entre les ordinateurs sur Internet
Note à l'article: L'abréviation "REST" est dérivée du terme anglais développé correspondant "REpresentational State Transfer".

[SOURCE: Roy Fielding – Representational State Transfer (REST)]

3.1.19
format d'échange de plans de route
RTZ
format basé sur la normalisation d'un plan de route qui est constitué de points de route où chaque point de route contient des informations relatives au segment issu du point de route précédent

Note 1 à l'article: Le format d'échange de routes est un fichier RTZ contenant une version encodée au format XML du plan de route.

[SOURCE: IEC 61174:2015]

3.1.20
administrateur de schéma
SA
organisme seul responsable du maintien et de la coordination du schéma de protection, en contrôlant l'adhésion au schéma au moyen du registre des identités afin d'assurer que tous les participants agissent conformément aux procédures définies

Note 1 à l'article: Le SA maintient le certificat racine numérique de premier niveau utilisé pour faire fonctionner le schéma de protection et il est le seul organisme pouvant certifier l'identité des autres participants au schéma. Le SA est chargé de distribuer le certificat directement à tous les utilisateurs enregistrés qui participent au schéma de protection.

Note 2 à l'article: L'abréviation "SA" est dérivée du terme anglais développé correspondant "scheme administrator".

3.1.21
service
mécanisme permettant d'accéder à une ou plusieurs capacités, dans lequel l'accès s'effectue par le biais d'une interface prescrite et conformément aux contraintes et politiques spécifiées par la description de service

[SOURCE: OASIS Reference Model for Service Oriented Architecture 1.0]

3.1.22
utilisateur de service
utilisateur qui utilise des instances de service fournies par des fournisseurs de services

Note 1 à l'article: Tous les utilisateurs du domaine maritime peuvent être des utilisateurs de services, par exemple les navires et leur équipage, les autorités, les stations VTS, les organisations (par exemple météorologiques), les fournisseurs de services commerciaux, etc.

[SOURCE: Lignes directrices de l'Association internationale de signalisation maritime (AISM) sur la documentation des services, G1128]

3.1.23
instance de service
implémentation de service qui peut être déployé à différents emplacements par des fournisseurs de services identiques ou différents

3.1.24
fournisseur de services
utilisateur qui fournit des instances de services conformément à une spécification de service et à une description d'instance de service

Note 1 à l'article: Tous les utilisateurs du domaine maritime peuvent être des fournisseurs de services, par exemple les autorités, les centres VTS, les organisations (par exemple météorologiques), les fournisseurs de services commerciaux, etc.

[SOURCE: Lignes directrices de l'AISM sur la documentation des services, G1128]

3.1.25
registre de services
registre (par exemple base de données) qui conserve les descriptions publiques (métadonnées) des services

3.1.26
nonce
valeur aléatoire ou non répétée incluse dans les données échangées par le biais d'un protocole, généralement dans le but de garantir la transmission de données en direct plutôt que de données retransmises, ceci afin de détecter et de protéger les données des attaques lors des retransmissions

[SOURCE: NIST, Computer security resource center]

3.2 Termes abrégés

AES Advanced Encryption Standard (norme de chiffrement avancé)
API Application Programming Interface (interface de programmation d'application)
AC autorité de certification
CBC Cipher Block Chaining (enchaînement de blocs de chiffrement)
CRL Certificate Revocation List (liste de révocation de certificats)
CSR Certificate Signing Request (demande de signature de certificat)
DSA Digital Signature Algorithm (algorithme de signature numérique)
DSS Digital Signature Standard (norme de signature numérique)
ENC Electronic Navigational Chart (carte électronique de navigation)
AISM Association internationale de signalisation maritime
OHI Organisation hydrographique internationale
OMI Organisation maritime internationale
ISO International Standards Organization (Organisation internationale de normalisation)
MRN Maritime Resource Name (nom de ressource maritime)
OCSP Online Certificate Status Protocol (protocole de statut de certificat en ligne)
FEO fabricant de l'équipement d'origine
PEM Privacy Enhanced Mail (courrier PEM)
PKCS Public Key Cryptography Standards (normes de cryptographie à clé publique)
PKI Public Key Infrastructure (infrastructure de clé publique)
SA Scheme Administrator (administrateur de schéma)
SSK Self Signed Key (clé autosignée)
TLS Transport Layer Security (sécurité de la couche de transport)
UML Unified Modelling Language (langage de modélisation unifié)
URI Uniform Resource Identifier (identifiant uniforme de ressource)
URL Uniform Resource Locator (localisateur uniforme de ressource) (adresse Web)
VTS Vessel Traffic Services (services du trafic maritime)
WKT Well-Known Text (représentation textuelle connue)

4 Description générale du SECOM

4.1 Généralités

Le SECOM comprend des interfaces de services d'information (API) pour l'échange de données, des mesures de sécurité des informations par une PKI SECOM, ainsi que la sécurité des canaux de communication et la protection des données pour permettre une communication sécurisée. En outre, le SECOM comprend des interfaces pour la découvrabilité des services, comme représenté à la Figure 1. L'objectif de ces composants est de permettre au SECOM d'assurer l'interopérabilité technique, où la même interface de service est utilisée pour l'échange d'informations indépendamment de son utilisation opérationnelle, jusqu'au niveau de l'échange d'informations en ligne sécurisé. Bien que conçu pour les produits basés sur la S-100 de l'OHI, le SECOM ne dépend pas techniquement des données utiles et est également applicable à d'autres types de données.

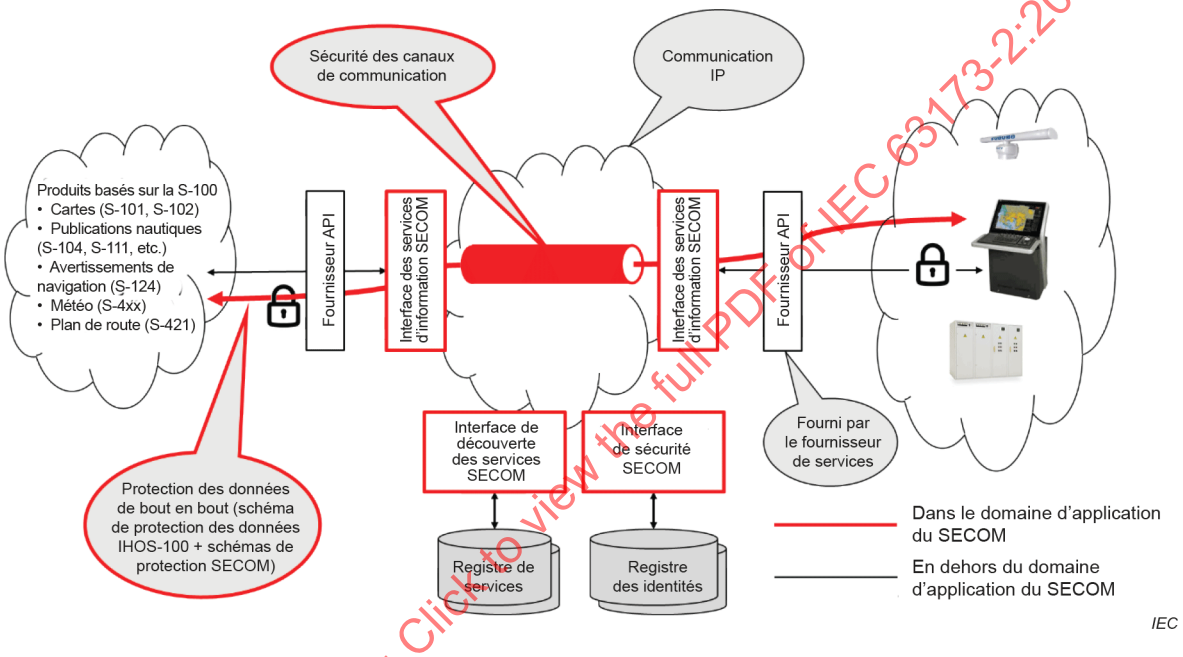


Figure 1 – Vue d'ensemble de SECOM

L'interface des services d'information SECOM comprend le côté public exposé sur l'internet comme représenté à la Figure 1. Les liens du "dernier kilomètre" entre une interface du service d'information SECOM et l'application d'utilisateur ne sont pas définis dans le présent document et, par conséquent, différentes solutions entre l'instance de service et le système à terre/navire sont possibles. L'Annexe D informative décrit une telle mise en œuvre.

La sécurité des informations SECOM comprend la sécurité des canaux de communication, une variante de PKI (Public Key Infrastructure) et des alternatives de schéma de protection des données pour l'échange d'informations avec une conformité totale ou partielle avec la S-100 de l'OHI. La protection des données est fournie entre les utilisateurs finaux. La PKI SECOM comprend la définition d'un ensemble d'interfaces de service pour la gestion des clés.

L'interface de découverte de services comprend des opérations permettant de rechercher des instances de services à partir d'un registre de services pour satisfaire certains critères, par exemple des mises à jour de cartes, des avertissements de navigation, l'heure estimée d'arrivée (ETA) actualisée ou des services d'optimisation des routes. L'interface de découverte de services permet à l'utilisateur de choisir une instance de service à utiliser.

L'échange d'informations entre les acteurs est bidirectionnel, ce qui signifie que tant le navire/la compagnie maritime que les autorités à terre et les fournisseurs de services privés peuvent initier l'échange d'informations et agir en tant que fournisseur d'informations. Le SECOM est donc applicable à la fois pour les communications entre le navire et la terre et entre la terre et le navire. L'Annexe D fournit des recommandations pour la mise en œuvre.

4.2 Interface des services d'information

La normalisation des interfaces de service pour l'échange d'informations permet à un utilisateur qui a mis en œuvre l'interface, par exemple pour l'échange de Plan de route S-421 (IEC 63173-1), d'utiliser un ensemble de services opérationnels ayant des objectifs différents, mais tous basés sur l'échange (push) de Plan de route S-421. L'interface de service normalisée contient également des interfaces pour l'abonnement, la demande d'accès, l'extraction (pull) et la demande dynamique de la capacité, de l'état et de la description de l'instance. L'Annexe B décrit en détail les besoins pour chaque interface et opération.

La mise en œuvre d'une interface normalisée présente des avantages considérables pour un navire, par opposition à l'obligation de mettre en œuvre un grand nombre d'interfaces différentes afin d'adhérer aux interfaces propriétaires de divers fournisseurs de services et utilisateurs. Le fait de permettre aux fabricants de matériel maritime, tels que les fabricants d'ECDIS, de se concentrer sur une interface de service normalisée facilite la mise en œuvre de produits d'information à bord des navires et génère moins d'exigences de changements à bord en cas d'introduction de nouveaux services.

Les fournisseurs de services profitent des mêmes avantages en produisant des services basés sur l'échange de produits d'information normaux, ce qui leur permet de bénéficier d'une interopérabilité technique avec tous les navires ayant mis en œuvre l'interface de service normalisée. Comme la plupart des interfaces SECOM sont facultatives, un fournisseur de services de la mise en œuvre d'un service d'information SECOM peut décider et choisir ce qu'il veut prendre en charge, ce qui lui donne la possibilité d'adapter le service à ses besoins opérationnels. Chaque capacité de service d'information SECOM peut être récupérée à partir du service d'information SECOM lui-même.

L'interface des services d'information SECOM est conçue pour les produits basés sur la S-100 de l'OHI, mais elle supporte également des données utiles arbitraires en utilisant le schéma de protection des données SECOM pour l'échange. Les données peuvent être envoyées soit ouvertes et signées, soit chiffrées et signées.

4.3 Sécurité des informations

4.3.1 Mesures

Le SECOM comporte plusieurs mesures visant à satisfaire aux attributs de qualité suivants:

- l'authentification est un processus par lequel un système vérifie l'identité d'un utilisateur (humain ou machine) qui souhaite y accéder (c'est-à-dire qu'il confirme que l'utilisateur est bien celui qu'il prétend être, à l'instar d'un passeport). Elle correspond à une signature écrite et à un cachet officiel sur papier. L'authentification en SECOM est obtenue à deux niveaux: 1) l'authentification des données par la signature (protection des données SECOM) et 2) l'authentification des services par la sécurité des canaux de communication et les certificats X.509 (RFC 5280 et RFC 2459) (sécurité des canaux de communication SECOM). L'authentification dépend d'identités uniques qui sont réalisées par la PKI SECOM;
- l'intégrité en SECOM est obtenue par la signature de données où la somme de contrôle fait partie de la signature, décrite dans la protection des données SECOM. La signature et la vérification de la signature dépendent d'identités uniques qui sont réalisées par la PKI SECOM;
- la confidentialité en SECOM est obtenue à deux niveaux: 1) le chiffrement des données selon la protection des données SECOM et S-100 de l'OHI, et 2) le canal de communication sécurisé (sécurité du transport) pour IP et HTTP par TLS et certificats X.509, décrit dans la sécurité des canaux de communication SECOM;

- le contrôle d'accès utilise l'autorisation qui est le processus de détermination d'un ensemble de permissions accordées à une identité de confiance spécifique. Elle est réalisée en SECOM par le propriétaire des informations qui autorise l'accès à des identités choisies dans le registre d'identité commun, décrit dans la PKI SECOM. Seules les interfaces publiques sont définies en SECOM (par exemple Request Access);
- l'identification en SECOM est obtenue par le registre commun des identités uniques, les liaisons avec les paires de clés asymétriques et l'API normalisée de la PKI SECOM;
- la non-répudiation en SECOM, c'est-à-dire un service destiné à protéger contre le faux démenti de l'expéditeur d'avoir créé le contenu d'un message et d'avoir envoyé un message, est obtenue en signant les données avec la clé privée.

4.3.2 PKI SECOM

La PKI SECOM (infrastructure de clé publique) décrit l'interface commune de gestion des clés et la fonctionnalité attendue assurant le lien entre l'identité et la clé utilisée pour la signature des données et l'authentification dans l'interaction avec le service.

Une PKI SECOM peut être fournie par tout fournisseur de PKI internationalement reconnu conforme aux exigences de PKI SECOM. Une PKI SECOM comprend une interface et un registre des identités régi par son administrateur de schéma. Plusieurs instances de la PKI SECOM peuvent être fournies par plusieurs administrateurs de schéma. Un acteur peut participer ou être enregistré dans plusieurs instances de la PKI SECOM. Le matériel compatible SECOM est supposé prendre en charge plusieurs instances de la PKI SECOM.

Le SECOM ne précise pas quelle instance du registre des identités utiliser et il ne définit ni ne limite le nombre de registres des identités.

4.3.3 Sécurité des canaux de communication

En utilisant des instances de service conformément au SECOM, le transport Internet est protégé par TLS et des certificats valides de parties de confiance, comme indiqué dans la sécurité des canaux de communication SECOM. La protection du canal est un complément à la protection des données proprement dites et il est nécessaire de l'inclure pour sécuriser également les autres demandes de service, telles que les demandes d'abonnement, les demandes d'accès et les notifications. La sécurité des canaux de communication SECOM décrit l'utilisation d'un certificat obtenu à partir d'un registre des identités de confiance et permet ainsi l'authentification sur l'interaction du service proprement dit, comme représenté à la Figure 2.

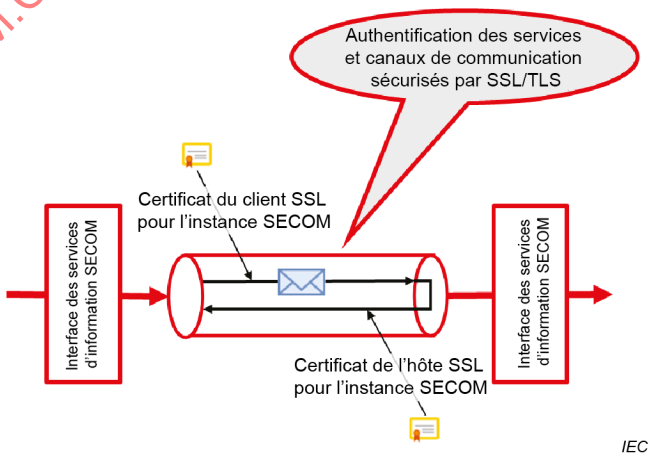


Figure 2 – Canaux de communication sécurisés

La sécurité des canaux de communication SECOM repose sur une infrastructure de clé publique SECOM (PKI SECOM) ou une gestion de clé publique-privée utilisant des certificats X.509 pour échanger les clés.

Le schéma de sécurité des canaux de communication SECOM ne couvre pas les liens du "dernier kilomètre" entre l'interface des services d'information SECOM et l'application d'utilisateur. L'Annexe D contient des exemples informatifs et des recommandations sur la manière dont cette protection peut être obtenue.

4.3.4 Protection des données

La protection des données comprend à la fois la signature des données pour l'authentification et l'intégrité (signature numérique), et éventuellement le chiffrement des données pour la confidentialité. La protection des données est alignée sur le schéma de protection des données de l'OHI (OHI S-100, Partie 15).

La signature numérique contient à la fois la somme de contrôle utilisée pour vérifier l'intégrité des données reçues et l'identité revendiquée pour l'authentification des données. La vérification de la signature comprend l'accès à une infrastructure à clé publique (PKI) commune où la clé publique de l'identité revendiquée peut être téléchargée. Lorsque des données sont échangées, la signature numérique doit toujours être jointe et vérifiée aussi près que possible de l'utilisateur final. Les données de base sont ainsi protégées de bout en bout, voir Figure 1.

Le chiffrement facultatif des données utilise un schéma de protection commun et offre deux alternatives pour la gestion des clés de chiffrement: 1) la gestion des clés de chiffrement existante avec des autorisations pour les données ENC, telles que celles de l'OHI, peut être utilisée avec l'interface des services d'information SECOM et 2) le SECOM inclut également sa propre gestion des clés de chiffrement qui est spécialement conçue pour un échange de données plus dynamique entre le navire et la terre et entre la terre et le navire, tel que l'échange de plans de route. Dans le schéma de protection des données SECOM, l'expéditeur de données classifiées crée une clé de chiffrement temporaire qui est transférée de façon sécurisée avant le transfert des données chiffrées. La clé de chiffrement temporaire est chiffrée avec la clé publique du destinataire, la même clé que celle utilisée pour l'authentification, et ne peut donc être déchiffrée que par l'acteur ayant la clé privée en sa possession. La clé publique du destinataire, si elle n'est pas déjà présente, peut être récupérée à partir de l'interface SECOM PublicKey du destinataire, voir 5.7.16, ou de l'interface GetPublicKey de la PKI SECOM, voir 8.6.3. Une paire de clés fortes et suffisamment longues pour permettre le chiffrement et le déchiffrement de la clé de chiffrement, est exigée.

La protection des données décrite ci-dessus ne protège que la partie du message contenant les données. Pour sécuriser le message complet, une signature supplémentaire est ajoutée, une signature d'enveloppe qui comprend toutes les informations de l'enveloppe contenue dans l'uploadObject, y compris les données et leur dataSignature. Les parties du message qui sont protégées par les différentes signatures sont représentées à la Figure 3.

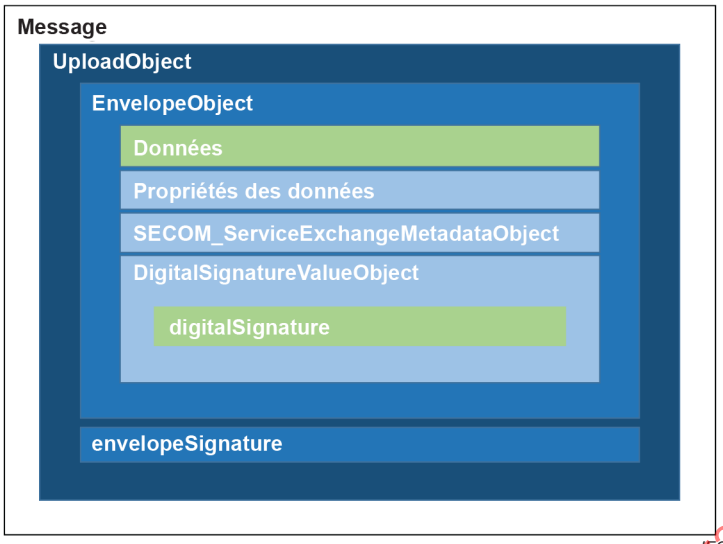


Figure 3 – Représentation des parties du message qui sont protégées par les deux signatures

Une autre raison justifiant l'introduction d'une signature d'enveloppe est de faciliter la vérification complète de l'intégrité des messages du côté du service d'information SECOM destinataire, ce qui interdit la transmission de messages corrompus au cours du dernier kilomètre jusqu'au destinataire, voir Figure 4.

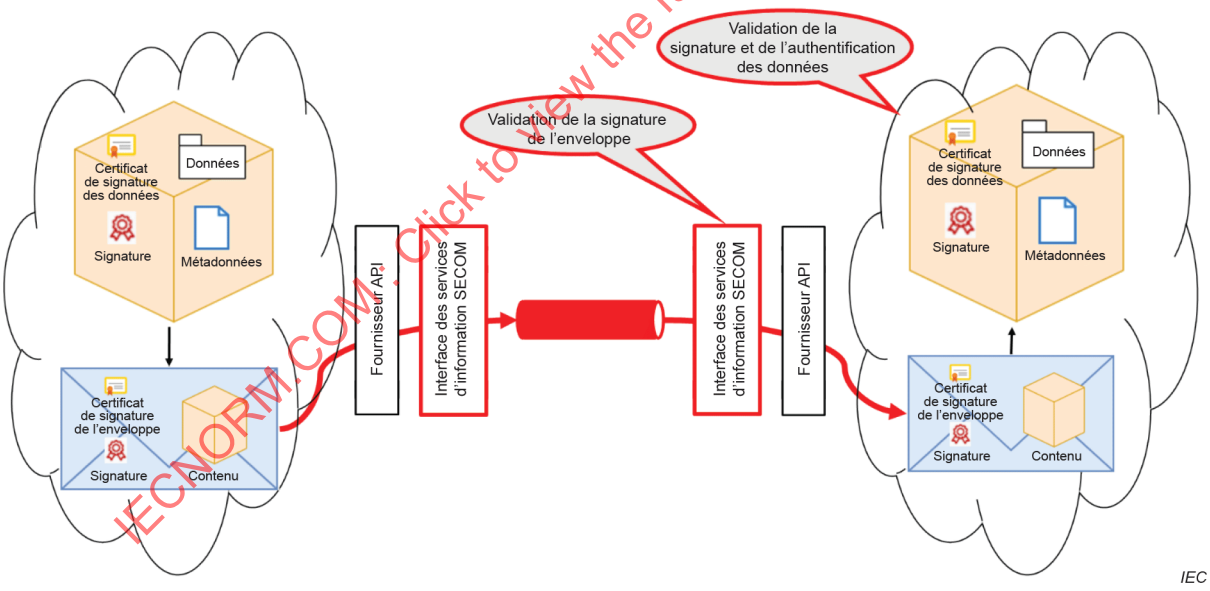


Figure 4 – Validation de l'enveloppe et des données

Le schéma de protection des données SECOM repose sur une infrastructure à clé publique SECOM (PKI SECOM) utilisant des certificats X.509 pour échanger les clés. La clé privée est générée par l'acteur et stockée en interne, alors que la clé publique est chargée de façon sécurisée vers une PKI SECOM et associée à l'identité.

Le schéma de protection des données et l'empreinte numérique du certificat racine indiquent le registre des identités utilisé pour l'échange de données spécifique. Les certificats racines de confiance identifiés avec leurs empreintes numériques sont censés être stockés dans un magasin de confiance local.

4.3.5 Etat de révocation du certificat

Une liste de révocation de certificats (CRL) est une liste certificats révoqués qui est téléchargée auprès de l'autorité de certification (AC). Le protocole de statut de certificat en ligne (OCSP) est un protocole permettant de vérifier la révocation d'un certificat unique de manière interactive à l'aide d'un service en ligne appelé répondeur OCSP (voir 8.5).

4.4 Découvrabilité des services

Pour rendre en charge l'utilisation dynamique des services, une interface de découverte de services SECOM a été définie. L'utilisation opérationnelle prévue est de permettre aux officiers à bord des navires et aux opérateurs à terre de rechercher des instances de service avec lesquelles interagir. Un navire qui veut trouver un fournisseur de services d'optimisation des routes ou des VTS sur la route prévue avec qui partager le plan de route ou des fournisseurs d'avertissements de navigation ou de cartes électroniques de navigation (ENC) sont des exemples du point de vue du navire. Du point de vue de la terre, il est possible de rechercher des navires immatriculés, ciblés pour des demandes d'information telles que les informations dont la déclaration est exigée, la route attendue ou l'heure estimée d'arrivée. La découverte de services SECOM contient des interfaces communes pour la découverte d'instances de services.

Une interface normalisée pour trouver des instances de service réduit la nécessité de prendre en charge plusieurs interfaces différentes, propriétaires ou propres à une entreprise. En outre, elle facilite la stabilité accrue de la mise en œuvre, c'est-à-dire qu'aucune nouvelle mise en œuvre et certification n'est nécessaire si un nouveau registre de services est utilisé.

Le SECOM ne précise pas quel registre de services utiliser et il ne définit ni ne limite le nombre de registres de services, de sorte qu'il peut y avoir plusieurs fournisseurs de registres de services.

Le SECOM définit uniquement l'interface de découverte des instances de service, et non l'enregistrement des instances de service dans le registre des services. Cela nécessite d'être décrit par le fournisseur du registre de services choisi.

4.5 Structure du présent document

Les articles suivants sont inclus en SECOM:

- article 5 Interface des services d'information SECOM,
- article 6 Sécurité des canaux de communication SECOM,
- article 7 Protection des données SECOM,
- article 8 PKI SECOM,
- article 9 Interface du service de découverte de services SECOM,
- article 10 Cas d'erreur SECOM,
- article 11 Méthodes d'essai et résultats escomptés.

5 Interface des services d'information SECOM

5.1 Généralités

Le présent Article 5 décrit l'interface des services d'information SECOM pour l'échange de produits basés sur la S-100 de l'OHI et d'autres données utiles. La mise en œuvre de l'interface des services d'information SECOM prend en charge la signature numérique et le chiffrement facultatif des données. L'interface des services d'information SECOM exige l'utilisation de certificats communs de la PKI SECOM pour l'authentification entre les instances de service, comme décrit dans la sécurité des canaux de communication SECOM.

La définition ci-dessous contient une compilation de plusieurs interfaces de service qui, combinées ensemble, constituent l'interface des services d'information SECOM.

Dans la définition, l'expression "utilisateur" désigne le client des données, et l'expression "fournisseur" désigne le serveur de données:

- quatre interfaces de service sont utilisées pour l'échange réel de données: Upload, Upload Link, Get et Get By Link. L'interface du service Upload est utilisée par les fournisseurs d'informations pour envoyer des données à un utilisateur, d'une taille maximale de 350 ko, codées en Base64 (voir RFC 2045). Elle couvre la nécessité primaire de l'échange de données. L'interface Get est utilisée lorsqu'un utilisateur extrait des données du fournisseur. Pour des quantités de données plus conséquentes pour lesquelles le téléchargement montant (push) ne peut être utilisé, l'interface Upload Link est utilisée pour envoyer un lien vers les données, puis les données sont récupérées avec l'interface Get By Link;
- l'interface Get Summary est utilisée pour recevoir un bref résumé des données disponibles à partir duquel l'utilisateur peut sélectionner un ou plusieurs éléments à récupérer en utilisant l'interface Get. L'interface Subscription est utilisée lorsqu'un utilisateur souhaite créer un abonnement aux données du fournisseur. Le fournisseur envoie ensuite les données à l'utilisateur en utilisant l'interface de service Upload. L'utilisateur utilise l'interface Remove Subscription pour supprimer l'abonnement. Le fournisseur peut également "désigner" un utilisateur en créant un abonnement pour l'utilisateur à l'aide de méthodes propriétaires au sein du système du fournisseur. Le fournisseur utilise ensuite l'interface Subscription Notification pour informer l'utilisateur de l'existence d'un abonnement actif. Le fournisseur peut également supprimer un abonnement en utilisant des méthodes propriétaires. Le fournisseur utilise alors l'interface Subscription Notification pour en informer l'utilisateur;
- si un utilisateur n'a pas accès aux données, par exemple si l'extraction de données spécifiques n'est pas autorisée, l'utilisateur utilise l'interface Access du fournisseur. Le fournisseur répond ensuite par Access Notification à l'utilisateur en lui communiquant sa décision, c'est-à-dire acceptation ou refus;
- le fournisseur et l'utilisateur utilisent tous deux l'interface Capability pour demander des informations sur les interfaces accessibles et les types de données disponibles. Cela permet par exemple aux fonctions de demander quelle version de la spécification des produits basés sur la norme S-100 est acceptée par un fournisseur ou un utilisateur, et aux fonctions de ne présenter que les interfaces accessibles pour une certaine instance de service, évitant ainsi une utilisation incorrecte de l'instance de service;
- l'interface Ping est utilisée pour vérifier l'état technique de l'instance de service;
- l'interface EncryptionKey est utilisée pour l'échange sécurisé de la clé de chiffrement symétrique;
- l'interface PublicKey est utilisée pour échanger des certificats publics utilisés pour le chiffrement des données et les signatures numériques. Au moyen de cette interface, un utilisateur peut à la fois demander un certificat (pull) et charger un certificat (push).

5.2 Comment lire les descriptions de la définition des interfaces de service

L'interface de service est décrite conformément au modèle de définition des services figurant dans la S-100:2018 de l'OHI, Partie 14, "Echange de données en ligne".

L'interface des services d'information SECOM constitue un ensemble d'interfaces de service, où chaque interface de service est d'abord décrite comme une spécification (qui ne dépend pas de la technologie) et ensuite une conception technique de l'interface de service en technologie REST.

La Figure 5 décrit comment lire l'interface de service spécifiée dans le diagramme UML, en représentant le nom de l'interface avec la ou les opérations définies, l'entrée et la sortie. L'entrée et la sortie sont définies dans le modèle d'échange de données. Chaque interface fournit (expose) au moins une interface de service (exploiter) et peut exiger (utiliser) une ou plusieurs autres interfaces de service.

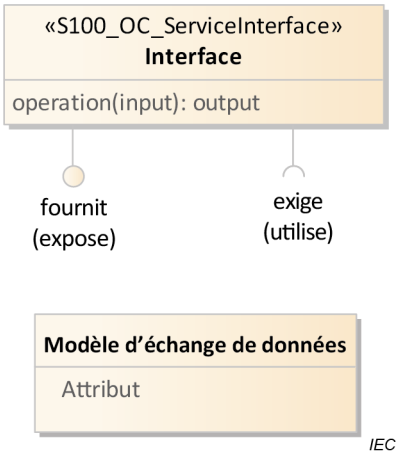


Figure 5 – Modèle de définition de service pour les définitions d'interface de service

La spécification de service de la définition de l'interface de service décrit les informations d'entrée et de sortie dans des tableaux. La colonne Multiplicité ("Mult.") indique si les données de l'attribut sont exigées ou non, où 0..1 et 0..* indiquent que la valeur peut être envoyée, et 1..* et 1 indiquent que la valeur doit être envoyée (c'est-à-dire envoi exigé). La colonne Traitement indique si le traitement des données reçues est obligatoire ou facultatif. Si elles sont obligatoires, les données doivent être traitées conformément à la description du comportement dynamique du service. Le traitement peut inclure l'envoi d'une réponse asynchrone. Voir Tableau 1.

Tableau 1 – Lire les instructions pour les tableaux dans les définitions des interfaces de service

<objet d'information>				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
Nom de l'attribut	Indique au responsable de la mise en œuvre de l'appel de service (c'est-à-dire l'expéditeur initial de la demande) s'il est exigé que l'attribut soit envoyé dans l'appel de service ou si l'attribut peut être omis. Indique au responsable de la mise en œuvre de l'interface de service (c'est-à-dire le répondeur de la demande initiale) si l'attribut est exigé ou non. Si un attribut est exigé par la multiplicité, mais que cet attribut n'est pas présent, l'appel de service est considéré comme une "Demande incorrecte" et l'appel de service est à ignorer. Alternatives à la multiplicité définie dans l'UML. 0..1 0..* 1..* 1	Indique au responsable de la mise en œuvre de l'appel de service si le traitement de l'attribut par le destinataire des données envoyées (c'est-à-dire le responsable de la mise en œuvre de l'interface de service) est obligatoire ou facultatif. Indique au responsable de la mise en œuvre de l'interface de service si le traitement de l'attribut est obligatoire ou facultatif. Options de traitement: Obligatoire Facultatif	Décrit le codage logique (indépendant de la conception du service) de l'attribut	Définition et description de l'attribut

Chaque interface de service décrit également une conception d'interface de service REST. Un tableau décrit chaque opération dans l'interface conçue selon la technologie REST. Pour l'interface de service conçue conformément à REST, le type de données est JSON. Voir Annexe A pour une définition formelle et détaillée de l'ensemble de l'interface des services d'information SECOM en tant que service REST.

5.3 Technologie des services et protocole de transport des services

La technologie (style architectural) choisie est REST (REpresentational State Transfer) sur HTTP/1.1 (RFC 7231).

REST est un style architectural et une approche des communications souvent utilisés dans le développement de services Web. L'utilisation de REST en SECOM est préférée à d'autres protocoles plus lourds comme SOAP (Simple Object Access Protocol), car REST n'utilise pas autant de bande passante, ce qui le rend plus adapté à la communication entre les navires et leur représentation à terre.

REST, généralement exécuté sur HTTP (Hypertext Transfer Protocol), présente plusieurs contraintes:

- découplage: il découple les utilisateurs des producteurs, ce qui convient bien à l'architecture décentralisée SECOM;
- existence sans état: également une bonne condition préalable à la conception d'une architecture décentralisée;
- capacité d'exploiter un cache: probablement moins important en SECOM puisque la plupart des interactions se font entre machines, bien que pour les services avec des interfaces homme-machine, cela soit important;
- exploitation d'un système à couches: le SECOM dépend de bonnes capacités de mise à l'échelle, ce que REST prend en charge;
- exploitation d'une interface uniforme: là encore, le SECOM définissant les services disponibles de manière centralisée dans un ou plusieurs registres de services, cette contrainte permet de découpler les implémentations des services qu'elles fournissent.

La définition de chaque opération définit des messages d'erreur supplémentaires avec lesquels l'opération doit spécifiquement répondre en complément des codes de réponse HTTP définis par HTTP/1.1 (RFC 7231).

5.4 Versions de l'interface de service

La version de la définition de l'interface de service suit la version du présent document.

La version de la conception en tant qu'interface de service REST est reflétée dans le chemin d'accès à l'instance de service. La version est définie dans la définition formelle des services REST en tant que spécification OpenAPI (Swagger) à l'Annexe A.

Le Tableau 2 décrit la structure des versions de l'interface de service SECOM.

Tableau 2 – Versions de l'interface de service SECOM

Méthode REST	URL	URL de base avec version	Opération REST SECOM
POST	https://iec.ch	/v1	/object

Exemple
POST https://iec.ch/v1/object{…}

NOTE La version décrite ici est celle de l'interface des services d'information SECOM définie dans le présent document. La version de l'instance de service elle-même ou d'autres versions du fabricant peuvent être incluses dans l'URL.

5.5 Pagination

Les interfaces Get et Get Summary peuvent renvoyer plusieurs résultats. Afin de limiter la taille de la réponse, ces interfaces prennent en charge la pagination.

Lorsque des demandes sont adressées à ces interfaces de service, les paramètres facultatifs "page" et "pageSize" peuvent être spécifiés par le client. S'ils sont fournis, ils permettent au client de demander combien d'éléments sont à renvoyer et le point de départ de ces éléments. Par exemple, si une requête particulière produit un résultat composé de 30 éléments, les paramètres page = 1 et pageSize = 20 renvoient les vingt premiers éléments, et page = 2, pageSize = 20 renvoient les dix éléments restants.

Que ce soit page ou pageSize qui est spécifié, le serveur doit toujours renvoyer le nombre total d'éléments dans l'attribut totalItems, et le nombre maximal d'éléments par page dans l'attribut maxItemsPerPage.

Des erreurs doivent être renvoyées par le serveur si une page non valide est demandée ou si trop d'éléments sont demandés en même temps.

Si le paramètre page n'est pas présent, le serveur doit renvoyer la première page du résultat.

Si le paramètre pageSize n'est pas présent, le serveur doit renvoyer le nombre maximal d'éléments par page qu'il prend en charge.

Si le paramètre page a la valeur 0, le serveur ne doit renvoyer aucun élément, mais doit renvoyer maxItemsPerPage. Cette valeur doit être constante quelle que soit la demande effectuée sur le serveur (de sorte qu'il n'est nécessaire pour le client de la demander qu'une seule fois).

5.6 Objets d'information communs et types de données

5.6.1 Généralités

Les objets d'information partagés entre plusieurs interfaces sont énumérés du Tableau 3 au Tableau 14.

5.6.2 Types de données de base

Tableau 3 – Types de données de base

Nom	Description
Integer (Entier)	Nombre entier signé. La représentation d'un nombre entier dépend de l'encapsulation et de l'utilisation. EXEMPLE: 29, –65547.
PositiveInteger (EntierPositif)	Nombre entier signé supérieur à 0.
Boolean (Booléen)	Valeur représentant la logique binaire. La valeur peut être vrai (true) ou faux (false).
CharacterString (ChaîneDeCaractères)	Un CharacterString est une séquence de caractères de longueur arbitraire comprenant des accents et des caractères spéciaux tirés d'un répertoire de l'un des jeux de caractères adoptés
Date	Une date fournit des valeurs pour l'année, le mois et le jour conformément au calendrier grégorien. Le codage de caractères d'une date est une chaîne qui doit suivre le format de date du calendrier (représentation complète, format de base). EXEMPLE: 19980918 (AAAAMMJJ).

Nom	Description
Time (Heure)	L'heure est exprimée en heures, minutes et secondes selon un système horaire sur 24 heures. Le codage des caractères d'une heure doit être une représentation complète du format de base. Une représentation complète signifie que les heures, les minutes et les secondes doivent être utilisées. Le format de base signifie que les caractères de séparation sont omis. L'heure est de préférence exprimée en temps universel coordonné (TUC). EXEMPLE: 183059Z. L'heure peut être exprimée en heure locale, avec un décalage donné par rapport au TUC. EXEMPLE: 183059+0100. L'heure peut être exprimée en heure locale, sans spécification de décalage par rapport au TUC. EXEMPLE: 183059. La représentation complète de l'heure locale de 15 h, 27 min et 46 s à Genève (en hiver une heure avant le TUC) et à New York (en hiver cinq heures après le TUC), ainsi que l'indication de la différence entre les échelles de temps de l'heure locale et le TUC, sont utilisées comme exemples. Genève: 152746+0100 New York: 152746-0500 Les heures de service pour un service disponible toute l'année dans une zone où l'heure d'été affecte le décalage par rapport au TUC peuvent être exprimées en heure locale, sans spécification de décalage. Ouverture: 074500 Fermeture: 161500
DateTime (DateHeure)	Combinaison d'un type de date et d'heure. Le codage des caractères d'une DateHeure doit être conforme à ce qui précède. EXEMPLE: 19850412T101530.
byte[] (octet[])	Tableau d'octets

5.6.3 SECOM_ExchangeMetadataObject

Les objets décrits dans le Tableau 4 et le Tableau 5 contiennent des attributs de métadonnées contenant des informations sur la protection, la compression et les clés publiques utilisées pour la signature numérique des données. L'ordre d'ajout des attributs dans le cadre de la génération et de la vérification de la signature d'enveloppe est réalisé en parcourant les attributs dans l'ordre du premier au dernier, c'est-à-dire dataProtection, protectionScheme, etc. (voir 7.3.4).

Tableau 4 – SECOM_ExchangeMetadataObject

SECOM_ServiceExchangeMetadataObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
dataProtection	1	Obligatoire	Boolean	(S-100) Indique si les données sont chiffrées. 0 indique des données non chiffrées 1 indique des données chiffrées
protectionScheme	1	Obligatoire	CharacterString	(S-100) Spécification ou méthode utilisée pour la protection des données Par exemple S-63, SECOM
digitalSignatureReference	1	Obligatoire	CharacterString	(S-100) Spécifie l'algorithme utilisé pour calculer digitalSignatureValue Par exemple "dsa"
digitalSignatureValue	1	Obligatoire	DigitalSignatureValueObject	Empreinte numérique de la clé publique signée plus signature numérique Tableau 5
compressionFlag	1	Obligatoire	Boolean	(S-100) Indique si les données sont compressées. 0 indique non compressées 1 indique compressées

Tableau 5 – DigitalSignatureValueObject

DigitalSignatureValueObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
publicRootCertificate Thumbprint	0..1	Facultatif	CharacterString	Empreinte numérique déclarée pour clé racine signée (certificat X.509)
publicCertificate	1	Obligatoire	CharacterString	Clé publique (certificat X.509), chaîne Base64 codée au format PEM
digitalSignature	1	Obligatoire	CharacterString	(S-100) Signature numérique au format HEX sur une ligne, sans retour de fin/nouvelle ligne

5.6.4 Transfert de clé publique

5.6.4.1 Généralités

Dans tous les cas, lorsque les données utiles comprennent des clés publiques déclarées, utilisées, par exemple pour les signatures et les données classifiées, il est nécessaire de convertir les clés de manière à les faire tenir dans un format JSON et à garantir l'interopérabilité. Les certificats au format PEM contiennent des sauts de ligne et un en-tête/un pied de page BEGIN/END. Les caractères de codage du saut de ligne diffèrent selon les systèmes d'exploitation (SE). Par exemple, Unix utilise uniquement le caractère \n pour le saut de ligne, tandis que Windows utilise les caractères retour chariot + saut de ligne (\r\n). Afin de garantir l'interopérabilité entre les différents systèmes d'exploitation, une clé publique réduite est introduite pour être utilisée dans les différentes données utiles des objets JSON. D'autres exemples sur la manière d'intégrer des clés codées au format PEM dans les données utiles peuvent être consultés dans la S-100:2018 de l'OHI, Partie 15-7, Chiffrement des données et licences.

5.6.4.2 Producteur

Lors de la construction d'un objet JSON et avant d'ajouter la clé publique au même objet, la clé originale est réduite à une chaîne de caractères d'une seule ligne de la façon suivante:

1) Supprimer tous les sauts de ligne
2) Supprimer l'en-tête -----BEGIN CERTIFICATE-----
3) Supprimer le pied de page -----END CERTIFICATE-----

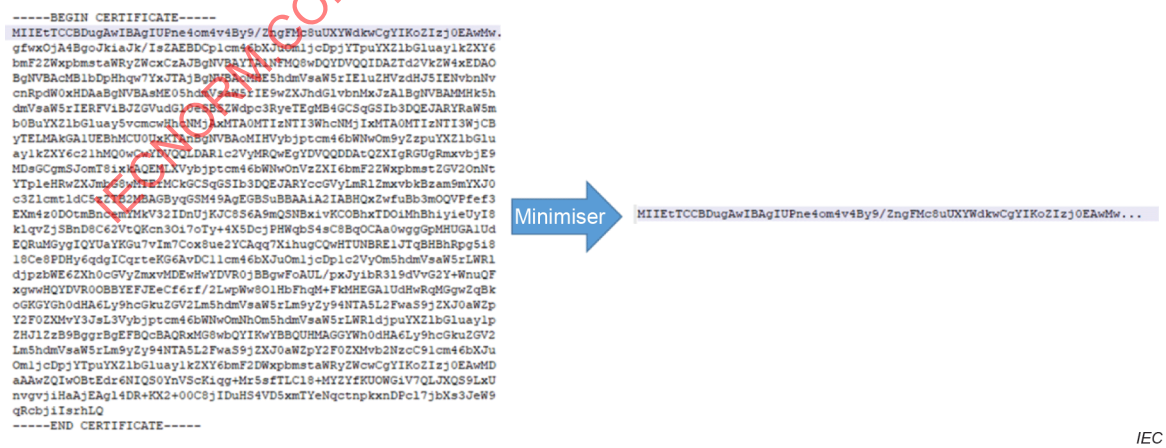
La Figure 6 représente un exemple de la minimisation ci-dessus à l'aide de C# .NET. La routine spécifique *Environment.NewLine* [la routine correspondante en Java est *System.lineSeparator()*] permet de s'assurer que le caractère de saut de ligne correct est utilisé en fonction de la machine sur laquelle le système est installé.

```
public string GetMinifiedPemFromCert(X509Certificate2 cert)
{
    var sb = new StringBuilder();
    using (var sw = new StringWriter(sb))
    {
        var writer = new PemWriter(sw);
        writer.WriteObject(DotNetUtilities.FromX509Certificate(cert));
    }
    //1.
    sb.Replace(Environment.NewLine, "");
    //2.
    sb.Replace("-----BEGIN CERTIFICATE-----", "");
    //3.
    sb.Replace("-----END CERTIFICATE-----", "");
    return sb.ToString();
}
```

IEC

Figure 6 – Exemple en C# de conversion du format PEM en clé publique minimisée

La Figure 7 représente un exemple de clé publique avant et après la conversion.



IEC

Figure 7 – Exemple de clé publique au format PEM convertie en chaîne d'une seule ligne

5.6.4.3 Utilisateur

Pour le destinataire, la conversion est inverse. Lors de l'utilisation des données utiles transférées, la clé publique est reconvertie au format d'origine:

1) ajout de l'en-tête -----BEGIN CERTIFICATE-----;
2) ajout d'un caractère de saut de ligne spécifique au système d'exploitation;
3) division de la chaîne de la clé publique provenant des données utiles en un tableau de 64 caractères au maximum par ligne;
4) pour chaque élément du tableau:
   a) ajout de l'élément en tant que nouvelle ligne;
   b) ajout d'un caractère de saut de ligne spécifique au système d'exploitation;
5) ajout du pied de page -----END CERTIFICATE-----;
6) ajout d'un caractère de saut de ligne spécifique au système d'exploitation;

La Figure 8 représente un exemple de conversion utilisant C# .NET où les sauts de ligne sont ajoutés à l'aide de .NET.

```
public X509Certificate GetCertFromPem(string publicKeyMinified)
{
    //1.
    string publicKey = "-----BEGIN CERTIFICATE-----";
    //2.
    publicKey += Environment.NewLine;
    //3.
    IEnumerable<string> keyArray = SplitIntoArray(publicKeyMinified, 64);
    //4.
    foreach(var row in keyArray)
    {
        //4a.
        publicKey += row;
        //4b.
        publicKey += Environment.NewLine;
    }
    //5.
    publicKey += "-----END CERTIFICATE-----";
    //6.
    publicKey += Environment.NewLine;

    return new X509Certificate(Encoding.UTF8.GetBytes(publicKey));
}
```

IEC

Figure 8 – Exemple en C# de conversion de clé publique minimisée au format PEM

La Figure 9 représente un exemple de résultat lors de la restauration d'une clé publique au format PEM à partir d'une chaîne minimisée.

IEC

Figure 9 – Exemple de chaîne de clé publique minimisée restaurée au format PEM original

5.6.5 PaginationObject

Les interfaces de service "Get" et "Get Summary" peuvent renvoyer plusieurs résultats. Pour limiter la taille de la réponse, ces interfaces prennent en charge la pagination, pour plus de détails, voir 5.5. Le Tableau 6 indique les attributs inclus dans PaginationObject.

Tableau 6 – PaginationObject

PaginationObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
totalItems	1	Obligatoire	PositiveInteger	Nombre total d'éléments qui satisfont à la requête. Il est toujours renvoyé.
maxItemsPerPage	1	Obligatoire	PositiveInteger	Nombre maximal d'éléments que le service doit renvoyer par page.

5.6.6 ContainerTypeEnum

Types de conteneurs utilisés dans les définitions d'interface SECOM. Le Tableau 7 indique les valeurs de l'énumération.

Tableau 7 – ContainerTypeEnum

ContainerTypeEnum	
Valeur	**Définition**
0	S100_DataSet
1	S100_ExchangeSet
2	AUCUNE

5.6.7 SECOM_DataProductType

Le Tableau 8 contient les types de produits pris en charge utilisés dans les interfaces d'information SECOM 5.7.5, 5.7.6, 5.7.8, 5.7.10 et 5.7.13.

Tableau 8 – SECOM_DataProductType

SECOM_DataProductType	
Nom	**Description**
AUTRE	Autres types de données non couverts par ce tableau
S57	S-57 Cartes électroniques de navigation (ENC)
S101	S-101 Cartes électroniques de navigation (ENC)
S102	S-102 Surface bathymétrique
S104	S-104 Information de hauteur d'eau pour la navigation de surface
S111	S-111 Courants de surface
S122	S-122 Aires marines protégées (AMP)
S123	S-123 Services de radio maritimes
S124	S-124 Avertissements de navigation
S125	S-125 Services de navigation maritime
S126	S-126 Environnement physique maritime
S127	S-127 Gestion du trafic maritime
S128	S-128 Catalogue de produits nautiques
S129	S-129 Gestion de la profondeur d'eau sous quille (UKCM)
S131	S-131 Infrastructure portuaire maritime
S210	S-210 Format d'échange inter-STM
S211	S-211 Format de messages relatifs aux escales
S212	S-212 Service d'information numérique VTS
S401	S-401 ENC intérieures
S402	S-402 ENC bathymétriques intérieures
S411	S-411 Information sur la glace
S412	S-412 Couche d'information météorologique
S413	S-413 Conditions météorologiques et de la houle
S414	S-414 Observations météorologiques et de la houle
S421	S-421 Plan de route
RTZ	Plan de route
EPC	Electronic Port Clearance (opérations portuaires assistées par systèmes électroniques)

5.6.8 SECOM_ResponseCodeEnum

L'énumération du Tableau 9 est ajoutée à l'objet réponse HTTP synchrone si une erreur de validation se produit au niveau de l'instance SECOM pour l'information de l'utilisateur de services. Elle est utilisée dans les interfaces push telles que Upload, UploadLink et Acknowledgement pour fournir un message d'erreur plus spécifique.

Tableau 9 – SECOM_ResponseCodeEnum

SECOM_ResponseCodeEnum	
Valeur	**Définition**
0	Données exigées pour le service introuvables
1	Echec de la vérification de la signature
2	Certificat non valide
3	Erreur de validation de schéma

5.6.9 AckRequest Enum

Fanion indiquant qu'une acceptation est attendue lorsque les données sont livrées à l'utilisateur final, et/ou lorsque le contenu des données est traité (ouvert) par l'utilisateur final. La valeur de cette énumération fait office de condition s'il convient d'envoyer une acceptation (valeur = 1, 2, 3) ou non (valeur = 0). Il est obligatoire de mettre en œuvre la fonctionnalité ci-dessus. Le Tableau 10 indique les valeurs d'énumération.

Tableau 10 – AckRequest Enum

AckRequestEnum	
Valeur	**Définition**
0 (par défaut)	Pas d'acceptation (ACK) demandée
1	Delivered ACK demandée
2	Opened ACK demandée
3	Delivered ACK + Opened ACK demandées

5.6.10 Codes de réponse HTTP communs

Le Tableau 11 énumère les codes de réponse HTTP communs à toutes les interfaces à mise en œuvre facultatives. Pour les interfaces obligatoires telles que Ping et Capability, le code 501 n'est pas applicable. Pour les interfaces utilisant le SECOM_ResponseCodeEnum du Tableau 9, la valeur de la colonne correspondante est définie comme nulle.

Tableau 11 – Codes HTTP communs

Code HTTP	Message
401	Non autorisé
405	Méthode non autorisée
500	Erreur de serveur interne
501	Non mis en œuvre

5.6.11 Représentation textuelle connue (WKT)

Type de paramètre de requête "geometry" utilisé dans les interfaces Get, Get Summary et Subscription.

WKT est une représentation lisible par l'homme d'objets spatiaux tels que des points, des lignes ou des zones fermées sur une carte. SECOM utilise un sous-ensemble des objets en WKT CRS 2 (voir ISO 19162) avec projection EPSG 4326 – WGS 84.

Les objets utilisés sont énumérés dans le Tableau 12.

Tableau 12 – Objets géométriques WKT pris en charge

Objets géométriques WKT pris en charge		
Objet	**Définition**	**Exemple**
Point	Géométrie à zéro dimension qui représente un emplacement unique dans l'espace de coordonnées sur une carte. La coordonnée (x,y) est représentée dans l'ordre (longitude, latitude) et exprimée en valeur réelle, au format degré/dixième de degré.	POINT (30 10)
LineString	Courbe avec interpolation linéaire entre les points. Chaque paire de points consécutifs définit un segment de ligne.	LINESTRING (30 10, 10 30, 40 40)
Polygon	Surface plane, c'est-à-dire objet géométrique à deux dimensions, défini par 1 limite extérieure et 0 ou plusieurs limites intérieures. Chaque limite intérieure définit un trou dans le polygone.	POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))
MultiPoint	Matrice d'objets point	MULTIPOINT ((10 40), (40 30), (20 20), (30 10))
MultiLineString	Matrice d'objets LineString	MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))
MultiPolygon	Matrice d'objets polygon	MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))
GeometryCollection	Collection d'un ou plusieurs objets géométriques	POINT (40 10) LINESTRING (10 10, 20 20, 10 40), POLYGON ((40 40, 20 45, 45 30, 40 40)))

5.6.12 Identificateur unique universel (UUID)

Un UUID (voir RFC 4122) est un numéro de 128 bits (16 octets) utilisé pour identifier une entité dans un système d'information. Lorsqu'ils sont générés selon des méthodes standard, les UUID sont uniques avec une probabilité de duplication assez proche de zéro. En SECOM, le type UUID est utilisé comme références de données et comme identifiants de transaction créés par le propriétaire des informations dans le système du FEO.

La chaîne de 128 bits est représentée par 32 caractères hexadécimaux affichés en cinq groupes séparés par des traits d'union, sous la forme 8-4-4-4-12, soit un total de 36 caractères, voir Figure 10.

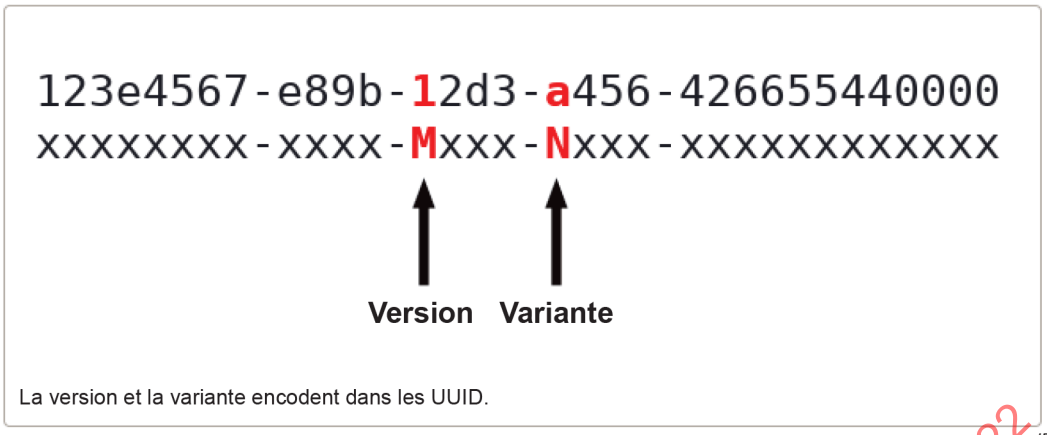


Figure 10 – Version UUID et variante

Les champs de quatre bits M (13e caractère) et des 1 à 3 premiers bits les plus significatifs N (17e caractère) codent le format de l'UUID proprement dit. Dans l'exemple à la Figure 10, M = 1 c'est-à-dire la version 1 et N = a c'est-à-dire la variante 1 puisque les seules valeurs hexadécimales possibles sont 8 (1000), 9 (1001), a (1010) ou b (1011). Il existe cinq versions d'UUID et quatre variantes, la variante 1 (gros-boutiste) étant la plus courante. Voir Tableau 13 et Tableau 14.

Tableau 13 – Variantes d'UUID

Variantes d'UUID			
Nom	**Binaire**	**Chiffre hexadécimal**	**Description**
0	0xxx	0-7	Réservé, rétrocompatibilité NCS
1	10xx	8-b	UUID RFC 4122/DCE 1.1
2	110x	c-d	Réservé, rétrocompatibilité Microsoft Corporation
réservé	111x	e-f	Réservé pour une utilisation future

Tableau 14 – Versions d'UUID

Versions d'UUID		
Nom	**Description du nom**	**Description**
0	nul	Cas particulier où tous les bits sont mis à zéro.
1	date-heure et adresse MAC	Généré à partir d'une heure et d'un identifiant de nœud.
2	date-heure et adresse MAC, version de sécurité	Généré à partir d'un identifiant (généralement un identifiant de groupe ou d'utilisateur), de l'heure et d'un identifiant de nœud. La RFC 4122 réserve la version 2 pour les UUID "à sécurité DCE (Distributed Computing Environment, environnements informatiques distribués)", mais sans fournir de détails. Pour cette raison, de nombreuses mises en œuvre d'UUID omettent la version 2.
3	basé sur le nom	Généré par le hachage MD5 (128 bits) d'un identifiant et d'un nom d'espace de nom.
4	aléatoire	Généré à l'aide d'un nombre aléatoire ou pseudo-aléatoire.
5	basé sur le nom	Généré par le hachage SHA-1 (160 bits) d'un identifiant et d'un nom d'espace de nom.

La version et la variante utilisées lors de la génération de l'UUID sont du ressort du propriétaire des informations et ne relèvent donc pas du SECOM, pour autant que la chaîne formatée qui en résulte soit conforme au format normal de la RFC 4122 et puisse être décodée par l'utilisateur d'informations.

5.6.13 UN/LOCODE

Le "Code des Nations Unies pour le commerce et les lieux pour le transport" est plus connu sous le nom de "UN/LOCODE". Il est géré et tenu à jour par la CEE-ONU. Tous les détails sont disponibles en suivant le lien ci-dessous:

https://unece.org/trade/uncefact/unlocode

Code de localisation selon la liste de codes UN/Locode limitée à une expression régulière [a-zA-Z]{2}[a-zA-Z2-9]{3}. Par exemple SEGOT, USAA7.

5.7 Définitions des interfaces de service

5.7.1 Généralités

Le Tableau 15 donne une vue d'ensemble des interfaces de service qui constituent l'interface de service d'information SECOM. Les interfaces de service Capability et Ping doivent être prises en charge, tandis que la mise en œuvre des autres interfaces est facultative.

Tableau 15 – Vue d'ensemble des interfaces de service

Interface de service	Schéma d'échange	Définition	Cas d'utilisation, voir Annexe B
Upload	ONE_WAY	Interface permettant d'envoyer (push) des informations à l'utilisateur	B.2.1, B.2.2, B.2.3, B.2.4, B.2.7
UploadLink	ONE_WAY	Interface permettant d'envoyer (push) un lien vers des informations pour qu'un utilisateur les obtienne	B.2.6, B.2.8
Acknowledgement	ONE_WAY	Interface permettant d'envoyer une acceptation des informations chargées	B.2.1
Get	REQUEST_RESPONSE	Interface pour demander (pull) des informations au fournisseur	B.2.2
Get Summary	REQUEST_RESPONSE	Interface pour demander (pull) une liste d'informations au fournisseur	B.2.2
Get By Link	ONE_WAY	Interface permettant de récupérer des informations à partir d'un lien chargé	B.2.6, B.2.8
Access	REQUEST_CALLBACK	Interface permettant de demander l'accès aux informations	B.2.1, B.2.4
Access Notification	ONE_WAY	Interface permettant de notifier une demande d'accès	B.2.1, B.2.4
Subscription	PUBLISH_SUBSCRIBE	Interface permettant de créer un abonnement d'informations	B.2.1, B.2.4, B.2.7
Remove Subscription	ONE_WAY	Interface permettant de supprimer un abonnement	B.2.1, B.2.4, B.2.7

Interface de service	Schéma d'échange	Définition	Cas d'utilisation, voir Annexe B
Subscription Notification	ONE_WAY	Interface permettant de notifier des événements d'abonnement	B.2.1, B.2.4, B.2.7
Capability	REQUEST_RESPONSE	Interface permettant d'obtenir les capacités de l'interface. Sa mise en œuvre est obligatoire	B.2.1
Ping	REQUEST_RESPONSE	Interface permettant de vérifier l'état de l'instance de service. Sa mise en œuvre est obligatoire	B.2.4
Encryption Key	ONE_WAY, REQUEST_CALLBACK	Interface permettant d'envoyer de manière sécurisée une clé symétrique pour le chiffrement des données	B.2.3 B.2.6
PublicKey	ONE_WAY, REQUEST_RESPONSE	Interface pour demander (pull) et envoyer (push) un certificat public	B.2.3

5.7.2 Interface de service – Upload

5.7.2.1 Spécification

Le rôle de cette interface est de charger (push) des informations qui ne doivent pas dépasser un poids maximal de 350 ko (codées en Base64) vers un utilisateur d'informations. Un utilisateur d'informations doit mettre en œuvre cette interface afin de recevoir des informations, tandis qu'un fournisseur d'informations peut la mettre en œuvre.

Lors du téléchargement montant (envoi) d'informations, une acceptation peut être demandée. Voir l'interface Acknowledgement pour les détails de l'envoi d'une acceptation.

Le téléchargement montant (envoi) d'informations peut se faire dans le cadre d'un abonnement actif d'informations ou être chargé (envoyé) une fois en tant que message unique, comme l'indique l'attribut fromSubscription.

Les données peuvent être en format XML, XML compressé ou binaire chiffré, généralement compressé, comme décrit dans l'objet de métadonnées d'échange. Les données peuvent contenir soit un message, tel qu'un ensemble de données de plan de route S-421 (IEC 63173-1), soit un ensemble de messages (fichiers) comprimés dans un conteneur ZIP et décrits par un ExchangeSet comme décrit dans la S-100 de l'OHI. Les données sont codées sur une ligne en Base64 avant d'être transférées.

Un ensemble de métadonnées décrivant la protection des données est associé à ces dernières. Les métadonnées font référence au contenu de l'attribut des données, indépendamment du type. Pour l'échange d'un DataSet S-100 de l'OHI, cela signifie que l'ensemble de données peut être en XML (compressionFlag = false) ou en ZIP (compressionFlag = true), ou éventuellement compressé et chiffré (compressionFlag = true et dataProtection = true) et les données sont signées (hachage calculé et signé). Pour l'échange d'un ExchangeSet (catalogue d'échange) S-100 de l'OHI, la même procédure est utilisée. Le ExchangeCatalogue de la S-100 de l'OHI contient un ensemble de fichiers, le dossier est comprimé, et éventuellement chiffré et les données sont signées (hachage calculé et signé).

La Figure 11 représente l'interface en UML.

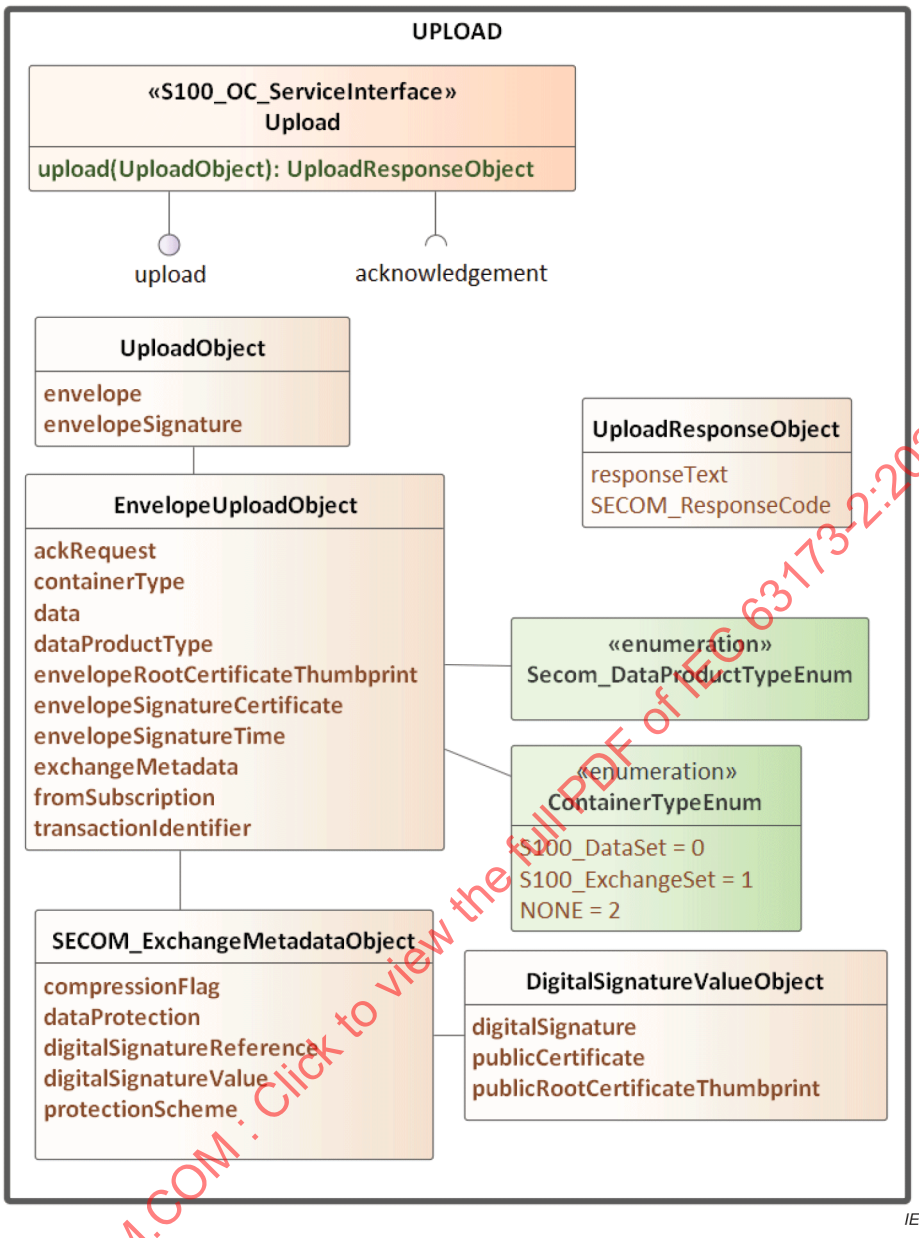


Figure 11 – Diagramme UML de l'interface Upload

5.7.2.2 Modèle d'échange de données

Le Tableau 16 et le Tableau 17 décrivent les données échangées et l'interface.

Tableau 16 – Entrée d'informations pour l'interface Upload

UploadObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
envelope	1	Obligatoire	EnvelopeUploadObject	EnvelopeObject complet en cours de chargement vers le destinataire, y compris les données et les propriétés du message.
envelopeSignature	1	Obligatoire	CharacterString	Signature d'EnvelopeObject au format HEX, sans espace ni saut de ligne.
EnvelopeUploadObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
data	1	Obligatoire	Base64	Données utiles XML dataProductType, codées en base64 (par exemple dans un S100_ExchangeSet, S100_DataSet), ZIP ou binaires. Les données peuvent être ouvertes, protégées et/ou compressées.
containerType	1	Obligatoire	ContainerTypeEnum	Type de conteneur du message dans l'objet de données Tableau 7
dataProductType	1	Obligatoire	SECOM_DataProductType.Name	Colonne Nom du SECOM_DataProductType dans le Tableau 8
exchangeMetadata	1	Obligatoire	SECOM_ExchangeMetadataObject	Les métadonnées d'échange contiennent des informations concernant le schéma de protection, la compression, la signature et l'identité revendiquée, conformément au Tableau 4
fromSubscription	1	Facultatif	Boolean	Fanion indiquant si les données ont été chargées dans le cadre d'un abonnement actif ou non.
ackRequest	1	Obligatoire	AckRequestEnum	Fanion indiquant qu'une acceptation est attendue lorsque les données sont livrées à l'utilisateur final, et/ou lorsque le contenu des données est traité (ouvert) par l'utilisateur final.
transactionIdentifier	1	Obligatoire	UUID	Identifiant de transaction à utiliser, par exemple dans l'acceptation.
envelopeSignatureCertificate	1	Obligatoire	CharacterString	Certificat public de l'expéditeur, utilisé pour vérifier la signature d'enveloppeObject.
envelopeRootCertificateThumbprint	1	Facultatif	CharacterString	Empreinte numérique déclarée pour clé racine signée (certificat X.509)
envelopeSignatureTime	1	Facultatif	DateTime	Horodateur de la signature de l'enveloppe

Tableau 17 – Sortie d'informations pour l'interface Upload

UploadResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
SECOM_ResponseCode	0..1	Obligatoire	SECOM_ResponseCodeEnum	Code d'erreur supplémentaire pour les erreurs internes Tableau 9
message	1	Facultatif	CharacterString	Message de réponse de succès ou d'erreur

5.7.2.3 Conception REST

5.7.2.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

Voir Annexe A pour une définition formelle et détaillée de l'interface de service au format OpenAPI.

5.7.2.3.2 Opération – POST /object

L'interface doit être utilisée pour télécharger (push) de données vers un utilisateur. L'opération attend un objet de données unique et ses métadonnées. Les détails sont décrits dans le Tableau 18. Pour une description détaillée du modèle d'échange de données, voir 5.7.2.2.

Tableau 18 – Mise en œuvre REST de Upload

Opération REST			
POST URL/v1/object {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
UploadObject	application/json	1	Paquet de données (données utiles) avec ses métadonnées
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
UploadResponseObject	application/json	1	Confirmation du téléchargement montant ou message d'erreur, y compris la description des valeurs incorrectes ou introuvables dans les données utiles pour le service spécifique.

5.7.2.3.3 Réponse du service

L'instance de service REST doit répondre avec des messages et des codes HTTP conformes au Tableau 19.

Le Tableau 19 décrit les codes d'erreur internes et les messages générés par le responsable de la mise en œuvre du service. Les codes et messages d'erreur générés par l'environnement de déploiement doivent suivre la HTTP/1.1 (RFC 7231) et ne relèvent pas du domaine d'application du SECOM.

Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces. Pour cette interface avec un attribut supplémentaire, SECOM_ResponseCodeEnum, cet attribut est défini comme nul pour les codes de réponse communs.

Pour les essais liés à ces codes d'erreur, voir 10.14.

Tableau 19 – Codes de réponse et messages HTTP dans l'objet réponse

Code HTTP	SECOM_ResponseCodeEnum	Exemples de messages
200	null	Message chargé avec succès
400	0	Données exigées pour le service introuvables
400	1	Echec de la vérification de la signature
400	2	Certificat non valide
400	3	Erreur de validation de schéma
403	null	Non autorisé à effectuer de téléchargement montant
413	null	Entité demandée trop longue

5.7.2.4 Comportement dynamique

La Figure 12 décrit l'utilisation dynamique de l'interface Upload et sa relation avec l'interface Acknowledgement.

L'interface Upload est utilisée lorsque des données sont envoyées (push) à un autre acteur. La taille des données est techniquement limitée par la technologie REST choisie à des messages de moins de 350 ko. Dans le diagramme de séquence, l'acteur de gauche, l'utilisateur du service SECOM, produit les données à envoyer à l'acteur de droite. L'utilisateur de service SECOM invoque l'interface Upload (il utilise l'interface de service Upload) et envoie les données, la signature des données et les métadonnées complémentaires à l'acteur destinataire. Les métadonnées complémentaires contiennent des attributs pour la demande d'acceptation (facultative), l'identifiant de la transaction, le fanion d'abonnement et la signature du paquet de données complet transféré.

Le destinataire vérifie la taille des données reçues (normalement par le serveur web) et répond par une réponse HTTP 413 si la taille est trop importante pour la configuration actuelle. Un utilisateur de service doit toujours être prêt à traiter une telle réponse d'erreur. Le destinataire des données vérifie l'intégrité dans le cadre de l'étape de vérification de la signature du paquet de données et répond en conséquence par la réponse HTTP 200 si elle est correcte ou 400 et SECOM_ResponseCodeEnum = 1 si elle ne l'est pas, voir Tableau 19.

Les données sont ensuite transmises à une application d'utilisateur, soit directement, soit par le biais de canaux de communication spécifiques au fournisseur. Si une acceptation a été demandée, "delivered ACK" est envoyée de manière asynchrone au point de terminaison d'acceptation désigné dans les métadonnées lorsque les données sont correctement transmises à l'utilisateur final.

L'application d'utilisateur vérifie les droits d'accès et vérifie les signatures reçues, à la fois comme contrôle d'intégrité et comme authentification des données reçues.

Si une acceptation ouverte a été demandée, le message "opened ACK" doit être envoyé de manière asynchrone au point de terminaison d'acceptation désigné dans les métadonnées lorsque les données sont correctement ouvertes ou traitées par l'utilisateur final. Le SECOM ne spécifie pas de délai entre la réception d'un message et l'envoi de "Opened ACK".

Pour plus d'informations sur les schémas d'échange de messages, voir Annexe C.

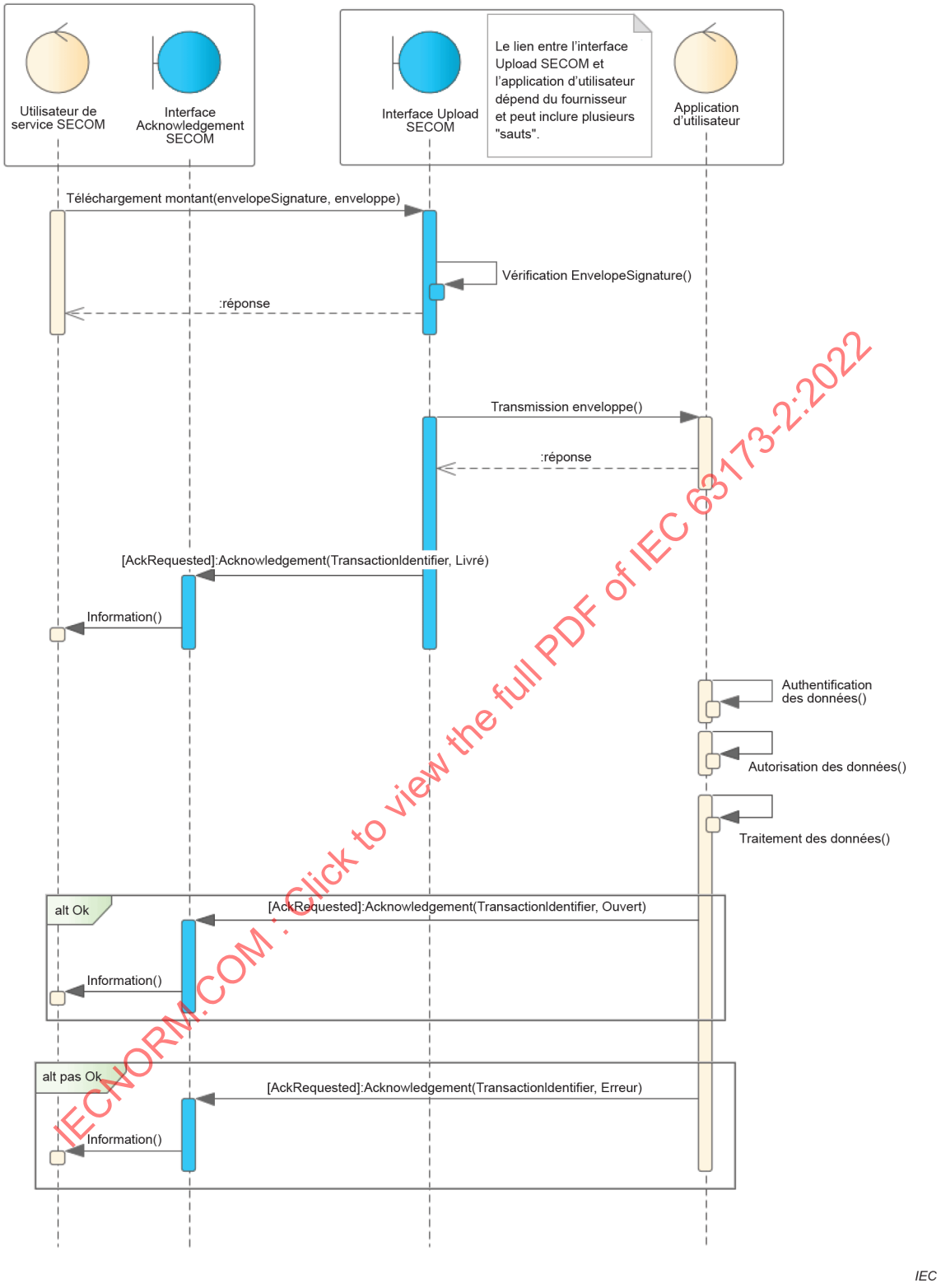


Figure 12 – Diagramme de séquence pour le téléchargement montant de données non classifiées signées avec acceptation

NOTE Les acceptations peuvent être utilisées à des fins de supervision et de diagnostic.

5.7.3 Interface de service – Upload Link

5.7.3.1 Spécification

Le rôle de cette interface est de charger vers un utilisateur (push) un lien vers des informations. Par conséquent, un utilisateur met en œuvre cette interface afin de recevoir un lien vers les informations qui peuvent être récupérées.

Cette interface est utilisée lorsque de grandes quantités de données sont à échanger. Le fournisseur d'informations charge un lien vers un utilisateur, qui utilise ensuite l'interface Get by Link pour extraire les données du fournisseur.

Le processus de téléchargement montant des données en deux étapes (Upload Link suivi de Get By Link) ajoute une complexité à la vérification de la signature. transactionIdentifier se rapporte aux données réelles, mais ce sont toujours les données sous-jacentes qui sont signées, comme dans le cas de l'interface Upload. Par conséquent, la vérification de la signature des données ne peut être effectuée qu'après le téléchargement des données à l'aide de Get By Link. La Figure 13 décrit l'interface en UML.

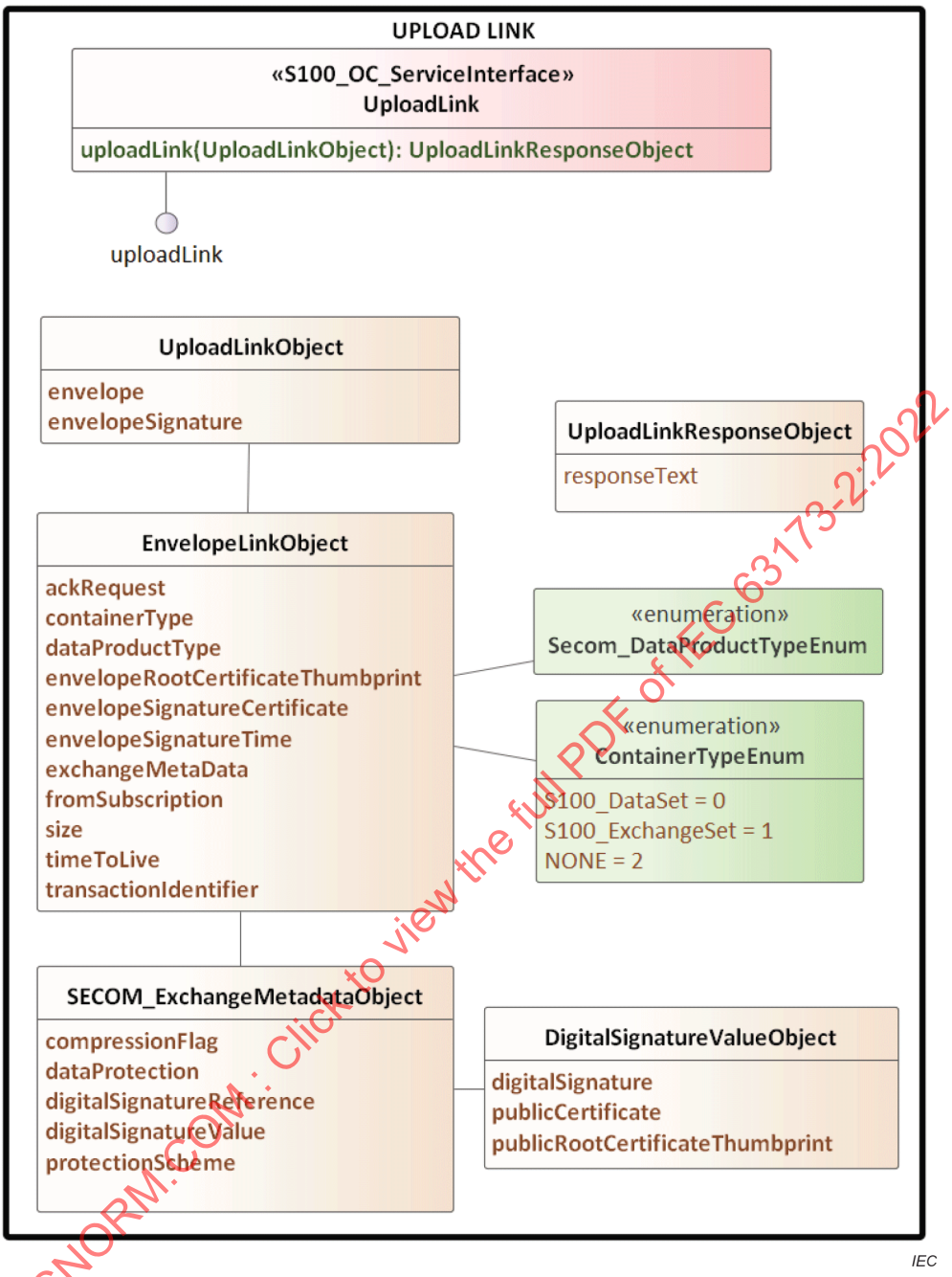


Figure 13 – Diagramme UML de l'interface de lien de téléchargement

5.7.3.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 20 et le Tableau 21.

Tableau 20 – Entrée d'informations pour l'interface Upload Link

UploadLinkObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
envelope	1	Obligatoire	EnvelopeLinkObject	EnvelopeLinkObject complet en cours de chargement vers le destinataire, y compris les propriétés du message.
envelopeSignature	1	Obligatoire	CharacterString	Signature d'EnvelopeLinkObject au format HEX, sans espace ni saut de ligne.
EnvelopeLinkObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
containerType	1	Obligatoire	ContainerTypeEnum	Type de conteneur du message dans l'objet de données, donné dans le Tableau 7
dataProductType	1	Obligatoire	SECOM_DataProductType.Name	Colonne Nom du SECOM_DataProductType dans le Tableau 8
exchangeMetadata	1	Obligatoire	SECOM_ExchangeMetadataObject	Métadonnées d'échange de services avec signature, identification, etc.
fromSubscription	1	Facultatif	Boolean	Fanion indiquant si les données ont été chargées dans le cadre d'un abonnement actif ou non.
ackRequest	1	Obligatoire	AckRequestEnum	Fanion indiquant qu'une acceptation est attendue lorsque les données sont livrées à l'utilisateur final, et/ou lorsque le contenu des données est traité (ouvert) par l'utilisateur final.
transactionIdentifier	1	Obligatoire	UUID	Identifiant de la transaction à utiliser dans l'acceptation et lors de la récupération du message à l'aide de Get By Link.
envelopeSignatureCertificate	1	Obligatoire	CharacterString	Certificat public de l'expéditeur, utilisé pour vérifier la signature d'envelopeLinkObject.
envelopeRootCertificateThumbprint	1	Facultatif	CharacterString	Empreinte numérique déclarée pour clé racine signée (certificat X.509)
size	1	Facultatif	PositiveInteger	Taille des données en ko à télécharger.
timeToLive	1	Obligatoire	DateTime	Date et heure auxquelles les données seront supprimées sur le serveur. Il est nécessaire de récupérer les données avant cette heure.
envelopeSignatureTime	1	Facultatif	DateTime	Horodateur de la signature de l'enveloppe

Tableau 21 – Sortie d'informations pour l'interface Upload Link

UploadLinkResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
SECOM_ResponseCode	0..1	Obligatoire	SECOM_ResponseCodeEnum	Code d'erreur supplémentaire pour les erreurs internes, voir Tableau 9
message	1	Facultatif	CharacterString	Message de réponse de succès ou d'erreur

5.7.3.3 Conception REST

5.7.3.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.3.3.2 Opération – POST /object/link

Opération REST POST /object/link. L'interface doit être utilisée pour télécharger un lien vers des données vers un utilisateur (push). Les détails sont décrits dans le Tableau 22.

Tableau 22 – Mise en œuvre REST de Upload Link

Opération REST			
POST URL/v1/object/link {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
UploadLinkObject	application/json	1	Lien du message avec ses métadonnées
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
UploadLinkResponseObject	application/json	1	Confirmation du téléchargement montant ou message d'erreur

5.7.3.3.3 Réponse du service

L'instance de service doit répondre avec des messages et des codes HTTP conformes au Tableau 23.

Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces. Pour cette interface avec un attribut supplémentaire, SECOM_ResponseCodeEnum, cet attribut est défini comme nul pour les codes de réponse communs.

Pour les essais liés à ces codes d'erreur, voir 10.14.

Tableau 23 – Codes de réponse et messages HTTP dans l'objet réponse

Code HTTP	SECOM_ResponseCodeEnum	Message
200	null	Lien chargé avec succès
400	0	Données exigées pour le service introuvables
400	1	Echec de la vérification de la signature
400	2	Certificat non valide
403	null	Non autorisé à effectuer un téléchargement montant du lien

5.7.3.4 Comportement dynamique

La Figure 14 décrit le comportement dynamique de l'interface de service.

L'interface de service Upload Link est utilisée lorsque les données sont plus volumineuses qu'il n'est techniquement possible pour une opération POST dans un appel de service REST tel que Upload. L'acteur de gauche dans le diagramme de séquence doit envoyer un paquet de données plus important à l'utilisateur (côté droit à la Figure 14).

Les données sont préparées et compressées pour permettre la référence à un seul fichier. Le fichier compressé se voit attribuer un identifiant de transaction unique.

L'identifiant de la transaction, ainsi que les métadonnées contenant les données temporelles si elles sont disponibles, la taille des données et la signature, sont chargés (envoyés) vers l'utilisateur (utilisateur d'informations).

Le destinataire du lien (transactionIdentifier) vérifie la signature d'enveloppe et contrôle les métadonnées, par exemple la taille et la durée de vie, et décide si les données doivent être récupérées immédiatement ou s'il faut attendre qu'une connexion moins chère soit établie. Lorsqu'il est prêt, l'acteur destinataire invoque le service Get By Link pour récupérer les données et les transmettre à l'application d'utilisateur. Après réception dans l'application d'utilisateur de l'utilisateur, la signature des données peut être vérifiée.

La même procédure décrite dans l'interface de service Upload concernant l'acceptation est également mise en œuvre ici. L'acteur destinataire doit toujours envoyer l'acceptation sur demande. Le SECOM ne spécifie pas de délai entre la réception d'un message et l'envoi de "Opened ACK".

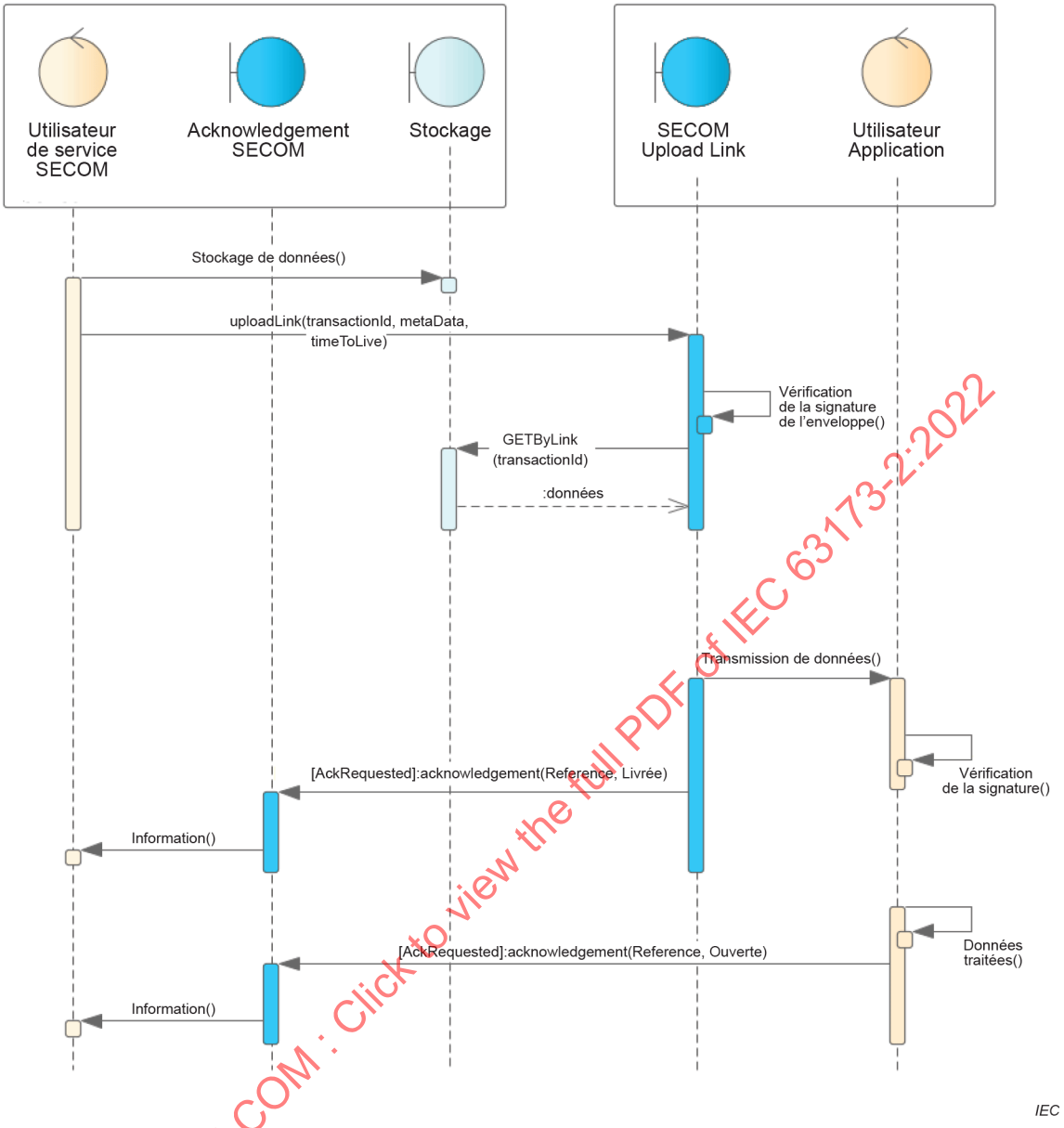


Figure 14 – Diagramme de séquence pour Upload Link vers des données volumineuses

5.7.4 Interface de service – Acknowledgement

5.7.4.1 Spécification

Pendant le téléchargement montant d'informations, une acceptation peut être demandée, et censée être reçue de manière asynchrone lorsque le message chargé a été livré au système final (acceptation technique), et une acceptation lorsque le message a été ouvert et/ou traité par l'utilisateur final (acceptation opérationnelle). L'acceptation contient une référence à l'objet livré et n'a pas de limite de temps.

La Figure 15 décrit l'interface Acknowledgement dans le diagramme UML.

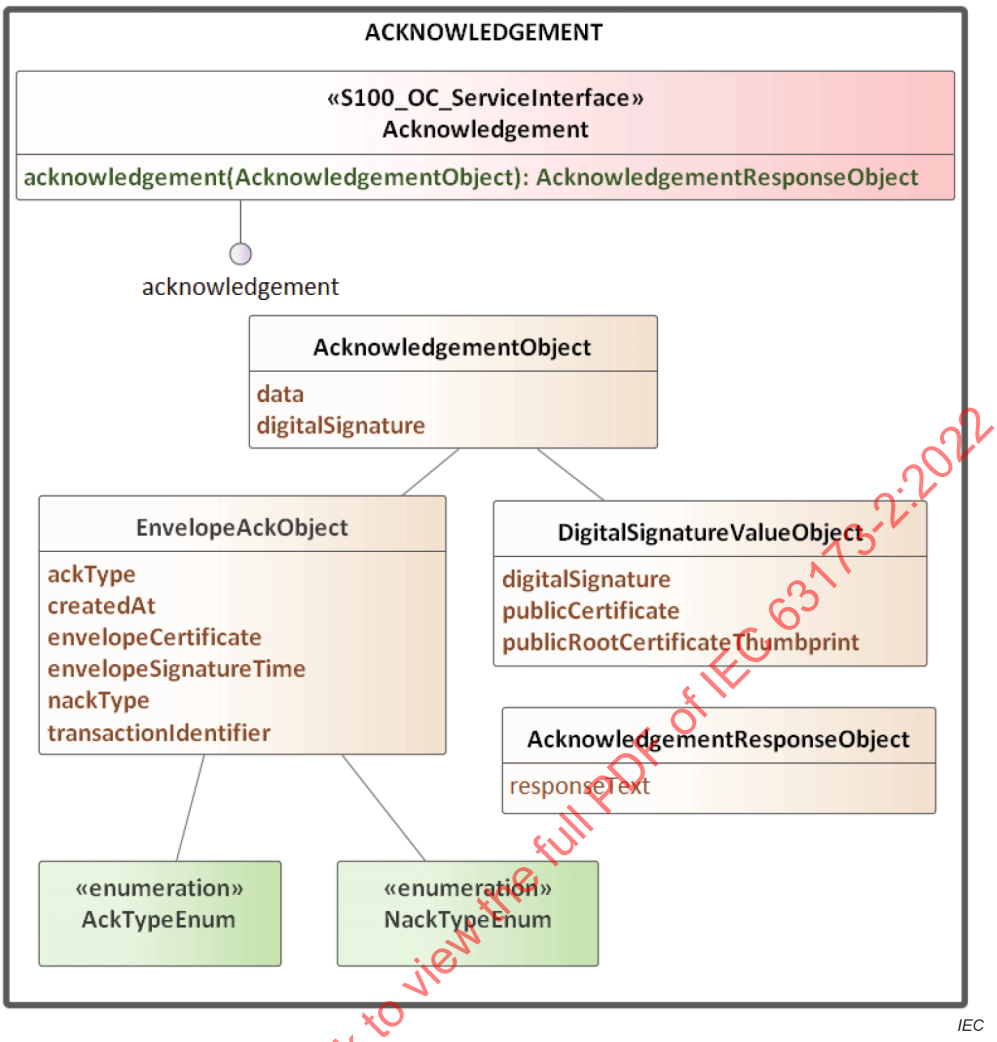


Figure 15 – Diagramme UML de l'interface Acknowledgement

5.7.4.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail du Tableau 24 au Tableau 27.

Tableau 24 – Entrée d'informations pour l'interface Acknowledgement

AcknowledgementObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
envelope	1	Obligatoire	EnvelopeAckObject	Données utiles de l'acceptation à signer
digitalSignature	1	Facultatif	CharacterString	Signature d'EnvelopeAckObject au format HEX, sans espace ni saut de ligne.
EnvelopeAckObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
createdAt	1	Obligatoire	DateTime	Heure de création de l'acceptation
envelopeCertificate	1	Obligatoire	CharacterString	Certificat public de l'expéditeur, utilisé pour vérifier la signature d'enveloppeObject.
envelopeRootCertificateThumbprint	1	Facultatif	CharacterString	Empreinte numérique déclarée pour clé racine signée (certificat X.509)
transactionIdentifier	1	Obligatoire	UUID	Identifiant de référence donné en téléchargement montant
ackType	1	Obligatoire	AckTypeEnum	Type d'acceptation selon le Tableau 27
nackType	0..1	Facultatif	NackTypeEnum	Détails de l'information en cas d'erreur selon le Tableau 25
envelopeSignatureTime	1	Facultatif	DateTime	Horodateur de la signature de l'enveloppe

Tableau 25 – Enumérations pour non accepté

NackTypeEnum	
Valeur	**Définition**
0	Erreur de validation de schéma XML
1	Type de données ou version inconnus
2	Echec de la vérification de la signature des données
3	Echec du déchiffrement.
4	Echec de la décompression

Tableau 26 – Sortie d'informations pour l'interface Acknowledgement

AcknowledgementResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
message	1	Facultatif	chaîne	Message de réponse de succès ou d'erreur
SECOM_ResponseCode	0..1	Obligatoire	SECOM_ResponseCodeEnum	Code d'erreur supplémentaire pour les erreurs internes, voir Tableau 9

Tableau 27 – Enumérations pour l'interface Acknowledgement

AckTypeEnum	
Valeur	**Définition**
1	Delivered ACK Acceptation technique émise lors de la livraison au système final
2	Opened ACK Acceptation opérationnelle émise lors de l'ouverture/lecture par l'utilisateur final
3	Erreur

5.7.4.3 Conception REST

5.7.4.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.4.3.2 Opération – POST /acknowledgement

Les détails relatifs à l'acceptation des opérations dans l'interface sont décrits dans le Tableau 28.

Tableau 28 – Mise en œuvre REST de Acknowledgement

Opération REST			
POST URL/v1/acknowledgement {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
AcknowledgementObject	application/json	1	Objet avec référence à l'information et à l'heure de la livraison
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
AcknowledgementResponseObject	application/json	1	Message de confirmation ou d'erreur selon Tableau 26

5.7.4.3.3 Réponse du service

L'instance de service doit répondre avec des messages et des codes HTTP conformes au Tableau 29.

Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces. Pour cette interface avec un attribut supplémentaire, SECOM_ResponseCodeEnum, cet attribut est défini comme nul pour les codes de réponse communs.

Pour les essais liés à ces codes d'erreur, voir 10.12.

Tableau 29 – Codes et messages de réponse HTTP

Code HTTP	SECOM_ResponseCodeEnum	Message
200	null	Acceptation bien reçue pour <référence>.
400	null	Demande incorrecte
400	0	Données exigées pour le service introuvables
400	1	Echec de la vérification de la signature
400	2	Certificat non valide
403	null	Non autorisé à effectuer un téléchargement montant de l'acceptation

5.7.4.4 Comportement dynamique

La Figure 16 décrit le comportement dynamique d'Acknowledgement. Voir également le comportement dynamique de l'interface Upload et de l'interface Upload Link pour le contexte opérationnel de l'interface Acknowledgement.

L'interface Acknowledgement est utilisée conjointement avec Upload, Upload Link et Get. L'expéditeur (téléchargement montant) de données peut demander une acceptation au destinataire, ce qui permet à l'expéditeur de suivre (tracer) les données envoyées. Cela est particulièrement important lors du téléchargement (envoi) de données vers un navire qui peut être hors connexion au moment de l'envoi, mais qui recevra les données la prochaine fois qu'il sera en ligne. De même, l'expéditeur (répondeur) des données après une demande Get peut demander une acceptation au destinataire.

NOTE Lorsque "Opened ACK" est demandé conjointement avec Upload Link, l'acceptation est envoyée après réception des données à la suite de la demande Get By Link.

Il existe deux acceptations différentes définies en SECOM. La première est "delivered ACK" qui est envoyée par le destinataire lorsque les données ont été acceptées et transmises à l'utilisateur final. La deuxième est "Opened ACK" qui est envoyée par l'utilisateur final lorsque les données reçues sont ouvertes et traitées correctement.

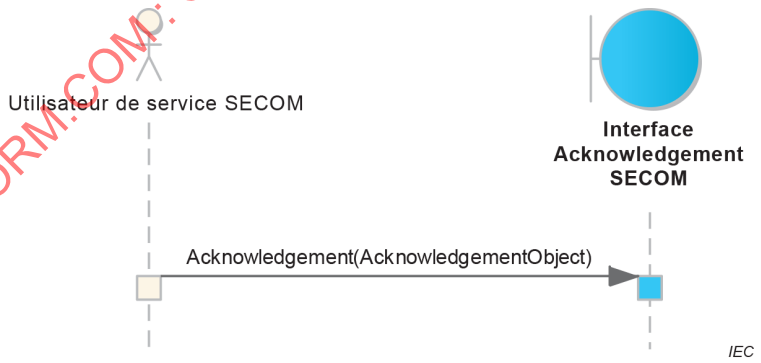


Figure 16 – Diagramme de séquence pour l'interface Acknowledgement

5.7.5 Interface de service – Get

5.7.5.1 Spécification

L'interface Get est utilisée pour extraire des informations d'un fournisseur de services. Le propriétaire de l'information (fournisseur) est responsable de la procédure d'autorisation avant de renvoyer l'information.

L'utilisateur peut demander des informations en fonction de leur référence, de leur géométrie, de l'heure ou d'une requête arbitraire concernant, l'état du produit d'information par exemple. Si aucun paramètre de filtrage n'est donné, toutes les informations autorisées doivent être envoyées. Le propriétaire des informations décide des informations auxquelles l'utilisateur est autorisé à accéder en fonction de l'identité figurant dans le certificat client TLS, c'est-à-dire l'identité à laquelle appartient l'instance de service.

Cette interface peut renvoyer de nombreux objets d'information et prend en charge la pagination décrite en 5.5. Si aucun paramètre de pagination n'est donné, le résultat est tous les messages retournés pour la première page. La Figure 17 représente l'interface Get en UML.

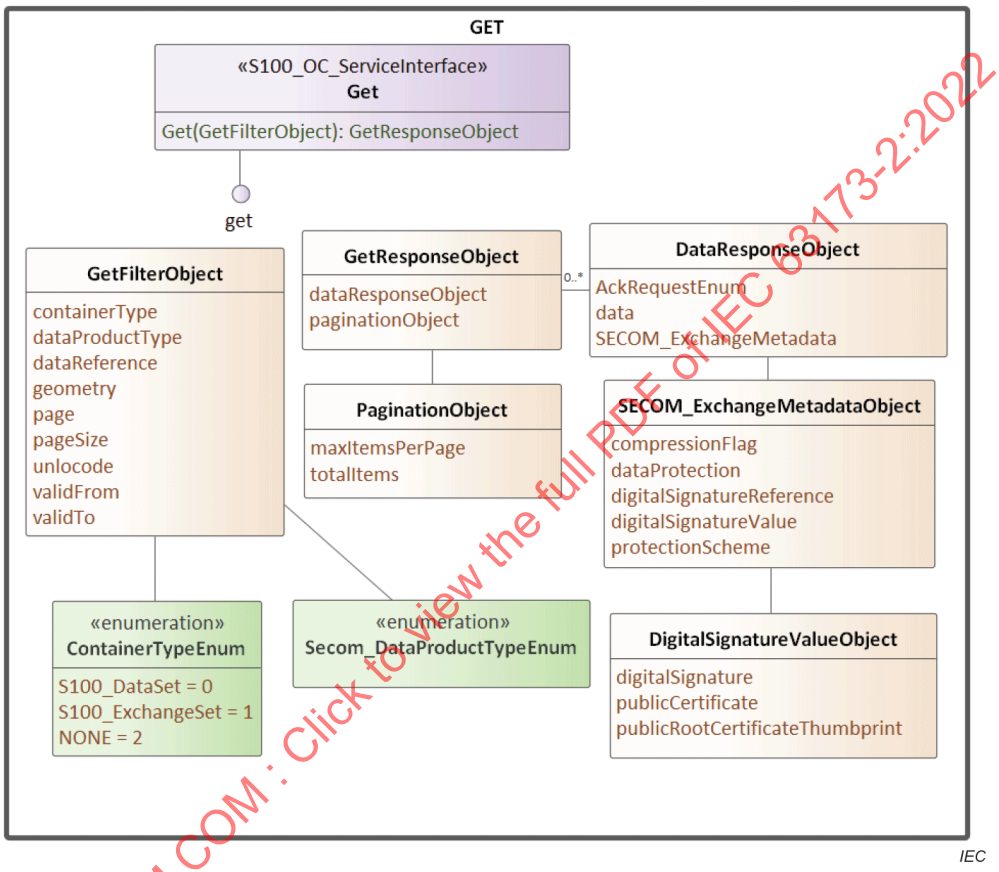


Figure 17 – Diagramme UML de l'interface Get

5.7.5.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 30 et le Tableau 31.

Tableau 30 – Entrée d'informations pour l'interface Get

GetFilterObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
dataReference	0..1	Obligatoire	UUID	Référence à l'objet d'information, par exemple à partir de Get Summary
containerType	0..1	Obligatoire	ContainerTypeEnum	Type de conteneur demandé, voir Tableau 7
dataProductType	0..1	Obligatoire	SECOM_DataProductType.Name	Colonne Nom du SECOM_DataProductType dans le Tableau 8
productVersion	0..1	Facultatif	CharacterString	Version du type de produit basée sur la S-100, par exemple 1.0.0
geometry	0..1	Facultatif	CharacterString	Condition de géométrie pour les objets d'information géolocalisés, voir Tableau 12
unlocode	0..1	Facultatif	CharacterString	Code de l'objet défini, voir 5.6.13
validFrom	0..1	Facultatif	DateTime	Valide à partir de l'heure indiquée
validTo	0..1	Facultatif	DateTime	Valide jusqu'à l'heure indiquée
page	0..1	Obligatoire	PositiveInteger	Page de pagination demandée
pageSize	0..1	Obligatoire	PositiveInteger	Taille de la page de pagination demandée

Tableau 31 – Sortie d'informations pour l'interface Get

GetResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
dataResponseObject	0..*	Obligatoire	DataResponseObject	Liste contenant les données
pagination	1	Obligatoire	PaginationObject	Informations sur la pagination
DataResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
data	1	Obligatoire	Base64	Tableau d'octets codés en Base64. Données conformes à la spécification de produit, intégrées en JSON. Les données peuvent être chiffrées et signées.
exchangeMetadata	1	Obligatoire	SECOM_ExchangeMetadataObject	Métadonnées d'échange de services avec signature, identification, etc.
ackRequest	1	Obligatoire	AckRequestEnum	Fanion indiquant qu'une acceptation est attendue lorsque les données sont livrées à l'utilisateur final, et/ou lorsque le contenu des données est traité (ouvert) par l'utilisateur final.

5.7.5.3 Conception REST

5.7.5.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.5.3.2 Opération – GET /object

Cette opération reçoit une demande Get pour information. Si elle est autorisée, les données sont renvoyées dans la réponse. Il incombe au fournisseur de services d'appliquer la procédure d'autorisation et le contrôle d'accès pertinents aux informations.

Le Tableau 32 décrit les paramètres logiques fournis dans l'interface.

Tableau 32 – Mise en œuvre REST de Get

Opération REST			
GET URL/v1/object?parameters: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
dataReference	String	0..1	Informations récupérées en utilisant la référence donnée dans Get Summary ou Upload Link (UUID).
containerType	DataTypeEnum	0..1	Type de conteneur demandé, voir Tableau 7
dataProductType	SECOM_DataProductType.Name	0..1	Type de produit demandé, par exemple S-122
productVersion	String	0..1	Version du type de produit demandée, par exemple 1.0.0
geometry	String	0..1	Geometry comme paramètre de recherche, voir Tableau 12
unlocode	String	0..1	Code de l'objet défini, voir 5.6.13
validFrom	DateTime	0..1	Heure de début de la période de validité de l'objet d'information
validTo	DateTime	0..1	Heure de fin de la période de validité de l'objet d'information
page	PositiveInteger	0..1	Page de pagination demandée
pageSize	PositiveInteger	0..1	Taille de la page de pagination demandée
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Pas de corps défini	s.o.	s.o.	s.o.
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
GetResponseObject	application/json	1	Ensemble de paquets de messages avec métadonnées ou message d'erreur

5.7.5.3.3 Réponse du service

Le Tableau 33 décrit la réponse de l'instance de service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.15.

Tableau 33 – Codes et messages de réponse HTTP de Get

Code HTTP	Message
200	GetMessageResponseObject
400	Demande incorrecte
403	Non autorisé à accéder aux informations demandées
404	Informations introuvables

5.7.5.4 Comportement dynamique

La Figure 18 et la Figure 19 décrivent le comportement dynamique de l'interface Get.

L'interface Get est utilisée pour extraire des données des informations publiées par un acteur. La demande de service contient des paramètres de filtrage qui permettent au propriétaire des informations de rechercher et de préparer les données à renvoyer. Une fois que l'authentification du demandeur a été vérifiée, la préparation des données comprend un contrôle d'autorisation par rapport à la liste de contrôle d'accès interne et l'empaquetage des données avec des signatures de données. Si l'accès est accepté, les données demandées sont renvoyées, sinon, un message d'erreur est émis.

Si le propriétaire des informations décide de publier des données protégées, il est nécessaire que l'utilisateur ait la possibilité de demander la clé de chiffrement, si elle n'est pas déjà disponible, pour pouvoir déchiffrer les données protégées. L'utilisateur demande la clé de chiffrement par l'identifiant de référence des données et son certificat public utilisé pour la dérivation de la clé symétrique. Le propriétaire des informations chiffre la clé aléatoire utilisée lors du chiffrement des données avec une clé symétrique dérivée de la clé publique de l'utilisateur et de sa propre clé privée. Le propriétaire des informations envoie la clé aléatoire protégée par l'intermédiaire de l'interface SECOM Encryption Key de l'utilisateur.

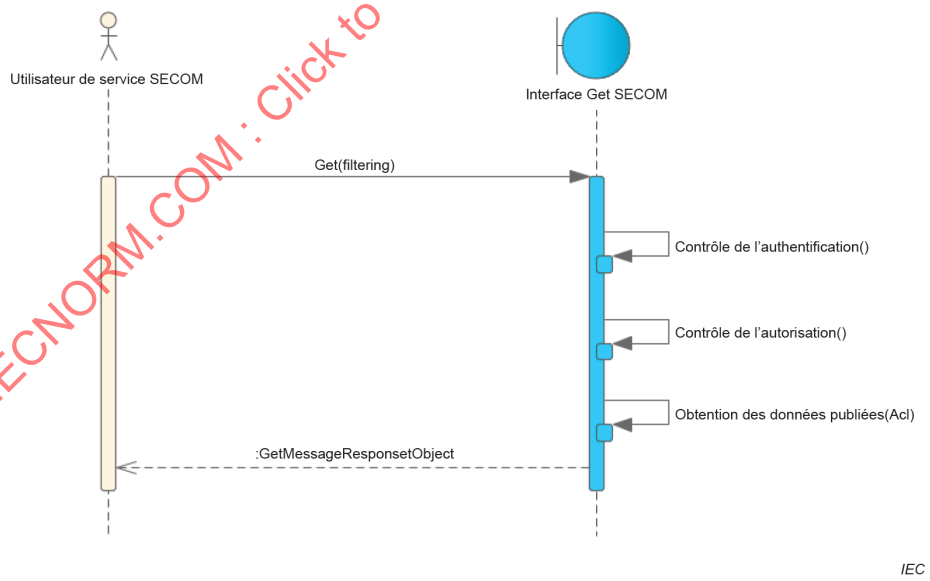


Figure 18 – Diagramme de séquence de l'interface Get

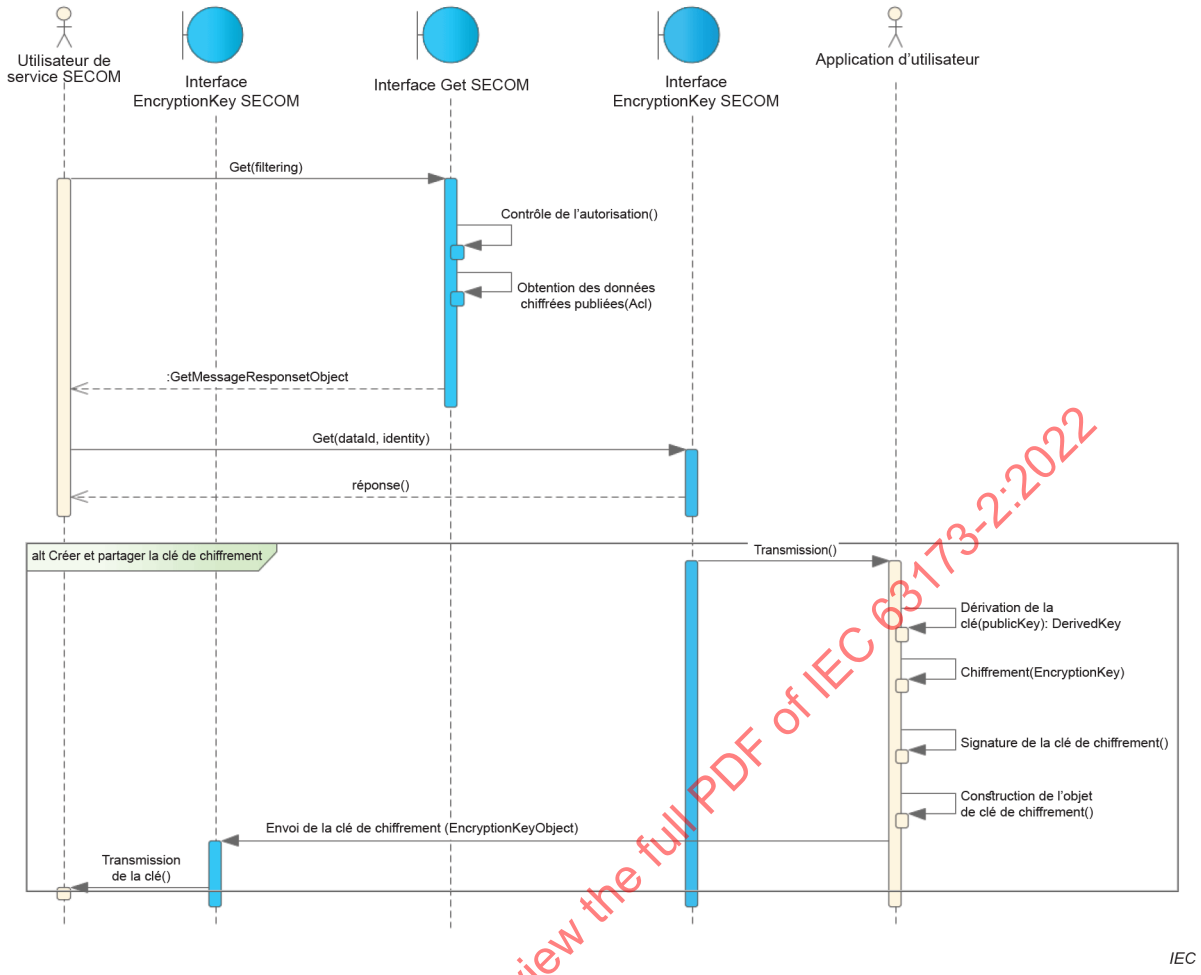


Figure 19 – Diagramme de séquence de l'interface Get et de données classifiées

5.7.6 Interface de service – Get Summary

5.7.6.1 Spécification

Une liste d'informations doit être renvoyée par cette interface. Le résumé contient l'identité, l'état, la taille et une courte description de chaque objet d'information. L'objet d'information réel doit être récupéré au moyen de l'interface Get. L'utilisateur peut demander des informations par géométrie, lieu et heure. Si aucun paramètre de filtrage n'est donné, les informations résumées disponibles doivent être renvoyées.

NOTE Les informations connues par le fournisseur d'informations peuvent être sensibles. Le fournisseur d'informations juge ce qui est disponible et qui est donc à inclure dans la réponse à Get Summary.

Cette interface peut renvoyer de nombreux objets d'information et prend en charge la pagination décrite en 5.5.

La Figure 20 décrit l'interface Get Summary dans le diagramme UML.

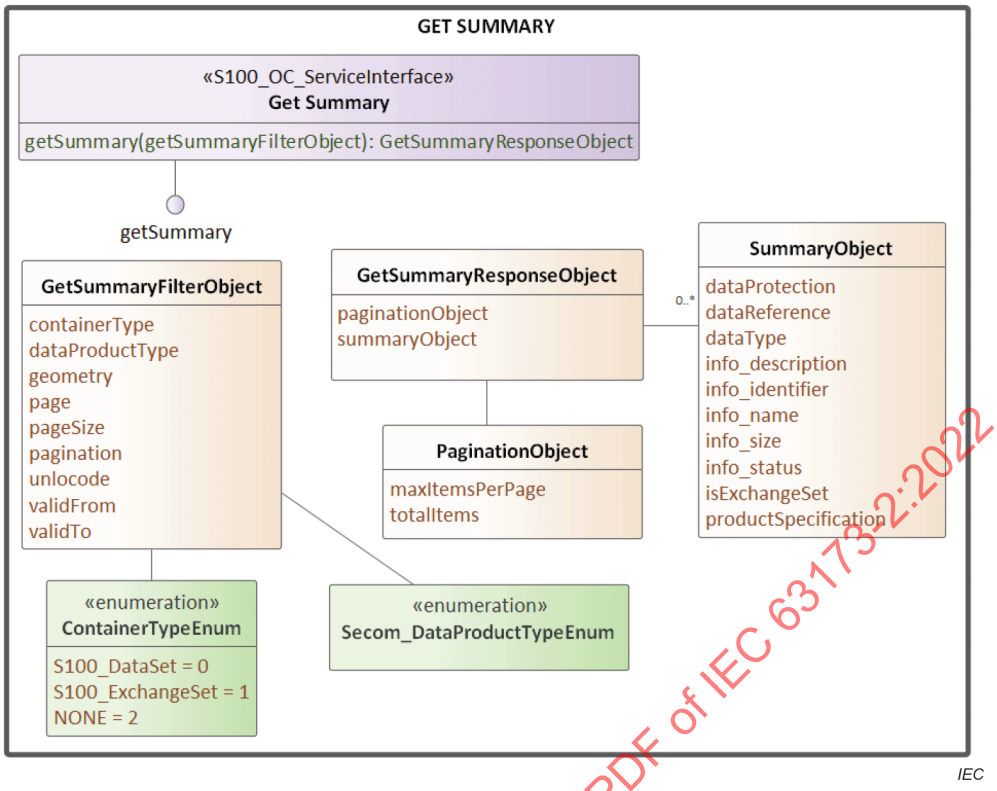


Figure 20 – Diagramme UML de l'interface Get Summary

5.7.6.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 34 et le Tableau 35.

Tableau 34 – Entrée d'informations pour l'interface Get Summary

GetSummaryFilterObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
containerType	0..1	Obligatoire	ContainerTypeEnum	Type de conteneur demandé, voir Tableau 7
dataProductType	0..1	Obligatoire	SECOM_DataProductType.Name	Colonne Nom du SECOM_DataProductType dans le Tableau 8
productVersion	0..1	Facultatif	CharacterString	Version du type de produit basée sur la S-100, par exemple 1.0.0
geometry	0..1	Facultatif	CharacterString	Condition de géométrie pour les objets d'information géolocalisés, voir Tableau 12
unlocode	0..1	Facultatif	CharacterString	Code de l'objet défini, voir 5.6.13
validFrom	0..1	Facultatif	DateTime	Valide à partir de l'heure indiquée
validTo	0..1	Facultatif	DateTime	Valide jusqu'à l'heure indiquée
page	0..1	Obligatoire	PositiveInteger	Page de pagination demandée
pageSize	0..1	Obligatoire	PositiveInteger	Taille de la page de pagination demandée

Tableau 35 – Sortie d'informations pour l'interface Get Summary

GetSummaryResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
summaryObject	0..*	Obligatoire	SummaryObject	Description de l'objet d'information
pagination	1	Obligatoire	PaginationObject	Informations sur la pagination
SummaryObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
dataReference	1	Obligatoire	UUID	Référence aux données
dataProtection	1	Facultatif	Boolean	Fanion indiquant si les données sont chiffrées ou non.
dataCompression	1	Facultatif	Boolean	Fanion indiquant si les données sont compressées ou non.
containerType	1	Facultatif	ContainerTypeEnum	Type de conteneur, voir Tableau 7
dataProductType	1	Facultatif	SECOM_DataProductType.Name	Colonne Nom du SECOM_DataProductType dans le Tableau 8
info_identifier	0..1	Facultatif	CharacterString	Identifiant de l'objet d'information, par exemple <S100XC:identifier>, gml:id
info_name	0..1	Facultatif	CharacterString	Nom de l'objet d'information, par exemple <S100XC:exchangeCatalogueName>, <S100:datasetFileIdentifier>
info_status	0..1	Facultatif	CharacterString	Etat de l'objet d'information, par exemple actif, inactif.
info_description	0..1	Facultatif	CharacterString	Description de l'objet d'information, par exemple <S100XC:description>, <S100:datasetTitle>
info_lastModifiedDate	0..1	Facultatif	DateTime	Date de la dernière modification, par exemple <S100XC:date>, <S100:datasetReferenceDate>
info_productVersion	0..1	Facultatif	CharacterString	Version du type de produit basée sur la S-100, par exemple 1.0.0
info_size	0..1	Facultatif	PositiveInteger	Taille des données en ko exprimée par int64 non signé long ou non signé avec une valeur maximale de 2^{64}.

5.7.6.3 Conception REST

5.7.6.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.6.3.2 Opération – GET /object/summary

Cette opération reçoit une demande Get Summary. La réponse contient un résumé des informations disponibles. Les informations réelles peuvent être récupérées par demande Subscription ou demande Get.

Le Tableau 36 décrit l'interface de service REST.

Tableau 36 – Mise en œuvre REST de Get Summary

Opération REST			
GET URL/v1/object/summary?parameters: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
geometry	String	0..1	Geometry comme paramètre de recherche, voir Tableau 12
unlocode	String	0..1	Code de l'objet défini, voir 5.6.13
validFrom	DateTime	0..1	Heure de début de la période de validité de l'objet d'information
validTo	DateTime	0..1	Heure de fin de la période de validité de l'objet d'information
page	PositiveInteger	0..1	Numéro de page demandé
pageSize	PositiveInteger	0..1	Taille de page demandée
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Pas de corps défini	s.o.	s.o.	s.o.
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
GetSummaryResponseObject	application/json	1	Liste des objets d'information disponibles, identifiés par une identité, un état et une brève description ou un message d'erreur.

5.7.6.3.3 Réponse du service

Le Tableau 37 décrit la réponse de l'instance de service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.15.

Tableau 37 – Codes et messages de réponse HTTP de Get Summary

Code HTTP	Message
200	GetSummaryResponseObject
400	Demande incorrecte
403	Non autorisé à accéder aux informations demandées
404	Informations introuvables

5.7.6.4 Comportement dynamique

La Figure 21 décrit le comportement dynamique de l'interface Get Summary.

L'interface Get Summary est utilisée pour extraire les métadonnées d'un acteur. La réponse reçue de métadonnées peut être utilisée pour sélectionner les données réelles à récupérer par une nouvelle demande utilisant l'interface Get en 5.7.5. La demande de service contient des paramètres de filtrage qui permettent au propriétaire des informations de rechercher et de préparer les métadonnées à renvoyer. Une fois l'authentification du demandeur vérifiée, la préparation des métadonnées comprend le jugement interne de la disponibilité des informations et l'empaquetage des métadonnées. Si les métadonnées sont disponibles, les métadonnées demandées sont renvoyées, sinon, un message d'erreur est émis.

Figure 21 – Diagramme de séquence de l'interface Get Summary

5.7.7 Interface de service – Get By Link

5.7.7.1 Spécification

L'interface Get By Link est utilisée pour extraire des informations d'un stockage de données géré par le propriétaire des informations. Le lien avec le stockage de données peut être échangé avec l'interface Upload Link. Le propriétaire de l'information (fournisseur) est responsable de la procédure d'authentification et d'autorisation pertinente avant de renvoyer l'information.

La Figure 22 décrit l'interface Get By Link en UML.

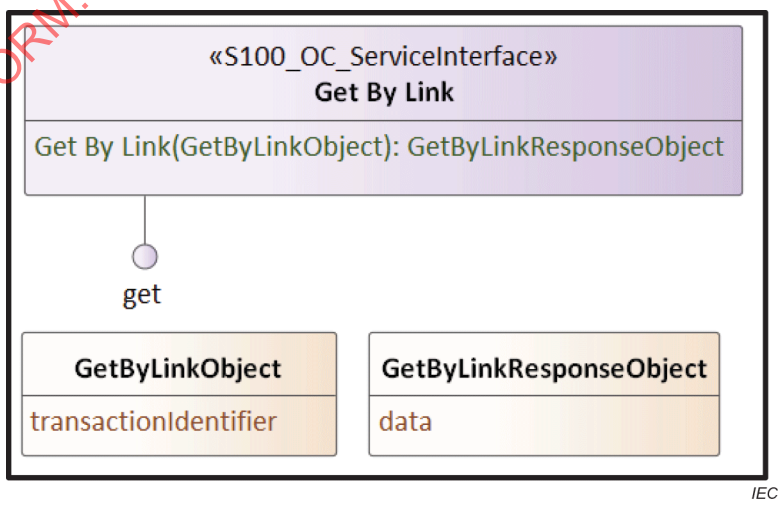


Figure 22 – Interface Get By Link en UML

5.7.7.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 38 et le Tableau 39.

Tableau 38 – Entrée d'informations pour l'interface Get By Link

GetByLinkObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
transactionIdentifier	1	Obligatoire	UUID	Identifiant chargé dans Upload Link

Tableau 39 – Sortie d'informations pour l'interface Get By Link

GetByLinkResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
data	1	Obligatoire	Base64	Données au format binaire codé en Base64

5.7.7.3 Conception REST

5.7.7.3.1 Généralités

L'interface du service est définie avec la technologie REST.

5.7.7.3.2 Opération – GET /object/link

Cette opération reçoit une demande d'information Get By Link. Si elle est autorisée, les données sont renvoyées dans la réponse. Il incombe au fournisseur de services d'appliquer la procédure d'autorisation et le contrôle d'accès pertinents aux informations.

Le Tableau 40 décrit les paramètres logiques fournis dans l'interface.

Tableau 40 – Mise en œuvre REST de Get By Link

Opération REST			
GET URL/v1/object/link?parameter: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
transactionIdentifier	String	1	Référence à la transaction de données de Upload Link, UUID
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Pas de corps défini	s.o.	s.o.	s.o.
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
data	application/octet-stream	1	Données binaires

5.7.7.3.3 Réponse du service

Le Tableau 41 décrit la réponse de l'instance de service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.11.

Tableau 41 – Codes et messages de réponse HTTP de Get By link

Code HTTP	SECOM_ResponseCodeEnum	Message
200	null	Réussite
400	null	Demande incorrecte
400	2	Certificat non valide
403	null	Non autorisé à accéder aux informations demandées
404	null	Informations avec <transactionIdentifier> introuvables.

5.7.7.4 Comportement dynamique

La Figure 23 décrit le comportement dynamique de l'interface Get By Link.

L'interface Get By Link est utilisée pour récupérer l'objet de données fourni par le propriétaire des informations après avoir invoqué l'interface Upload Link du destinataire. Le destinataire du lien (transactionIdentifier) contrôle les métadonnées, par exemple la taille et la durée de vie, et décide si les données doivent être récupérées immédiatement ou s'il faut attendre qu'une connexion moins chère soit établie. Lorsqu'il est prêt, l'acteur destinataire invoque l'opération Get By Link pour récupérer les données. Enfin, la vérification de la signature des données est effectuée en utilisant les données reçues correspondant à l'objet digitalSignatureValue de l'objet demande Upload Link reçu précédemment.

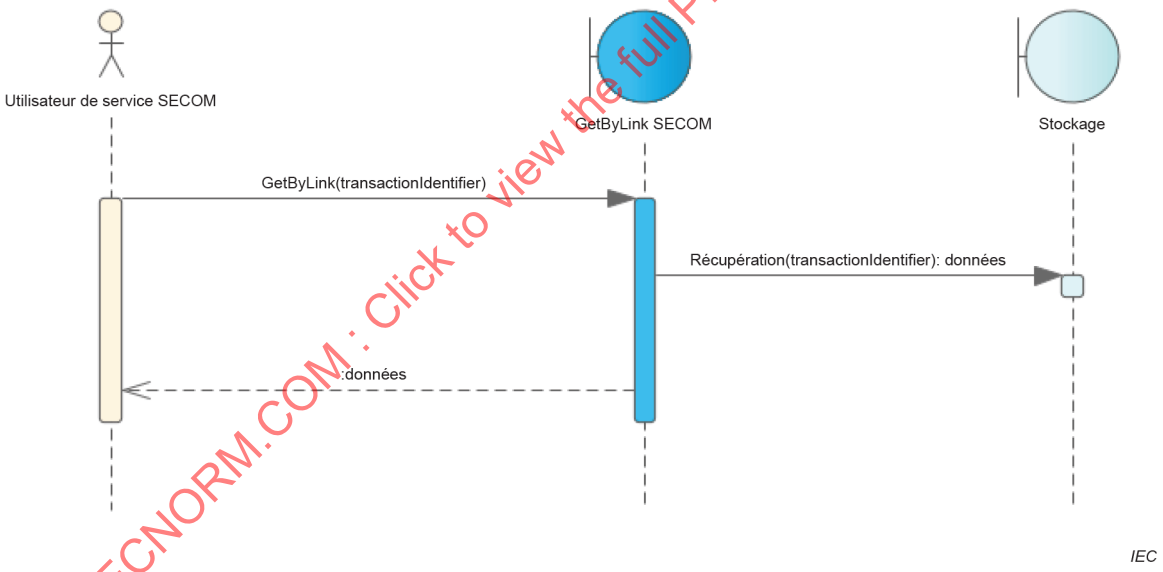


Figure 23 – Diagramme de séquence de l'interface Get By Link

5.7.8 Interface de service – Access

5.7.8.1 Spécification

L'accès aux informations peut être demandé par le biais de l'interface Access. Le résultat est envoyé de manière asynchrone par le biais de l'interface Access Notification.

La Figure 24 décrit l'interface Access dans le diagramme UML.

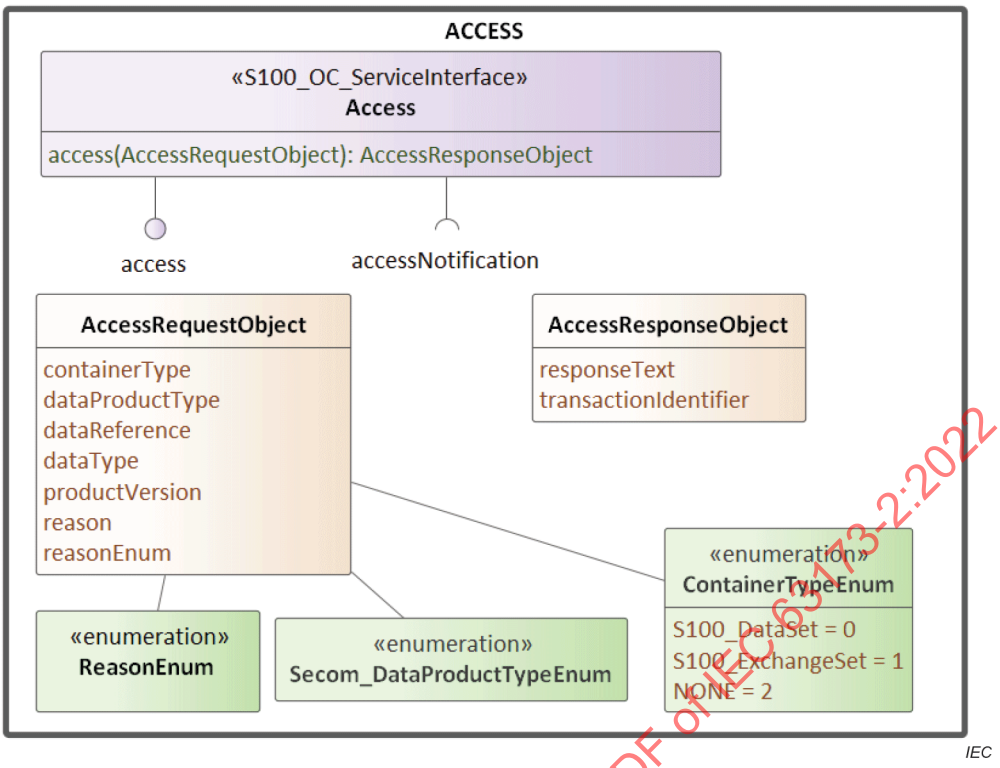


Figure 24 – Diagramme UML de l'interface Access

5.7.8.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 42, le Tableau 43 et le Tableau 44.

Tableau 42 – Entrée d'informations pour l'interface Access

AccessRequestObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
reason	1	Obligatoire	CharacterString	Motif de la demande d'accès, lisible par l'homme
reasonEnum	1	Obligatoire	ReasonEnum	Motif de la demande d'accès, exploitable par une machine
containerType	0..1	Obligatoire	ContainerTypeEnum	Type de conteneur demandé, voir Tableau 7
dataProductType	0..1	Obligatoire	SECOM_DataProduct Type.Name	Colonne Nom du SECOM_DataProductType dans le Tableau 8
dataReference	0..1	Obligatoire	UUID	Référence aux données dont l'accès est demandé
productVersion	0..1	Facultatif	CharacterString	Version du type de produit basée sur la S-100, par exemple 1.0.0

Tableau 43 – Sortie d'informations pour l'interface Access

AccessResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
message	1	Facultatif	CharacterString	Message de réponse de succès ou d'erreur

Tableau 44 – Enumérations pour l'interface Access

ReasonEnum	
Valeur	**Définition**
0	Exigé par les autorités
1	Exigé par le fournisseur de services

5.7.8.3 Conception REST

5.7.8.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.8.3.2 Opération – POST /access

Cette opération reçoit une demande d'accès par un utilisateur. Le résultat est envoyé de manière asynchrone.

Le Tableau 45 décrit les paramètres logiques fournis dans l'interface.

Tableau 45 – Liens avec les paramètres de l'opération

Opération REST			
POST URL/v1/access {body}; return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
AccessRequestObject	application/json	1	Description du motif de la demande d'accès aux informations
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
AccessResponsObject	application/json	1	Confirmation ou message d'erreur. Le résultat de la demande est envoyé de manière asynchrone par l'interface Access Notification

5.7.8.3.3 Réponse du service

Le Tableau 46 décrit la réponse de l'instance de service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.9.

Tableau 46 – Codes de réponse HTTP

Code HTTP	Message
200	Réussite
400	Demande incorrecte
403	Non autorisé à accéder aux informations demandées

5.7.8.4 Comportement dynamique

La Figure 25 décrit le comportement dynamique des interfaces Request Access et Access Notification.

L'interface Access est utilisée pour demander l'accès à un ou plusieurs objets d'information. L'utilisateur du service obtient une réponse asynchrone contenant la décision par le biais de l'interface Access Notification de l'utilisateur.

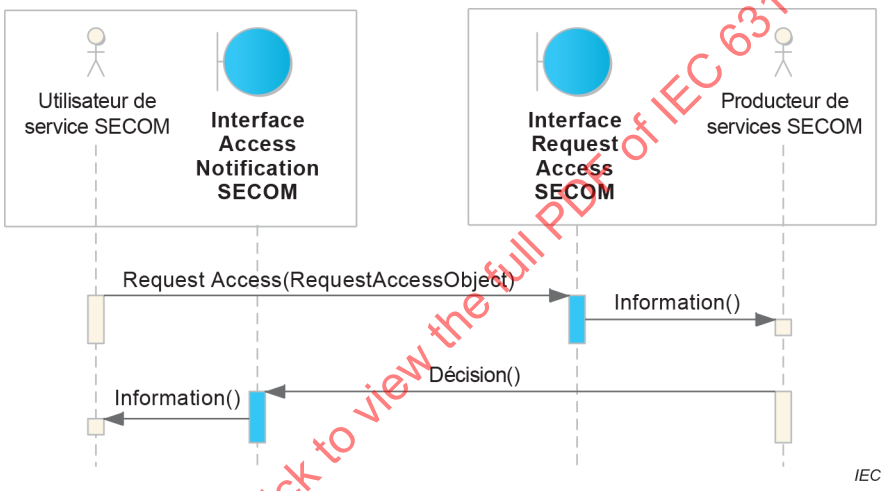


Figure 25 – Diagramme de séquence des interfaces Request Access et Access Notification

5.7.9 Interface de service – Access Notification

5.7.9.1 Spécification

Le résultat de la demande Access doit être envoyé de manière asynchrone par le biais de l'interface Access Notification.

La Figure 26 décrit l'interface Access Notification dans le diagramme UML.

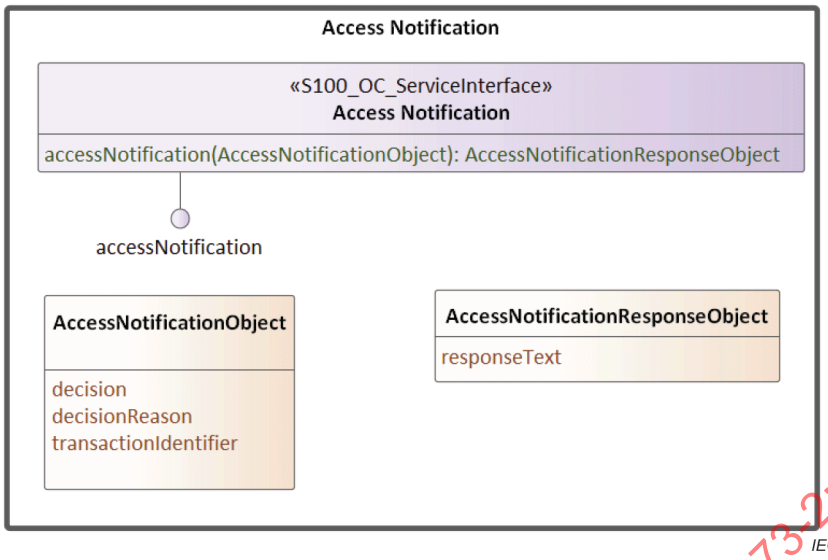


Figure 26 – Diagramme UML de l'interface Access Notification

5.7.9.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 47 et le Tableau 48.

Tableau 47 – Entrée d'informations pour l'interface Access Notification

AccessNotificationObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
decision	1	Obligatoire	Boolean	Décision concernant la demande Access, oui ou non
decisionReason	1	Facultatif	CharacterString	Motif de la décision lisible par l'homme
transactionIdentifier	1	Obligatoire	UUID	Identifiant de transaction correspondant à Request Access

Tableau 48 – Sortie d'informations pour l'interface Access Notification

AccessNotificationResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
message	1	Facultatif	CharacterString	Message de réponse de réussite

5.7.9.3 Conception REST

5.7.9.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.9.3.2 Opération – POST /access/notification

Cette opération reçoit le résultat de la demande Access.

Le Tableau 49 décrit les paramètres logiques fournis dans l'interface.

Tableau 49 – Liens avec les paramètres de l'opération

Opération REST			
POST URL/v1/access/notification {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
AccessNotificationObject	application/json	1	Résultat de la demande d'accès: Vrai ou Faux et motif.
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
AccessNotificationResponseObject	application/json	1	Confirmation ou message d'erreur

5.7.9.3.3 Réponse du service

Le Tableau 50 décrit la réponse du service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.9.

Tableau 50 – Codes de réponse HTTP

Code HTTP	Message
200	Réussite
400	Demande incorrecte

5.7.9.4 Comportement dynamique

Cette interface est une réponse asynchrone à l'interface de service Request Access. Voir Figure 25.

5.7.10 Interface de service – Subscription

5.7.10.1 Spécification

Le rôle de l'interface est de demander un abonnement à des informations, soit à des informations spécifiques en fonction de paramètres, soit à des informations accessibles sur décision du fournisseur d'informations. Chaque demande d'abonnement reflète un ensemble de requêtes de paramètres.

La Figure 27 décrit l'interface Subscribe dans le diagramme UML.

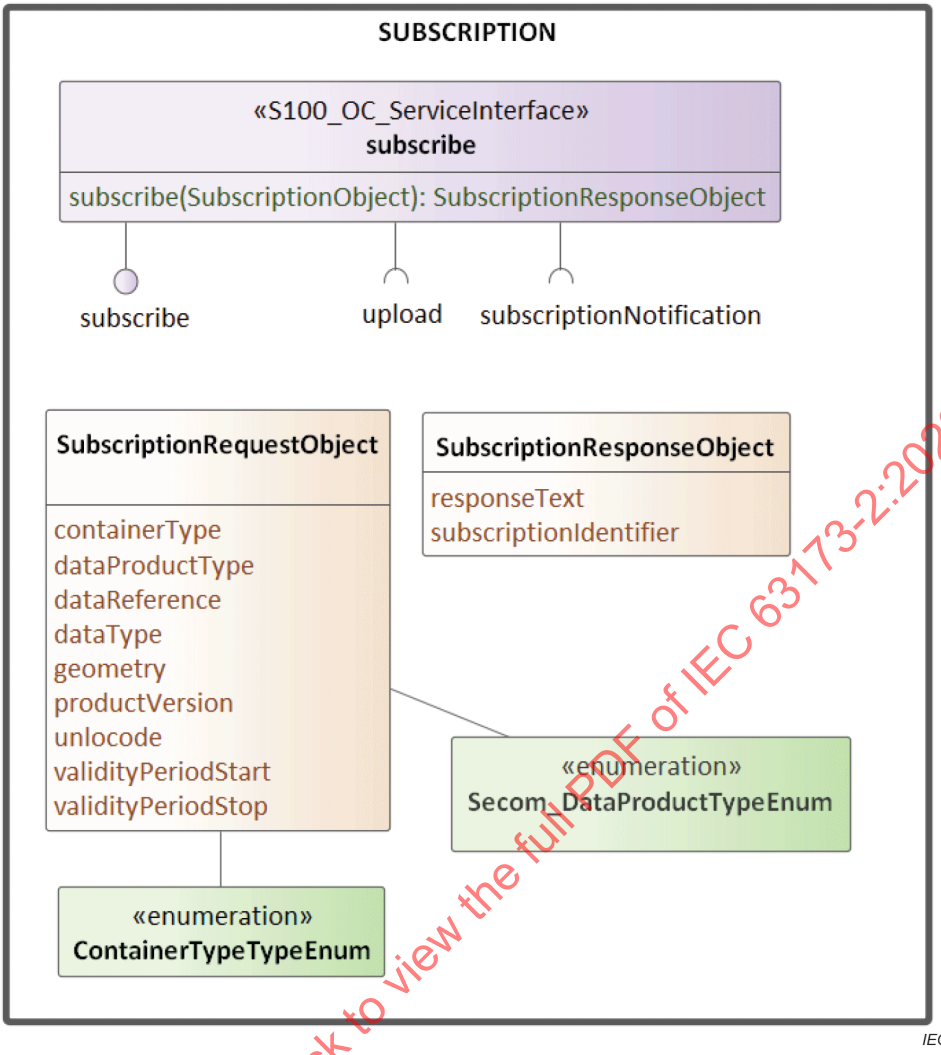


Figure 27 – Diagramme UML de l'interface Subscribe

5.7.10.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 51 et le Tableau 52.

Tableau 51 – Entrée d'informations pour l'interface Subscription

SubscriptionRequestObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
containerType	0..1	Obligatoire	ContainerTypeEnum	Type de conteneur demandé, voir Tableau 7
dataProductType	0..1	Obligatoire	SECOM_DataProductType.Name	Colonne Nom du SECOM_DataProductType dans le Tableau 8
dataReference	0..1	Obligatoire	UUID	Référence aux données
productVersion	0..1	Facultatif	CharacterString	Version du type de produit basée sur la S-100, par exemple 1.0.0
geometry	0..1	Facultatif	CharacterString	Geometry comme critère, voir Tableau 12
unlocode	0..1	Facultatif	CharacterString	Code de l'objet défini, voir 5.6.13
subscriptionPeriodStart	0..1	Facultatif	DateTime	Heure de début de l'abonnement
subscriptionPeriodEnd	0..1	Facultatif	DateTime	Heure de fin de l'abonnement

Tableau 52 – Sortie d'informations pour l'interface Subscription

SubscriptionResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
message	1	Facultatif	CharacterString	Message de réponse
subscriptionIdentifier	0..1	Facultatif	CharacterString	Identifiant de l'abonnement, si réussi (UUID)

5.7.10.3 Conception REST

5.7.10.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.10.3.2 Opération – POST /subscription

Cette opération reçoit une demande d'abonnement. Si le demandeur est autorisé, un abonnement est créé et le dernier message est envoyé.

Le Tableau 53 décrit les paramètres logiques fournis dans l'interface.

Tableau 53 – Mise en œuvre REST de Subscription

Opération REST			
POST URL/v1/subscription {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
SubscriptionObject	application/json	1	Demande d'abonnement incluant des paramètres de filtrage
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
SubscriptionResponseObject	application/json	1	Confirmation ou message d'erreur Identifiant de l'abonnement en retour, si ok.

5.7.10.3.3 Réponse du service

Le Tableau 54 décrit la réponse du service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.16.

Tableau 54 – Codes et messages de réponse HTTP de Subscription

Code HTTP	Message
200	Abonnement créé avec succès
400	Demande incorrecte
403	Non autorisé à accéder aux informations demandées

5.7.10.4 Comportement dynamique

La Figure 28, la Figure 29 et la Figure 30 décrivent le comportement dynamique de l'interface de service.

L'interface Subscription est utilisée pour demander un abonnement à des objets d'information. L'abonnement peut être à la fois demandé par l'utilisateur (Figure 30) et désigné par le producteur (Figure 29). De même, la suppression d'un abonnement peut être initiée par l'utilisateur comme par le producteur. La création et la suppression de l'abonnement demandé ou désigné sont notifiées à l'aide de l'interface Subscription Notification représentée à la Figure 29 et à la Figure 30.

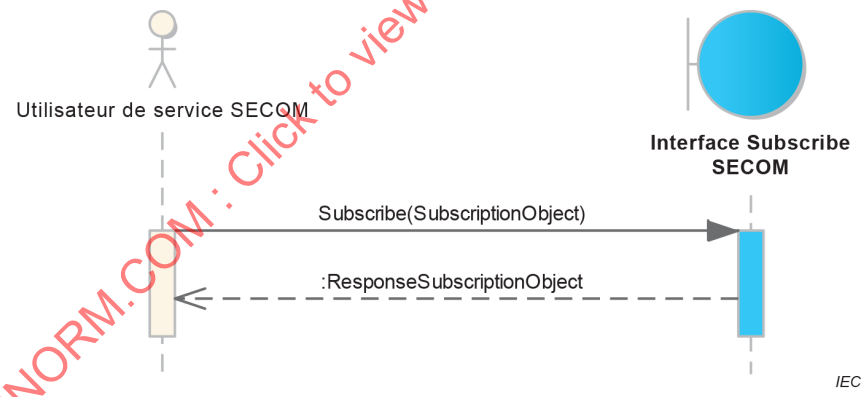


Figure 28––Diagramme de séquence de l'interface Subscribe

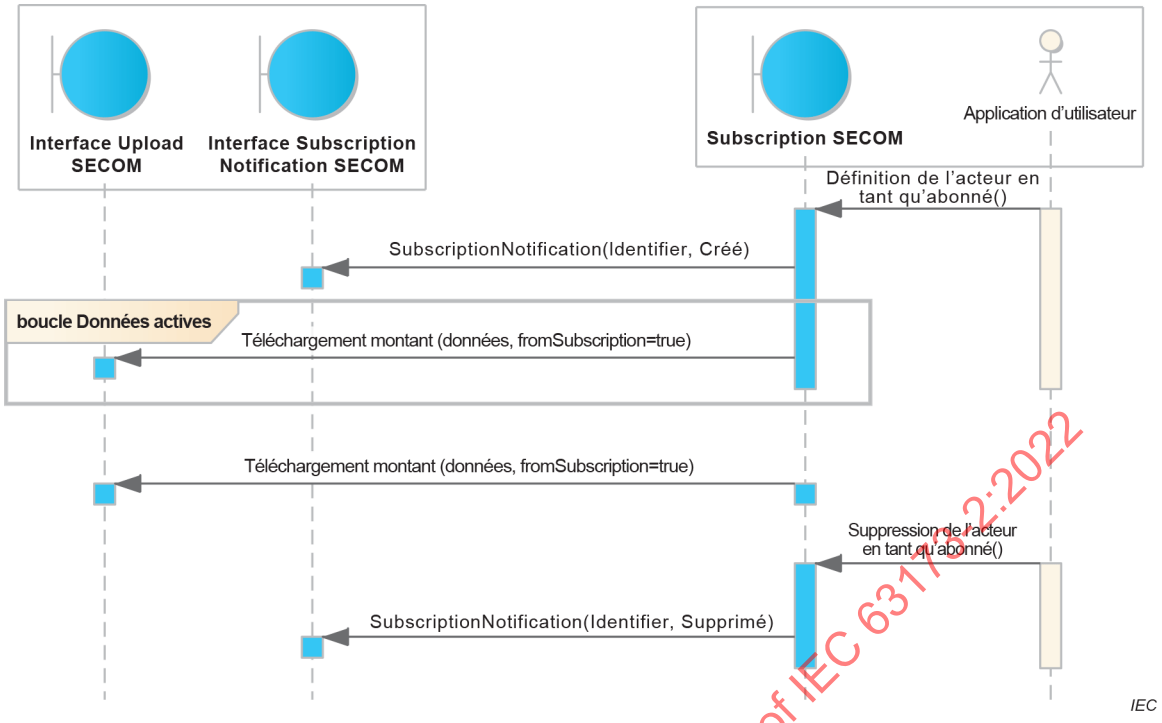


Figure 29 – Diagramme de séquence opérationnelle des interfaces Subscription

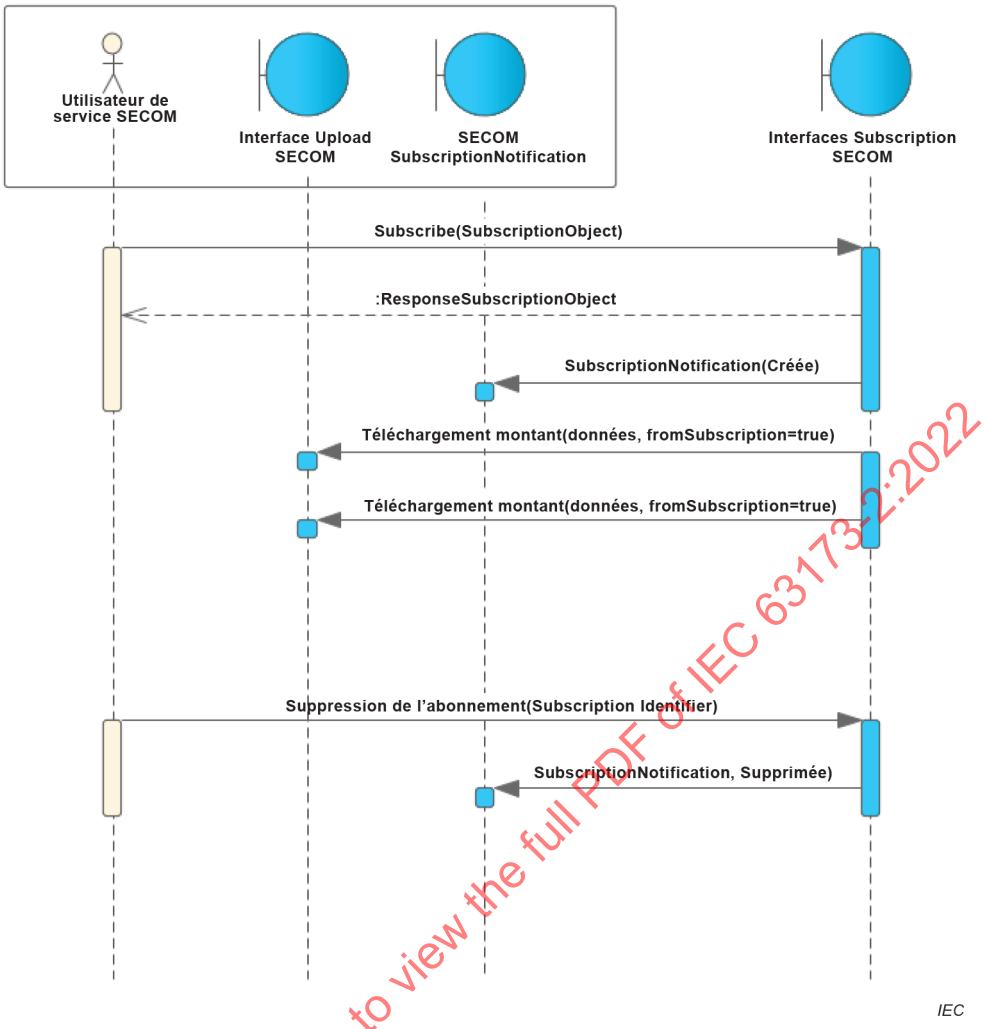


Figure 30 – Diagramme de séquence des interfaces Subscription avec demande d'abonnement externe

5.7.11 Interface de service – Remove Subscription

5.7.11.1 Spécification

Le ou les abonnements peuvent être supprimés soit en interne par le propriétaire des informations, soit en externe par l'utilisateur. Cette interface doit être utilisée par l'utilisateur pour demander la suppression de l'abonnement.

La Figure 31 décrit l'interface Remove Subscription en UML.

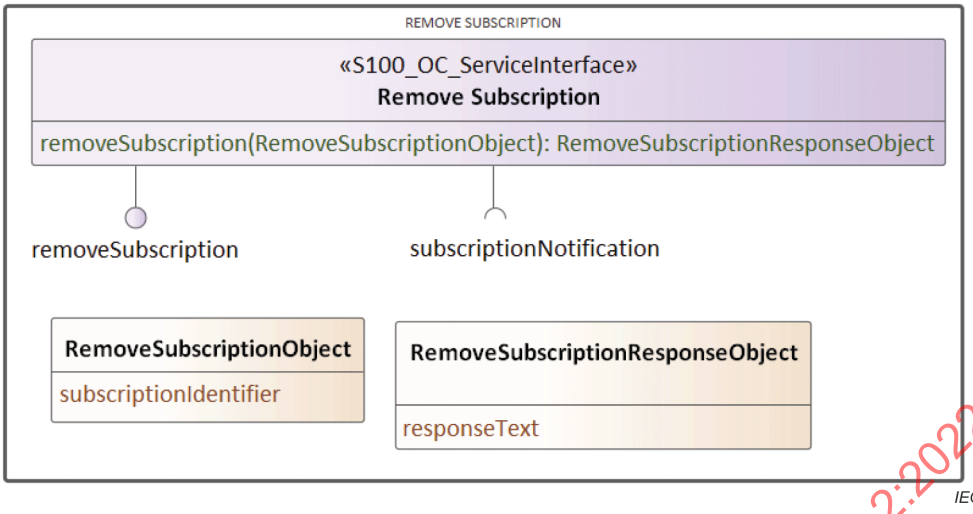


Figure 31 – Diagramme UML de l'interface Remove Subscription

5.7.11.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 55 et le Tableau 56.

Tableau 55 – Entrée d'informations pour l'interface Remove Subscription

RemoveSubscriptionObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
subscriptionIdentifier	1	Obligatoire	UUID	Identifiant de l'abonnement à supprimer. Reçu en demande d'abonnement ou en notification d'abonnement.

Tableau 56 – Sortie d'informations pour l'interface Remove Subscription

RemoveSubscriptionResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
message	1	Facultatif	CharacterString	Message de réponse

5.7.11.3 Conception REST

5.7.11.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.11.3.2 Opération – DELETE /subscription

Cette opération reçoit une demande de suppression d'abonnement.

Le Tableau 57 décrit les paramètres logiques fournis dans l'interface.

Tableau 57 – Mise en œuvre REST de Remove Subscription

Opération REST			
DELETE URL/v1/subscription {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Description**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Description**
RemoveSubscriptionObject	application/json	1	Identité spécifique de l'objet d'information pour lequel il faut supprimer l'abonnement. Si aucune entité d'identification n'est fournie, tous les abonnements de l'appelant sont supprimés.
Retour (sortie)	**Codage REST**	**Mult.**	**Description**
RemoveSubscriptionResponseObject	application/json	1	Confirmation ou message d'erreur

5.7.11.3.3 Réponse du service

Le Tableau 58 décrit la réponse du service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.16.

Tableau 58 – Codes et messages de réponse HTTP de Remove Subscription

Code HTTP	Message
200	Abonnement <identifiant> supprimé
400	Demande incorrecte
403	Non autorisé à supprimer l'abonnement
404	Identifiant de l'abonné introuvable

5.7.11.4 Comportement dynamique

La Figure 32 décrit le comportement dynamique de l'interface de service.

L'interface Remove Subscription permet de supprimer un ou plusieurs abonnements existants. Voir aussi 5.7.10.4 pour une description opérationnelle.

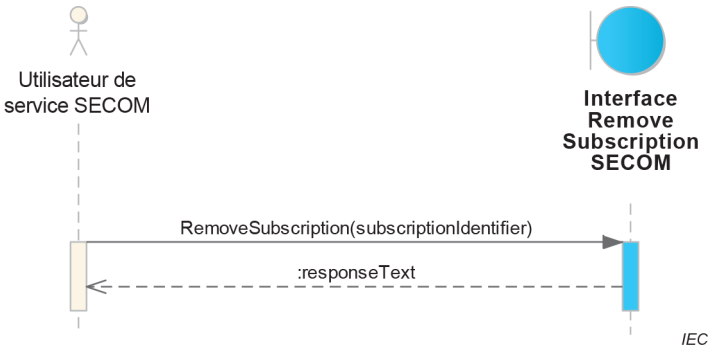


Figure 32 – Diagramme de séquence de l'interface Remove Subscription

5.7.12 Interface de service – Subscription Notification

5.7.12.1 Spécification

L'interface reçoit des notifications lorsqu'un abonnement est créé ou supprimé par le fournisseur d'informations.

La Figure 33 décrit la Subscription Notification dans le diagramme UML.

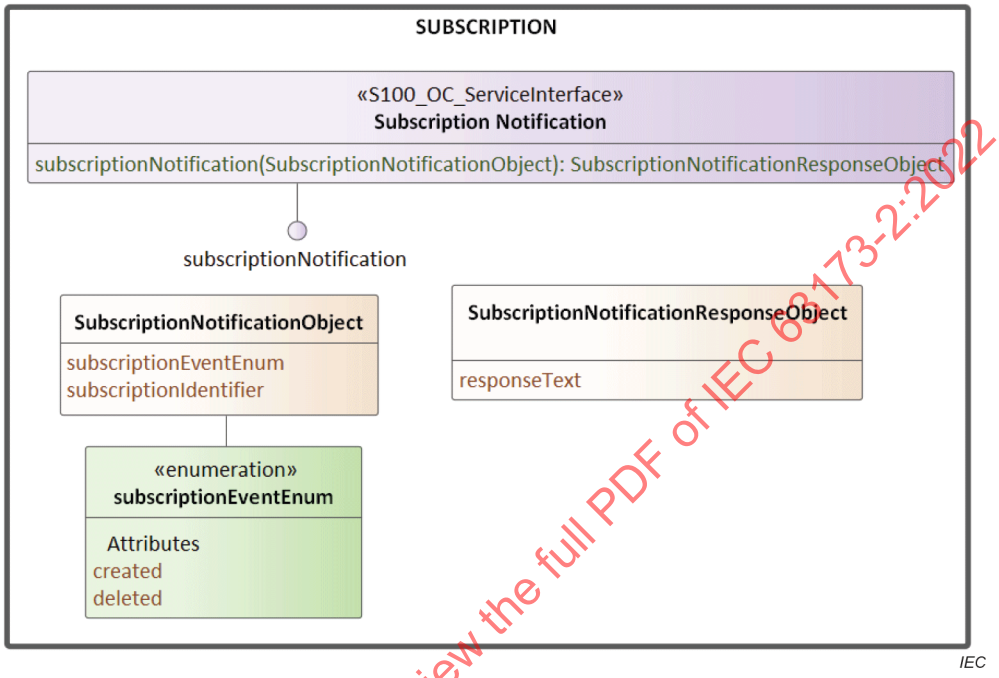


Figure 33 – Diagramme UML de l'interface Subscription Notification

5.7.12.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 59, le Tableau 60 et le Tableau 61.

Tableau 59 – Entrée d'informations pour l'interface Subscription Notification

SubscriptionNotificationObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
subscriptionIdentifier	1	Obligatoire	UUID	Identifiant de l'abonnement
eventEnum	1	Obligatoire	SubscriptionEventEnum	Enumération des événements d'abonnement selon le Tableau 61

Tableau 60 – Sortie d'informations pour l'interface Subscription Notification

SubscriptionNotificationResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
message	1	Facultatif	CharacterString	Message de réponse de succès ou d'erreur

Tableau 61 – Enumérations pour l'interface Subscription Notification

SubscriptionEventEnum	
Valeur	**Définition**
1	Abonnement créé
2	Abonnement supprimé

5.7.12.3 Conception REST

5.7.12.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.12.3.2 Opération – POST /subscription/notification

Cette opération reçoit une notification lorsque l'abonnement est créé, soit en interne par le fournisseur d'informations, soit en externe à la demande de l'utilisateur.

Le Tableau 62 décrit les paramètres logiques fournis dans l'interface.

Tableau 62 – Echange d'informations pour Subscription Notification

Opération REST			
POST URL/v1/subscription/notification {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Description**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Description**
SubscriptionNotificationObject	application/json	1	Contient l'identité de l'objet d'information concerné et le type d'événement: Créer ou Supprimer.
Retour (sortie)	**Codage REST**	**Mult.**	**Description**
SubscriptionNotificationResponseObject	application/json	1	Confirmation ou message d'erreur

5.7.12.3.3 Réponse du service

Le service doit répondre avec des messages et des codes HTTP conformes au Tableau 63. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.16.

Tableau 63 – Codes de réponse HTTP pour Subscription Notification

Code HTTP	Message
200	OK
400	Demande incorrecte

5.7.12.4 Comportement dynamique

La Figure 34 décrit le comportement dynamique de l'interface de service.

L'interface Subscription Notification est utilisée pour obtenir des notifications sur les abonnements créés et supprimés. Voir aussi 5.7.10.4 pour une description opérationnelle.

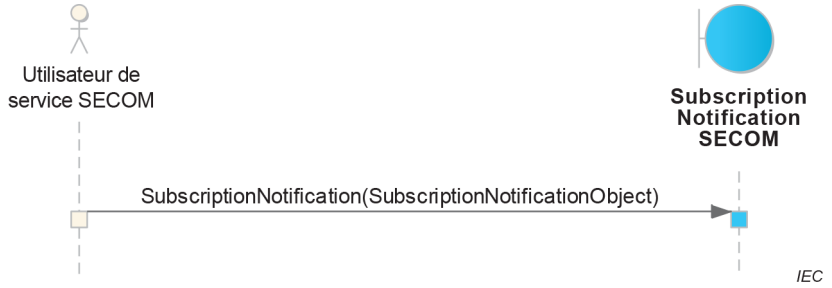


Figure 34 – Diagramme de séquence de l'interface Subscription Notification

5.7.13 Interface de service – Capability

5.7.13.1 Spécification

Le rôle de cette interface est de fournir une méthode dynamique pour demander à une instance de service au moment de l'exécution quelles interfaces sont accessibles pour les données utiles, formats et versions valides respectifs. Si une interface n'est pas mise en œuvre, il convient de renvoyer la réponse HTTP 501.

Le Tableau 64 représente un exemple de capacité pour un service d'information SECOM spécifique. Cette instance est capable de recevoir des types de produits S-124 encapsulés dans un S100_DataSet et un RTZ, de publier et d'envoyer un RTZ en réponse à des demandes Get, d'autoriser l'accès et de permettre l'abonnement au RTZ et au S-124, de prendre en charge le RTZ chiffré et le S-124 ouvert.

Tableau 64 – Exemple de capacité

containerType	S100_DataSet	AUCUNE
dataProductType	S124	RTZ
productSchemaUrl	http://iho.int/s124.xsd	http://cirm.org/rtz/index.html
upload	vrai	vrai
uploadLink	vrai	vrai
get	faux	vrai
getByLink	faux	vrai
getSummary	faux	vrai
subscription	vrai	vrai
access	vrai	vrai
encryptionKey	faux	vrai

La mise en œuvre d'une instance SECOM doit refléter ses capacités en validant toujours le type de produit dans le Tableau 8 des demandes entrantes. Si le type de conteneur est un seul S100_DataSet ou un S100_ExchangeSet, le contenu des données ou le type de produit est vérifié et les actions appropriées sont entreprises.

La Figure 35 décrit l'interface Capability en UML.

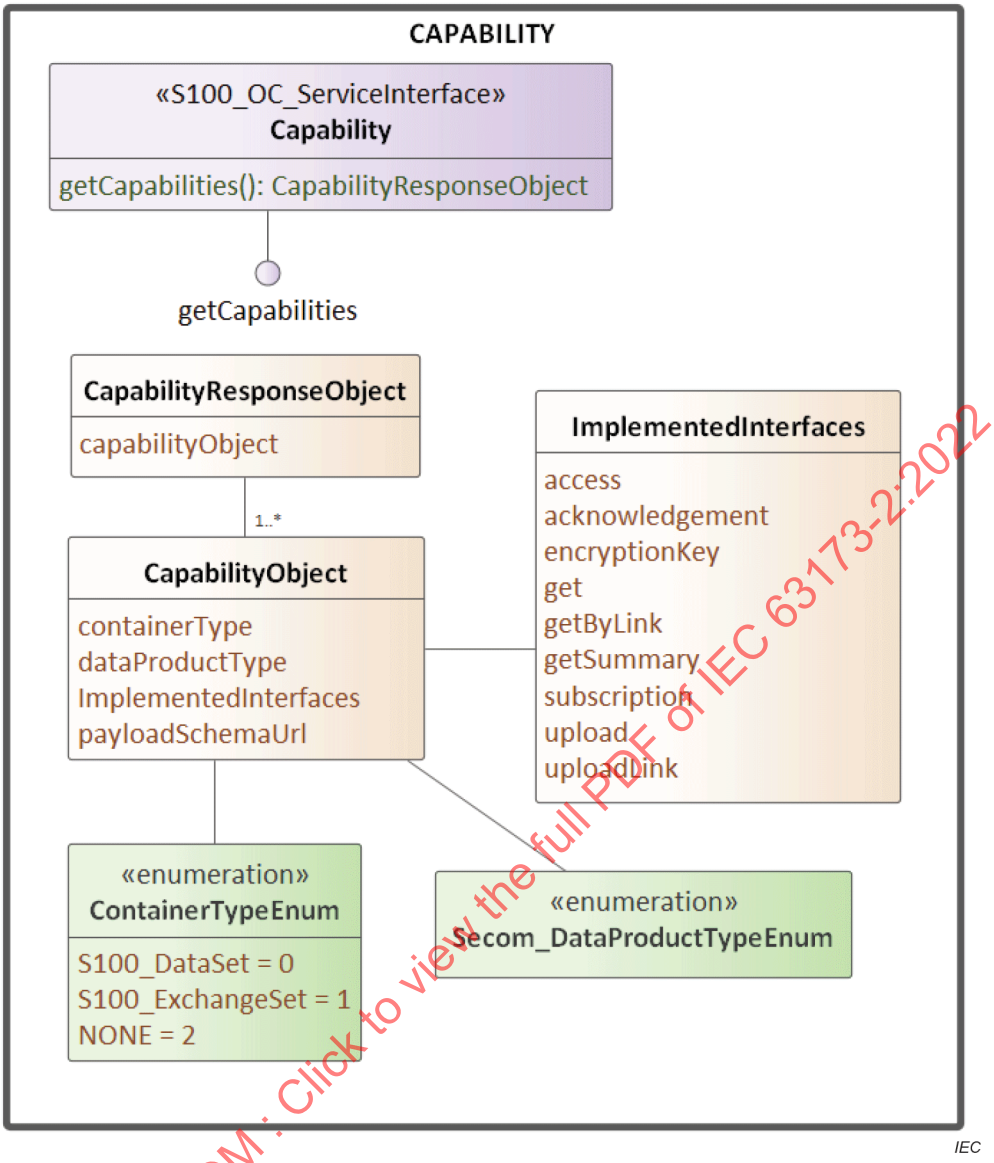


Figure 35 – Diagramme UML de l'interface Capability

5.7.13.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 65.

Tableau 65 – Sortie d'informations pour l'interface Capability

CapabilityResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
capability	1 ..*	Facultatif	CapabilityObject	Liste des objets Capability
CapabilityObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
containerType	1	Facultatif	ContainerTypeEnum	Type de conteneur de l'attribut de données, voir Tableau 7
dataProductType	1	Facultatif	SECOM_DataProductType.Name	Colonne Nom du SECOM_DataProductType dans le Tableau 8
productSchemaUrl	1	Facultatif	URL	L'attribut schemaUrl du produit S-100
implementedInterfaces	1	Facultatif	ImplementedInterfaces	Liste des interfaces de service mises en œuvre avec les données utiles (produit) concernées
serviceVersion	1	Facultatif	CharacterString	Version de l'instance de service, par exemple les nouvelles éditions désignées par n.0.0. Révisions désignées par n.n.0 Clarifications désignées par n.n.n
ImplementedInterfaces				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
upload	1	Facultatif	Boolean	Vrai si l'interface de service est mise en œuvre
uploadLink	1	Facultatif	Boolean	Vrai si l'interface de service est mise en œuvre
get	1	Facultatif	Boolean	Vrai si l'interface de service est mise en œuvre
getByLink	1	Facultatif	Boolean	Vrai si l'interface de service est mise en œuvre
getSummary	1	Facultatif	Boolean	Vrai si l'interface de service est mise en œuvre
subscription	1	Facultatif	Boolean	Vrai si l'interface de service est mise en œuvre
access	1	Facultatif	Boolean	Vrai si l'interface de service est mise en œuvre
encryptionKey	1	Facultatif	Boolean	Vrai si l'interface de service est mise en œuvre

5.7.13.3 Conception REST

5.7.13.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.13.3.2 Opération – GET /capability

Cette opération reçoit une demande de capacités d'interface de service. Le résultat contient les interfaces fournies par l'instance de service, et le type de produit d'information traité.

Le Tableau 66 décrit les paramètres logiques fournis dans l'interface.

Tableau 66 – Mise en œuvre REST de Capability

Opération REST			
GET URL/v1/capability {body}: return			
Paramètre REST (entrée)	**Type**	**Mult.**	**Définition**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Type**	**Mult.**	**Définition**
Pas de corps défini	s.o.	s.o.	s.o.
Retour (sortie)	**Type**	**Mult.**	**Définition**
CapabilityResponseObject	application/json	1..*	Description des capacités du service: quelles interfaces sont valables pour l'instance de service spécifique, le format et la version acceptés des données utiles, les exigences spécifiques des données utiles, etc.

5.7.13.3.3 Réponse du service

Le Tableau 67 décrit la réponse du service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.17.

Tableau 67 – Codes et messages de réponse HTTP de Capability

Code HTTP	Message
200	OK

5.7.13.4 Comportement dynamique

La Figure 36 décrit le comportement dynamique de l'interface de service.

L'interface Capability est utilisée pour récupérer des informations sur les interfaces et les données utiles prises en charge par une instance SECOM.

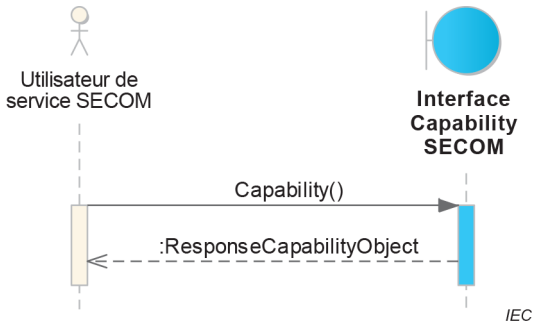


Figure 36 – Diagramme de séquence de l'interface Capability

5.7.14 Interface de service – Ping

5.7.14.1 Spécification

Le rôle de cette interface est de fournir une méthode dynamique pour demander le statut technique de l'instance de service spécifique.

La Figure 37 décrit l'interface Ping en UML.

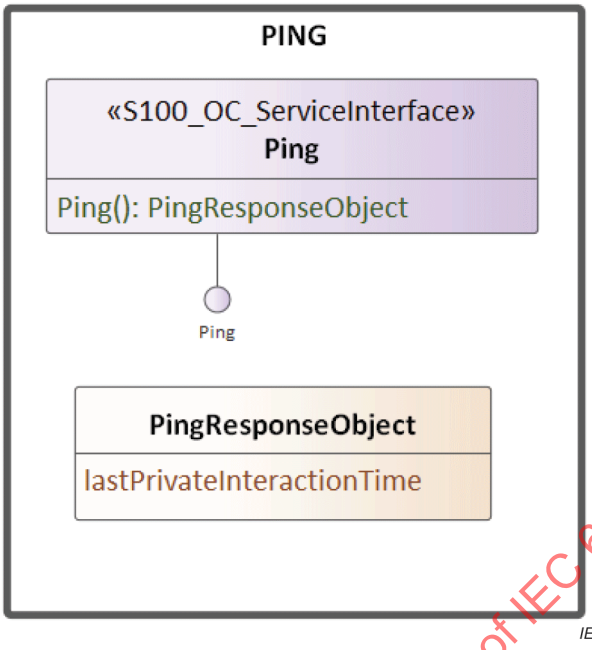


Figure 37 – Diagramme UML de l'interface Ping

5.7.14.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 68.

Tableau 68 – Sortie d'informations pour l'interface Ping

PingResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
lastPrivateInteractionTime	0..1	Facultatif	DateTime	Décrit l'heure de la dernière interaction avec l'application d'utilisateur

5.7.14.3 Conception REST

5.7.14.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.14.3.2 Opération – GET /ping

Cette opération reçoit une demande d'état sur l'instance de service.

Le Tableau 69 décrit les paramètres logiques fournis dans l'interface.

Tableau 69 – Mise en œuvre REST de Ping

Opération REST			
GET URL/v1/ping {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Description**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Description**
Pas de corps défini	s.o.	s.o.	s.o.
Retour (sortie)	**Codage REST**	**Mult.**	**Description**
PingResponseObject	application/json	1	Etat de l'instance de service.

5.7.14.3.3 Réponse du service

Le Tableau 70 décrit la réponse du service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.18.

Tableau 70 – Codes de réponse HTTP Ping

Code HTTP	Message
200	ResponsePingObject

5.7.14.4 Comportement dynamique

La Figure 38 décrit le comportement dynamique de l'interface de service Ping.

L'interface Ping est utilisée pour récupérer des informations sur l'état actuel (version et heure de la dernière interaction) d'une instance de service.

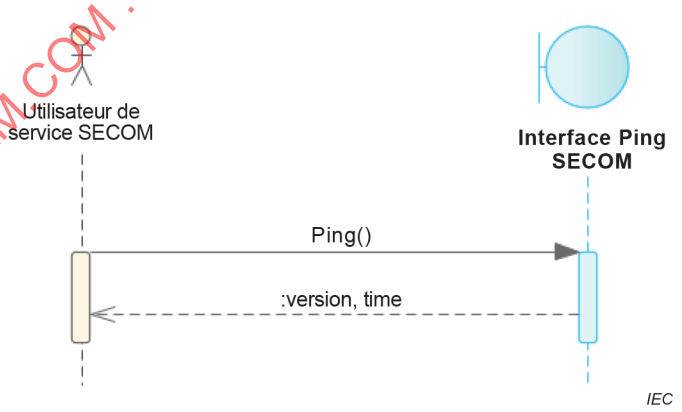


Figure 38 – Vérification de l'état du service

5.7.15 Interface de service – EncryptionKey

5.7.15.1 Spécification

Le rôle de l'interface est d'échanger une clé secrète temporaire.

La Figure 39 décrit l'interface Encryption Key en UML.

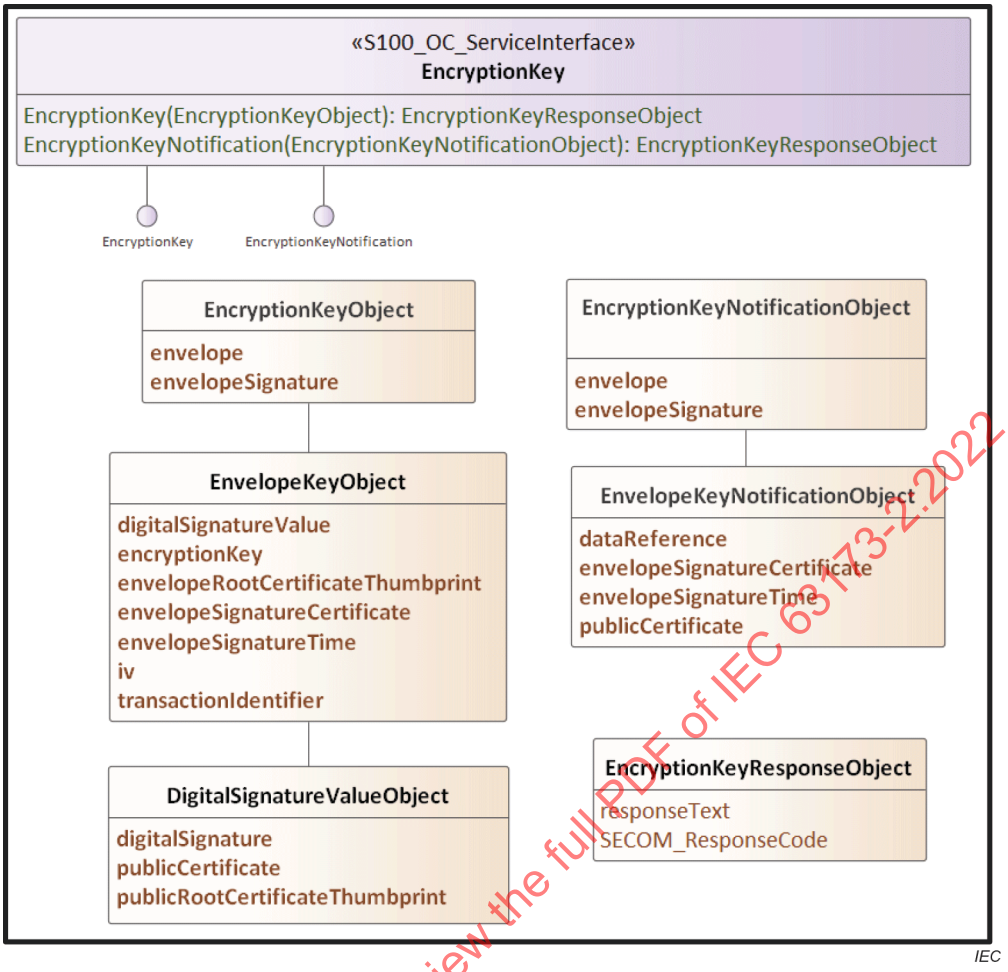


Figure 39 – Diagramme UML de l'interface Encryption Key

5.7.15.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 71, le Tableau 72 et le Tableau 73.

Tableau 71 – Entrée d'informations pour l'interface Encryption Key

EncryptionKeyObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
envelope	1	Obligatoire	EnvelopeKeyObject	EnvelopeKeyObject complet en cours de chargement vers le destinataire, y compris les données et les propriétés du message.
envelopeSignature	1	Obligatoire	CharacterString	Signature d'EnvelopeKeyObject au format HEX, sans espace ni saut de ligne.
EnvelopeKeyObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
encryptionKey	1	Obligatoire	Base64	Clé de chiffrement symétrique protégée
iv	1	Obligatoire	Base64	Vecteur d'initialisation
transactionIdentifier	1	Obligatoire	UUID	Identifiant de la transaction avec les données chiffrées
digitalSignatureValue	1	Obligatoire	DigitalSignatureValueObject	Voir Tableau 5
envelopeSignatureCertificate	1	Obligatoire	CharacterString	Certificat public de l'expéditeur, utilisé pour vérifier la signature d'envelopeLinkObject.
envelopeRootCertificateThumbprint	1	Facultatif	CharacterString	Empreinte numérique déclarée pour clé racine signée (certificat X.509)
envelopeSignatureTime	1	Facultatif	DateTime	Heure de la signature de l'enveloppe encryptionKey

Tableau 72 – Entrée d'informations pour l'interface Encryption Key Notification

EncryptionKeyNotificationObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
envelope	1	Obligatoire	EnvelopeKeyObject	EnvelopeKeyNotificationObject complet en cours de chargement vers le destinataire, y compris les données et les propriétés du message.
envelopeSignature	1	Obligatoire	CharacterString	Signature d'EnvelopeKeyNotificationObject au format HEX, sans espace ni saut de ligne.
EnvelopeKeyNotificationObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
dataReference	1	Obligatoire	UUID	Identifiant d'information récupéré en utilisant la référence de la réponse dans la demande Get Summary (UUID)
publicCertificate	1	Obligatoire	CharacterString	Certificat public de l'utilisateur des données pour la dérivation de la clé symétrique partagée.
envelopeSignatureCertificate	1	Facultatif	CharacterString	Certificat public de l'expéditeur, utilisé pour vérifier la clé de signature d'EnvelopeKeyNotificationObject (certificat X.509)
envelopeSignatureTime	1	Facultatif	DateTime	Heure de la signature de l'enveloppe

Tableau 73 – Sortie d'informations pour l'interface Encryption Key

EncryptionKeyResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
SECOM_ResponseCode	0..1	Obligatoire	SECOM_ResponseCodeEnum	Code d'erreur supplémentaire pour les erreurs internes
message	1	Facultatif	CharacterString	Message de réponse de succès ou d'erreur

5.7.15.3 Conception REST

5.7.15.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.15.3.2 Opération – POST /encryptionkey

Cette opération est utilisée pour charger (push) une clé secrète chiffrée vers un utilisateur.

Le Tableau 74 décrit les paramètres logiques fournis dans l'interface.

Tableau 74 – Mise en œuvre REST du téléchargement montant EncryptionKey

Opération REST			
POST URL/v1/encryptionkey {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Description**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Description**
EncryptionKeyObject	application/json	1	Clé de chiffrement protégée avec ses métadonnées
Retour (sortie)	**Codage REST**	**Mult.**	**Description**
EncryptionKeyResponseObject	application/json	1	Réponse du service

5.7.15.3.3 Réponse du service

Le Tableau 75 décrit la réponse du service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.6.

Tableau 75 – Codes de réponse HTTP du téléchargement montant EncryptionKey

Code HTTP	Message
200	EncryptionKeyResponseObject
400	Demande incorrecte

5.7.15.3.4 Opération – POST /encryptionkey/notify

Cette opération permet à un utilisateur de demander une clé secrète chiffrée à un producteur en fournissant une référence aux données chiffrées et un certificat public de dérivation de clé symétrique utilisé pour protéger la clé de chiffrement temporaire pendant le transfert. La

réponse est envoyée de manière asynchrone en utilisant l'opération de clé de chiffrement de l'utilisateur décrite en 5.7.15.3.2. Le Tableau 76 décrit la mise en œuvre REST.

Tableau 76 – Mise en œuvre REST de EncryptionKey notification

Opération REST			
POST URL/v1/encryptionkey/notify {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
EncryptionKeyRequestObject	application/json	1	Objet de demande de clé de chiffrement.
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
EncryptionKeyResponseObject	application/json	1	Code HTTP avec message

5.7.15.3.5 Réponse du service

Le Tableau 77 décrit la réponse du service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.6.

Tableau 77 – Codes de réponse HTTP de EncryptionKey notification

Code HTTP	Message
202	Notification reçue
403	Non autorisé

5.7.15.4 Comportement dynamique

La Figure 40 et la Figure 41 décrivent le comportement dynamique de l'interface de service EncryptionKey.

Le propriétaire des informations décide de la protéger et génère une clé aléatoire et un vecteur d'initialisation qui sont utilisés pour chiffrer les données. Pour que l'utilisateur des informations puisse déchiffrer les données, la clé aléatoire est protégée par une clé symétrique dérivée utilisant le certificat public de l'utilisateur des informations, ainsi que la clé privée du propriétaire des informations. La clé aléatoire chiffrée est signée et la clé aléatoire, le vecteur d'initialisation, la signature et la clé publique du propriétaire des informations sont placés dans l'objet de données utiles et chargés dans l'interface EncryptionKey de l'utilisateur des informations. Voir Figure 40.

Le propriétaire des informations décide de publier les informations protégées et stocke en interne la clé aléatoire et le vecteur d'initialisation générés. L'utilisateur du service récupère la clé de chiffrement protégée en notifiant le propriétaire des informations par le biais de son interface de notification SECOM EncryptionKey. Le propriétaire des informations protège la clé de chiffrement avec une clé symétrique dérivée, obtenue en utilisant la clé publique du corps de la demande de notification, ainsi que sa propre clé privée. La clé de chiffrement est chiffrée avec la clé dérivée et la clé de chiffrement chiffrée est signée et envoyée à l'interface SECOM EncryptionKey de l'utilisateur du service. Enfin, l'utilisateur obtient la clé de chiffrement de son instance SECOM et peut déchiffrer les informations protégées récupérées. Voir Figure 41.

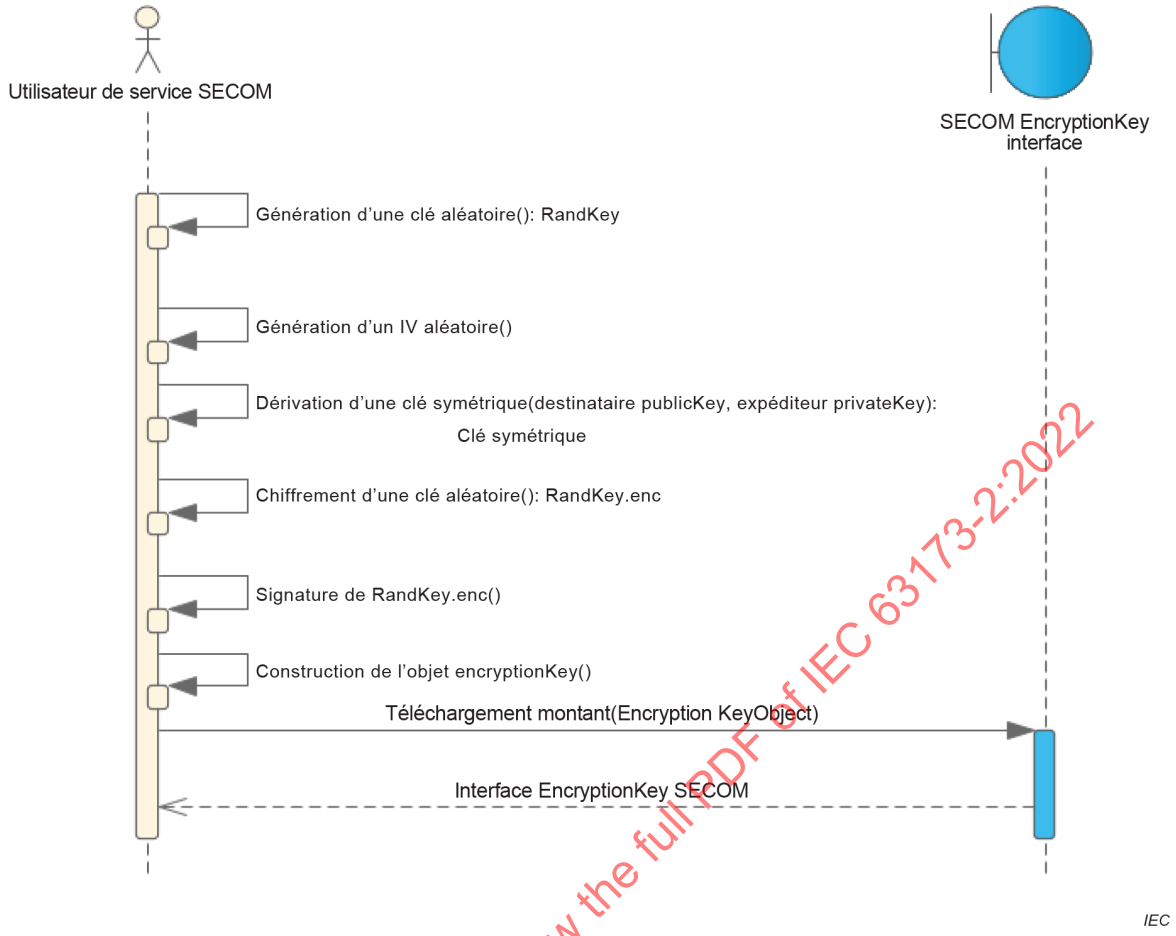


Figure 40 – Diagramme de séquence opérationnelle de l'interface de téléchargement montant EncryptionKey

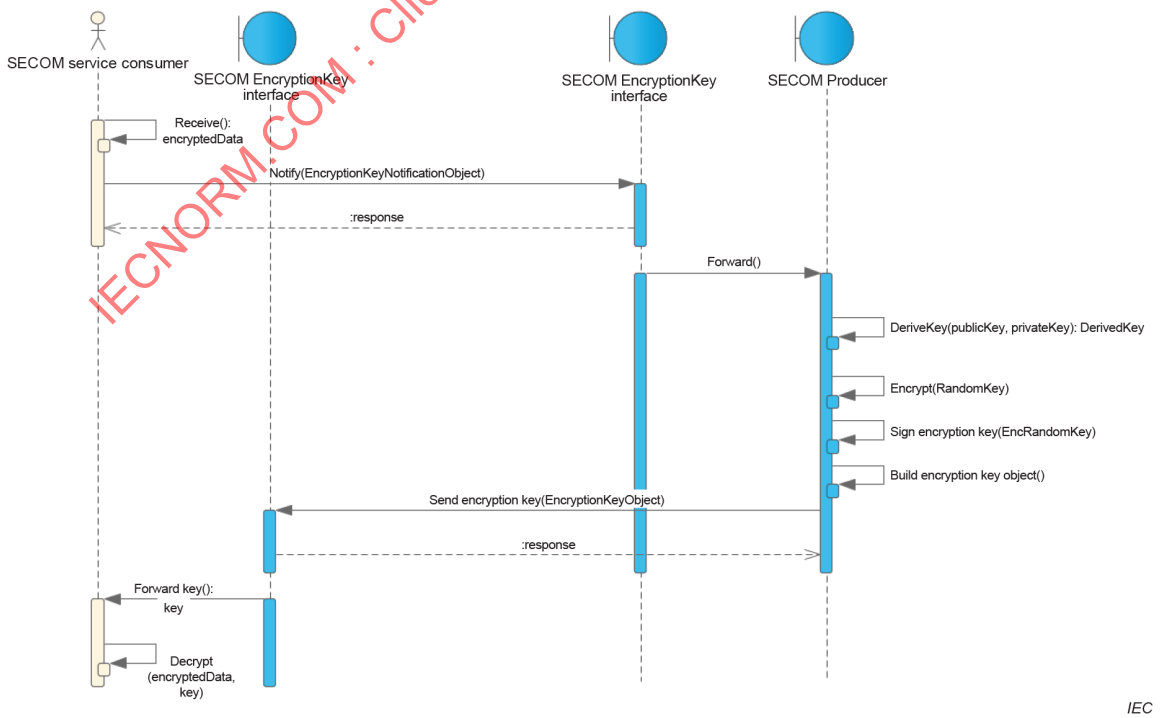


Figure 41 – Diagramme de séquence opérationnelle de l'interface EncryptionKey notification

5.7.16 Interface de service – PublicKey

5.7.16.1 Spécification

Le rôle de cette interface est de demander (pull) une clé publique et d'envoyer (push) une clé publique.

Le rôle de l'attribut booléen "dataProtection" défini sur vrai est de récupérer le certificat public du destinataire des données qui effectue le déchiffrement. Le certificat renvoyé est ensuite utilisé pour dériver une clé symétrique Diffie-Hellman afin de protéger le transfert d'une clé de chiffrement. La valeur faux de "dataProtection" indique une demande de certificat public utilisé pour la vérification de la signature numérique.

Le rôle de l'attribut "certificateThumbprint" est de récupérer le certificat public lié à cet identifiant.

La Figure 42 représente le diagramme UML PublicKey.

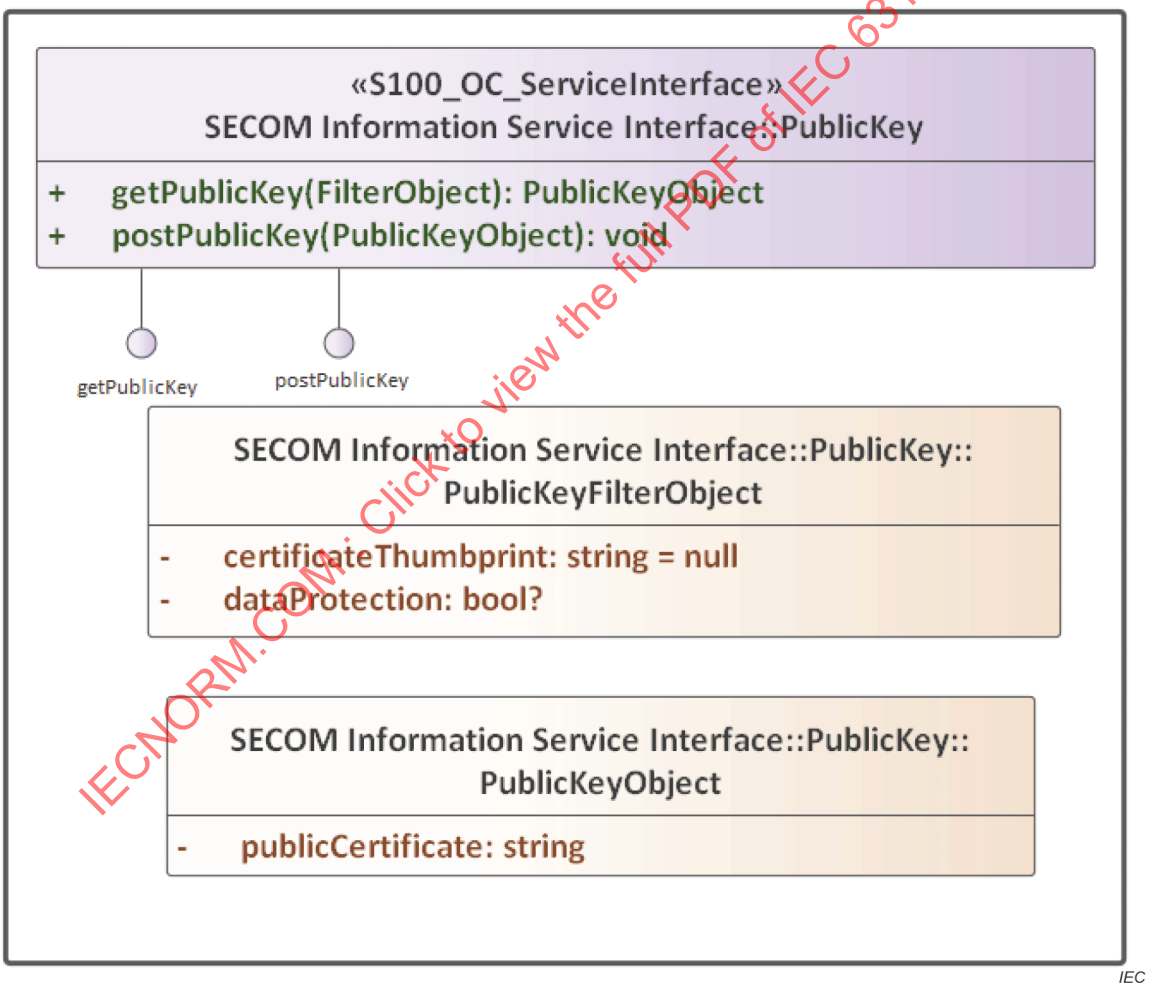


Figure 42 – Diagramme UML de l'interface PublicKey

5.7.16.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 78 et le Tableau 79.

Tableau 78 – Entrée d'informations pour l'interface PublicKey

PublicKeyFilterObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
dataProtection	0..1	Obligatoire	Boolean	Fanion indiquant le certificat public nécessaire pour l'échange de clés de chiffrement.
certificateThumbprint	0..1	Obligatoire	CharacterString	Empreinte numérique déclarée pour clé signée (certificat X.509)

Tableau 79 – Sortie d'information pour l'interface PublicKey GET et entrée d'informations pour l'interface PublicKey POST

PublicKeyObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
publicCertificate	1	Obligatoire	Base64	Certificat public X.509 au format PEM, tableau d'octets codés en Base64.

5.7.16.3 Conception REST

5.7.16.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

5.7.16.3.2 Opération – GET /publicKey

Cette opération reçoit une demande Get de clé publique. Si elle est autorisée, la clé est renvoyée dans la réponse. Il incombe au fournisseur de services d'appliquer la procédure d'autorisation et le contrôle d'accès pertinents aux informations.

Le Tableau 80 décrit les paramètres logiques fournis dans cette opération.

Tableau 80 – Mise en œuvre REST de PublicKey (GET)

Opération REST			
GET URL/v1/publicKey?parameters: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
dataProtection	Boolean	0..1	Fanion indiquant que la clé demandée est destinée à la dérivation d'une clé symétrique en échange d'une clé de chiffrement aléatoire.
certificateThumbprint	String	0..1	Empreinte numérique déclarée pour clé signée (certificat X.509)
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Pas de corps défini	s.o.	s.o.	s.o.
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
PublicKeyObject	application/json	1	Certificat public X.509 au format PEM, tableau d'octets codés en Base64.

5.7.16.3.3 Réponse du service

Le Tableau 81 décrit la réponse de l'instance de service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.7.

Tableau 81 – Codes et messages de réponse HTTP de PublicKey (GET)

Code HTTP	Message
200	PublicKeyObject
400	Demande incorrecte
403	Non autorisé à accéder aux informations demandées
404	Introuvable

5.7.16.3.4 Opération – POST /publicKey

Cette opération charge (push) une clé publique.

Le Tableau 82 décrit les paramètres logiques fournis dans cette opération.

Tableau 82 – Mise en œuvre REST de PublicKey (POST)

Opération REST			
POST URL/v1/publickey {body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Description**
Aucun paramètre défini	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage REST**	**Mult.**	**Description**
PublicKeyObject	application/json	1	Certificat public X.509 au format PEM, tableau d'octets codés en Base64.
Retour (sortie)	**Codage REST**	**Mult.**	**Description**
ResponseObject	application/json	1	Réponse du service

5.7.16.3.5 Réponse du service

Le Tableau 83 décrit la réponse de l'instance de service. Voir aussi le Tableau 11 pour les codes communs à toutes les interfaces.

Pour les essais liés à ces codes d'erreur, voir 10.7.

Tableau 83 – Codes et messages de réponse HTTP de PublicKey (POST)

Code HTTP	Message
200	OK
400	Demande incorrecte

5.7.16.4 Comportement dynamique

La Figure 43 décrit le comportement dynamique de l'interface de service PublicKey. Un utilisateur peut demander un certificat X.509 public par des combinaisons de filtres du Tableau 78 ou charger (push) un certificat X.509 public.

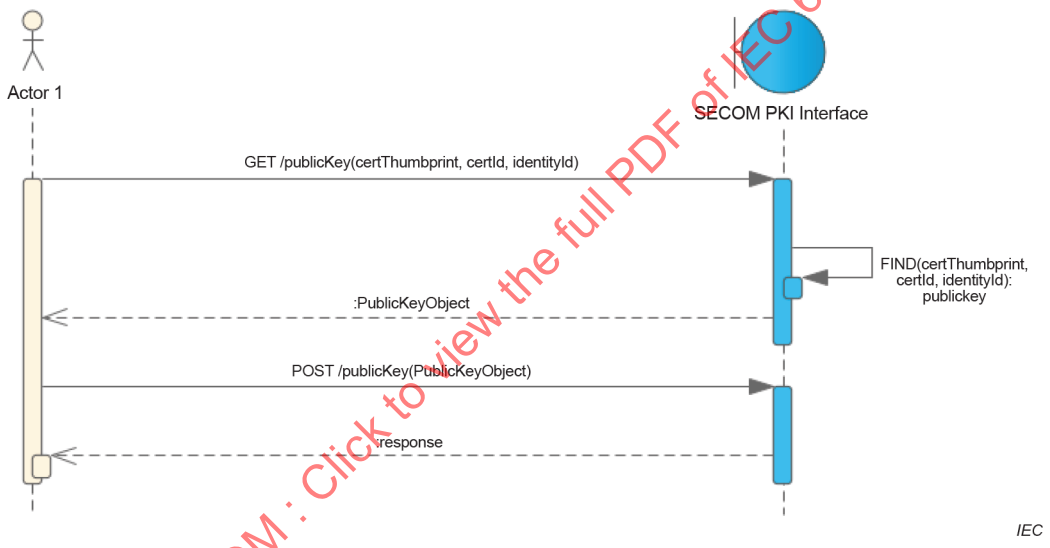


Figure 43 – Diagramme de séquence opérationnelle de l'interface PublicKey

6 Sécurité des canaux de communication SECOM

6.1 Généralités

La raison de la définition de la sécurité des canaux de communication est de protéger le transfert des données et de faciliter l'authentification sur le transfert lors de l'utilisation des services d'information SECOM. Il est nécessaire que les deux parties en communication suivent le même schéma de communication pour atteindre l'interopérabilité lors de l'échange d'informations avec des services web.

L'interface du service d'information SECOM est axée sur l'échange de données, et ces données doivent toujours être signées. L'interface de service d'information SECOM contient également des interfaces de service de prise en charge où aucune donnée opérationnelle n'est échangée, mais où une authentification est tout de même nécessaire pour vérifier qui a réellement fait la demande, appelée ici authentification de service. Lorsqu'un utilisateur demande par exemple à s'abonner à des informations ou à y accéder, il est essentiel de l'authentifier avant d'exécuter la demande. A cette fin, un certificat client émis par la PKI SECOM (voir Article 8) doit être échangé pour permettre au fournisseur de l'instance de service d'authentifier l'utilisateur de l'instance de service.

Dans ce contexte, le fournisseur d'instance de service est l'acteur qui expose une instance de l'interface du service d'information SECOM, c'est-à-dire qu'il agit comme un serveur qui attend une demande. L'utilisateur de l'instance de service est une autre instance de service ou application qui fait une demande à une instance de service d'information SECOM, c'est-à-dire qu'il agit en tant que client et initie la demande de service. Lorsqu'un navire envoie (télécharge vers l'amont) des informations, telles qu'un plan de route S-421 (IEC 63173-1), à un VTS, ce dernier fait office de fournisseur d'instance de service (serveur) et le navire fait office d'utilisateur d'instance de service (client). Inversement, lorsque le VTS envoie (charge) des informations, telles qu'un changement de route recommandé (S-421), le navire agit en tant que fournisseur d'instance de service (serveur) et le VTS agit en tant qu'utilisateur d'instance de service (client). Ainsi, lors de la conception de l'interface du service d'information SECOM, il est nécessaire que la plupart des acteurs aient la capacité d'agir à la fois comme fournisseur d'instance de service (serveur) et comme utilisateur d'instance de service (client).

L'Article 6 décrit la technologie et les procédures de protection des canaux de communication IP et HTTP pour les services web, tels que les instances de service basées sur les services d'information SECOM.

6.2 Transfert sécurisé

6.2.1 Canaux de communication sécurisés

Le chiffrement du canal par l'utilisation de la sécurité de la couche de transport, TLS version 1.2 (RFC 5246) et TLS version 1.3 (RFC 8446) doit être pris en charge.

HTTP sur TLS doit être utilisé conformément à la RFC 2818.

HTTP/1.1 doit être utilisé conformément à la RFC 7231.

Le présent document décrit les communications interentreprises avec authentification mutuelle, où le client et le serveur s'authentifient mutuellement à l'aide d'une authentification mutuelle TLS basée sur un certificat (authentification TLS côté client X.509). Voir Figure 44.

L'authentification du service doit reposer sur des certificats X.509 (RFC 5280 et RFC 2459).

Les certificats doivent pouvoir être révoqués, c'est pourquoi la procédure d'authentification doit contenir une vérification par rapport à la liste de révocation de certificats (CRL) ou à l'OCSP.

6.2.2 Procédure d'authentification

La procédure d'authentification dans une demande de service est définie à la Figure 44 et décrit comment il doit être vérifié que le certificat du client est émis par une PKI SECOM. Le certificat du serveur (hôte) doit être vérifié comme étant émis par une autorité de certification de confiance, qui ne doit pas nécessairement être la PKI SECOM.

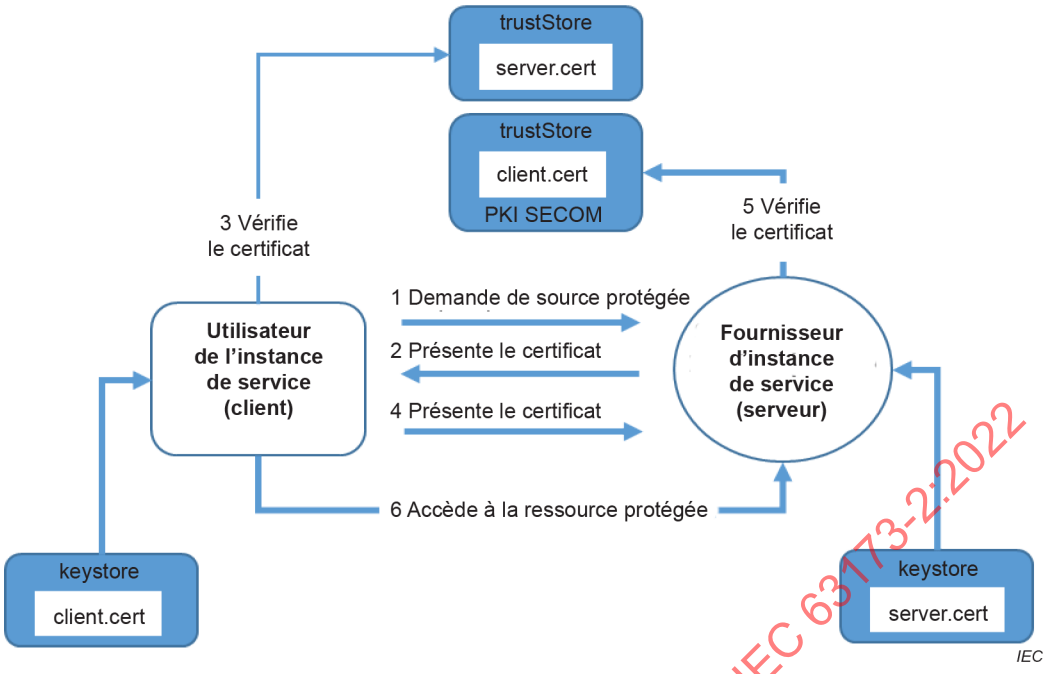


Figure 44 – Principe d'authentification des services

La procédure de vérification du certificat du serveur doit être la suivante:

1) le client doit vérifier que le certificat du serveur est valide;
2) le client doit vérifier que le certificat du serveur est émis par une AC de confiance, normalement émise par l'une des autorités de certification de confiance présentes sur le marché, mais qui peut également être émise par la PKI SECOM;
3) le client doit vérifier que le nom de domaine du serveur correspond au nom de domaine figurant dans le certificat.

La procédure de vérification du certificat du client doit être la suivante:

4) le serveur doit vérifier que le certificat du client est valide;
5) le serveur doit vérifier que le certificat du client est émis par la PKI SECOM.

7 Protection des données SECOM

7.1 Généralités

La protection des données SECOM comprend une signature numérique des données échangées qui facilite l'authentification des données de bout en bout. La signature numérique comprend une somme de contrôle et une identité revendiquée. La somme de contrôle facilite le contrôle de l'intégrité des données. La somme de contrôle doit être chiffrée avec l'identité revendiquée (clé privée de l'expéditeur), ce qui facilite l'authentification des données par le destinataire à l'aide de la clé publique de l'expéditeur.

La protection des données SECOM décrit également le chiffrement facultatif des données.

Pour des informations concernant les clés privées et publiques, voir Article 8.

7.2 Compression et empaquetage des données

L'utilisation de la compression peut être appliquée aux données. Les spécifications des produits individuels basés sur la S-100 de l'OHI fournissent les détails de l'utilisation de la compression.

Si la compression des données classifiées est appliquée, les données doivent être compressées avant d'être chiffrées.

La compression doit utiliser l'algorithme ZIP et la méthode DEFLATE. Les fonctions de chiffrement et de signature numérique de ZIP ne sont pas utilisées.

La compression et l'empaquetage des données SECOM sont alignés sur la compression et le conditionnement des données dans la S-100 de l'OHI. Voir la S-100:2018 de l'OHI, Partie 15-5, Compression et empaquetage des données.

NOTE L'édition 4.0.0 de la S-100:2018 de l'OHI constitue la dernière version publiée au moment de la rédaction du présent document.

7.3 Authentification et signature des données

7.3.1 Généralités

SECOM est basé sur la même technologie de signature des données que celle décrite dans la S-100:2018 de l'OHI, Partie 15-8, Authentification des données, et utilise un algorithme normal et un mécanisme d'échange de clés largement disponibles et utilisés.

La signature numérique utilise des algorithmes de clés publiques asymétriques dans le schéma PKI SECOM pour lier de manière irréfutable les données à l'identité de l'émetteur.

Le schéma repose sur le chiffrement asymétrique d'une somme de contrôle des données. En vérifiant la signature par rapport à la clé publique de l'émetteur et en vérifiant également la clé publique de l'émetteur par rapport à une identité de premier niveau, l'utilisateur est assuré de l'identité du signataire. Une description technique détaillée des signatures numériques sort du domaine d'application du présent document et le lecteur est invité à consulter la norme de signature numérique du NIST (DSS-FIPS Publication 186-3) pour une explication plus détaillée.

La différence entre le schéma d'authentification des données SECOM et le schéma d'authentification des données décrit dans la S-100:2018 de l'OHI, Partie 15-8 réside dans le fait que le SECOM est conçu pour l'échange d'informations duplex, donc un acteur tel qu'un navire est à la fois expéditeur et destinataire d'informations, à la fois serveur et client dans le schéma. Cela signifie que le nombre de serveurs de données augmente considérablement par rapport à l'hypothèse de la S-100 de l'OHI, qui est conçue principalement pour un serveur à terre fournissant des informations aux navires (clients). Cela signifie également que chaque client nécessite un accès à beaucoup plus de clés publiques pour l'authentification des données. Mais le principe reste le même.

Les données utiles et la signature à des fins d'authentification et d'intégrité sont toujours échangées, ce qui est également décrit dans l'interface du service d'information SECOM.

7.3.2 Formats de données et normes pour les signatures numériques, les clés et les certificats

Cette description est alignée techniquement sur la description correspondante de la S-100:2018 de l'OHI (éd 4.0.0, Partie 15-8.3).

Les catégories de contenu suivantes sont exigées pour l'authentification des données:

1) paires de clés, clés privées et publiques (voir Article 8);
2) demandes de signature de certificats et clés publiques signées numériquement. Lorsqu'une clé publique est elle-même signée numériquement, elle est appelée "certificat". Lorsque la clé publique est signée par sa clé privée correspondante, elle est appelée certificat "autosigné". Elles sont établies sous forme d'enregistrements X.509 (RFC 5280 et RFC 2459) et peuvent être codées au format DER (ISO/IEC 9594-8) ou au format PEM (RFC 2045) pour être envoyées à la PKI SECOM pour signature. Lorsqu'elles sont intégrées dans des fichiers XML, les clés doivent être codées au format PEM afin que le texte en clair puisse être inséré comme un élément XML;
3) le format numérique des clés publiques signées ("certificats") (voir Article 8).

Le format PEM définit un codage textuel des grands nombres multiples exigés par l'algorithme de signature numérique (DSA), ainsi que les paramètres DSA exigés par l'algorithme DSA. Le codage au format PEM permet d'intégrer des clés publiques dans des objets JSON et XML.

L'Article 8 offre davantage d'informations sur la création de clés et la demande de signature.

Les signatures numériques SECOM (et S-100) sont des mises en œuvre de la norme de signature numérique (DSS). Le DSS utilise l'algorithme de hachage sécurisé (SHA-256, voir ISO/IEC 10118-1) pour créer une empreinte de hachage (hachage) de 256 bits du contenu du fichier. L'empreinte de hachage est ensuite introduite dans le DSA pour générer la signature numérique du message à l'aide d'un algorithme de chiffrement asymétrique et de la "clé privée" de la paire de clés du signataire.

Dans l'algorithme DSA, une signature est une séquence de deux entiers. Par convention, ils sont nommés R et S (une "paire R,S"). Le format des signatures numériques lorsqu'elles sont intégrées dans des fichiers XML est le suivant:

```
<digitalSignature>

302C021433796C6647CC1C55A67DC72FA7C6E157A6594B2B02145D3768B44F3A6ABA1

1A77178B738AD3B6A0DE344

</digitalSignature>
```

La paire R,S est représentée par un codage hexadécimal (chiffres de 0 à 9, lettres de A à F). Le codage des deux grands entiers R,S est une séquence 2 de la notation syntaxique abstraite un (ASN.1). Ils sont produits de façon native par la mise en œuvre d'openssl et peuvent être générés et vérifiés sans avoir besoin de décompresser individuellement les entiers R et S. Ce codage permet d'encapsuler les deux entiers en une seule chaîne codée en caractères hexadécimaux. Par exemple, le schéma ASN.1 de l'exemple ci-dessus est le suivant:

```
SEQUENCE (2 elem)

  INTEGER (158 bit) 357903468461694385184092679318935791752802642315
  INTEGER (158 bit) 351274375691773793245737972991313380578601275707
```

7.3.3 Création d'une signature numérique

Cette description est alignée techniquement sur la description correspondante de la S-100:2018 de l'OHI (éd 4.0.0, Partie 15-8.6).

Comme indiqué en 4.1, le domaine d'application de la protection des données SECOM concerne les utilisateurs finals. Le transfert de données pouvant s'effectuer sur des canaux de communication non sécurisés, par exemple le "dernier kilomètre", il est nécessaire que les données originales soient signées par le propriétaire des informations. La signature est créée par:

1) le propriétaire des informations crée une signature numérique pour les données à l'aide de l'algorithme DSA et de sa clé privée, comme décrit en 7.3.2;
2) la paire R,S constituant la signature numérique est stockée sous la forme d'une séquence ASN.1 de 2 éléments dans le champ digitalSignature d'un DigitalSignatureValueObject.

Dans le json résultant, la signature se ressemble à la ligne ci-dessous:

"digitalSignature":"302C021433796C6647CC1C55A67DC72FA7C6E157A6594B2B02145D3768B44F3A6ABA11A77178B738AD3B6A0DE344"

La signature doit toujours être échangée conjointement au contenu des données.

7.3.4 Création d'une signature d'enveloppe

Pour les interfaces conçues pour envoyer (push) des données, c'est-à-dire avec POST par la méthode REST, une signature par couche supplémentaire est ajoutée, contenant non seulement les données, mais aussi les métadonnées, voir Tableau 86 pour les interfaces concernées. Le principal motif de l'ajout d'une autre couche de signature est d'assurer l'intégrité des données de bout en bout pour l'ensemble des données utiles où les données et les métadonnées sont regroupées dans un objet enveloppe. Pour assurer l'interopérabilité entre différents environnements, langages de programmation et systèmes d'exploitation de serveur, la signature est traitée à l'aide d'une procédure de sérialisation personnalisée décrite dans le présent document.

L'idée de base est d'analyser les attributs d'une enveloppe et de concaténer leurs valeurs dans une chaîne de valeurs séparées par des virgules (CSV). Un point a été choisi comme séparateur. L'utilisation d'un point de séparation est essentiellement motivée par le besoin d'indiquer le début et la fin des propriétés dans la longue chaîne combinée. La taille de la signature finale n'est pas affectée par ces caractères ajoutés.

L'ordre de chaque attribut ajouté (voir Tableau 16, Tableau 20, Tableau 24, Tableau 71 et Tableau 72) est obtenu en ajoutant les noms des attributs dans l'ordre où ils apparaissent dans chaque table de données utiles. Pour les objets enveloppe comprenant d'autres objets comme attributs, le même ordre est appliqué pour chaque sous-structure d'objet. L'étape suivante consiste à traverser l'objet ordonné et ses sous-objets et à ajouter les valeurs des attributs en suivant l'ordre. La chaîne CSV contient maintenant toutes les données nécessaires à la signature.

Il est nécessaire que la conversion de chaque type de données d'attribut en sa représentation sous forme de chaîne soit indépendante de l'environnement et de l'emplacement. Par conséquent, un ensemble de règles de conversion des chaînes de caractères est défini et nécessite d'être exécuté pour que la création et la vérification de la signature fonctionnent pour l'acteur expéditeur et l'acteur destinataire. Dans le Tableau 84, les différentes règles pour chaque conversion de type de données en chaîne sont énumérées et le Tableau 85 indique les interfaces auxquelles la conversion s'applique.

Tableau 84 – Règles de conversion

Datatype	Règle	Exemple
DateTime	1. Convertit l'objet datetime en un horodateur entier Unix en secondes 2 Analyse un nombre entier en tant que chaîne de caractères	Entrée: 2020-07-01T15:00:00 Sortie: "1593608400"
bool	Convertit la valeur bool en une chaîne de caractères minuscules	Entrée: Vrai Sortie: "vrai"
Guid	Convertit l'objet guid en une chaîne de caractères minuscules	Input:[0f8fad5b-d9cb-469f-a165-70867728950e] Sortie: "0f8fad5b-d9cb-469f-a165-70867728950e"
enum	1. Convertit enum en sa représentation entière 2 Analyse un nombre entier en tant que chaîne de caractères	Entrée: Troisième: 3 Sortie: "3"
byte[]	Convertit un tableau d'octets en une chaîne en Base64	Entrée: {10, 20, 30, 40, 0, 0, 0, 0} Sortie: "ChQeKDI8RIA="
Nombre entier	Analyse un nombre entier en tant que chaîne de caractères	Entrée: 17 Sortie: "17"
null	Représenter la valeur nulle comme une chaîne vide	Entrée: null Sortie: ""

EXEMPLE La signature d'une enveloppe d'acceptation s'effectue comme suit:

1) définir la valeur de départ du résultat; CSV = "";

2) ajouter des propriétés selon l'ordre du Tableau 24 pour produire un tableau: [createdAt, envelopeCertificate, transactionIdentifier, ackType, nackType, envelopeSignatureTime];

3) commencer à ajouter les valeurs du tableau traversé:

   a) convertir DateTime createdAt en une chaîne d'horodatage UNIX: CSV += "1593608400";

   b) envelopeCertificate est déjà une chaîne; CSV += "MIIE1zCCBFygAwIBAgIU";

   c) convertir Guid transactionIdentifier en une chaîne comprenant des lettres minuscules: CSV += "10f032cc-2c31-4441-b6e5-bc0511983cd0";

   d) convertir AckEnum ackType en un entier, puis en une chaîne: CSV += "2";

   e) convertir la valeur NackEnum nackType null en une chaîne vide: CSV += "";

   f) convertir DateTime envelopeSignatureTime en une chaîne d'horodatage UNIX: CSV += "1593608405";

4) supprimer le dernier séparateur de point;

5) la chaîne CSV contient alors: "1593608400.MIIE1zCCBFygAwIBAgIU.10f032cc-2c31-4441-b6e5-bc0511983cd0.2,,1593608405";

6) Enfin, convertir la chaîne générée en un tableau d'octets en utilisant le codage UTF8: CSV[] = ConvertToArray(CSV);

7) Utiliser CSV[] comme entrée de l'algorithme DSA.

Tableau 85 – Interfaces avec signature d'enveloppe

Paragraphe	Interface	Type
5.7.2	Interface de service – Upload	EnvelopeUploadObject
5.7.3	Interface de service – Upload Link	EnvelopeLinkObject
5.7.4	Interface de service – Acknowledgement	EnvelopeAckObject
5.7.15	Interface de service – EncryptionKey	EnvelopeKeyObject

7.3.5 Vérification de la signature numérique

Le rôle de la signature numérique est de vérifier l'intégrité des données et de les authentifier. La vérification de la signature numérique est un algorithme qui fonctionne sur trois éléments de données indépendants:

1) un contenu qui exige une validation (le format de ce contenu est arbitraire);
2) une clé publique, correctement codée. Dans l'algorithme DSA adopté, cette clé publique est composée d'un ensemble de paramètres DSA et d'une clé publique;
3) une signature. Dans l'algorithme DSA, une signature est composée de deux nombres, appelés par convention R et S (paire R,S).

Un processus de vérification de la signature permet de déterminer si la paire R,S authentifie le contenu par rapport à la clé publique donnée. Celui-ci ne peut aboutir qu'à un résultat vrai ou faux.

La vérification de la signature numérique DSA aboutit à deux résultats:

- authentification: le système de mise en œuvre compare la clé publique du serveur de données ("contenu") et la signature du certificat du serveur de données ("signature") à la clé publique du fournisseur ("clé publique") pour confirmer que la clé publique du fournisseur dans le certificat est valide et que le serveur de données est un membre de bonne foi du schéma de protection des données désigné;
- contrôle d'intégrité: le système de mise en œuvre compare la signature du fichier de données ("signature") et la clé publique du serveur de données dans le certificat du serveur de données ("clé publique") au fichier de données ("contenu"). Cette opération permet de vérifier le contenu du fichier de données.

Si cette vérification de la validation est concluante, elle prouve que le fichier de données n'a été corrompu d'aucune façon et que l'identité du serveur de données dans les signatures de l'ensemble de données est validée par l'identité de l'administrateur de schéma (SA) telle que définie dans le certificat racine du SA.

La signature est calculée sur les données originales, donc avant de calculer la signature, toutes les compressions, le chiffrement et le codage en Base64 nécessitent d'être inversés dans l'ordre suivant:

1) les données binaires codées en Base64 sont retransformées en octets;
2) si les données binaires sont compressées, il est nécessaire de les reconvertir;
3) si les données sont chiffrées, les données sont déchiffrées à l'aide de la clé appropriée (en supposant que le destinataire ait reçu la clé et ait pu la déchiffrer);
4) si les étapes précédentes sont réussies, il convient que les données soient alors un tableau d'octets et qu'un hachage SHA-256 puisse être calculé;
5) le hachage calculé à l'étape 4 est ensuite utilisé conjointement avec la clé publique dans la digitalSignatureValue comme entrée pour la vérification DSS.

NOTE Si les données du message sont chiffrées, l'instance SECOM n'est pas en mesure de déchiffrer la partie données du message, car cela exige l'accès à la clé privée du destinataire final.

7.3.6 Vérification d'une signature d'enveloppe

A la réception d'un message SECOM signé, l'instance SECOM destinataire peut vérifier que le message reste tel que l'expéditeur l'a envoyé (intégrité des données) en validant la signature d'enveloppe. Cela se fait en recalculant le hachage de la signature à partir des données du message et en le vérifiant par rapport au certificat de l'expéditeur inclus dans le message. Toutefois, cela ne fait que vérifier que le certificat contenu dans le message est celui qui a été utilisé pour le signer. Il est également important de vérifier que le certificat utilisé est émis par une autorité de certification de confiance pour prouver l'identité revendiquée de l'expéditeur.

Le processus de vérification est similaire au processus de signature:

1) toutes les propriétés sont sérialisées et concaténées sous forme de texte en utilisant le même principe que dans le processus de signature de 7.3.4;
2) le texte est converti en un tableau d'octets en utilisant le codage UTF8;
3) un hachage est calculé pour le tableau d'octets créé à l'étape 2), à l'aide de SHA-256;
4) le hachage calculé à l'étape 3 est vérifié par DSA en utilisant la clé publique du certificat de l'expéditeur.

Cette vérification peut être effectuée par toute personne recevant le message, ce qui signifie que si le message est envoyé par l'intermédiaire d'une API du fournisseur, l'instance SECOM destinataire peut effectuer cette vérification avant de transmettre le message à son destinataire final.

7.3.7 Exemple de commandes d'authentification des données

Le Tableau 86 donne des exemples de commandes de création et de vérification de signature.

Tableau 86 – Exemples de commandes

Description	Exemples de commandes

Signature des données	`openssl dgst -sha256 -sign privateKey.pem data.txt > data.sig`
	
	`xxd -u -ps data.sig > data.sig.hex`

Vérification de la signature	`xxd -r -u -ps data.sig.hex > data.sig`
	
	`openssl dgst -sha256 -verify publicKey.pem -keyform pem`
	
	`-signature data.sig data.txt`

7.4 Chiffrement des données

7.4.1 Généralités

L'utilisation du chiffrement des données est facultative et, si elle est disponible, elle est généralement spécifiée dans chaque spécification de produit S. S'il est décrit comme facultatif dans une spécification de produits basés sur la norme S, le serveur de données décide, en tant que propriétaire des informations, si les données doivent être chiffrées.

Le SECOM prend en charge l'utilisation de l'algorithme de la S-100:2018 de l'OHI, Partie 15-6, Chiffrement des données, pour les liaisons terre-navire et navire-terre.

7.4.2 Algorithme de chiffrement

L'algorithme de chiffrement utilisé en SECOM doit être celui décrit dans la S-100:2018 de l'OHI, Partie 15-6, Chiffrement des données; il est résumé ci-dessous.

Lorsque les données sont chiffrées, l'algorithme doit être la norme de chiffrement avancée (AES) avec le mode de fonctionnement enchaînement de blocs de chiffrement (CBC). Il est toujours supposé que la totalité des données (utiles) doit être chiffrée.

L'algorithme de chiffrement par blocs AES est à clé symétrique. Cela signifie que la même clé est utilisée pour le chiffrement et le déchiffrement. L'algorithme définit comment un bloc de texte en clair est converti en un bloc de texte chiffré et vice versa. La taille des blocs de l'AES est toujours de 16 octets (128 bits). La longueur de la clé peut être de 128 bits, 192 bits et 256 bits. Les variantes correspondantes sont appelées AES-128, AES-192 ou AES-256.

L'algorithme AES ne peut chiffrer qu'un seul bloc de texte en clair. Pour les messages plus longs, un mode de chiffrement par blocs doit être utilisé. Ce schéma de protection choisit le mode CBC (enchaînement de blocs de chiffrement) pour le chiffrement de plusieurs blocs de données. Dans ce mode de fonctionnement, il est exigé que la longueur du texte en clair doive être un multiple exact de la taille du bloc. Le remplissage est exigé.

La méthode de remplissage qui doit être utilisée est décrite dans le PKCS #7 (RFC 2315). Il ajoute N octets au message jusqu'à ce que sa longueur soit un multiple de 16 octets. La valeur de chaque octet est N. Noter que si le texte en clair d'origine a déjà une longueur multiple de 16, un bloc complet de 16 octets ayant chacun la valeur de 16 doit être ajouté.

7.5 Création et transfert de clé de chiffrement

7.5.1 Généralités

Le SECOM prend en charge à la fois la gestion des clés de chiffrement telle qu'elle est décrite dans la S-100:2018 de l'OHI, Partie 15-7, Chiffrement des données et licences, et telle qu'elle est décrite ici. Le fournisseur d'informations décide du schéma de clés de chiffrement (schéma de protection des données) à utiliser.

Le schéma décrit dans la S-100:2018 de l'OHI, Partie 15-7, est axé sur les informations transmises de la terre au navire, où le serveur de données a soumis les clés symétriques utilisées pour le chiffrement des données encapsulées dans des permis et liées à un ID matériel pour le client de données.

Si la S-100:2018, Partie 15-7, Chiffrement des données et licences, n'est pas appliquée, un autre schéma de gestion des clés de chiffrement SECOM peut être utilisé. Ce schéma convient lorsqu'un acteur joue à la fois le rôle de serveur de données et de client de données, par exemple lorsqu'un navire partage son plan de route avec plusieurs autres acteurs et que ces derniers peuvent renvoyer au navire des plans de route actualisés ou optimisés. Dans le schéma de gestion des clés de chiffrement de SECOM, la clé symétrique utilisée pour le chiffrement des données est générée en tant que clé temporaire et envoyée au client des données par l'intermédiaire d'une interface de service dédiée et protégée. La clé symétrique doit être utilisée uniquement dans un temps limité (session).

Pour plus de détails sur la gestion des clés de chiffrement conformément à la S-100:2018 de l'OHI, voir Partie 15-7, Chiffrement des données et licences.

7.5.2 Gestion des clés de chiffrement SECOM

La clé secrète doit être une clé symétrique temporaire qui est générée pour l'échange spécifique de données. La clé de chiffrement doit être transférée au client à l'aide de l'interface de service définie en 5.7.15. L'ordre de transfert des données et de la clé de chiffrement doit être indifférent et la clé de chiffrement peut être transférée avant ou après les données chiffrées.

La clé secrète doit être valide et utilisée uniquement pour une durée limitée. La Figure 45 et la Figure 46 représentent l'interaction principale entre le serveur et le client pour l'échange de données classifiées avec le schéma de gestion des clés SECOM. La procédure utilisée est une méthode d'échange de clés Diffie-Hellman. Le certificat public du destinataire est récupéré soit à partir de l'interface SECOM PublicKey du destinataire dans 5.7.16 (Figure 45), soit à partir de l'interface PKI SECOM GetPublicKey dans 8.6.3 (Figure 46).

Le serveur de données est chargé de décider quand créer une nouvelle clé de chiffrement.

Si le serveur de données a décidé de chiffrer ses données et que le client de données répond avec une édition de données mise à jour, le client de données doit également chiffrer les données répondues.

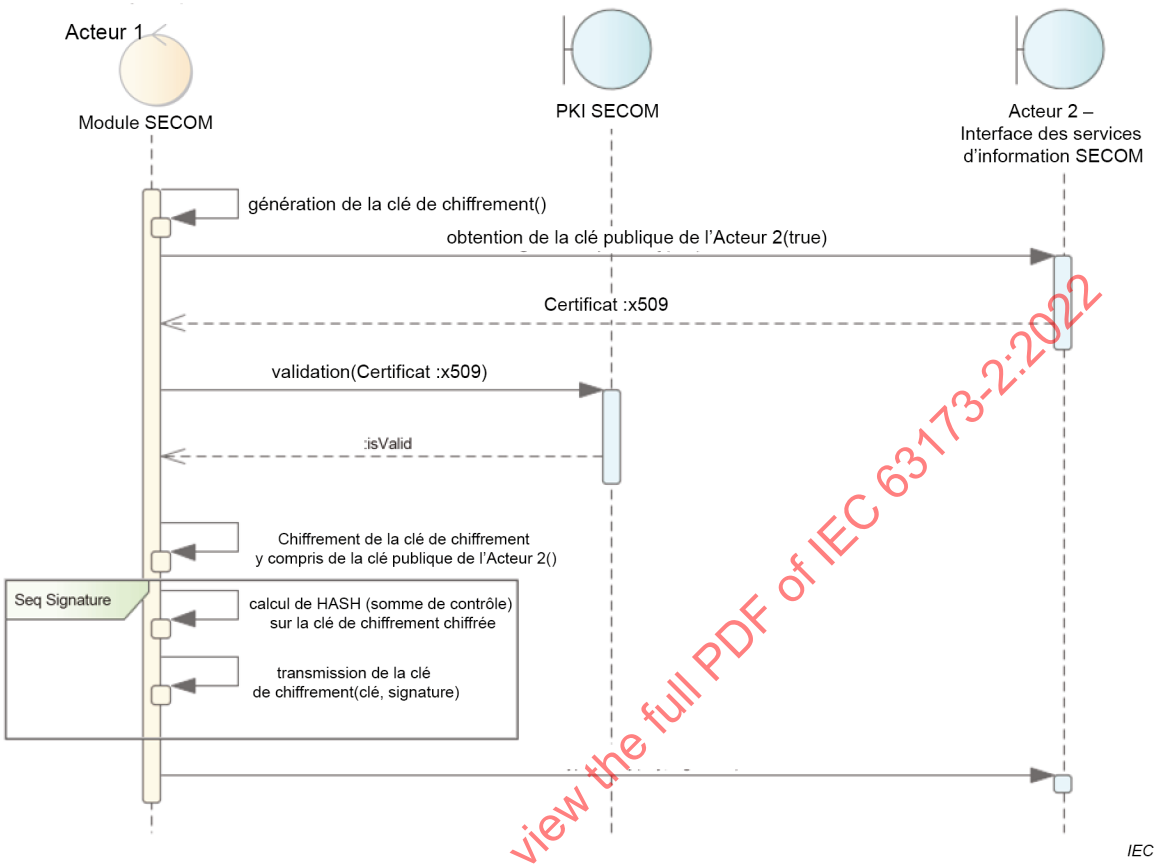


Figure 45 – Séquence de gestion des clés de chiffrement SECOM

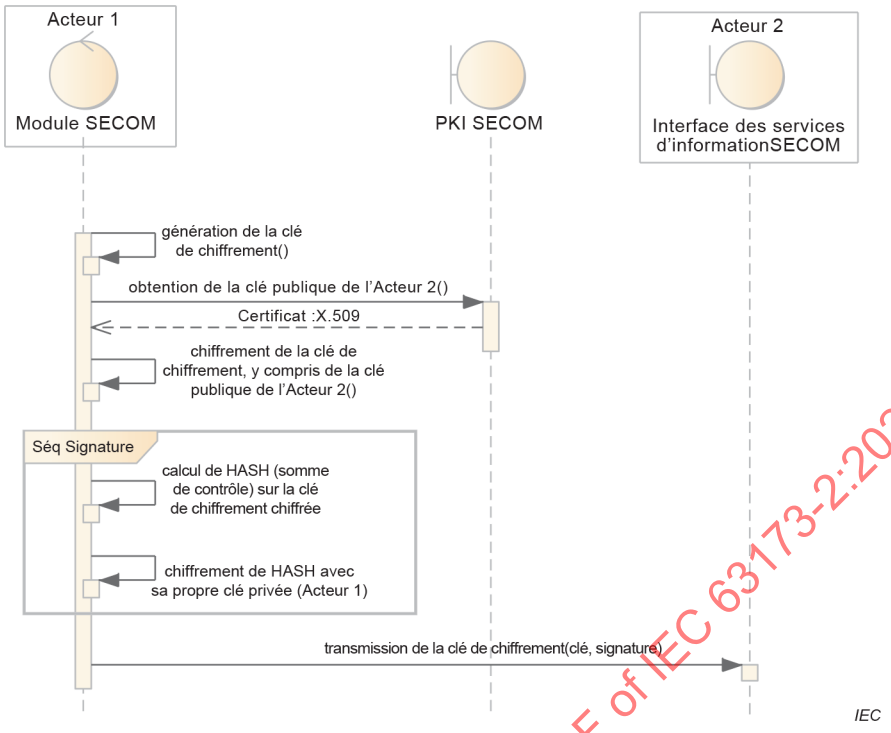


Figure 46 – Autre séquence de gestion des clés de chiffrement SECOM

7.5.3 Génération de la clé de chiffrement.

La clé de chiffrement doit être générée sous la forme d'un nombre aléatoire correspondant à l'algorithme de chiffrement et à la longueur de clé sélectionnés conformément à 7.4.2.

Un vecteur d'initialisation (IV, Initialization Vector) aléatoire doit être ajouté pour être utilisé dans la procédure de chiffrement et de déchiffrement. L'IV doit être généré comme un nombre aléatoire et n'être utilisé qu'une seule fois, c'est-à-dire qu'il est important de ne jamais réutiliser les IV avec la même clé de chiffrement. Pour l'AES, la taille de l'IV est toujours de 16 octets, c'est-à-dire qu'elle est égale à la taille du bloc décrite en 7.4.2. Le motif de l'utilisation d'un IV est de se conformer à la norme de chiffrement avancé (AES) actuelle et donc d'assurer la compatibilité avec les cadres commerciaux existants tels que .NET et Java, où les méthodes sous-jacentes exigent toujours un IV.

Avant le transfert de la clé de chiffrement, celle-ci doit être protégée (chiffrée). La protection de la clé de chiffrement doit comprendre l'utilisation de la clé asymétrique publique du destinataire (provenant de la PKI SECOM.

Etant donné que le destinataire et l'expéditeur utilisent tous deux l'algorithme de cryptographie sur les courbes elliptiques (ECC) dans les clés asymétriques, apparier les clés et chiffrer la clé de chiffrement avec la paire ECC (clé privée propre appariée avec la clé publique du destinataire).

7.5.4 Signature de la clé de chiffrement protégée

Avant le transfert, la clé de chiffrement protégée doit être signée avec la clé asymétrique privée de l'expéditeur (provenant de la PKI SECOM. La procédure de signature doit être la même que celle utilisée pour signer des données, telle que décrite en 7.3 conformément à la norme de signature numérique (DSS).

7.5.5 Transfert de la clé de chiffrement.

La clé chiffrée protégée et la signature doivent ensuite être transférées au destinataire à l'aide de l'interface de service d'information SECOM (5.7.15).

Le destinataire authentifie la signature au moyen de la clé asymétrique publique de l'expéditeur préalablement stockée, récupérée à partir de l'interface PublicKey, voir 5.7.16 ou récupérée à partir de l'interface GetPublicKey de la PKI SECOM, voir 8.6.3. Si l'authentification est concluante, le destinataire doit utiliser sa clé asymétrique privée (de PKI SECOM) avec la clé asymétrique publique de l'expéditeur pour dériver la clé symétrique partagée en utilisant un algorithme Diffie-Hellman (voir 7.5.2) et utiliser cette clé dérivée pour décrypter la clé de chiffrement.

7.5.6 Exemple

Un exemple avec openssl est donné dans le Tableau 87.

Tableau 87 – Exemple de commandes

Dériver la clé partagée à partir de la clé publique du destinataire et de la clé privée de l'expéditeur	`openssl pkeyutl -derive -inkey Actor1PrivateKey.pem -peerkey Actor2PublicKey.pem -out sharedSecret.tmp`
Calculer le hachage de la clé partagée	`openssl dgst -sha256 -out sharedSecret_hashed.tmp sharedSecret.tmp`

Analyser la clé dérivée à valeur brute	`$temp = (Get-Content -Path .\sharedSecret_hashed.tmp -Raw)`
	
	`$sharedKey = $temp.Substring($temp.IndexOf("=") + 2, 64)`

Chiffrer la clé secrète avec la clé partagée dérivée et le vecteur d'initialisation	`openssl enc -aes-256-cbc -nosalt -p -K $sharedKey -iv $iv -e -in encryptionKey.bin -out encryptionKey.enc`

Créer la signature numérique pour la clé secrète chiffrée	`openssl dgst -sha256 -sign actor1PrivateKey.pem encryptionKey.enc > encryptionKey.enc.sign`
	
	`xxd -u -ps encryptionKey.enc.sign > encryptionKey.enc.sign.hex`

La clé secrète signée et chiffrée, ainsi que le vecteur d'initialisation, doivent ensuite être chargés vers l'interface du service d'information SECOM du destinataire.

8 PKI SECOM

8.1 Généralités

L'Article 8 décrit la fonctionnalité prévue de la PKI SECOM et les interfaces publiques exposées aux utilisateurs pour la gestion de la clé.

La principale fonctionnalité consiste à fournir une interface pour accéder au stockage dans un registre des identités pour les clés publiques et les identités à des fins d'authentification. Les clés publiques sont signées et encapsulées dans des certificats X.509 (RFC 5280 et RFC 2459) lors de leur échange, voir 8.4. Il est possible de vérifier les certificats en utilisant une API normale à cet effet, OCSP (protocole de statut de certificat en ligne) et CRL (liste de révocation de certificats), voir 8.5. Il est possible de récupérer les clés publiques d'une certaine identité dans la PKI SECOM, et de récupérer l'identité d'une certaine clé publique, voir 8.6.3.

Une PKI SECOM est régie par un administrateur de schéma (SA). Il peut y avoir plusieurs instances de la PKI SECOM fournies par plusieurs SA, et un acteur peut être enregistré dans plusieurs instances de la PKI SECOM. Le matériel compatible SECOM est supposé prendre en charge plusieurs instances de la PKI SECOM.

La clé privée est générée par l'acteur et n'est jamais échangée sur Internet. La clé publique correspondante est chargée de manière sécurisée vers la PKI SECOM pour signature et stockage, voir 8.4. La clé publique est chargée vers la PKI SECOM par le biais d'une demande de signature de certificat (CSR), comme décrit en 8.2.5 et 8.4. Si la demande de signature est acceptée par la PKI SECOM, elle est signée et le certificat résultant est stocké dans la PKI SECOM et également renvoyé.

Le SECOM ne définit ni ne limite le nombre de registres des identités, il peut donc y avoir plusieurs fournisseurs de registres des identités. Le schéma de protection des données utilisé est décrit dans l'objet de métadonnées d'échange associé.

Le SECOM ne spécifie pas le registre des identités à utiliser. Voir Annexe D.

8.2 Schéma

8.2.1 Généralités

Le schéma est aligné sur la S-100:2018 de l'OHI, Partie 15.8, Authentification des données. La description et la terminologie des participants au schéma sont alignées sur la S-100 de l'OHI.

Il y a plusieurs types d'utilisateurs du schéma:

- les administrateurs de schéma (SA), un seul d'entre eux est dans une instance PKI SECOM;
- les serveurs de données (DS, Data Server), ils sont nombreux;
- les clients de données (DC, Data Client), ils sont nombreux;
- les fabricants d'équipements d'origine (FEO), ils sont nombreux.

Une explication plus détaillée de ces termes est donnée de 8.2.2 à 8.2.4.

8.2.2 Administrateur de schéma

L'administrateur de schéma (SA) est le seul responsable du maintien et de la coordination du schéma.

Le SA est chargé de contrôler l'adhésion au schéma et de s'assurer que tous les participants agissent selon les procédures définies. Le SA maintient le certificat racine numérique de premier niveau utilisé pour faire fonctionner le schéma et est le seul organisme pouvant certifier l'identité des autres participants au schéma.

8.2.3 Serveurs de données

Les serveurs de données (DS) sont responsables du chiffrement des données et des signatures numériques des ensembles de données conformément aux procédures et processus définis dans le schéma.

8.2.4 Clients de données

Les clients de données (DC) sont les utilisateurs finals des ensembles de données et reçoivent des informations protégées des serveurs de données pour accéder et utiliser les ensembles de données et les services. L'application logicielle du client de données (système FEO) est chargée d'authentifier les signatures numériques et de déchiffrer les informations des ensembles de données conformément aux procédures définies dans le schéma.

8.2.5 Procédure

Voici une liste des étapes suivies par chaque participant au schéma avant la signature numérique des fichiers de données:

1) création et configuration des schémas (lors de la création d'une PKI SECOM):
   - l'administrateur de schéma (SA) crée sa propre paire de clés publiques/privées et l'autosigne;
   - le SA met sa clé publique autosignée (appelée aussi "certificat racine") dans le domaine public;
   - la clé publique du SA ("certificat racine" et son empreinte numérique) est intégrée, lorsque cela est exigé, dans les systèmes du FEO ("magasin de confiance");
2) configuration du serveur de données (où n'importe qui peut agir en tant que serveur de données, par exemple navire, terre):
   - le serveur de données crée une paire de clés publique et privée;
   - le serveur de données signe sa clé publique (avec sa clé privée), créant ainsi une clé autosignée (SSK);
   - la SSK du serveur de données est envoyée au SA pour validation lors de la demande de signature des clés par un schéma de protection des données SECOM (parfois appelée "demande de signature de certificat". Le SA peut avoir ses propres exigences administratives ou techniques pour l'acceptation du serveur de données. Ces exigences administratives ou techniques dépassent le domaine d'application du SECOM;
3) vérification de l'identité du serveur de données:
   - si elle est acceptée, le SA vérifie la SSK et l'identité du serveur de données;
   - le SA signe la SSK du serveur de données avec sa propre clé privée pour produire un certificat de serveur de données signé par le SA. Le certificat du serveur de données est stocké et lié à l'identité du serveur de données dans le registre des identités;
   - le certificat du serveur de données est ensuite renvoyé au serveur de données;
   - le serveur de données vérifie que le certificat du serveur de données est authentifié par rapport à la clé publique du SA;
4) le serveur de données peut alors produire des signatures numériques des données.

8.3 Génération des clés publique et privée

La paire de clés asymétriques doit être générée selon l'un des couples algorithmes – longueur de clé suivants: RSA ≥ 2 048, DSA ≥ 1 024, EC ≥ 224 et EdDSA ≥ 256.

Pour la S-100:2018 de l'OHI, la paire de clés asymétriques est générée sous forme de clés à algorithme de signature numérique (DSA) d'une longueur de 1 024 bits.

La paire de clés asymétriques doit être générée par chaque participant au schéma.

Ce sont toutes des clés DSA codées au format PEM, ainsi que leurs paramètres de clé DSA. Voir Tableau 88 pour un exemple de commandes de base. Les commandes réelles nécessaires dépendent de la mise en œuvre du fournisseur de la PKI.

Tableau 88 – Création de paires de clés publiques et privées – exemple de commandes de base

Tâche	Commande
SA-1 crée les paramètres du DSA	`openssl dsaparam 1024 -out dsaparam.txt`
SA-2 crée la clé racine du SA et le certificat racine autosigné	`openssl req -x509 -sha256 -nodes -days 365 -newkey dsa:dsaparam.txt -keyout iho.key -out iho.crt`
SA-3 signe une demande de signature de certificat vérifiée	`openssl x509 -req -in CSR.csr -sha256 -CA iho.crt -CAkey iho.key -CAcreateserial -out signedicds.crt`

8.4 Demande de signature de certificat

La clé publique doit être codée au format PEM X509v3 et encapsulée dans une demande de signature de certificat PKCS #10 (RFC 2986).

La demande de signature de certificat est envoyée de manière sécurisée à PKI SECOM par le biais de l'interface de service définie en 8.6.2.

La clé publique est signée par le certificat de l'AC PKI SECOM.

8.5 Révocation de certificats

8.5.1 Généralités

L'administration des certificats se fait en dehors de l'interface SECOM et c'est le SA qui est responsable des processus de mise à jour, de création, de signature et de révocation.

Il existe deux techniques principales pour vérifier l'état de révocation d'un certificat: CRL et OCSP.

8.5.2 CRL Liste de révocation de certificats

La CRL dans la PKI SECOM doit être basée sur une CRL normale qui est publiée à intervalles définis. Entre les intervalles définis, une CRL delta peut être publiée qui fait toujours référence à la dernière CRL publiée. Il en résulte une gestion des CRL qui réduit le besoin global de transmettre des paquets de données volumineux aux navires en route.

La nécessité pour les utilisateurs finals de s'abonner à la CRL est le compromis avec l'OCSP qu'il convient de comparer à la possibilité de vérifier les informations d'identification localement sans réponse de l'OCSP. Le problème du transfert d'un fichier unique volumineux sur une transmission à faible bande passante pourrait être atténué par l'utilisation d'une CRL delta intermédiaire qui gère la CRL.

8.5.3 OCSP Protocole de statut de certificat en ligne

Le principal avantage de l'OCSP par rapport à la CRL est la possibilité de centraliser la liste de révocation et la validation des certificats en un seul point. Cependant, une fonction centralisée de vérification des certificats augmente le besoin de communication et de transmission. Si une latence est introduite dans la vérification d'un certificat et qu'elle s'ajoute à la latence d'un transfert de données utiles, le délai de livraison pourrait être important.

Le principal avantage de l'utilisation d'OCSP est la centralisation de la révocation et le fait qu'un certificat révoqué n'a pas besoin d'être ajouté et mis à jour dans plusieurs listes CRL locales. Cela réduit également la nécessité pour les systèmes d'extrémité d'extraire, de mettre à jour et de conserver une CRL. La diminution de l'utilisation des points de terminaison est répartie sur plusieurs systèmes: besoin de transmission et appels/demandes d'informations vers des clusters informatiques centralisés.

8.6 Interface des services PKI SECOM

8.6.1 Généralités

L'interface du service de sécurité comprend les opérations indiquées dans le Tableau 89. Alors que les interfaces CRL et OCSP sont ouvertes, les autres interfaces CSR, GetPublicKey et Revoke exigent une authentification et une autorisation spécifique du fournisseur de la PKI.

Tableau 89 – Vue d'ensemble des interfaces PKI

Interface PKI	Schéma d'échange	Définition	Fournisseur des informations	Utilisateur des informations
CSR	REQUEST_RESPONSE	Interface permettant d'envoyer des clés publiques dans une demande de signature de certificat et de les obtenir signées par le fournisseur PKI SECOM	Obligatoire	Obligatoire
GetPublicKey	REQUEST_RESPONSE	Interface permettant de demander une clé publique signée à l'administrateur de schéma (SA)	Obligatoire	Obligatoire
CRL	REQUEST_RESPONSE	Interface permettant de récupérer une liste de révocation de certificats (CRL), complète ou delta	Obligatoire	Obligatoire
OCSP	REQUEST_RESPONSE	Interface permettant de vérifier l'état de révocation des certificats avec le protocole de statut de certificat en ligne (OCSP)	Obligatoire	Obligatoire
Revoke	ONE_WAY	Interface permettant de révoquer un certificat	Obligatoire	Obligatoire

8.6.2 Interface de service – CSR

8.6.2.1 Spécification

Le rôle de cette interface est d'envoyer des clés publiques dans une demande de signature de certificat et de les obtenir signées par la PKI SECOM. La demande de signature doit être conforme à la PKCS #10 décrite dans la RFC 2986.

La Figure 47 représente l'interface en UML; le Tableau 90 et le Tableau 91 décrivent les données échangées dans l'interface.

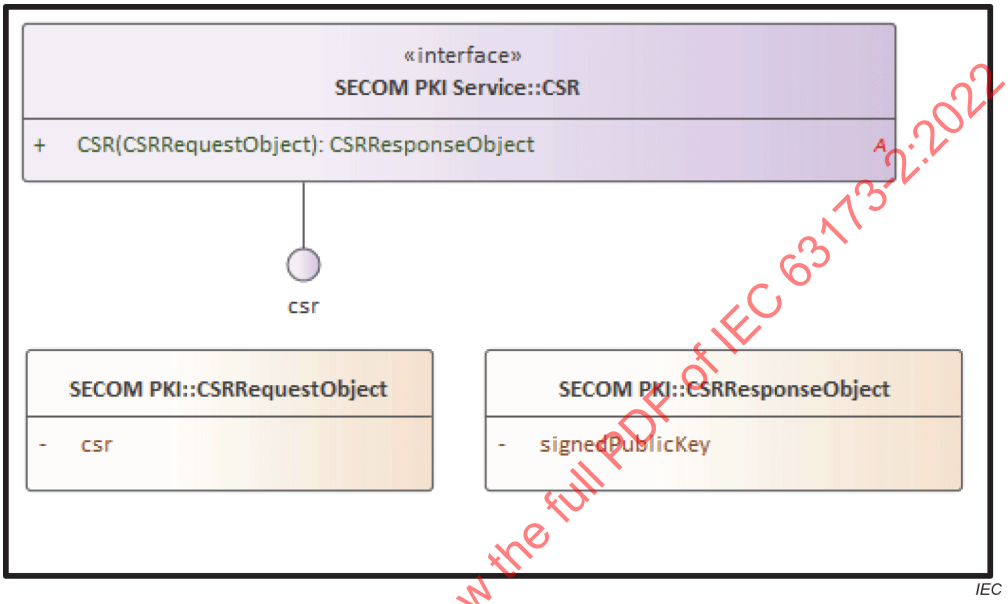


Figure 47 – Diagramme UML de l'interface CSR

8.6.2.2 Modèle d'échange de données

Tableau 90 – Entrée d'informations pour l'interface CSR

SignRequestObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
csr	1	Obligatoire	chaîne	CSR PKCS #10 codée au format PEM

Tableau 91 – Sortie d'informations pour l'interface CSR

SignRequestResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
signedPublicKey	1	Obligatoire	chaîne	PKCS #10 codée au format PEM

8.6.2.3 Conception REST

8.6.2.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

8.6.2.3.2 Opération – POST /csr

Cette opération est utilisée pour demander la signature de la clé publique par le PKI SECOM. La clé publique signée est renvoyée dans la réponse.

Le Tableau 92 décrit les paramètres logiques fournis dans l'interface.

Tableau 92 – Mise en œuvre REST de CSR

Opération REST			
POST URL/v1/csr{body}: return			
Paramètre REST (entrée)	**Codage REST**	**Mult.**	**Définition**
Non applicable (aucun paramètre accepté)			
Corps REST (entrée)	**Codage REST**	**Mult.**	**Définition**
SignRequestObject	texte en clair	1	Demande de signature en tant que chaîne, une CSR (demande de signature de certificat) PKCS #10 codée au format PEM
Retour (sortie)	**Codage REST**	**Mult.**	**Définition**
SignRequestResponseObject	application/pem-certificate-chain	1	Confirmation ou message d'erreur

8.6.2.3.3 Réponse du service

Le service doit répondre avec des messages et des codes HTTP conformes au Tableau 93.

Tableau 93 – Codes de réponse et messages HTTP dans l'objet réponse

Code HTTP	Message
200	OK
201	Créée
401	Non autorisé
403	Non autorisé
404	Introuvable

8.6.2.4 Comportement dynamique

La Figure 48 représente le comportement dynamique d'une demande de signature de certificat. L'Acteur 1 crée ou met à jour des certificats autosignés pour son application finale qui émet une requête à la PKI SECOM. La PKI renvoie une paire de clés publique-privée signée.

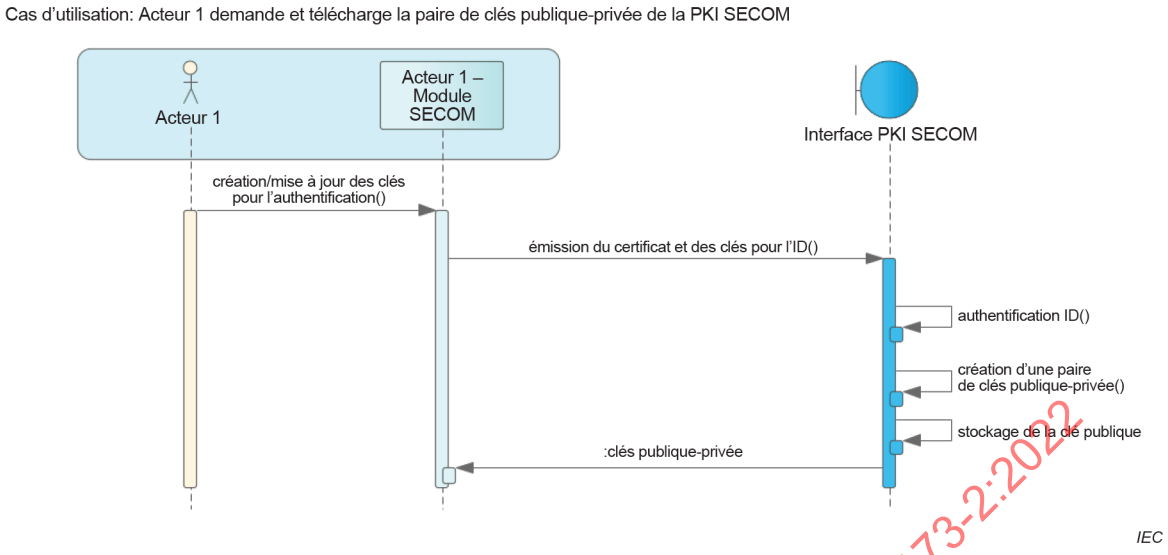


Figure 48 – Diagramme de séquence opérationnelle de CSR

8.6.3 Interface de service – GetPublicKey

8.6.3.1 Spécification

Le rôle de cette interface est de faciliter la récupération d'une clé publique à partir d'une PKI fournie par une empreinte numérique. Afin d'optimiser la bande passante, les interfaces SECOM échangent des empreintes de certificats au lieu du ou des certificats complets. L'utilisateur de l'interface de service nécessite un moyen de récupérer le certificat public correspondant à l'aide de l'empreinte numérique du certificat échangé. La Figure 49 représente l'interface en UML.

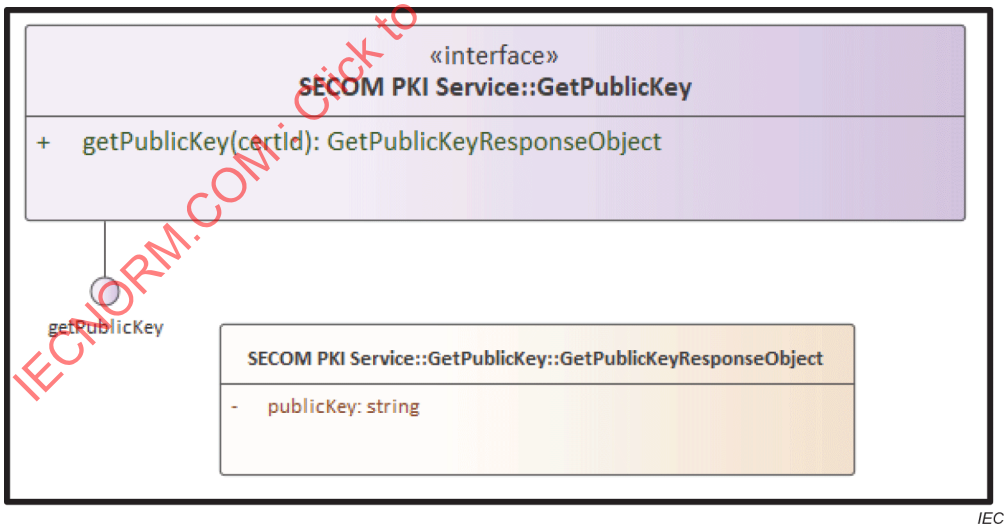


Figure 49 – Diagramme UML de l'interface GetPublicKey

8.6.3.2 Modèle d'échange de données

Les objets d'information dans l'échange de données sont indiqués dans le Tableau 94 et le Tableau 95.

Tableau 94 – Entrée d'informations pour l'interface GetPublicKey

GetPublicKeyObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
certThumbprint	0..1	Obligatoire	chaîne	Empreinte numérique de la clé publique déclarée, codée en Base64
certId	0..1	Obligatoire	chaîne	Numéro de série du certificat donné en caractères décimaux
identityId	0..1	Obligatoire	chaîne	Identifiant de l'identité liée au certificat, par exemple un MRN urn:mrn:org:ca:env:identity

Tableau 95 – Sortie d'informations pour l'interface GetPublicKey

GetPublicKeyResponseObject				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
publicKey	0..1	Obligatoire	chaîne	Clé publique (certificat X.509)

8.6.3.3 Conception REST

8.6.3.3.1 Généralités

L'interface du service et son modèle d'échange de données sont définis avec la technologie REST.

8.6.3.3.2 Opération – GET /publicKey

Cette opération est utilisée pour récupérer le contenu complet d'une clé publique par de la PKI SECOM. La clé publique est renvoyée dans la réponse.

Le Tableau 96 décrit les paramètres logiques fournis dans l'interface.

Tableau 96 – Mise en œuvre REST de l'interface GetPublicKey

Opération REST			
GET URL/v1/publicKey/parameter: return			
Paramètre REST (entrée)	**Codage**	**Mult.**	**Définition**
certThumbprint	chaîne	0..1	Empreinte numérique de la clé publique déclarée, codée en Base64
certId	chaîne	0..1	Numéro de série du certificat donné en caractères décimaux
identityId	chaîne	0..1	Identifiant de l'identité liée au certificat, par exemple un MRN urn:mrn:org:ca:env:identity
Corps REST (entrée)	**Codage**	**Mult.**	**Définition**
s.o.	s.o.	s.o.	s.o.
Retour (sortie)	**Codage**	**Mult.**	**Définition**
GetPublicKeyResponseObject	application/x-pem-file	1	Certificat X.509 codé au format PEM binaire

8.6.3.3.3 Réponse du service

Le service doit répondre avec des messages et des codes HTTP conformes au Tableau 97.

Tableau 97 – Codes de réponse et messages HTTP dans l'objet réponse

Code HTTP	Message
200	OK
401	Non autorisé
403	Non autorisé
404	Introuvable

8.6.3.4 Comportement dynamique

La Figure 50 représente le comportement dynamique de la demande Get pour récupérer un certificat public X.509.

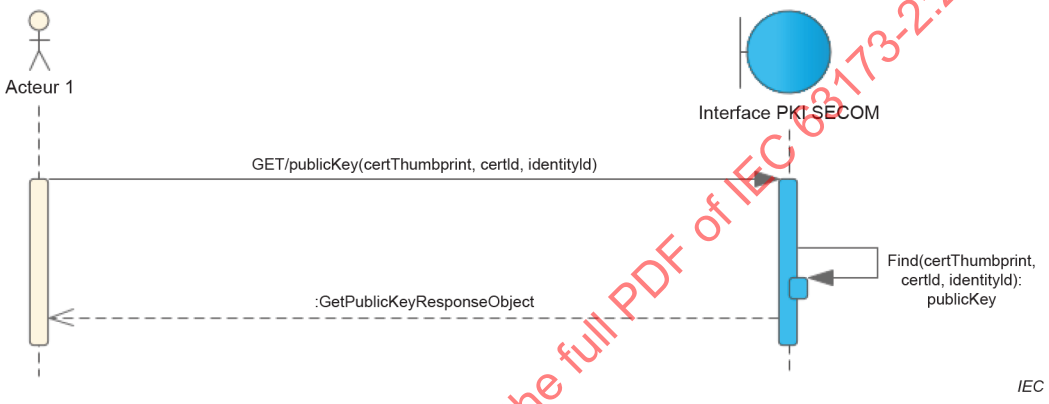


Figure 50 – Diagramme de séquence opérationnelle de GetPublicKey

8.6.4 Interface de service – CRL

8.6.4.1 Spécification

Le rôle de cette interface est de récupérer une liste de révocation de certificats. La Figure 51 représente l'interface en UML.

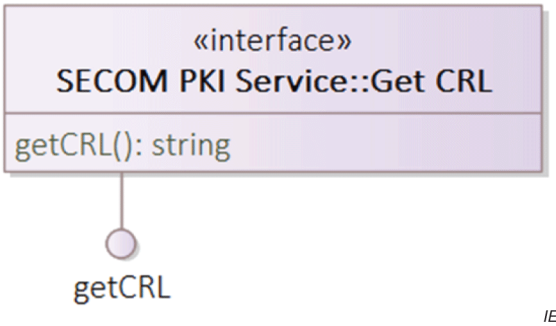


Figure 51 – Diagramme UML de l'interface GetCRL

8.6.4.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans la RFC 5280.

8.6.4.3 Conception REST

8.6.4.3.1 Généralités

La CRL doit être basée sur la RFC 5280.

8.6.4.3.2 Opération – GET /crl

L'opération REST crl est décrite dans la RFC 5280. Le Tableau 98 indique la mise en œuvre REST de l'interface CRL.

Tableau 98 – Mise en œuvre REST de CRL

Opération REST			
GET URL/v1/crl/parameter: return			
Paramètre REST (entrée)	**Codage**	**Mult.**	**Définition**
caAlias	chaîne	1	L'alias de l'autorité de certification, par exemple un MRN urn:mrn:org:ca:env:identity
Corps REST (entrée)	**Codage**	**Mult.**	**Définition**
s.o.	s.o.	s.o.	s.o.
Retour (sortie)	**Codage**	**Mult.**	**Définition**
CRL	application/x-pem-file	1	CRL X.509 codée au format PEM binaire

8.6.4.3.3 Réponse du service

Le service doit répondre avec des messages et des codes HTTP conformes au Tableau 99.

Tableau 99 – Codes de réponse et messages HTTP dans l'objet réponse

Code HTTP	Message
200	OK
401	Non autorisé
403	Non autorisé
404	Introuvable

8.6.4.4 Comportement dynamique

La Figure 52 représente le comportement dynamique pour récupérer une liste de révocation de certificats.

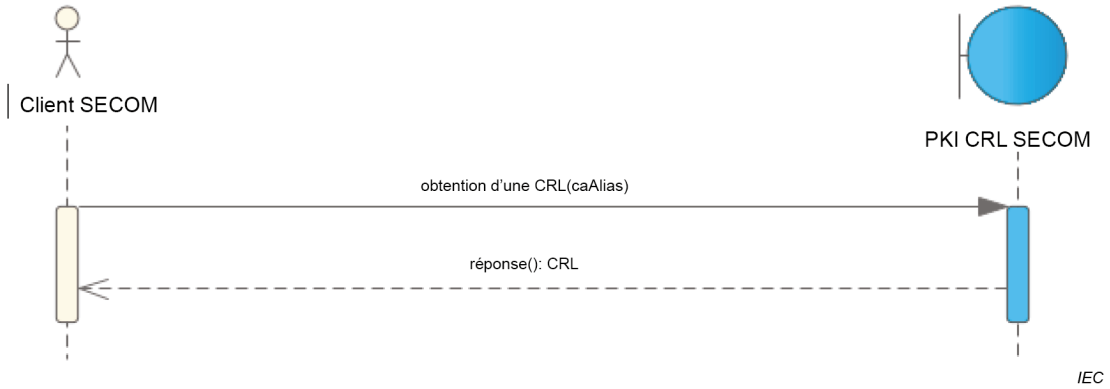


Figure 52 – Diagramme de séquence opérationnelle de CRL

8.6.5 Interface de service – OCSP

8.6.5.1 Spécification

Le rôle de cette interface est de vérifier l'état de révocation des certificats avec le protocole OCSP. La Figure 53 représente l'interface en UML.

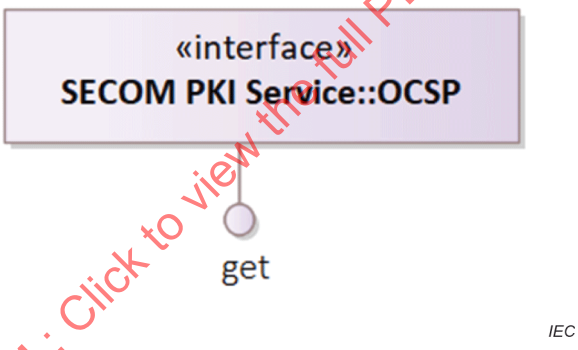


Figure 53 – Diagramme UML de l'interface GetOCSP

8.6.5.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans la RFC 6960.

8.6.5.3 Conception REST

8.6.5.3.1 Généralités

L'OCSP doit être basée sur la RFC 6960.

8.6.5.3.2 Opération – GET /ocsp

Les opérations REST pour l'OCSP sont décrites dans la RFC 6960. Le Tableau 100 indique la mise en œuvre REST de l'interface OCSP.

Tableau 100 – Mise en œuvre REST de OCSP

Opération REST			
GET URL/v1/ocsp/parameter: return			
Paramètre REST (entrée)	**Codage**	**Mult.**	**Définition**
caAlias	chaîne	1	L'alias de l'autorité de certification, par exemple un MRN urn:mrn:org:ca:env:identity
Corps REST (entrée)	**Codage**	**Mult.**	**Définition**
s.o.	s.o.	s.o.	s.o.
Retour (sortie)	**Codage**	**Mult.**	**Définition**
ocsp-response	application/ocsp-response	1	RFC 6960

8.6.5.3.3 Réponse du service

Le service doit répondre avec des messages et des codes HTTP conformes au Tableau 101.

Tableau 101 – Codes de réponse et messages HTTP dans l'objet réponse

Code HTTP	Message
200	OK
401	Non autorisé
403	Non autorisé
404	Introuvable

8.6.5.3.4 Opération – POST /ocsp

Les opérations REST pour l'OCSP sont décrites dans la RFC 6960. Le Tableau 102 indique la mise en œuvre REST de l'interface OCSP.

Tableau 102 – Mise en œuvre REST de OCSP

Opération REST			
POST URL/v1/ocsp/{body}: return			
Paramètre REST (entrée)	**Codage**	**Mult.**	**Définition**
s.o.	s.o.	s.o.	s.o.
Corps REST (entrée)	**Codage**	**Mult.**	**Définition**
ocsp-request	application/ocsp-request	1	RFC 6960
Retour (sortie)	**Codage**	**Mult.**	**Définition**
ocsp-response	application/ocsp-response	1	RFC 6960

8.6.5.3.5 Réponse du service

Le service doit répondre avec des messages et des codes HTTP conformes au Tableau 103.

Tableau 103 – Codes de réponse et messages HTTP dans l'objet réponse

Code HTTP	Message
200	OK
201	Créée
401	Non autorisé
403	Non autorisé
404	Introuvable

8.6.5.4 Comportement dynamique

La Figure 54 représente le comportement dynamique de la vérification en ligne de la validité d'un certificat.

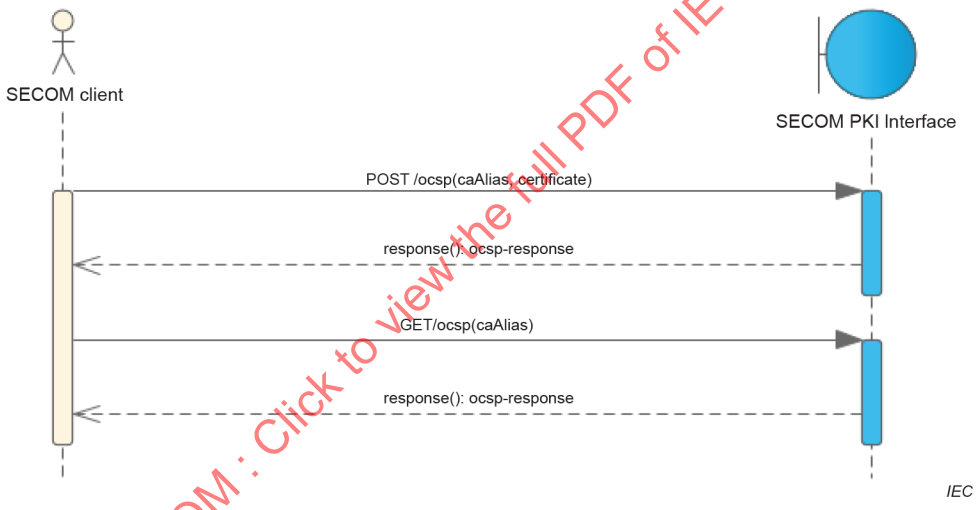


Figure 54 – Diagramme de séquence opérationnelle de OCSP

8.6.6 Interface de service – Revoke

8.6.6.1 Spécification

Le rôle de cette interface est de révoquer un certificat. La Figure 55 représente l'interface en UML.

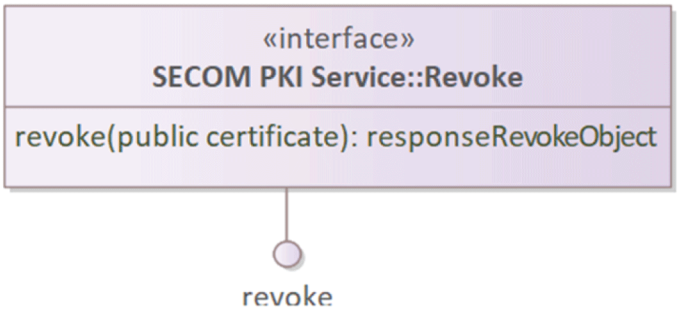


Figure 55 – Diagramme UML de l'interface PostRevoke

8.6.6.2 Modèle d'échange de données

Les données échangées dans l'interface de service sont décrites en détail dans le Tableau 104, le Tableau 105 et le Tableau 106.

Tableau 104 – Entrée d'informations pour l'interface Revoke

CertificateRevocation				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
RevocationReason	0..1	Obligatoire	RevocationReason Enum	Motif pour lequel le certificat a été révoqué
RevokedAt	0..1	Obligatoire	DateTime	Date à laquelle il convient d'activer la révocation du certificat

Tableau 105 – Enumérations pour l'interface Revoke

RevocationReasonEnum		
Index	**Valeur**	**Description**
1	unspecified	Enum UnspecifiedEnum pour unspecified
2	keycompromise	Enum KeycompromiseEnum pour keycompromise
3	cacompromise	Enum CacompromiseEnum pour cacompromise
4	affiliationchanged	Enum AffiliationchangedEnum pour affiliationchanged
5	superseded	Enum SupersededEnum pour superseded
6	cessationofoperation	Enum CessationofoperationEnum pour cessationofoperation
7	certificatehold	Enum CertificateholdEnum pour certificatehold
8	privilegewithdrawn	Enum PrivilegewithdrawnEnum pour privilegewithdrawn
9	aacompromise	Enum AacompromiseEnum pour aacompromise

Tableau 106 – Sortie d'informations pour l'interface Revoke

RevocationResponse				
Attribut	**Mult.**	**Traitement**	**Type**	**Définition**
Message	0..1	Facultatif	chaîne	Message de réponse

8.6.6.3 Conception REST

8.6.6.3.1 Généralités

Revoke doit être basé sur la RFC 6960.

8.6.6.3.2 Opération – POST /revoke

Le Tableau 107 indique la mise en œuvre REST de l'interface Revoke.

Tableau 107 – Mise en œuvre REST de Revoke

Opération REST			
POST URL/v1/revoke/parameter {body}: return			
Paramètre REST (entrée)	**Codage**	**Mult.**	**Définition**
certId	application/json	1	Numéro de série du certificat donné en caractères décimaux
Corps REST (entrée)	**Codage**	**Mult.**	**Définition**
CertificateRevocation	application/json	1	Motif de la révocation et heure d'activation
Retour (sortie)	**Codage**	**Mult.**	**Définition**
RevocationResponse	application/json	1	Message de réponse

8.6.6.3.3 Réponse du service

Le service doit répondre avec des messages et des codes HTTP conformes au Tableau 108.

Tableau 108 – Codes de réponse et messages HTTP dans l'objet réponse

Code HTTP	Message
200	OK
201	Créée
401	Non autorisé
403	Non autorisé
404	Introuvable

8.6.6.4 Comportement dynamique

La Figure 56 représente le comportement dynamique d'une demande de révocation.

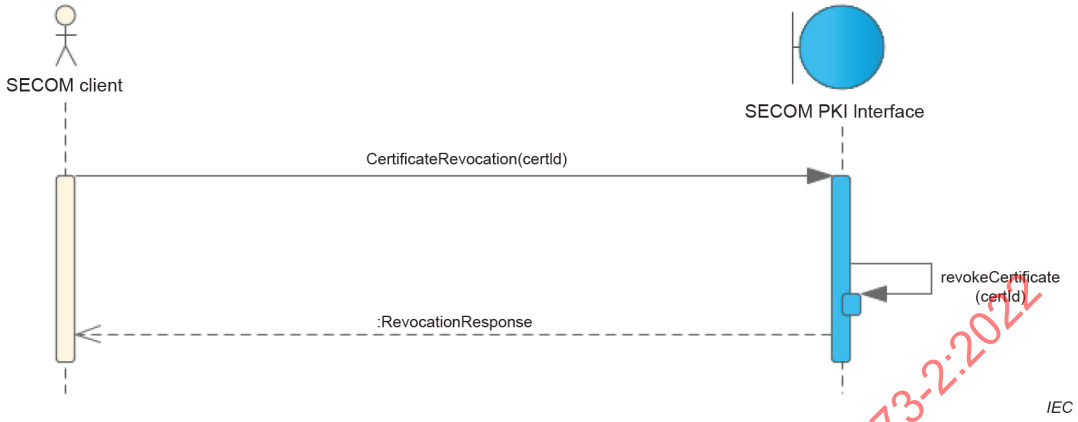


Figure 56 – Diagramme de séquence opérationnelle de Revoke

9 Interface du service de découverte de services SECOM

9.1 Généralités

La raison de la normalisation de la découvrabilité des services est d'obtenir une interopérabilité pour la recherche du service à utiliser. Pour encourager une approche orientée services et obtenir la flexibilité et l'évolutivité souhaitées, il est prévu de définir une interface de service commune permettant de trouver d'autres services à utiliser. Il est nécessaire d'enregistrer les services à utiliser dans un registre de services, ce qui permet de rechercher le point de terminaison du service. L'interface de découverte des services SECOM est placée devant un registre de services. Voir Annexe B.

Le SECOM ne définit pas le nombre de registres de services, ainsi le SECOM prend en charge plusieurs fournisseurs de registres de services, chacun étant mis en œuvre avec l'interface de découverte de services SECOM définie.

Le SECOM ne spécifie pas le registre de service à utiliser.

9.2 Interface de service – Search Service

9.2.1 Spécification

Le rôle de cette interface est de rechercher des instances de service à utiliser. La Figure 57 représente le diagramme UML du modèle d'information.

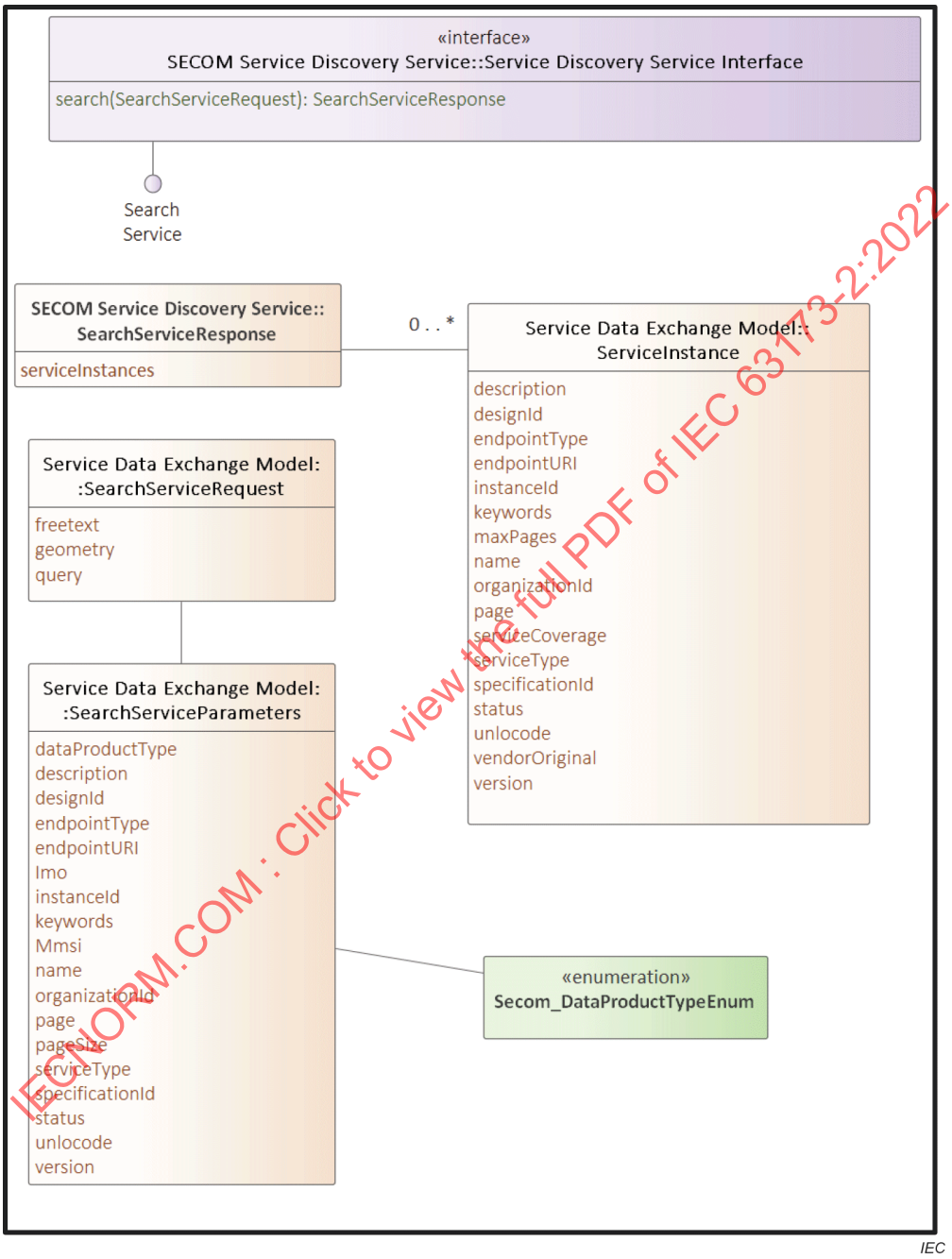


Figure 57 – Diagramme d'information UML de Search Service