INTERNATIONAL STANDARD

ISO 22376

First edition
2023-08

Security and resilience — Authenticity, integrity and trust for products and documents — Specification and usage of visible digital seal (VDS) data format for authentication, verification and acquisition of data carried by a document or object

Sécurité et résilience — Authenticité, intégrité et confiance pour les produits et les documents — Spécifications relatives aux formats de données et l'utilisation du Cachet Électronique Visible (CEV) aux fins d'authentification, de vérification et d'acquisition des données véhiculées par un document ou un objet

Reference number
ISO 22376:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 292, *Security and resilience*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

A visible digital seal (VDS) is a presentation of a structured data set, often in the form of a machine-readable code (MRC), used to ensure the authenticity and integrity of key data associated with a document or object at a relatively low cost and with a high level of security through asymmetrical cryptography.

Systems based on a VDS format can enable enhanced and interoperable secure track and trace with related anti-counterfeiting authentication.

The trustworthiness of the data carried by a VDS is dependent on the confidence that can be granted to the certificate related to its cryptographic environment. The VDS specified in this document is one possible presentation of electronically signed encoded data sets (ESEDSs), such as, for example, uniform resource identifier (URI) formatted in accordance with GS1 Digital Link Standard and protected by digital signature in accordance with ISO/IEC 20248. The expected confidence for this VDS is provided via a related ESEDS scheme.

This document will not interfere with existing authentication, traceability and identification systems. It enables interoperability between technologically heterogenous track and trace and anti-counterfeit environments.

Implementation of a trusted entry point (TEP) is enabled, helping to reduce the number of object identification applications by combining them into a universal and use-case resilient reader for professional inspectors and consumers. This document is intended for developers and users wishing to enable secure and interoperable track and trace and authentication systems. Since the method is technology agnostic, it is available for all industrial sectors where unique identifiers (UIDs) or data need to be secured and inspected in a global and multilingual environment.

Security and resilience — Authenticity, integrity and trust for products and documents — Specification and usage of visible digital seal (VDS) data format for authentication, verification and acquisition of data carried by a document or object

1 Scope

This document defines the conditions necessary for the interoperable deployment of visible digital seals (VDSs). It describes the structure, possible forms of representation, production process and verification process applicable to VDSs, for any type of document or object to which they relate.

This document does not establish requirements for users that issue and verify documents or for users that implement and deploy VDSs.

This document does not apply to detailed response formatting functions (RFFs). These requirements and functions are defined by the trust service operator (TSO) and generally cover functionalities such as the security levels of certificates and governance rules to be applied to document issuers and trust service providers (TSPs) intervening in the VDS ecosystem.

This document does not apply to the governance related to the operation of the VDS scheme. It is not intended to replace the specifications from Agence Nationale des Titres Sécurisés (ANTS), Bundesamt für Sicherheit in der Informationstechnik (BSI) and International Civil Aviation Organization (ICAO) documents.

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.

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions — Part 1: Country code*

ISO 14533-1, *Processes, data elements and documents in commerce, industry and administration — Long term signature — Part 1: Profiles for CMS Advanced Electronic Signatures (CAdES)*

ISO 14533-2, *Processes, data elements and documents in commerce, industry and administration — Long term signature — Part 2: Profiles for XML Advanced Electronic Signatures (XAdES)*

ISO 22300, *Security and resilience — Vocabulary*

ISO/IEC 8825-1, *Information technology — ASN.1 encoding rules — Part 1: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 9834-1, *Information technology — Procedures for the operation of object identifier registration authorities — Part 1: General procedures and top arcs of the international object identifier tree*

ISO/IEC 9834-8, *Information technology — Procedures for the operation of object identifier registration authorities — Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers*

ISO/IEC 15459-2, *Information technology — Automatic identification and data capture techniques — Unique identification — Part 2: Registration procedures*

ETSI TS 102 853, *Electronic Signatures and Infrastructures (ESI); Signature verification procedures and policies Technical Specification*

ETSI TS 119 612, *Electronic Signatures and Infrastructures (ESI); Trusted Lists*

IETF RFC 3339, *Date and Time on the Internet: Timestamps*

IETF RFC 3986, *Uniform Resource Identifiers*

IETF RFC 4387, *Internet X.509 Public Key Infrastructure, Operational Protocols: Certificate Store Access via HTTP*

IETF RFC 4648:2006, *The Base16, Base32, and Base64 Data Encodings*

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

NIST FIPS 180-4, *Secure Hash Standard (SHS)*

NIST FIPS 186-5, *Digital Signature Standard (DSS)*

PCRE, *Perl Compatible Regular Expressions* [online]. Available at: https://www.pcre.org/

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 22300 and the following apply.

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

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
machine-readable code
MRC
graphical symbol or electronic device, or a combination of the two, containing a set of signs, characters, patterns or signals that can be interpreted by an acquisition system

Note 1 to entry: Examples of MRC include, but are not limited to, 2D barcodes and radio-frequency identification (RFID) tags.

3.2
electronically signed encoded data set
ESEDS
structured data set containing the header, payload, signature and optional auxiliary data block

Note 1 to entry: The payload type and issuer identity are included in the header.

Note 2 to entry: ESEDS can often be expressed as *machine-readable code* (3.1).

3.3
visible digital seal
VDS
structured data set, often in the form of a *machine-readable code* (3.1), containing a payload and its signature from the issuer

Note 1 to entry: A header identifies the type of payload and the issuer. An optional auxiliary data block may be added after the signature

Note 2 to entry: VDS specified in this document is one of possible various *electronically signed encoded data sets* (3.2).

3.4
asymmetrical cryptography
encryption/decryption operations performed using a key pair: a private key used by the issuer to sign *documents* (3.10) and a public key used to verify the signature

Note 1 to entry: The two keys have an "asymmetric" role, hence the term.

3.5
base36
notation for encoding arbitrary data using a restricted set of symbols made of 36 characters: digits 0 to 9 and letters A to Z

Note 1 to entry: Symbol "Z" has a value of 35

3.6
C40 encoding
encoding to reduce the number of bytes required to encode a string of characters

Note 1 to entry: C40 encoding is explained in ISO/IEC 16022:2006.

3.7
certificate authority
CA
service offered by a *trust service provider* (3.21) to create, issue and produce *certificates* (3.11) on behalf of users, and ensure the integrity of the electronic identification of signatories

Note 1 to entry: The CA signs the certificate (with its own private key) to guarantee the integrity of the certificate and the accuracy of the data contained in the certificates that it issues.

3.8
certificate revocation list
CRL
list of *certificates* (3.11) that have been revoked by the issuing *certificate authority* (3.7)

Note 1 to entry: The CRL shall be in in accordance with IETF RFC 5280.

3.9
digital signature algorithm
DSA
mathematical technique used to validate the authenticity and integrity of a message, software or digital document

Note 1 to entry: These include, but are not limited to, the RSA and ECDSA algorithms defined in NIST FIPS 186-4.

3.10
document
information and the medium on which it is contained

Note 1 to entry: The medium can be paper or any other substrate, magnetic, electronic or optical computer disc, photograph or master sample, or a combination thereof.

3.11
certificate
electronic certificate
X.509 certificate
electronic file attesting that a cryptographic key pair belongs to either a physical or legal person, a hardware component or a software component as identified in the certificate

Note 1 to entry: Certificates are issued by a *certificate authority (CA)* (3.7). By signing the certificate, the CA certifies the association between the key pair with the person, hardware component or software component. A certificate may be revoked if this association can no longer be established. A certificate is valid for a limited amount of time.

3.12
hash
operation that consists of applying a mathematical function to create a digital fingerprint on a data block, transforming the data block into a fixed-size code for authentication and storage purposes

Note 1 to entry: Any change to the original data block results in a change in the hash value.

3.13
manifest
external resource containing information in extensible markup language (XML) format about the *visible digital seal* (3.3) use case, its *schema* (3.18), validation policies and optional extensions

3.14
message pack
binary data exchange format used to represent simple data structures and tables

Note 1 to entry: The purpose of message pack is to be as compact and simple as possible. It is defined in Reference [10].

3.15
online certificate status protocol
OCSP
protocol to validate a *certificate's* (3.11) status, usually to determine if the certificate has been revoked

Note 1 to entry: OCSP is explained in IETF RFC 6960.

Note 2 to entry: OCSP is an alternative to a *certificate revocation list* (3.8).

3.16
regular expression
character string that describes, using a specific syntax, a set of allowed strings or characters

Note 1 to entry: The regular expression shall conform to the Perl Compatible Regular Expressions (PCRE) specification.

3.17
response formatting function
RFF
function specifying how to format and present the output with *visible digital seal* (3.3) verification results

3.18
schema
payload data structure that allows for data encoding, decoding and verification

3.19
symbology
description of numeric, text or binary data encoding in a *machine-readable code* (3.1) defining the redundancy, error correction code mechanisms and specifying the quiet zone around the barcode

3.20
trust service operator
TSO
entity that defines the governance structure and technical requirements of the trust service, and oversees the overall operations

Note 1 to entry: In some industries, the TSO acts as the authentication service body (ASB).

3.21
trust service provider
TSP
entity tasked with defining the *certificate authority (CA)* (3.7) trust framework and governance structure, offering certificate service(s), operating the CA and ensuring compliance with said governance

3.22
trusted entry point
TEP
method provided and/or certified by the *trust service operator* (3.20) having support for the *response formatting function* (3.17), open for additional object identification and authentication systems (OIAS), and able to resolve without ambiguity any unique identifier (UID)

3.23
trust service list
TSL
list containing compliant information about the *trust service operator (TSO)* (3.20), the *trust service providers (TSPs)* (3.21) and the TSP's *certificate authority* (3.7) authorized to issue *certificates* (3.11) to sign *electronically signed encoded data sets* (3.2)

Note 1 to entry: ETSI TS 119 612 sets out what is a compliant TSL.

Note 2 to entry: TSLs are extensible using extensible markup language (XML) defined by the TSO.

3.24
uniform resource identifier
URI
character string that unambiguously identifies a resource

Note 1 to entry: The syntax shall conform to IETF RFC 3986.

4 General concepts

A VDS is usually represented in the form of one or a combination of MRC(s). It contains data to be protected and the electronic signature of the data by the document issuer. VDSs are read using an optoelectronic or electronic device, e.g. a scanner, 2D barcode reader, RFID reader or a combination of devices.

The detailed implementation of a VDS is always conditioned by a specific use case. Defined by a group of experts, each use case determines the key data fields to be included in the schema, as well as the field constraints. If required, the use case may also include additional verification policies and features defined in TSO extensions to satisfy the business rules for the use case. Use cases are then translated into a machine-readable format (secure XML-format manifest file) so that those who produce and verify the VDSs can interpret and process the information in a standard and deterministic way, while respecting the rules defined for each use case. The manifest is extensible through XML extensions defined by the TSO, with the aim of enabling a richer set of functionalities, such as additional VDS verification policies (e.g. authorized symbologies, signer's legitimacy, validity period), the RFF and post-verification business rules.

Many steps have been taken to minimize the amount of space used by the data carrier and to maximize the amount of space that can be used for data. As such, the VDS does not contain the certificate used for the signature or the definition of the use case; rather, the VDS contains identifiers allowing for their retrieval (see 5.3.2). The VDS payload and signature are also structured to minimize size.

To ensure the reliability of the service, verification of the different elements of the VDS and the chain of trust is essential. Therefore, verification is a normative component to this specification (see Clause 7). Each element supporting the VDS is protected and signed by the TSO to ensure its trustworthiness.

In this document, in order to unambiguously differentiate hexadecimal numbers from others, a prefixed notation “0x” is used.

5 Structures and resources

5.1 General

All URI and endpoints shall be in lower case.

5.2 Trust service list and extensions

5.2.1 General

The TSL shall conform to ETSI TS 119 612, with the exception of the field <SchemeTerritory> which is optional, and include information regarding the TSO and authorized TSP’s certificate authority (CA) to enable trusted service operation.

5.2.2 Extensions

The operation of a VDS trust service requires adding the following three attributes that are not defined in ETSI TS 119 612:

— URI to locate manifests;

— CA reference for each TSP Service;

— URI to locate signing certificates for each CA.

These shall be defined through extensions.

5.2.3 TSO identity

The TSO shall be identified in accordance with ISO/IEC 15459-2 by an Issuing Agency Code (IAC). This IAC code shall be used to locate the TSL.

5.2.4 TSO manifest location

The VDS header includes a reference to the manifest. To locate the manifest, the TSL shall include a URI to the manifest directory.

The manifest directory URI is defined by the extension attribute <VDSManifestResource> in the scheme extensions of the TSL using the following xPath:

```
/TrustServiceStatusList/SchemeInformation/SchemeExtensions/Extension/
vdsext:VDSManifestResource
```

The manifest URI is a concatenation of the manifest directory URI and the manifest ID (in hexadecimal format, always in lower case letters). An example to retrieve manifest 0x89AB01 with the <VDSManifestResource> extension value is as follows:

```
“https://manifest.otentik.codes”:
https://manifest.otentik.codes/89ab01.xml
```

5.2.5 CA reference

The CA reference is a four-character string included in the VDS header to locate the signing certificate and to ensure the CA validity in the TSL. It is composed of:

— CA issuing country (ISO 3166-1 Alpha2), in upper case letters;

— CA identifier: two digits assigned by the TSO.

The CA reference is defined by the extension attribute <VDSAuthorityID> in the TSL's service information extensions using the following xPath:

```
/TrustServiceStatusList/TrustServiceProviderList/TrustServiceProvider/TSPInformation/
ServiceInformationExtensions/Extension/vdsext:VDSAuthorityID
```

5.2.6 Public certificate directory

The VDS header includes a reference to the signing certificate. To locate the certificate, the TSL shall include, for each CA, a URI to the certificate directory. The TSP's CA identifier should be present in the URI.

The certificate endpoint is defined by the attribute <VDSCertResource> in the service information extensions of the TSL using the following syntax:

```
/TrustServiceStatusList/TrustServiceProviderList/TrustServiceProvider/
TSPServices/TSPService/ServiceInformation/ServiceInformationExtensions/Extension/
vdsext:VDSCertResource
```

The certificate URI is a concatenation of the certificate endpoint and the certificate reference number (in base36 format). This URI format shall conform to IETF RFC 4387. An example to retrieve certificate "09HZ" issued by a CA that is assigned the identifier "FR01" and the <VDSCertResource> to "`https://trust.otentik.codes/fr01`" is as follows:

```
https://trust.otentik.codes/fr01/09hz.cer
```

5.2.7 XML security

To ensure its integrity, the TSL shall be electronically signed by the TSO in accordance with ISO 14533-1 CAdES or ISO 14533-2 XAdES-B format. The TSO may detail the required parameters and characteristics of the signature.

5.2.8 Example and verification

See Annex B for a TSL example.

The TSL shall include a xsi:schemaLocation directive to facilitate automated XML validation (see the example in Annex D).

5.3 Manifest

5.3.1 General

The manifest is a machine-readable definition of a use case. It contains various sections: use case information, data schema and optional extensions (defined by the TSO). The VDS header includes a reference to a manifest. The reference number shall be assigned by the TSO.

5.3.2 Information section

This section links the machine-readable manifest identifier to a human-readable use case name and description. The list of attributes and their description is given in Table 1.

Table 1 — Information section attributes

Attribute	Description
`Id`	Use case identifier (six hexadecimal characters). Corresponds to the manifest ID field in the VDS header.
`Version`	Use case version (informative only). Allows for non-breaking changes in the data schema, such as changes in the name or description of the use case, or non-breaking changes to extension information.
`Name`	Use case name. Multilingual.
`Description`	Use case description. Multilingual.

An example of the information section is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Manifest version="1" xmlns="https://otentik.codes/" xsi:schemaLocation="https://otentik.
codes/ https://otentik.codes/schemas/otentik-baseline-manifest-v1.xsd">
  <Id>89AB01</Id>
  <Version>1</Version>
  <Name>
    <Value xml:lang="en">Use Case Name</Value>
    <Value xml:lang="fr">Nom du cas d'usage</Value>
  </Name>
  <Description>
    <Value xml:lang="en">Use case description</Value>
    <Value xml:lang="fr">Description du cas d'usage</Value>
  </Description>
...
</Manifest>
```

5.3.3 Schema section

This section defines, for a specific use case, the structure of the information in the VDS payload and auxiliary data sections as well as the field validation rules. The schema is used both for VDS production and verification.

The default Payload and AuxData encoding format is MessagePack. The TSO may define support for other encoding formats. When using MessagePack encoding, the VDS Payload and AuxData data shall be encoded in the same order as the manifest's schema definition. The list of attributes and their description is given in Table 2.

Table 2 — Schema section attributes

Attribute	Description
`Payload`	Defines the data structures, whether optional or mandatory, that will be signed. The optional parameter **encoding** is used for formats other than MessagePack (CBOR, JSON, TLV or any other), e.g. `<Payload encoding="cbor">`)
`AuxData`	Defines the data structures that are not be signed. The auxiliary data section is optional and uses the same XML structure and encoding principles as the payload.
`Types`	Defines custom field types to be referenced by objects and object arrays.

The schema can define data types and constraints, such as permitted characters and minimum/maximum lengths or values. The schema supports typical data fields as well as objects and data arrays.

When a TSO authorizes a specific data structure, it shall include the ability to decode that data structure in the TEP reader application.

An example of the schema section is as follows:

```
<Schema>
  <Payload>
    <Fields>
```

```
      <String name="stringSimple"/>
      <String name="stringC40">
        <StringConstraints>
          <Nillable/>
          <Pattern>[A-Z0-9]</Pattern>
          <Encoding>C40</Encoding>
        </StringConstraints>
      </String>
      <Integer name="intExample">
        <IntegerConstraints>
          <Nillable/>
          <Min>1</Min>
          <Max>9999</Max>
        </IntegerConstraints>
      </Integer>
      <Object name="objectExample" type="object1">
        <ObjectConstraints>
          <Nillable/>
        </ObjectConstraints>
      </Object>
    </Fields>
  </Payload>
  <Types>
    <Type name="object1">
      <Fields>
        <String name="string">
          <StringConstraints>
            <Nillable/>
            <Pattern>[A-Za-z]</Pattern>
          </StringConstraints>
        </String>
        <Date name="date">
          <DateConstraints>
            <Nillable/>
            <From>2000-01-01</From>
          </DateConstraints>
        </Date>
      </Fields>
    </Type>
  </Types>
</Schema>
```

See Annex C for more detailed examples.

The manifest uses common field types, as well as objects, arrays and C40-encoded strings (see Table 3).

NOTE String encoding generally reduces the data size of the encoded string (albeit with a reduced character set) and is available for any symbology.

Table 3 — Schema parameters and related constraints

Schema parameters	Optional parameter constraints
`Integer`	`<IntegerConstraints>` `Nillable`: the field is optional `Min`: minimum value of the integer `Max`: maximum value of the integer
`Boolean`	`<BooleanConstraints>` `Nillable`: the field is optional
`Float`	`<FloatConstraints>` `Nillable`: the field is optional `Min`: minimum value of the float `Max`: maximum value of the float

Table 3 *(continued)*

Schema parameters	Optional parameter constraints
`String`	`<StringConstraints>` `Nillable`: the field is optional `MinLength`: minimum number of characters `MaxLength`: maximum number of characters `Pattern`: regular expression in accordance with Perl Compatible Regular Expressions (PCRE) defining the accepted characters in the string `Encoding`: alternative encoding type if the default UTF-8 isn't adequate. Only value C40 is accepted.
`Binary`	`<BinaryConstraints>` `Nillable`: the field is optional `MinLength`: minimum number of bytes `MaxLength`: maximum number of bytes
`Timestamp`	`<TimestampConstraints>` `Nillable`: the field is optional
`Date`	Encodes the number of days (positive or negative) between a date and a reference date. `<DateConstraints>` `Nillable`: the field is optional `From`: reference date using the format YYYY-MM-DD. Default reference date is 1900-01-01 NotBefore: minimum valid date NotAfter: maximum valid date
`Object`	`<ObjectConstraints>` `Nillable`: the field is optional Objects are defined in a custom type, and are encoded as an array of fixed size, using one entry of the array for each field defined in the custom type.

`IntegerArray`
`BooleanArray`
`FloatArray`
`StringArray`
`BinaryArray`
`TimestampArray`
`DateArray`
`ObjectArray`
	`<ArrayConstraints>`
	`Nillable`: the array is optional
	`MinSize`: minimum array size
	`MaxSize`: maximum array size
	In addition to the ArrayConstraints, arrays should have constraints that correspond to the field type in the array (Integer, Boolean, Float, etc.)

5.3.4 Extensions section

The manifest's XML shall be structured to allow extensions through the `/Manifest/Extensions` attribute.

5.3.5 XML security

To ensure its integrity, the manifest shall be electronically signed by the TSO in accordance with ISO 14533-1 CAdES or ISO 14533-2 XAdES-B format. TSO may detail the required parameters and characteristics of the signature.

5.3.6 Example and verification

See Annex C for an example of a manifest.

For automatic verification, the manifest shall include the xsi:schemaLocation directive. See the example in Annex E for more details.

5.4 Manifest extensions

5.4.1 General

The TSO is responsible to define the default validation policies in accordance with ETSI TS 102 853. Some policies can be added or overridden by a use case and defined in the manifest.

5.4.2 Policies extension

The policies extension is defined as follows:

```
<Extension xsi:type="ext:PoliciesExtension">
...
</Extension>
```

5.4.3 Authorized usage policy

The TSO shall put in place the policy, ensuring that the signing certificate is legitimately authorized to sign a VDS for a specific use case.

This policy is tagged `Authorized Usage`.

When this policy is defined, VDS verification shall fail if the universally unique identifier (UUID) defined in the manifest is not listed in the signing certificate's usage list extension identified by the object identifier (OID) (see 5.6.2 for more details). The UUID shall be 128-bit long, in accordance with ISO/IEC 9834-8.

```
<ext:AuthorizedUsage>
  <ext:oid>1.3.6.1.4.1.51528.1.1</ext:oid>
  <ext:uuid>57c19de1cbe74605ba74deb773f97042</ext:uuid>
</ext:AuthorizedUsage>
```

NOTE Additional extensions that can be defined and specified by the TSO are outside the scope of this document.

5.5 VDS

5.5.1 General

A VDS is a data structure that may be preceded by ISO/IEC 15459-2 IAC code and is composed of up to four blocks of data delivered by the data carrier reader to the TEP. The functional structure of a VDS

is given in Figure 1. Received data are organized or reorganized via the mechanisms included in the manifest resulting in the order shown in Table 4.

NOTE The role of the ISO/IEC 15459-2 IAC prefix is to enable common readers and application programming interfaces (APIs) to recognize the TSO exploiting the VDS scheme carried by ISO-standardized MRCs including, but not limited to, RFID, QR-code, data matrix and Han Xin. Since the ISO/IEC 15459-2 IAC prefix is mandatory in the VDS header, the VDS implementations within URIs and private data carriers can be optimized for VDS-dedicated readers and APIs. In such a case, there is no need to precede the VDS with the IAC code because once VDS blocks are decoded, the data can be reorganized by the RFF (via manifest) as data in accordance with ISO/IEC 15459-2.

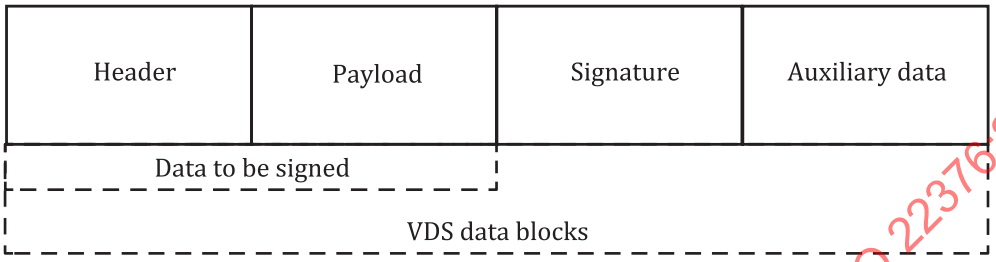

Figure 1 — General format of a visible digital seal

Table 4 — Data blocks in VDS structure

Attribute	Description
`Header`	The header provides the information required to decode and verify the VDS.
`Payload`	The payload contains the data specific to each VDS, whether mandatory or optional. The size of this block is variable. The header and payload are together referred to as the “Data”.
`Signature`	The signature is used to verify the integrity of the two parts mentioned above (the header and the payload) and to identify/authenticate the issuer of the document onto which the VDS is placed. The signature is computed using the header and payload. See 5.5.5 for calculation details.
`AuxData`	The auxiliary data contains additional information about the document that is not signed upon VDS creation. The auxiliary data section is optional. It applies the same encoding principles as the payload.

5.5.2 Binary encoding

The MRC should be encoded using a format that optimizes code size and readability for a specific use case, keeping in mind that not all MRC readers support Base256 (binary). Other encoding methods should include Base32 and Base45. The TSO may define an extension in the manifest to restrict the type of encoding allowed.

To ensure conformity to ISO/IEC 15459-2, the IAC prefix shall not be encoded.

If needed, a VDS shall be padded with binary zeros to accommodate encoding methods that do not support padding. When encoding in Base32, the alphabet used shall be the standard alphabet as defined in IETF RFC 4648:2006, section 6. However, since some symbologies cannot efficiently store the “=” (EQUAL) character (the padding character defined in IETF RFC 4648), the “+” (PLUS) character shall also be an accepted padding character.

5.5.3 Header section

The header provides the information required to decode and verify the VDS. All binary fields are using Big Endian convention. The header structure and description shall be as shown in Table 5.

Table 5 — Header information

Pos	Bytes	Description	Example
0x00	1	**VDS marker** with the hard-coded value 0xDE	0xDE
0x01	1	**Payload length field type** (bits 7 to 6) 01: uint8 — header's payload length is 8 bits 00: uint16 — header's payload length is 16 bits 10: uint32 — header's payload length is 32 bits 11: reserved Reserved bits (bits 5 to 4) Shall be 00 **VDS version ID** (bits 3 to 0) Currently at version 3 NOTE Bit 7 is the most significant bit.	0x03
0x02	2	**ISO/IEC 15459-2 Issuing Agency Code (IAC)** (three C40-encoding characters) The IAC code shall be filled with ASCII [0] until it reaches a length of three characters.	"XXX"
0x04	6	**Certificate reference** (nine C40-encoded characters) CA reference (four characters) — CA issuing country (ISO 3166-1 Alpha2) - two upper case characters [A-Z] — CA identifier - two digits [0–9] Certificate identifier (four upper case characters) — four characters in base36 [0–9][A-Z] Reserved (one C40 character) Always ASCII 0 (C40 code 4)	"FR99" "09HZ" "0"
0x0A	3	**Manifest ID**	0x89AB01
0x0D	4	**Timestamp** of the VDS signature Unsigned 32-bit Unix timestamp as defined in IETF RFC 3339 EXAMPLE Aug 22 2017 08:00:00 GMT+0000	Date: Aug 22 2017 08:00:00 UTC Unix timestamp: 1503388800 Header value: 0x599BE480
0x11	1, 2 or 4	**Payload length** The payload length is encoded according to the field Payload length field type, and has the following size: — 1 byte (uint 8 - lengths of up to 255 bytes) — 2 bytes (uint 16 - lengths of up to 64 kB) — 4 bytes (uint 32 - lengths of up to 4 GB) Default encoding size should be 2 bytes.	0x0044

For example, using the data in the table above, the header would be as follows.

VDS marker	DE
Payload length field type and VDS version	03
IAC identifier (“XXX” in C40)	ED2E
Certificate reference (“FR9909HZ0” in C40)	7BA651EE895D
Manifest ID	89AB01
Signature date and time	599BE480
Payload Length	0044
Header (19 bytes)	0xDE03ED2E7BA651EE895D89AB01599BE4800044

5.5.4 Payload section

To minimize the data size, the data schema is defined in the manifest and referenced through the VDS header. Data encoded in the payload section can be of any format supported by the TSO and indicated in the manifest.

Table 6 gives an example of data encoded using the message pack format. The official source of information pertaining to this format is given in Reference [10].

Table 6 — Schema parameters and corresponding message pack format family

Schema parameters	Message pack format family
Integer	nil, int
Boolean	nil, bool
Float	nil, float
String	nil, string
Binary	nil, binary
Timestamp	nil, uint32[a]
Date	nil, int
Object	nil, array
IntegerArray BooleanArray FloatArray StringArray BinaryArray TimestampArray DateArray ObjectArray	nil, array

[a] To optimize encoding size, timestamp fields use uint32 (4 bytes) instead of MessagePack’s Timestamp32 format (6 bytes).

All defined fields in the schema are mandatory, unless they have the `nillable` constraint. In this case, the field becomes optional and shall be encoded as `NIL` if not used.

Due to UTF-8 encoding, the number of encoded bytes in the payload may exceed the maximum number of characters allowed in the string.

NOTE C40 encoding generally minimizes the number of bytes needed to encode a string. However, the number of bytes required for encoding can sometimes exceed the size of the string, especially when the latter is composed of several lower case characters.

Some examples of payload message pack encoding using fields defined in the manifest are as shown in Table 7.

Table 7 — Message pack encoding examples based on manifest's field definition

Field definition in payload section of manifest	Field encoding in message pack
`<String name="stringSimple"/>`	**Value to encode**: Évaluation **Message pack format**: fixstr11 (0xAB) **Data value**: 0xC38976616C756174696F6E NOTE Message pack encodes strings using UTF-8. As such, the letter "É" is encoded in 2 bytes (0xC389).

`<String name="stringC40">`
`  <StringConstraints>`
`  <Nillable/>`
`  <Pattern>[A-Z0-9]</Pattern>`
`  <Encoding>C40</Encoding>`
`  </StringConstraints>`
`</String>`
	Value to encode: A1B2C3D4E5F6
	Message pack format: fixstr18 (0xA8)
	Data value: 0x585828086B933B43

`< Integer name="intExample">`
`  <IntegerConstraints>`
`  <Nillable/>`
`  <Min>1</Min>`
`  <Max>9999</Max>`
`  </IntegerConstraints>`
`</Integer>`
	Value to encode: 200
	Message pack format: uint8 (0xCC)
	Data value: 0xC8

See Annex C for more examples.

NOTE An empty string (fixstr0) is not the same as NIL. An empty string (fixstr0) defines a 0-character string. NIL refers to an optional string that is not used in a VDS.

Message pack allows the encoding of the decimal value "23" using either fixint, int8, int16, int32, uint8, uint16 or uint32. However, even though all these are valid numeral formats to store the value "23," the encoder should optimize and aim for the most efficient data size.

5.5.5 Signature section

The signature data are in raw format (r|s), not in ASN.1 DER format.

The certificate defines the digital signature algorithm (DSA) used and incidentally the signature size, which is typically based on the signature's expected life span. The hashing algorithm, whose robustness should also correlate to the signature's expected life span, shall be as stated in Table 8.

Table 8 — Hashing algorithm and signature size

Digital signature algorithm	Hashing algorithm	Signature size (bytes)
ECC P-192	SHA-224	48
ECC P-224	SHA-224	56
ECC P-256	SHA-256	64
ECC P-384	SHA-256	96
ECC P-521	SHA-512	132
RSA-1024	SHA-256	128
RSA-2048	SHA-256	256
RSA-3072	SHA-512	384
RSA-4096	SHA-512	512
Other	Defined by TSO	Defined by TSO

The hash calculation for the signature shall be performed in accordance with NIST FIPS 180-4 in the following two-step hash calculation:

— calculate the payload hash and append it to the header;

— calculate the hash of (header + payload's hash).

This two-step approach improves privacy and security: a trusted party (or a TSP) can provide a VDS signature service without having access to sensitive data contained in the payload.

NOTE An extension defined in the manifest can require external payload data, not present in the VDS payload section, to be included in the hash calculation.

5.5.6 Auxiliary data section

The auxiliary data and the payload sections use the same schema description, encoding and validation. However, data in the auxiliary data section is not signed. This section is mainly used in the context of anti-counterfeiting and anti-fraud, where at the time of the generation of the VDS some information about the production process cannot be known.

This section can also be used as a necessary secret/encrypted data enabling the implicit authentication of the header/payload or authentication of the object related to the payload.

5.5.7 Example

See Annex C for a byte-level example of an encoded VDS.

5.6 Signing certificate

5.6.1 General

The signing certificate is referenced by the CA reference and the certificate ID. Access to the certificate is done using a URI. The URI is a concatenation of the CA's directory URI (defined in 5.2.6), the CA reference and the certificate ID. For example:

```
https://vds.certifio.com/certificates/ca01/09hz.cer
```

The published certificate shall not include the private key and shall conform to Canonical Encoding Rules (CER) in accordance with ISO/IEC 8825-1.

To ensure that all VDSs issued are properly validated, a certificate reference shall not be reused with a different signing certificate or with a different public/private key pair. In addition, the signing certificate shall conform to the TSO's digital signature service in accordance with NIST FIPS 186-5 (DSS) policies.

The signing certificate may support X.509 extensions defined by the TSO to support specific policies. The X509 extensions shall be identified by ISO/IEC 9834-1-compliant OIDs.

5.6.2 Usage list extensions

The usage list extension is a X.509 extension which contains the list of UUIDs used to verify that the certificate is legitimately authorized to sign a VDS for a specific use case (see 5.4.1 for more details).

The list is DER encoded as defined by the following ASN.1 specification:

```
UUID::= OCTET STRING
        -- contains the DER encoded UUID (128-bit long) as defined in the ISO 9834-8
UsageList::= SEQUENCE {
   oid     OBJECT IDENTIFIER
           -- OID defined by the TSO
   uuids   SET OF UUID
}
```

6 Production process

VDSs can be produced using the following process:

— selection of a use case and relevant data;

— encoding of data in the payload section in accordance with the requirements specified in the manifest;

— creation of a header;

— signing of the VDS;

— appending of auxiliary data (optional);

— encoding of the VDS data block (optional, see 5.5.2 for more details);

— prepending the ISO/IEC 15459-2 IAC code;

— affixing the VDS on the object or document.

This process requires the following inputs:

— TSL;

— current date and time;

— manifest;

— payload data;

— auxiliary data (optional);

— certificate from the issuer (and credentials for signing).

7 Verification process

7.1 General concepts

If a TEP cannot interpret a statement defined in the manifest or strictly comply with it, the verification shall fail. This applies to statements within this specification and to extensions defined by other parties, most notably the TSO.

If verification fails, the data contained in the VDS shall not be shown to the user or used in an automated process.

Every signature verification shall be completed in accordance with ETSI TS 102 853, unless specified otherwise by the TSO.

7.2 Acquisition of VDS data

A VDS can be applied on documents, objects or an electronic data set, generally in the form of MRC (including, but not limited to, a printed 2D barcode, a 2D barcode in the form of an electronic image or an RFID tag).

The VDS structured data set is acquired by reading and interpreting one or more MRCs and converted into the functional VDS data blocks: header, payload, signature and (optional) auxiliary data.

7.3 Header structure analysis

The raw data are analysed and converted to Base256 if necessary. If the header does not conform to the header defined in 5.4 (including, but not limited to, the VDS marker byte 0xDE, the reserved bits and the VDS version ID 0x3), the verification shall fail.

Verification shall also fail if the VDS timestamp is after the current time, in coordinated universal time (UTC).

The IAC in the header shall match the IAC preceding the VDS data blocks.

7.4 Reference retrieval and verification

7.4.1 General

To verify a VDS, multiple external resources shall be retrieved and verified. It is possible to perform a local verification (also called an "offline verification") if the references are already cached on the TEP and are still valid at the time of verification, according to the policies defined in the manifest and by the TSO.

7.4.2 TSL

The IAC may be used to locate the TSL.

The TSL shall conform to ETSI TS 119 612, with the exception of the field <SchemeTerritory>, which is optional. The TSL shall also conform to the specifications defined in 5.2 and with the TSL XSD (XML schema definition) published by the TSO. It shall carry a valid CAdES or XAdES-B signature from the TSO and the TSO's signature certificate shall be valid. The signature shall conform to the TSO's required parameters and characteristics.

An example of TSL XSD is given in Annex D.

The CA reference present in the VDS header shall match the extension attribute <VDSAuthorityID> of a TSP's CA defined in the TSL. The CA shall have been valid at the time defined in the timestamp of the VDS header, with the attribute ServiceStatus set to "granted" in the TSL at that time (e.g. https://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/granted).

In the event the TSO does not specify a TSL extensions for a historical ServiceStatus list, the CA shall currently be valid in the TSL, and the <StatusStartingTime> shall be prior to the timestamp in the VDS header.

7.4.3 Manifest

The manifest shall conform to the specifications defined in 5.3 and with the manifest XSD published by the TSO. It shall carry a valid CAdES or XAdES-B signature from the TSO and the TSO's signature certificate shall also be valid. The signature shall conform to the TSO's required parameters and characteristics.

An example of manifest XSD is given in Annex E.

The manifest ID in the XML file retrieved shall match the manifest ID in the VDS header.

7.4.4 Signing certificate and certificate revocation list

The signing certificate shall be signed by the CA's public certificate listed in the TSL, valid in accordance with ETSI TS 102 853, and conform to the TSO's digital signature service in accordance with NIST FIPS 186-5 (DSS) policies. Some of those policies, and well as some validation parameters in ETSI TS 102 853, can be superseded by TSO-defined extensions, including, but not limited to, the CRL or online certificate status protocol (OCSP) response validity period, the assessment of the signature time and the URI to determine the certificate's revocation status.

The signing certificate's validation shall use the time of signature present in the timestamp field defined in the VDS header. However, since the timestamp in the VDS header cannot be considered reliable in the event the certificate is revoked or expired, a second validity test shall be performed using the present time as the time of signature, unless an extension indicates otherwise.

7.5 Payload processing

Verify that the payload data conform to the field constraints defined in the manifest.

7.6 Extensions processing

Verify conformance with each extension listed in the XML manifest in 7.4, if applicable.

7.7 Signature verification

Verify that the signature present in the VDS is valid, based on the specifications defined in 5.5.5 and using the signing certificate retrieved in 7.4.4.

7.8 Document data presentation

If all verifications are successful, present the information to the user or the automated system, strictly respecting an RFF(s) as defined in the manifest or by the TSO. When appropriate, present the information adapted to the user's language and guidelines established by the experts that have defined a given use case.

Annex A
(informative)

VDS encoding example

Tables A.1 to A.4 give a byte-level example of a VDS, based on the manifest documented in Annex C.

Table A.1 — VDS header section

Pos	Bytes	Description	Data encoding
0	1	VDS marker	0xDE
1	1	Payload length field type = uint16, version ID = 3	0x03
2	2	IAC ref="XXX"	0xED2E
4	6	Certificate ref: CA ref="FR99", Cert ID="09HZ"	0x7BA651EE895D
10	3	Manifest ID = 0x89AB01	0x89AB01
13	4	Timestamp = 2019-07-14 00:00:00 UTC	0x5D2A7080
17	2	Payload length	0x0058

Table A.2 — Payload section (message pack encoding)

Pos	Bytes	Field name	Value	msgpack format	Format encoding	Data encoding
19	12	stringSimple[a]	Évaluation	fixstr11	0xAB	0xC38976616C756174696F6E
31	13	stringPattern	A1b2C3d4E5f6	fixstr12	0xAC	0x413162324333643445356636
44	3	stringPattern2	en	fixstr2	0xA2	0x656E
47	9	stringC40	A1B2C3D4E5F6	fixstr8	0xA8	0x585828086B933B43
56	1	stringNil	*n.d.*	nil	0xC0	
57	1	stringArray Example		fixarray2	0x92	
58	7	[0]	A1b2C3	fixstr6	0xA6	0x413162324333
65	5	[1]	d4E5	fixstr4	0xA4	0x64344535
70	2	intExample	200	uint8	0xCC	0xC8
72	2	dateExample	2019-09-10	uint16	0xCD	0xFC
74	1	objectExample		fixarray2	0x92	
75	11	string	Evaluation	fixstr10	0xAA	0x4576616C756174696F6E
86	1	bool	TRUE	Bool-True	0xC3	
87	1	objectArray Example		fixarray3	0x93	
88	1	[0]		fixarray2	0x92	
89	6	string	aBcDe	fixstr5	0xA5	0x6142634465
95	3	date	2019-09-10	uint16	0xCD	0x1C18
98	1	[1]		fixarray2	0x92	
99	1	string	*n.d.*	nil	0xC0	
100	1	date	*n.d.*	nil	0xC0	
101	1	[2]		fixarray2	0x92	

[a] The field "stringSimple" has the character "É" encoded in UTF-8 in 2 bytes. The length of the encoded string reflects the actual length in bytes.

Table A.2 *(continued)*

Pos	Bytes	Field name	Value	msgpack format	Format encoding	Data encoding
102	4	string	aBc	fixarray3	0xA3	0x614263
106	1	date	1999-12-31	fix-numNeg1	0xFF	

[a] The field "stringSimple" has the character "É" encoded in UTF-8 in 2 bytes. The length of the encoded string reflects the actual length in bytes.

Table A.3 — Signature section (raw)

Pos	Bytes	Value (assuming a P-256 certificate)
107	64	0x934ff8d7a19bdd61df430c9a6a4bba14a42cdeb83e4715a2471ebef3a55b9c70 d84473e00453b177ef494e9f0b0e39c4f55e591694170e56a736764372fa0b70

Table A.4 — AuxData section (message pack encoding)

Pos	Bytes	Field name	Value	msgpack format	Format encoding	Data encoding
171	5	intExampleAuxData	90210	uint32	0xCE	0x00016062

Annex B
(informative)

TSL example

This annex presents a simple example of a TSL with the extensions referenced in this specification.

```
<?xml version="1.0" encoding="UTF-8"?><TrustServiceStatusList xmlns="http://uri.etsi.
org/02231/v2#" xmlns:ns2="https://www.w3.org/2000/09/xmldsig#" xmlns:vdsext="https://
otentik.codes" TSLTag = "https://uri.etsi.org/02231/TSLTag" >
  <SchemeInformation>
    <TSLVersionIdentifier>5</TSLVersionIdentifier>
    <TSLSequenceNumber>1</TSLSequenceNumber>
    <TSLType>
      https://uri.etsi.org/TrstSvc/TrustedList/TSLType/EUgeneric/
    </TSLType>
    <SchemeOperatorName>
      <Name xml:lang="en">Otentik Network</Name>
      <Name xml:lang="fr">Réseau Otentik</Name>
    </SchemeOperatorName>
    <SchemeOperatorAddress>
      <PostalAddresses>
        <PostalAddress xml:lang="fr">
          <StreetAddress>15 rue Dulac</StreetAddress>
          <Locality>Paris</Locality>
          <PostalCode>75015</PostalCode>
          <CountryName>FR</CountryName>
        </PostalAddress>
      </PostalAddresses>
      <ElectronicAddress>
        <URI xml:lang="en">mailto:contact@aigcev.org</URI>
        <URI xml:lang="fr">mailto:contact@aigcev.org</URI>
      </ElectronicAddress>
    </SchemeOperatorAddress>
    <SchemeName>
      <Name xml:lang="en">Otentik</Name>
    </SchemeName>
    <SchemeInformationURI>
      <URI xml:lang="en">https://otentik.codes</URI>
    </SchemeInformationURI>
    <StatusDeterminationApproach>
      https://uri.etsi.org/TrstSvc/TrustedList/StatusDetn/EUappropriate
    </StatusDeterminationApproach>
    <SchemeTypeCommunityRules>
      <URI xml:lang="en">
       https://uri.etsi.org/TrstSvc/TrustedList/schemerules/EUcommon
      </URI>
    </SchemeTypeCommunityRules>
    <PolicyOrLegalNotice>
      <TSLLegalNotice xml:lang="en">
        The Otentik Network trust list is governed by the
        Visible Digital Seal International Council (VDSIC).
        Details at https://otentik.codes
      </TSLLegalNotice>
      <TSLLegalNotice xml:lang="fr">
        La liste de confiance du réseau Otentik est régie par
        l'Association Internationale de Gouvernance du
        Cachet Électronique Visible (AIGCEV).
        Plus de détails à https://otentik.codes/fr/
      </TSLLegalNotice>
    </PolicyOrLegalNotice>
    <HistoricalInformationPeriod>0</HistoricalInformationPeriod>
    <ListIssueDateTime>2019-09-01T00:00:00.000Z</ListIssueDateTime>
    <NextUpdate>2019-12-01T00:00:00.000Z</NextUpdate>
    <SchemeExtensions>
```

```
    <Extension Critical="true">
      <vdsext:VDSManifestResource>
        https://manifest.otentik.codes/
      </vdsext:VDSManifestResource>
    </Extension>
  </SchemeExtensions>
</SchemeInformation>
<TrustServiceProviderList>
  <TrustServiceProvider>
    <TSPInformation>
      <TSPName>
        <Name xml:lang="en">Certigna</Name>
        <Name xml:lang="fr">Certigna</Name>
      </TSPName>
      <TSPTradeName>
        <Name xml:lang="en">Certigna</tsl:Name>
        <Name xml:lang="fr">Certigna</tsl:Name>
      <TSPTradeName>
      <TSPAddress>
        <PostalAddresses>
          <PostalAddress xml:lang="fr">
            <StreetAddress>20 allée de la râperie</StreetAddress>
            <Locality>Villeneuve d'Ascq</Locality>
            <CountryName>FR</CountryName>
          </PostalAddress>
        </PostalAddresses>
        <ElectronicAddress>
          <URI xml:lang="en">mailto:contact@certigna.com</URI>
          <URI xml:lang="fr">mailto:contact@certigna.com</URI>
        </ElectronicAddress>
      </TSPAddress>
      <TSPInformationURI>
        <URI xml:lang="fr">https://www.certigna.com/&#x003C;/URI>
      </TSPInformationURI>
    </TSPInformation>
    <TSPServices>
      <TSPService>
        <ServiceInformation>
          <ServiceTypeIdentifier>
            https://uri.etsi.org/TrstSvc/Svctype/CA/PKC
          </ServiceTypeIdentifier>
          <ServiceDigitalIdentity>
            <DigitalId>
              <X509Certificate>
                (removed to facilitate readability of this example)
              </X509Certificate>
            </DigitalId>
          </ServiceDigitalIdentity>
          <ServiceStatus>
            https://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/granted
          </ServiceStatus>
          <StatusStartingTime>
            2015-07-16T22:00:00.000Z
          </StatusStartingTime>
          <SchemeServiceDefinitionURI>
            <URI xml:lang="en">
              https://uri.etsi.org/TrstSvc/TrustedList/schemerules/otentik
            </URI>
            <URI xml:lang="fr">
              https://uri.etsi.org/TrstSvc/TrustedList/schemerules/otentik
            </URI>
          </SchemeServiceDefinitionURI>
          <ServiceInformationExtensions>
            <Extension Critical="true">
              <vdsext:VDSAuthorityID>FR03</vdsext:VDSAuthorityID>
              <vdsext:VDSCertResource>
                https://certificates.certigna.fr/certificates/fr03/
              </vdsext:VDSCertResource>
            </Extension>
          </ServiceInformationExtensions>
        </ServiceInformation>
```

```
      </TSPService>
    </TSPServices>
  </TrustServiceProvider>
  <TrustServiceProvider>
    <TSPInformation>
      <TSPName>
        <Name xml:lang="en">Notarius Solutions inc.</Name>
        <Name xml:lang="fr">Solutions Notarius inc.</Name>
      </TSPName>
      <TSPTradeName>
        <Name xml:lang="en">Notarius</tsl:Name>
        <Name xml:lang="fr">Notarius</tsl:Name>
      <TSPTradeName>
      <TSPAddress>
        <PostalAddresses>
          <PostalAddress xml:lang="en">
            <StreetAddress>465 McGill St., suite 300</StreetAddress>
            <Locality>Montreal</Locality>
            <StateOrProvince>QC</StateOrProvince>
            <CountryName>CA</CountryName>
          </PostalAddress>
          <PostalAddress xml:lang="fr">
            <StreetAddress>465 rue McGill, bureau 300</StreetAddress>
            <Locality>Montréal</Locality>
            <StateOrProvince>QC</StateOrProvince>
            <CountryName>CA</CountryName>
          </PostalAddress>
        </PostalAddresses>
        <ElectronicAddress>
          <URI xml:lang="en">mailto:support@notarius.com</URI>
          <URI xml:lang="fr">mailto:support@notarius.com</URI>
        </ElectronicAddress>
      </TSPAddress>
      <TSPInformationURI>
        <URI xml:lang="en">https://notarius.com/en/&#x003C;/URI>
        <URI xml:lang="fr">https://notarius.com/&#x003C;/URI>
      </TSPInformationURI>
    </TSPInformation>
    <TSPServices>
      <TSPService>
        <ServiceInformation>
          <ServiceTypeIdentifier>
            https://uri.etsi.org/TrstSvc/Svctype/CA/PKC
          </ServiceTypeIdentifier>
          <ServiceDigitalIdentity>
            <DigitalId>
              <X509Certificate>
                (removed to facilitate readability of this example)
              </X509Certificate>
            </DigitalId>
          </ServiceDigitalIdentity>
          <ServiceStatus>
            https://uri.etsi.org/TrstSvc/TrustedList/Svcstatus/granted
          </ServiceStatus>
          <StatusStartingTime>
            2019-02-26T00:00:00.000Z
          </StatusStartingTime>
          <SchemeServiceDefinitionURI>
            <URI xml:lang="en">
              https://uri.etsi.org/TrstSvc/TrustedList/schemerules/otentik
            </URI>
            <URI xml:lang="fr">
              https://uri.etsi.org/TrstSvc/TrustedList/schemerules/otentik
            </URI>
          </SchemeServiceDefinitionURI>
          <ServiceInformationExtensions>
            <Extension Critical="true">
              <vdsext:VDSAuthorityID>CA01</vdsext:VDSAuthorityID>
              <vdsext:VDSCertResource>
                https://vds.certifio.com/certificates/ca01/
              </vdsext:VDSCertResource>
```

```
              </Extension>
            </ServiceInformationExtensions>
          </ServiceInformation>
        </TSPService>
      </TSPServices>
    </TrustServiceProvider>
  </TrustServiceProviderList>
  <ds:Signature xmlns:ds="https://www.w3.org/2000/09/xmldsig#" Id="id-407b823ae15c8125">
    (removed to facilitate readability of this example)
  </ds:Signature>
</TrustServiceStatusList>
```

Annex C
(informative)

Manifest example

This annex presents an example of a manifest, with various examples of fields and types.

```
<?xml version="1.0" encoding="UTF-8"?>
<Manifest version="1" xmlns="http://otentik.codes/" xmlns:ext="http://otentik.
codes/extensions/" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://otentik.codes/xsd/otentik-base-1.0.xsd" >
  <Id>89AB01</Id>
  <Version>1</Version>
 <Name>
    <Value xml:lang="en">Name of use case</Value>
    <Value xml:lang="fr">Nom du cas d'usage</Value>
  </Name>
  <Description>
    <Value xml:lang="en">English use case's description</Value>
    <Value xml:lang="fr">Description française du cas d'usage</Value>
  </Description>
  <Schema>
    <Payload>
      <Fields>
        <String name="stringSimple"/>
        <String name="stringPattern">
          <StringConstraints>
            <Nillable/>
            <MaxLength>12</MaxLength>
            <Pattern>[A-Za-z0-9]</Pattern>
          </StringConstraints>
        </String>
        <String name="stringPattern2">
          <StringConstraints>
            <Pattern>^(en|fr)$</Pattern>
          </StringConstraints>
        </String>
        <String name="stringC40">
          <StringConstraints>
            <Nillable/>
            <Pattern>[A-Z0-9]</Pattern>
            <Encoding>C40</Encoding>
          </StringConstraints>
        </String>
        <String name="stringNil">
          <StringConstraints>
            <Nillable/>
          </StringConstraints>
        </String>
        <StringArray name="stringArrayExample">
          <ArrayConstraints>
            <MinSize>1</MinSize>
            <MaxSize>3</MaxSize>
          </ArrayConstraints>
          <StringConstraints>
            <Nillable/>
            <Pattern>[A-Za-z0-9]</Pattern>
          </StringConstraints>
        </StringArray>
        <Integer name="intExample">
          <IntegerConstraints>
            <Nillable/>
            <Min>1</Min>
            <Max>9999</Max>
          </IntegerConstraints>
```

```
        </Integer>
        <Date name="dateExample">
          <DateConstraints>
            <Nillable/>
            <From>2019-01-01</From>
          </DateConstraints>
        </Date>
        <Object name="objectExample" type="object1">
          <ObjectConstraints>
            <Nillable/>
          </ObjectConstraints>
        </Object>
        <ObjectArray name="objectArrayExample" type="object2">
          <ArrayConstraints>
            <MinSize>1</MinSize>
            <MaxSize>3</MaxSize>
          </ArrayConstraints>
          <ObjectConstraints>
            <Nillable/>
          </ObjectConstraints>
        </ObjectArray>
      </Fields>
    </Payload>
    <AuxData>
      <Fields>
        <Integer name="intExampleAuxData">
          <IntegerConstraints>
            <Nillable/>
            <Min>-1</Min>
            <Max>999999</Max>
          </IntegerConstraints>
        </Integer>
      </Fields>
    </AuxData>
    <Types>
      <Type name="object1">
        <Fields>
          <String name="string">
            <StringConstraints>
              <Nillable/>
            </StringConstraints>
          </String>
          <Boolean name="bool">
            <BooleanConstraints>
              <Nillable/>
            </BooleanConstraints>
          </Boolean>
        </Fields>
      </Type>
      <Type name="object2">
        <Fields>
          <String name="string">
            <StringConstraints>
              <Nillable/>
              <Pattern>[A-Za-z]</Pattern>
            </StringConstraints>
          </String>
          <Date name="date">
            <DateConstraints>
              <Nillable/>
              <From>2000-01-01</From>
            </DateConstraints>
          </Date>
        </Fields>
      </Type>
    </Types>
  </Schema>
  <Extensions/>
</Manifest>
```

Annex D
(informative)

TSL XML schema definition example

This annex presents an example of the TSL XSD specific to this specification to automate verification. Another XSD provided by the TSO is needed to automate conformance with ETSI TS 119 612.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://otentik.codes/" xmlns:xsd="https://www.w3.org/2001/
XMLSchema" >
  <xsd:simpleType name="VDSAuthorityID">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9A-Z]{4}" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="VDSCertResource">
    <xsd:restriction base="xsd:anyURI" />
  </xsd:simpleType>
  <xsd:simpleType name="VDSManifestResource">
    <xsd:restriction base="xsd:anyURI" />
  </xsd:simpleType>
</xsd:schema>
```

Annex E
(informative)

Manifest XML schema definition example

This annex presents an example of the manifest XSD to automate verification. It should also serve as a guide when declaring field types.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns="http://otentik.codes/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://otentik.codes/">
  <xs:import namespace="https://www.w3.org/XML/1998/namespace" />
  <xs:element name="Manifest" type="ManifestType">
    <xs:annotation>
      <xs:documentation>Root element for a Use Case descriptor.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="Id">
    <xs:annotation>
      <xs:documentation>
        Use Case unique identifier. Restriction: 4 chars hexadecimal.
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="[0-9a-fA-F]{4}" />
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="Version">
    <xs:annotation>
      <xs:documentation>
        Use Case version. To be incremented when a non-breaking change is introduced.
        Restriction: 1-255
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="1" />
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
 <xs:element name="Name" type="NameType">
    <xs:annotation>
      <xs:documentation>Use Case name. Multilingual.
        Restriction: At least 1 name must be defined.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="Description" type="DescriptionType">
    <xs:annotation>
      <xs:documentation>Use Case description. Multilingual.
        Restriction: At least 1 description must be defined.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="Schema" type="SchemaType">
    <xs:annotation>
      <xs:documentation>Use Case data schema. Used to decode a VDS payload.
        Restriction: Exactly 1 schema MUST be defined.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="Extensions" type="ExtensionsType">
    <xs:annotation>
      <xs:documentation>
        Defines an extension point where new features (proprietary or not) can be added.
```

```
      Default extensions are:
      - Policies: Validation policies
      - Presentation views: Templates used to render VDS content
    </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:complexType name="ValueType">
  <xs:annotation>
    <xs:documentation>
      Element used to define a translatable value. Translation is based on the
      xml:lang provide, ie a 2 letter language code.
    </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute ref="xml:lang" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name="NameType">
  <xs:sequence>
    <xs:element type="ValueType" name="Value" maxOccurs="unbounded" minOccurs="1" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="DescriptionType">
  <xs:sequence>
    <xs:element type="ValueType" name="Value" maxOccurs="unbounded" minOccurs="1" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="BaseConstraintsType" abstract="true">
  <xs:sequence>
    <xs:element type="xs:string" name="Nillable" nillable="true" minOccurs="0">
      <xs:annotation>
        <xs:documentation>
          Sets an element as Nillable. By default, an element is not nillable.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="StringConstraintsType">
  <xs:annotation>
    <xs:documentation>Sets constraints on a String field.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="BaseConstraintsType">
      <xs:sequence>
        <xs:element type="xs:unsignedInt" name="MinLength" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Minimum string length. Optional. Values from 0 to 4,294,967,295.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element type="xs:unsignedInt" name="MaxLength" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Maximum string length. Optional. Values from 0 to 4,294,967,295.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element type="xs:string" name="Pattern" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Allowed string pattern. Supports JavaScript RegEx. Optional.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="Encoding" minOccurs="0">
          <xs:annotation>
```

```
              <xs:documentation>
                The field value encoding. If specified, the restrictions are
                validated against the decoded value.
              </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="BASE64" />
                <xs:enumeration value="C40" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="TypeDef" abstract="true">
    <xs:annotation>
      <xs:documentation>Base entity for a schema field.
              </xs:documentation>
    </xs:annotation>
    <xs:attribute type="xs:string" name="name" use="required">
      <xs:annotation>
        <xs:documentation>Field name. Required
              </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
  <xs:complexType name="StringType">
    <xs:annotation>
      <xs:documentation>
        Field representing a sequence of characters.
        Restrictions can be applied using StringConStraints element.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="TypeDef">
        <xs:sequence>
          <xs:element type="StringConstraintsType" name="StringConstraints" minOccurs="0">
            <xs:annotation>
              <xs:documentation>Value restrictions. Optional.
              </xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="BooleanConstraintsType">
    <xs:annotation>
      <xs:documentation>Boolean field restrictions.
              </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="BaseConstraintsType" />
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="BooleanType">
    <xs:annotation>
      <xs:documentation>
        Boolean field type.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="TypeDef">
        <xs:sequence>
          <xs:element type="BooleanConstraintsType" name="BooleanConstraints"
minOccurs="0">
            <xs:annotation>
              <xs:documentation>
                Boolean field restrictions.
```

```
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="IntegerConstraintsType">
  <xs:annotation>
    <xs:documentation>Integer field restrictions.
            </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="BaseConstraintsType">
      <xs:sequence>
        <xs:element type="xs:long" name="Min" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Minimum value. Optional.
Values from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element type="xs:long" name="Max" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Maximum value. Optional.
Values from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="IntegerType">
  <xs:annotation>
    <xs:documentation>Integer type field.
            </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="TypeDef">
      <xs:sequence>
        <xs:element type="IntegerConstraintsType" name="IntegerConstraints"
minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Integer field restrictions. Optional.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="FloatConstraintsType">
  <xs:annotation>
    <xs:documentation>
      Float field restrictions.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="BaseConstraintsType">
      <xs:sequence>
        <xs:element type="xs:decimal" name="Min" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Minimum value. Optional.
              The decimal separator is always a point (.), and no separation at the
              thousand mark may be added. There is no support for scientific notation.
              Allows any number of insignificant leading and trailing zeros
              (after the decimal point).
```