Digital Signature Service Metadata
Version 1.0

Committee Specification Draft 02/
Public Review Draft 02

20 September 11 December 2019

This stage:
https://docs.oasis-open.org/dss-x/dss-md/v1.0/cs02/dss-md-v1.0-cs02.docx (Authoritative)
https://docs.oasis-open.org/dss-x/dss-md/v1.0/cs02/dss-md-v1.0-cs02.html
https://docs.oasis-open.org/dss-x/dss-md/v1.0/cs02/dss-md-v1.0-cs02.pdf

Previous stage:
https://docs.oasis-open.org/dss-x/dss-md/v1.0/csprd02/dss-md-v1.0-csprd02.docx (Authoritative)
https://docs.oasis-open.org/dss-x/dss-md/v1.0/csprd02/dss-md-v1.0-csprd02.html
https://docs.oasis-open.org/dss-x/dss-md/v1.0/csprd02/dss-md-v1.0-csprd02.pdf

Previous version:
https://docs.oasis-open.org/dss-x/dss-md/v1.0/dss-md-v1.0.docx (Authoritative)
https://docs.oasis-open.org/dss-x/dss-md/v1.0/dss-md-v1.0.html
https://docs.oasis-open.org/dss-x/dss-md/v1.0/dss-md-v1.0.pdf

Technical Committee:
OASIS Digital Signature Services eXtended (DSS-X) TC

Chairs:
Andreas Kuehne (kuehne@trustable.de), Individual
Ernst Jan van Nigtevecht (EJvN@Sonnenglanz.net), Sonnenglanz Consulting

Editors:
Detlef Hühnlein (detlef.huehnlein@ecsec.de), Individual
Andreas Kuehne (kuehne@trustable.de), Individual

Additional artifacts:
This prose specification is one component of a Work Product that also includes:
• JSON and XML schemas: https://docs.oasis-open.org/dss-x/dss-md/v1.0/cs02/schema/

Related work:
This specification is related to:
• Digital Signature Service Core Protocols, Elements, and Bindings Version 2.0. Edited by Andreas Kuehne and Stefan Hagen. Latest version: https://docs.oasis-open.org/dss-x/dss-core/v2.0/dss-core-v2.0.html.

Declared XML namespace:
• http://docs.oasis-open.org/dss-x/ns/metadata

Abstract:
This document defines JSON and XML structures and discovery mechanisms for metadata related to digital signature services.
Status:
This document was last revised or approved by the OASIS Digital Signature Services eXtended (DSS-X) TC on the above date. The level of approval is also listed above. Check the “Latest version stage” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dss-x#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/dss-x/.

This specification is provided under the RF on Limited Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC’s web page (https://www.oasis-open.org/committees/dss-x/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:
When referencing this specification, the following citation format should be used:

[DSS-MD-v1.0]

Notices

Copyright © OASIS Open 2019. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the “OASIS IPR Policy”). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an “AS IS” basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.
Table of Contents

1 Introduction .......................................................................................................................... 5
  1.1 IPR Policy ....................................................................................................................... 5
  1.2 Terminology .................................................................................................................... 5
    1.2.1 Terms and Definitions ............................................................................................. 5
    1.2.2 Abbreviated Terms ................................................................................................. 5
  1.3 Normative References ...................................................................................................... 5
  1.4 Non-Normative References ............................................................................................ 6
  1.5 Typographical Conventions ........................................................................................... 7
  1.6 Motivation and related work (Non-normative) ............................................................... 7

2 Overview ................................................................................................................................ 8

3 Data Structure Models ......................................................................................................... 9
  3.1 Data Structure Models defined in this document ............................................................. 9
    3.1.1 Component Provider .................................................................................................. 9
      3.1.1.1 Provider – JSON Syntax .................................................................................... 9
      3.1.1.2 Provider – XML Syntax ................................................................................... 11
    3.1.2 Component Protocol .................................................................................................. 11
      3.1.2.1 Protocol – JSON Syntax .................................................................................. 12
      3.1.2.2 Protocol – XML Syntax ................................................................................... 13
    3.1.3 Component Profile ..................................................................................................... 13
      3.1.3.1 Profile – JSON Syntax ..................................................................................... 14
      3.1.3.2 Profile – XML Syntax ..................................................................................... 15
    3.1.4 Component Operation ............................................................................................... 15
      3.1.4.1 Operation – JSON Syntax ............................................................................... 16
      3.1.4.2 Operation – XML Syntax ............................................................................... 17
    3.1.5 Component Parameter ............................................................................................... 18
      3.1.5.1 Parameter – JSON Syntax .............................................................................. 18
      3.1.5.2 Parameter – XML Syntax .............................................................................. 19
    3.1.6 Component Format ..................................................................................................... 20
      3.1.6.1 Format – JSON Syntax .................................................................................... 20
      3.1.6.2 Format – XML Syntax .................................................................................... 21
    3.1.7 Component Policy ...................................................................................................... 22
      3.1.7.1 Policy – JSON Syntax ..................................................................................... 22
      3.1.7.2 Policy – XML Syntax ..................................................................................... 23
    3.1.8 Component PolicyByRef ............................................................................................. 23
      3.1.8.1 PolicyByRef – JSON Syntax .......................................................................... 24
      3.1.8.2 PolicyByRef – XML Syntax .......................................................................... 24
    3.1.9 Component Extension ............................................................................................... 24
      3.1.9.1 Extension – JSON Syntax .............................................................................. 25
      3.1.9.2 Extension – XML Syntax .............................................................................. 25
  3.2 Element / JSON name lookup tables ............................................................................... 26

4 Metadata Discovery .............................................................................................................. 29

Appendix A. Acknowledgments ............................................................................................... 30
Appendix B. List of Figures ....................................................................................................... 31
Appendix C. Revision History .................................................................................................. 32
1 Introduction

[All text is normative unless otherwise labeled]

1.1 IPR Policy

This specification is provided under the RF on Limited Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC’s web page (https://www.oasis-open.org/committees/dss-x/ipr.php).

1.2 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119] and [RFC8174].

1.2.1 Terms and Definitions

For the purposes of this document no specific terms or definitions have been identified as deviating from the usual meaning in the context of XML / JSON schema, digital signatures or transport.

1.2.2 Abbreviated Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>URI</td>
<td>(IETF) Uniform Resource Identifier according to [RFC3986]</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>XML</td>
<td>(W3C) Extensible Markup Language</td>
</tr>
<tr>
<td>XSD</td>
<td>(W3C) XML Schema</td>
</tr>
</tbody>
</table>

1.3 Normative References


1.4 Non-Normative References


[CSC-v1.0] Cloud Signature Consortium, “Architectures and protocols for remote signature applications”, Published version 1.0.3.0, 2018


[TS119512] ETSI, “Electronic Signatures and Infrastructures (ESI); Protocols for trust service providers providing long-term data preservation services”, Draft ETSI TS 119 512, V0.0.9 (2019-05).

1.5 Typographical Conventions

Keywords defined by this specification use this monospaced font.

Normative source code uses this paragraph style.

Text following the special symbol («) – an opening Guillemet (or French quotation mark) – within this specification identifies automatically testable requirements to aid assertion tools. Every such statement is separated from the following text with the special end symbol (») – a closing Guillemet and has been assigned a reference that follows that end symbol in one of the three patterns:

1. [DSS-section#-local#] if it applies regardless of syntax
2. [JDSS-section#-local#] if it applies only to JSON syntax
3. [XDSS-section#-local#] if it applies only to XML syntax

Some sections of this specification are illustrated with non-normative examples.

Example 1: text describing an example uses this paragraph style

```
Non-normative examples use this paragraph style.
```

All examples in this document are non-normative and informative only.

Representation-specific text is indented and marked with vertical lines.

**Representation-Specific Headline**

Normative representation-specific text

All other text is normative unless otherwise labelled e.g. like:

**Non-normative Comment:**

This is a pure informative comment that may be present, because the information conveyed is deemed useful advice or common pitfalls learned from implementer or operator experience and often given including the rationale.

1.6 Motivation and related work (Non-normative)

Based on existing [DSS-v1.0] and emerging [DSS-v2.0] standards for digital signature services as well as the [eIDAS] regulation on electronic identification and trust services, there is a growing ecosystem consisting of providers and consumers of a variety of digital signature related services, which raises the demand for a normalised discovery and provision of service-related metadata.

While there are already standards for the handling of service-related metadata for services for exchanging business documents (see [BDX-SMP-v1.0] and [BDX-SMP-v2.0]) or identity management services (see [SAML-MD], [RFC8414] and [OIDC-MD]), there is currently no comprehensive metadata standard for digital signature services, but only first steps towards filling this gap (see [CSC-v1.0], [TS119432] and [TS119512]).

Against this background, the present document aims at providing a generic and extensible structure (see clauses 2 and 3) and simple discovery mechanism (see clause 4) for digital signature service-related metadata, which is intended to be used in conjunction with [DSS-v2.0] and related profiles and extensions, such as [TS119432], [TS119442] and [TS119512] for example.
2 Overview

As depicted in Figure 1, the main components of the service-related metadata structure specified in the present document comprise Provider, Protocol, Profile, Operation and Policy.

![Diagram of main components within the service-related metadata structures]

Figure 1: Overview of main components within the service-related metadata structures

The main component is the Provider element (see clause 3.1.1), which contains general metadata related to the provider of the service(s). As a service provider may support one or more protocols, for signature generation, signature validation or long-term preservation for example, and a provider may support different profiles of the supported protocols, the Provider element may contain one or more Protocol elements (see clause 3.1.2), which in turn may contain one or more Profile elements (see clause 3.1.3), which describe the supported profiles. A Profile element may in turn contain among other elements one or more Operation elements (see clause 3.1.4) and zero or more Policy elements (see 3.1.7), in which the applicable policies are specified or referenced.
3 Data Structure Models

3.1 Data Structure Models defined in this document

The XML elements of this section are defined in the XML namespace 'http://docs.oasis-open.org/dss-x/ns/metadata'.

3.1.1 Component Provider

The component Provider is the main element of the metadata structure and contains information about the provider of the digital signature services. The structure of this component has been inspired by the content provided by the info call defined in [CSC-v1.0].

Below follows a list of the sub-components that constitute this component:

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>name</td>
</tr>
<tr>
<td>Logo</td>
<td>logo</td>
</tr>
</tbody>
</table>

3.1.1.1 Provider – JSON Syntax

The ProviderType JSON object SHALL implement in JSON syntax the requirements defined in the Provider component.

Properties of the JSON object SHALL implement the sub-components of Provider using JSON-specific names mapped as shown in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>name</td>
</tr>
<tr>
<td>Logo</td>
<td>logo</td>
</tr>
</tbody>
</table>
The `ProviderType` JSON object is defined in the JSON schema [DSSMD-JSON] and is provided below as a service to the reader.

```json
"md-ProviderType": {
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "logo": {
      "type": "string"
    },
    "region": {
      "type": "string"
    },
    "lang": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "description": {
      "type": "array",
      "items": {"$ref": "/definitions/dsb-InternationalStringType"}
    },
    "authinfo": {
      "type": "string"
    },
    "protocol": {
      "type": "array",
      "items": {
        "$ref": "/definitions/md-ProtocolType"
      }
    },
    "ext": {
      "type": "array",
      "items": {
        "$ref": "/definitions/md-ExtensionType"
      }
    }
  },
  "required": ["name", "logo", "region", "protocol"]
}
```
3.1.1.2 Provider – XML Syntax

The XML type ProviderType SHALL implement the requirements defined in the Provider component. The ProviderType XML element is defined in XML Schema [DSSMD-XML], and is copied below for information.

```xml
<xs:complexType name="ProviderType">
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Logo" type="xs:anyURI"/>
    <xs:element name="Region" type="xs:string"/>
    <xs:element name="SupportedLanguage" type="xs:language" maxOccurs="unbounded" minOccurs="0"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="md:Description"/>
    <xs:element name="AuthInfo" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
    <xs:element name="Protocol" type="md:ProtocolType" maxOccurs="unbounded" minOccurs="1"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="md:Extension"/>
  </xs:sequence>
</xs:complexType>
```

Each child element of ProviderType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name.

3.1.2 Component Protocol

The Protocol component is part of the Provider component specified in clause 3.1.1 and provides information about a digital signature related protocol supported by the service provider. Below follows a list of the sub-components that constitute this component:
The OPTIONAL `Server` element, if present, MUST contain one instance of a URI, which SHOULD be the URL of the target host of the service supporting the protocol. For REST-based services this is the URL of `Server Object` component within [OpenAPI] and for SOAP-based services this is the `soap:address` within [WSDL].

The OPTIONAL `Specification` element, if present, MAY occur zero or more times containing a URI, which points to a specification document describing the digital signature related protocol. Examples of digital signature related protocols include the generation [DSS-v1.0, DSS-v2.0, TS119432], validation [TS119442] and preservation [TS119512] of digital signatures.

The OPTIONAL `Version` element, if present, MUST contain a string, which indicates the version of the protocol specification, if it is not specified within the specification document mentioned above.

The OPTIONAL `Description` element, if present, MAY occur zero or more times containing a sub-component, which provides additional information with respect to the supported protocol. If present, each instance MUST satisfy the requirements specified in [DSS-v2.0] for the `InternationalString` component.

The `Profile` element MUST occur 1 or more times containing a sub-component, which further describes the specific profile of the supported digital signature related protocol. Each instance MUST satisfy the requirements specified in this document in section 3.1.3.

The OPTIONAL `Extension` element, if present, MAY occur zero or more times containing a sub-component, which extends the semantics of the `Protocol` component. If present each instance MUST satisfy the requirements specified in this document in section 3.1.9.

### 3.1.2.1 Protocol – JSON Syntax

The `ProtocolType` JSON object SHALL implement in JSON syntax the requirements defined in the `Protocol` component.

Properties of the JSON object SHALL implement the sub-components of `Protocol` using JSON-specific names mapped as shown in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td><code>srv</code></td>
</tr>
<tr>
<td>Specification</td>
<td><code>spec</code></td>
</tr>
<tr>
<td>Version</td>
<td><code>version</code></td>
</tr>
<tr>
<td>Description</td>
<td><code>description</code></td>
</tr>
<tr>
<td>Profile</td>
<td><code>profile</code></td>
</tr>
<tr>
<td>Extension</td>
<td><code>ext</code></td>
</tr>
</tbody>
</table>

The `ProtocolType` JSON object is defined in the JSON schema [DSSMD-JS0N] and is provided below as a service to the reader.

```json
"md-ProtocolType": {
  "type": "object",
  "properties": {
    "srv": {
      "type": "string"
    },
    "spec": {
      "type": "array",
      "items": {
```
3.1.2.2 Protocol – XML Syntax

The XML type ProtocolType SHALL implement the requirements defined in the Protocol component. The ProtocolType XML element is defined in XML Schema [DSSMD-XML], and is copied below for information.

```xml
<xs:complexType name="ProtocolType">
    <xs:sequence>
        <xs:element name="Server" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
        <xs:element name="Specification" type="xs:anyURI" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element name="Version" type="xs:string" maxOccurs="1" minOccurs="0"/>
        <xs:element name="Profile" type="md:ProfileType" maxOccurs="unbounded" minOccurs="1"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="md:Description"/>
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="md:Extension"/>
    </xs:sequence>
</xs:complexType>
```

Each child element of ProtocolType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name.

3.1.3 Component Profile

The Profile component is part of the Protocol component specified in clause 3.1.2 and provides information about the specific profile of the supported digital signature related protocol.

Below follows a list of the sub-components that constitute this component:
The **ProfileIdentifier** element MUST contain one instance of a URI, which uniquely identifies the profile of the digital signature related protocol.

The OPTIONAL **Specification** element, if present, MAY occur zero or more times containing a URI, which points to a specification document describing the specific profile of the digital signature related protocol.

The OPTIONAL **Description** element, if present, MAY occur zero or more times containing a sub-component, which satisfies the requirements specified in [DSS-v2.0] for the **InternationalString** component and can be used to provide descriptions of the profile in multiple languages.

The **Operation** element MUST occur 1 or more times containing a sub-component, which describes a specific operation supported by the profile of the digital signature related protocol. For each supported operation there MUST be an **Operation** component and each instance MUST satisfy the requirements specified in this document in section 3.1.4.

The OPTIONAL **Policy** element, if present, MAY occur zero or more times containing a sub-component, which specifies the set of policies, which are applicable for the specific profile of the digital signature related protocol. If present each instance MUST satisfy the requirements specified in this document in section 3.1.7.

The OPTIONAL **Extension** element, if present, MAY occur zero or more times containing a sub-component, which extends the semantics of the **Profile** component. If present each instance MUST satisfy the requirements specified in this document in section 3.1.9.

### 3.1.3.1 Profile – JSON Syntax

The **ProfileType** JSON object SHALL implement in JSON syntax the requirements defined in the **Profile** component.

Properties of the JSON object SHALL implement the sub-components of **Profile** using JSON-specific names mapped as shown in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProfileIdentifier</td>
<td>pid</td>
</tr>
<tr>
<td>Specification</td>
<td>spec</td>
</tr>
<tr>
<td>Description</td>
<td>description</td>
</tr>
<tr>
<td>Operation</td>
<td>op</td>
</tr>
<tr>
<td>Policy</td>
<td>pol</td>
</tr>
<tr>
<td>Extension</td>
<td>ext</td>
</tr>
</tbody>
</table>

The **ProfileType** JSON object is defined in the JSON schema [DSSMD-JSON] and is provided below as a service to the reader.

```json
"md-ProfileType": {
  "type": "object",
  "properties": {
    "pid": {
      "type": "string"
    },
    "spec": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "description": "...
  }
}
```
3.1.3.2 Profile – XML Syntax

The XML type ProfileType SHALL implement the requirements defined in the Profile component. The ProfileType XML element is defined in XML Schema [DSSMD-XML], and is copied below for information.

```xml
<xs:complexType name="ProfileType">
  <xs:sequence>
    <xs:element name="ProfileIdentifier" type="xs:anyURI" maxOccurs="unbounded" minOccurs="0"/>
    <xs:element name="Specification" type="xs:anyURI" maxOccurs="unbounded" minOccurs="0"/>
    <xs:element name="Operation" type="md:OperationType" maxOccurs="unbounded" minOccurs="1"/>
    <xs:element name="Policy" type="md:PolicyType" maxOccurs="unbounded" minOccurs="0"/>
    <xs:element name="Extension" type="md:ExtensionType" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Each child element of ProfileType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name.

3.1.4 Component Operation

The Operation component is part of the Profile component specified in clause 3.1.3 and provides information about an operation supported by a specific profile of the supported digital signature related protocol.

Below follows a list of the sub-components that constitute this component:
The Name element MUST contain one instance of a string, which MUST reflect the name of the request to invoke the operation. For REST-based services this corresponds to the Paths Object component within [OpenAPI] and the Name element SHOULD contain the relative path the endpoint at which the operation can be invoked, which is appended to the URL of the Server component within the Protocol element specified in clause 3.1.2. For SOAP-based services the Name element corresponds to the soap:operation within [WSDL].

The OPTIONAL Specification element, if present, MUST contain a URI, which points to a specification document describing the specific operation under consideration.

The OPTIONAL Description element, if present, MAY occur zero or more times containing a sub-component, which satisfies the requirements specified in [DSS-v2.0] for the InternationalString component and can be used to provide additional information with respect to the specific operation under consideration.

The OPTIONAL Input element, if present, MAY occur zero or more times containing a sub-component, which specifies details of a specific input parameter. If present each instance MUST satisfy the requirements specified in this document in section 3.1.5.

The OPTIONAL Option element, if present, MAY occur zero or more times containing a sub-component, which specifies details of a specific optional input parameter. If present each instance MUST satisfy the requirements specified in this document in section 3.1.5.

The OPTIONAL Output element, if present, MAY occur zero or more times containing a sub-component, which specifies details of a specific output parameter. If present each instance MUST satisfy the requirements specified in this document in section 3.1.5.

The OPTIONAL Schema element, if present, MUST contain a URI, which points to the applicable schema document, which defines the detailed syntax of the component implementing the operation under consideration.

The OPTIONAL Extension element, if present, MAY occur zero or more times containing a sub-component, which extends the semantics of the Operation element. If present each instance MUST satisfy the requirements specified in this document in section 3.1.9.

3.1.4.1 Operation – JSON Syntax

The OperationType JSON object SHALL implement in JSON syntax the requirements defined in the Operation component.

Properties of the JSON object SHALL implement the sub-components of Operation using JSON-specific names mapped as shown in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>name</td>
</tr>
<tr>
<td>Specification</td>
<td>spec</td>
</tr>
<tr>
<td>Description</td>
<td>desc</td>
</tr>
<tr>
<td>Input</td>
<td>in</td>
</tr>
<tr>
<td>Option</td>
<td>opt</td>
</tr>
<tr>
<td>Output</td>
<td>out</td>
</tr>
<tr>
<td>Schema</td>
<td>xsd</td>
</tr>
<tr>
<td>Extension</td>
<td>ext</td>
</tr>
</tbody>
</table>
The `OperationType` JSON object is defined in the JSON schema [DSSMD-JSON] and is provided below as a service to the reader.

```json
"md-OperationType": {
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "spec": {
      "type": "string"
    },
    "desc": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/dsb-InternationalStringType"
      }
    },
    "in": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/md-ParameterType"
      }
    },
    "opt": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/md-ParameterType"
      }
    },
    "out": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/md-ParameterType"
      }
    },
    "xsd": {
      "type": "string"
    },
    "ext": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/md-ExtensionType"
      }
    }
  },
  "required": ["name"]
}
```

### 3.1.4.2 Operation – XML Syntax

The XML type `OperationType` SHALL implement the requirements defined in the Operation component.

The `OperationType` XML element is defined in XML Schema [DSSMD-XML], and is copied below for information.

```xml
<xs:complexType name="OperationType">
  <xs:sequence>
  
  </xs:sequence>
</xs:complexType>
```
Each child element of OperationType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name.

3.1.5 Component Parameter

The Parameter component defines the syntax and semantics of the child components Input, Option and Output of the Operation component specified in clause 3.1.4 and allows to provide additional information with respect to specific input and output parameters as well as the available options for an operation, if this is not yet unambiguously specified by the document referenced in the child element Specification of the Operation according to clause 3.1.4.

Below follows a list of the sub-components that constitute this component:

- The Name element MUST contain one instance of a string, which reflects the name of the parameter under consideration.
- The OPTIONAL Specification element, if present, MUST contain a URI, which points to a specification document describing additional details with respect to the parameter under consideration.
- The OPTIONAL Description element, if present, MAY occur zero or more times containing a sub-component, which satisfies the requirements specified in [DSS-v2.0] for the InternationalString component and can be used to provide additional information with respect to the specific (optional) input or output parameter under consideration.
- The OPTIONAL Format element, if present, MAY occur zero or more times containing a sub-component, which can be used to specify the format of the (optional) input or output parameter under consideration. If present each instance MUST satisfy the requirements specified in this document in section 3.1.6.
- The OPTIONAL Schema element, if present, MUST contain a URI, which points to the applicable schema document, which defines the detailed syntax of the component implementing the specific (optional) input or output parameter under consideration.
- The OPTIONAL Extension element, if present, MAY occur zero or more times containing a sub-component, which extends the semantics of the Parameter component. If present each instance MUST satisfy the requirements specified in this document in section 3.1.9.

3.1.5.1 Parameter – JSON Syntax

The ParameterType JSON object SHALL implement in JSON syntax the requirements defined in the Parameter component.

Properties of the JSON object SHALL implement the sub-components of Parameter using JSON-specific names mapped as shown in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Specification</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Schema</td>
<td></td>
</tr>
<tr>
<td>Extension</td>
<td></td>
</tr>
</tbody>
</table>
The `ParameterType` JSON object is defined in the JSON schema [DSSMD-JSON] and is provided below as a service to the reader.

```
"md-ParameterType": {
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "spec": {
      "type": "string"
    },
    "desc": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/dsb-InternationalStringType"
      }
    },
    "form": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/md-FormatType"
      }
    },
    "xsd": {
      "type": "string"
    },
    "ext": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/md-ExtensionType"
      }
    }
  }
}
```

### 3.1.5.2 Parameter – XML Syntax

The XML type `ParameterType` SHALL implement the requirements defined in the `Parameter` component.

The `ParameterType` XML element is defined in XML Schema [DSSMD-XML], and is copied below for information.

```xml
<xs:complexType name="ParameterType">
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```
Each child element of ParameterType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name.

### 3.1.6 Component Format

The **Format** component is part of the Parameter component specified in clause 3.1.5 and allows to provide additional information with respect to format of the specific input and output parameters or options of an operation, if this is not yet unambiguously specified by the document referenced in the child element **Specification** of the Operation according to clause 3.1.4.

Below follows a list of the sub-components that constitute this component:

- **FormatID** element MUST contain one instance of a URI, which identifies the format of the parameter.
- The OPTIONAL **Specification** element, if present, MUST contain a URI, which points to a specification document describing additional details with respect to the format under consideration.
- The OPTIONAL **Description** element, if present, MAY occur zero or more times containing a sub-component, which satisfies the requirements specified in [DSS-v2.0] for the InternationalString component and can be used to provide additional information with respect to the format under consideration.
- The OPTIONAL **Parameter** element, if present, MAY occur zero or more times containing a sub-component, which provides more information with respect to a specific parameter under consideration. If present each instance MUST satisfy the requirements specified in this document in section 3.1.5.
- The OPTIONAL **Extension** element, if present, MAY occur zero or more times containing a sub-component, which extends the semantic of the Format component. If present each instance MUST satisfy the requirements specified in [DSS-v2.0] for the Any component.
- The OPTIONAL **IsDefault** element, if present, MUST contain one instance of a boolean and indicates whether the format under consideration is the default format. Its default value is ‘false’. The precise semantics what it means that a format is considered to be “the default format” MUST be defined by profiles or extensions of [DSS-v2.0].

#### 3.1.6.1 Format – JSON Syntax

The **FormatType** JSON object SHALL implement in JSON syntax the requirements defined in the Format component.

Properties of the JSON object SHALL implement the sub-components of Format using JSON-specific names mapped as shown in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormatID</td>
<td>fid</td>
</tr>
<tr>
<td>Specification</td>
<td>spec</td>
</tr>
<tr>
<td>Description</td>
<td>desc</td>
</tr>
<tr>
<td>Parameter</td>
<td>format</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
</tr>
<tr>
<td>Extension</td>
<td>ext</td>
</tr>
<tr>
<td>IsDefault</td>
<td>def</td>
</tr>
</tbody>
</table>

The `FormatType` JSON object is defined in the JSON schema [DSSMD-JSON] and is provided below as a service to the reader.

```
"md-FormatType": {
  "type": "object",
  "properties": {
    "fid": {
      "type": "string"
    },
    "spec": {
      "type": "string"
    },
    "desc": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/dsb-InternationalStringType"
      }
    },
    "format": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/md-ParameterType"
      }
    },
    "ext": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/dsb-AnyType"
      }
    },
    "def": {
      "type": "boolean",
      "default": "false"
    }
  },
  "required": ["fid"]
}
```

### 3.1.6.2 Format – XML Syntax

The XML type `FormatType` SHALL implement the requirements defined in the `Format` component. The `FormatType` XML element is defined in XML Schema [DSSMD-XML], and is copied below for information.

```
<xs:complexType name="FormatType">
  <xs:sequence>
    <xs:element name="FormatID" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
    <xs:element name="Specification" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
    <xs:element name="Parameter" type="md:ParameterType" maxOccurs="unbounded" minOccurs="0"/>
    <xs:element name="Extension" type="dsb:AnyType" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```
Each child element of FormatType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name.

### 3.1.7 Component Policy

The Policy component appears within the Profile component specified in clause 3.1.3 and provides information about an applicable policy of the profile of the supported digital signature related protocol.

Below follows a list of the sub-components that constitute this component:

- The OPTIONAL PolicyByRef element, if present, MUST contain one instance of a sub-component, which provides a reference to a human readable policy document. This element MUST satisfy the requirements specified in this document in section 3.1.8.
- The OPTIONAL PolicyByDef element, if present, MUST contain one instance of a sub-component, which contains a machine readable policy document. This element MUST satisfy the requirements specified in [DSS-v2.0] for the Any component. The detailed syntax and semantics of the machine readable policy document MUST be defined by profiles or extensions of [DSS-v2.0] or specifications referenced in such documents.
- The OPTIONAL EarlierPolicy element, if present, MAY occur zero or more times containing a URI, which refers to an earlier policy document.
- The OPTIONAL Extension element, if present, MAY occur zero or more times containing a sub-component, which extends the semantics of the Policy component. If present each instance MUST satisfy the requirements specified in this document in section 3.1.9.
- The OPTIONAL Type element, if present, MUST contain one instance of a URI. The admissible or recommended values for the policy types SHOULD be defined by profiles or extensions of [DSS-v2.0].

#### 3.1.7.1 Policy – JSON Syntax

The PolicyType JSON object SHALL implement in JSON syntax the requirements defined in the Policy component.

Properties of the JSON object SHALL implement the sub-components of Policy using JSON-specific names mapped as shown in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolicyByRef</td>
<td>pbrf</td>
</tr>
<tr>
<td>PolicyByDef</td>
<td>pbdef</td>
</tr>
<tr>
<td>EarlierPolicy</td>
<td>ep</td>
</tr>
<tr>
<td>Extension</td>
<td>ext</td>
</tr>
<tr>
<td>Type</td>
<td>type</td>
</tr>
</tbody>
</table>

The PolicyType JSON object is defined in the JSON schema [DSSMD-JSON] and is provided below as a service to the reader.

```json
"md-PolicyType": {
  "type": "object",
  "properties": {
    "type": { 
```
3.1.7.2 Policy – XML Syntax

The XML type PolicyType SHALL implement the requirements defined in the Policy component. The PolicyType XML element is defined in XML Schema [DSSMD-XML], and is copied below for information.

```xml
<xs:complexType name="PolicyType">
    <xs:sequence>
        <xs:choice>
            <xs:element name="PolicyByRef" type="md:PolicyByRefType"/>
            <xs:element name="PolicyByDef" type="dsb:AnyType"/>
        </xs:choice>
        <xs:element name="EarlierPolicy" type="xs:anyURI" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Type" type="xs:anyURI" use="optional"/>
</xs:complexType>
```

Each child element of PolicyType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name.

3.1.8 Component PolicyByRef

The PolicyByRef component appears within the Policy component specified in clause 3.1.7 and provides a reference to a human readable policy document, which is applicable for a profile of the supported digital signature related protocol.

Below follows a list of the sub-components that constitute this component:
The PolicyID element MUST contain one instance of a URI, which uniquely identifies the policy under consideration.

The OPTIONAL PolicyLocation element, if present, MUST contain a URI, which SHOULD refer to the location where the policy document can be retrieved. If the PolicyID is already a retrievable URL, the PolicyLocation MAY be omitted.

### 3.1.8.1 PolicyByRef – JSON Syntax

The PolicyByRefType JSON object SHALL implement in JSON syntax the requirements defined in the PolicyByRef component.

Properties of the JSON object SHALL implement the sub-components of PolicyByRef using JSON-specific names mapped as shown in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolicyID</td>
<td>polid</td>
</tr>
<tr>
<td>PolicyLocation</td>
<td>polloc</td>
</tr>
</tbody>
</table>

The PolicyByRefType JSON object is defined in the JSON schema [DSSMD-JSON] and is provided below as a service to the reader.

```
"md-PolicyByRefType": {
  "type": "object",
  "properties": {
    "polid": {
      "type": "string"
    },
    "polloc": {
      "type": "string"
    }
  },
  "required": ["polid"]
}
```

### 3.1.8.2 PolicyByRef – XML Syntax

The XML type PolicyByRefType SHALL implement the requirements defined in the PolicyByRef component.

The PolicyByRefType XML element is defined in XML Schema [DSSMD-XML], and is copied below for information.

```xml
<xs:complexType name="PolicyByRefType">
  <xs:sequence>
    <xs:element name="PolicyID" type="xs:anyURI"/>
    <xs:element name="PolicyLocation" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Each child element of PolicyByRefType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name.

### 3.1.9 Component Extension

The Extension component defined in the present document is used in several other components and provides a lightweight possibility for extending the semantics of other components.

Below follows a list of the sub-components that constitute this component:
The **Name** element MUST contain one instance of a string and specifies the name of the extension element. The **Value** element MUST contain one instance of a string and specifies the value of the extension element.

**NOTE:** In contrast to the **Any** component defined in [DSS-v2.0], the Extension element defined here only consists of a simple **Name** and **Value** pair, which maintains the direct readability by humans, but is less powerful than the **Any** component, which also allows features transformations for example.

### 3.1.9.1 Extension – JSON Syntax

The **ExtensionType** JSON object SHALL implement in JSON syntax the requirements defined in the **Extension** component.

Properties of the JSON object SHALL implement the sub-components of **Extension** using JSON-specific names mapped as shown in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>name</td>
</tr>
<tr>
<td>Value</td>
<td>value</td>
</tr>
</tbody>
</table>

The **ExtensionType** JSON object is defined in the JSON schema [DSSMD-JSON] and is provided below as a service to the reader.

```json
"md-ExtensionType": {
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "value": {
      "type": "string"
    }
  },
  "required": ["name", "value"]
}
```

### 3.1.9.2 Extension – XML Syntax

The XML type **ExtensionType** SHALL implement the requirements defined in the **Extension** component.

The **ExtensionType** XML element is defined in XML Schema [DSSMD-XML], and is copied below for information.

```xml
<xs:complexType name="ExtensionType">
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Value" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```
Each child element of `ExtensionType` XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name.

### 3.2 Element / JSON name lookup tables

The subsequent table allows to find the names of a component's element for a given JSON member name.

<table>
<thead>
<tr>
<th>JSON member name</th>
<th>mapped from element name</th>
</tr>
</thead>
<tbody>
<tr>
<td>authinfo</td>
<td>AuthInfo</td>
</tr>
<tr>
<td>def</td>
<td>IsDefault</td>
</tr>
<tr>
<td>ep</td>
<td>EarlierPolicy</td>
</tr>
<tr>
<td>ext</td>
<td>Extension</td>
</tr>
<tr>
<td>fid</td>
<td>FormatID</td>
</tr>
<tr>
<td>form</td>
<td>Format</td>
</tr>
<tr>
<td>format</td>
<td>Parameter</td>
</tr>
<tr>
<td>ID</td>
<td>Id</td>
</tr>
<tr>
<td>in</td>
<td>Input</td>
</tr>
<tr>
<td>lang</td>
<td>SupportedLanguage</td>
</tr>
<tr>
<td>logo</td>
<td>Logo</td>
</tr>
<tr>
<td>name</td>
<td>Name</td>
</tr>
<tr>
<td>op</td>
<td>Operation</td>
</tr>
<tr>
<td>opt</td>
<td>Option</td>
</tr>
<tr>
<td>out</td>
<td>Output</td>
</tr>
<tr>
<td>pbdef</td>
<td>PolicyByDef</td>
</tr>
<tr>
<td>pbref</td>
<td>PolicyByRef</td>
</tr>
<tr>
<td>pid</td>
<td>ProfileIdentifier</td>
</tr>
<tr>
<td>pol</td>
<td>Policy</td>
</tr>
<tr>
<td>polid</td>
<td>PolicyID</td>
</tr>
<tr>
<td>polloc</td>
<td>PolicyLocation</td>
</tr>
<tr>
<td>pre</td>
<td>NamespacePrefix</td>
</tr>
<tr>
<td>profile</td>
<td>Profile</td>
</tr>
<tr>
<td>protocol</td>
<td>Protocol</td>
</tr>
<tr>
<td>region</td>
<td>Region</td>
</tr>
<tr>
<td>spec</td>
<td>Specification</td>
</tr>
<tr>
<td>uri</td>
<td>NamespaceURI</td>
</tr>
</tbody>
</table>
The subsequent table allows to find the abbreviated JSON member names for a given element name.

<table>
<thead>
<tr>
<th>Element</th>
<th>Implementing JSON member name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthInfo</td>
<td>authinfo</td>
</tr>
<tr>
<td>DigestMethod</td>
<td>alg</td>
</tr>
<tr>
<td>DigestValue</td>
<td>val</td>
</tr>
<tr>
<td>EarlierPolicy</td>
<td>ep</td>
</tr>
<tr>
<td>Extension</td>
<td>ext</td>
</tr>
<tr>
<td>Format</td>
<td>form</td>
</tr>
<tr>
<td>FormatID</td>
<td>fid</td>
</tr>
<tr>
<td>Id</td>
<td>ID</td>
</tr>
<tr>
<td>Input</td>
<td>in</td>
</tr>
<tr>
<td>IsDefault</td>
<td>def</td>
</tr>
<tr>
<td>Logo</td>
<td>logo</td>
</tr>
<tr>
<td>Name</td>
<td>name</td>
</tr>
<tr>
<td>NamespacePrefix</td>
<td>pre</td>
</tr>
<tr>
<td>NamespaceURI</td>
<td>uri</td>
</tr>
<tr>
<td>Operation</td>
<td>op</td>
</tr>
<tr>
<td>Option</td>
<td>opt</td>
</tr>
<tr>
<td>Output</td>
<td>out</td>
</tr>
<tr>
<td>Parameter</td>
<td>format</td>
</tr>
<tr>
<td>Policy</td>
<td>pol</td>
</tr>
<tr>
<td>PolicyByDef</td>
<td>pbdef</td>
</tr>
<tr>
<td>PolicyByRef</td>
<td>pbrf</td>
</tr>
<tr>
<td>PolicyID</td>
<td>polid</td>
</tr>
<tr>
<td>PolicyLocation</td>
<td>polloc</td>
</tr>
<tr>
<td>Profile</td>
<td>profile</td>
</tr>
<tr>
<td>ProfileIdentifier</td>
<td>pid</td>
</tr>
<tr>
<td>Protocol</td>
<td>protocol</td>
</tr>
<tr>
<td>Region</td>
<td>region</td>
</tr>
<tr>
<td>Schema</td>
<td>xsd</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Specification</td>
<td>spec</td>
</tr>
<tr>
<td>SupportedLanguage</td>
<td>lang</td>
</tr>
<tr>
<td>Value</td>
<td>value</td>
</tr>
<tr>
<td>Version</td>
<td>version</td>
</tr>
</tbody>
</table>
4 Metadata Discovery

Unless other discovery mechanisms are specified (by profiles or extensions of [DSS-v2.0] for example), it is RECOMMENDED that digital signature service providers make available a JSON or XML document using the appropriate content type (i.e. application/json or application/xml) with the digital signature service metadata at the path formed by concatenating the string /well-known/dss-info to a suitable “basic information URL” of the service provider, which is intended to provide relevant information about the provided services (see also “TSP information URI” in clause 5.4.4 of [TS119612]).
Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Andreas Kuehne, Individual
Detlef Hühnlein, Individual
Ernst Jan van Nigtevecht, Sonnenglanz Consulting
Appendix B. List of Figures

Figure 1: Overview of main components within the service-related metadata structures ........................................8
## Appendix C. Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>WD01</td>
<td>2019-03-17</td>
<td>Detlef Hühnlein and Andreas Kuehne</td>
<td>Draft for discussion within DSS-X and potential ballot public review</td>
</tr>
<tr>
<td>CSD01</td>
<td>2019-03-18</td>
<td>Detlef Hühnlein and Andreas Kuehne</td>
<td>Version for public review</td>
</tr>
<tr>
<td>CS01</td>
<td>2019-05-07</td>
<td>Detlef Hühnlein and Andreas Kuehne</td>
<td>Committee Specification</td>
</tr>
<tr>
<td>CSD02</td>
<td>2019-09-09</td>
<td>Andreas Kuehne</td>
<td>Version for public review, no changes within this document but in the related schema files</td>
</tr>
<tr>
<td>CS02</td>
<td>2019-10-27</td>
<td>Andreas Kuehne</td>
<td>Version for publication as CS. Only editorial changes, as no comments received for CSD02.</td>
</tr>
</tbody>
</table>