Digital Signature Service Metadata Version 1.0

Committee Specification Draft 01 / Public Review Draft 01

27 March 2019

This version:

https://docs.oasis-open.org/dss-x/dss-md/v1.0/csprd01/dss-md-v1.0-csprd01.docx (Authoritative)
https://docs.oasis-open.org/dss-x/dss-md/v1.0/csprd01/dss-md-v1.0-csprd01.html
https://docs.oasis-open.org/dss-x/dss-md/v1.0/csprd01/dss-md-v1.0-csprd01.pdf

Previous version:

N/A

Latest 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/csprd01/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/info

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" 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]

Digital Signature Service Metadata Version 1.0. Edited by Detlef Hühnlein and Andreas Kuehne. 27 March 2019. OASIS Committee Specification Draft 01 / Public Review Draft 01. https://docs.oasis-open.org/dss-x/dss-md/v1.0/csprd01/dss-md-v1.0-csprd01.html. Latest version: https://docs.oasis-open.org/dss-x/dss-md/v1.0/dss-md-v1.0.html.

Notices

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 6
1.1 IPR Policy 6
1.2 Terminology 6
1.2.1 Terms and Definitions 6
1.2.2 Abbreviated Terms 6
1.3 Normative References 6
1.4 Non-Normative References 7
1.5 Typographical Conventions 8
1.6 Motivation and related work (Non-normative) 8
2 Overview 9
3 Data Structure Models 10
3.1 Data Structure Models defined in this document 10
3.1.1 Component Provider 10
3.1.1.1 Provider – JSON Syntax 10
3.1.1.2 Provider – XML Syntax 12
3.1.2 Component Protocol 12
3.1.2.1 Protocol – JSON Syntax 12
3.1.2.2 Protocol – XML Syntax 13
3.1.3 Component Profile 14
3.1.3.1 Profile – JSON Syntax 14
3.1.3.2 Profile – XML Syntax 15
3.1.4 Component Operation 16
3.1.4.1 Operation – JSON Syntax 16
3.1.4.2 Operation – XML Syntax 18
3.1.5 Component Parameter 18
3.1.5.1 Parameter – JSON Syntax 19
3.1.5.2 Parameter – XML Syntax 20
3.1.6 Component Format 20
3.1.6.1 Format – JSON Syntax 21
3.1.6.2 Format – XML Syntax 22
3.1.7 Component Policy 22
3.1.7.1 Policy – JSON Syntax 23
3.1.7.2 Policy – XML Syntax 24
3.1.8 Component PolicyByRef 24
3.1.8.1 PolicyByRef – JSON Syntax 24
3.1.8.2 PolicyByRef – XML Syntax 25
3.1.9 Component Extension 25
3.1.9.1 Extension – JSON Syntax 25
3.1.9.2 Extension – XML Syntax 26
3.1.10 Component TypedLocator 26
3.1.10.1 TypedLocator – JSON Syntax 27
3.1.10.2 TypedLocator – XML Syntax 27
3.2 Element / JSON name lookup tables 28
4 Metadata Discovery 31
Appendix A. Acknowledgments 32

Appendix B. List of Figures 33
Appendix C. Revision History 34

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

JSON — JavaScript Object Notation

URI — (IETF) Uniform Resource Identifier according to [RFC3986]

URL — Uniform Resource Locator

XML — (W3C) Extensible Markup Language

XSD — (W3C) XML Schema

1.3 Normative References

[DSS2-JSON] A. Kuehne, S. Hagen. *DSS 2.0 Core JSON Schema*. OASIS.

[DSS2-XSD] A. Kuehne, S. Hagen. *DSS 2.0 Core XML Schema*. OASIS.

[DSSMD-JSON] D. Hühnlein, A. Kuehne. *Digital Signature Service Metadata JSON Schema.* OASIS.

[DSSMD-XML] D. Hühnlein, A. Kuehne. *Digital Signature Service Metadata XML Schema.* OASIS.

[ISO3166-1] ISO 3166-1:2013: "Codes for the representation of names of countries and their subdivisions — Part 1: Country codes".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, http://www.rfc-editor.org/info/rfc2119.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, https://www.rfc-editor.org/info/rfc3986.

[RFC5646] Phillips, A., Ed., and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, https://www.rfc-editor.org/info/rfc5646

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, http://www.rfc-editor.org/info/rfc8174.

1.4 Non-Normative References

[BDX-SMP-v1.0] Service Metadata Publishing (SMP) Version 1.0. Edited by Jens Aabol, Kenneth Bengtsson, Erlend Klakegg Bergheim, Sander Fieten, and Sven Rasmussen. 01 August 2017. OASIS Standard. http://docs.oasis-open.org/bdxr/bdx-smp/v1.0/os/bdx-smp-v1.0-os.html

[BDX-SMP-v2.0] Service Metadata Publishing (SMP) Version 2.0. Edited by Kenneth Bengtsson, Erlend Klakegg Bergheim, Sander Fieten, and G. Ken Holman. 30 January 2019. OASIS Committee Specification Draft 02 / Public Review Draft 02. https://docs.oasis-open.org/bdxr/bdx-smp/v2.0/csprd02/bdx-smp-v2.0-csprd02.html**.** Latest version: https://docs.oasis-open.org/bdxr/bdx-smp/v2.0/bdx-smp-v2.0.html

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

[DSS-v1.0] *Digital Signature Service Core Protocols, Elements, and Bindings Version 1.0*. Edited by Stefan Drees. 11 April 2007. OASIS Standard. http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-spec-v1.0-os.html.

[DSS-v2.0] *Digital Signature Service Core Protocols, Elements, and Bindings Version 2.0*. Edited by Andreas Kuehne and Stefan Hagen. 20 February 2019. OASIS Committee Specification Draft 02 / Public Review Draft 02. http://docs.oasis-open.org/dss-x/dss-core/v2.0/csprd02/dss-core-v2.0-csprd02.html. Latest version: http://docs.oasis-open.org/dss-x/dss-core/v2.0/dss-core-v2.0.html.

[eIDAS] Regulation (EU) No 910/2014 of the European Parliament and of the Council of of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC, http://data.europa.eu/eli/reg/2014/910/oj.

[OIDC-MD] OpenID Connect Discovery 1.0. Edited by N. Sakimura, J. Bradley, M. Jones and E. Jay, 8 November 2014, https://openid.net/specs/openid-connect-discovery-1_0.html

[OpenAPI] The OpenAPI Specification, https://github.com/OAI/OpenAPI-Specification

[RFC8414] M. Jones, N. Sakimura, J. Bradley. *OAuth 2.0 Authorization Server Metadata.* IETF RFC 8414, June 2018. http://www.ietf.org/rfc/rfc8414.txt.

[SAML-MD] *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. Edited by Scott Cantor, Jahan Moreh, Rob Philpott and Eve Maler. 15 March 2005, OASIS Standard. https://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf

[TS119432] ETSI, “Electronic Signatures and Infrastructures (ESI); Protocols for remote digital signature creation”, Draft ETSI TS 119 432, V0.0.10 (2019-03).

[TS119442] ETSI, “Electronic Signatures and Infrastructures (ESI); Protocol profiles for trust service providers providing AdES digital signature validation services”, ETSI TS 119 442, V1.1.1 (2019-02), https://www.etsi.org/deliver/etsi_ts/119400_119499/119442/01.01.01_60

[TS119512] ETSI, “Electronic Signatures and Infrastructures (ESI); Protocols for trust service providers providing long-term data preservation services”, Draft ETSI TS 119 512, V0.0.8 (2019-03).

[TS119612] ETSI, “Electronic Signatures and Infrastructures (ESI); Trusted Lists”, ETSI TS 119 612, V2.2.1 (2016-04), https://www.etsi.org/deliver/etsi_ts/119600_119699/119612/02.02.01_60/.

[WSDL] Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001, https://www.w3.org/TR/2001/NOTE-wsdl-20010315

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`.

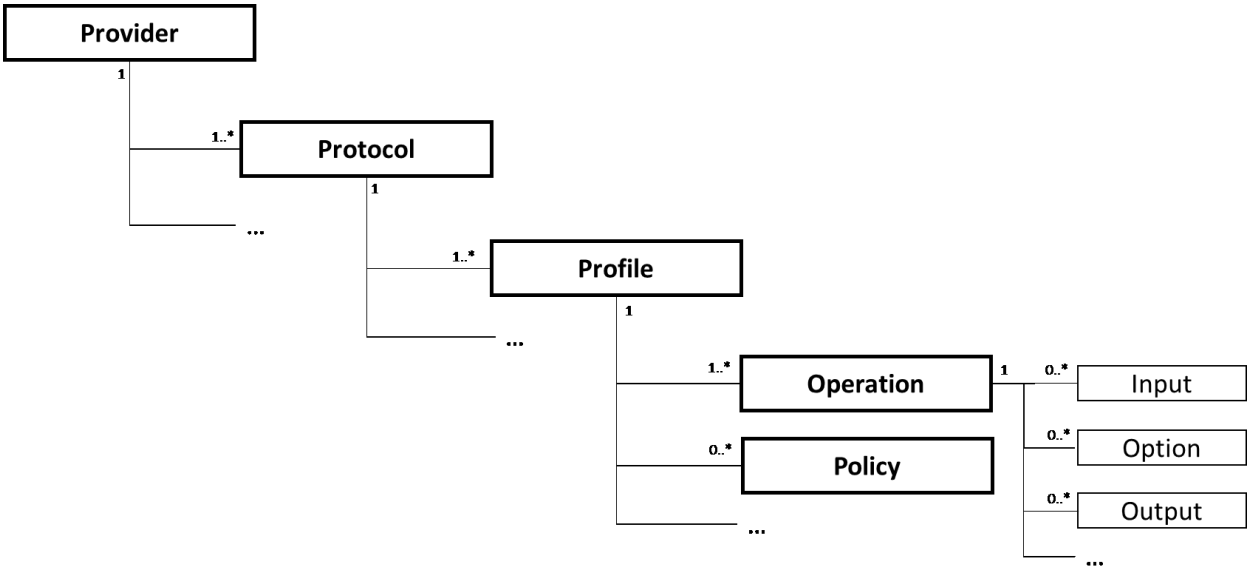

Figure 1: Overview of main components within the service-related metadata structures

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/info`'.

3.1.1 Component Provider

The component `Provider` is the main element of the metadata structure and contains information about the provider of the related service. 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:

> The `Name` element MUST contain one instance of a string, which contains the commercial name of the service provider. It is RECOMMENDED to limit the size of this string to 255 characters.
>
> The `Logo` element MUST contain one instance of a URI, which refers to an image file containing the logo of the service provider. This image file MUST be published online and SHOULD either be in JPEG or PNG format and SHOULD NOT be larger than 256x256 pixels.
>
> The `Region` element MUST contain one instance of a string with the [ISO3166-1] Alpha-2 code of the country in which the service provider is established.
>
> The OPTIONAL `SupportedLanguage` element, if present, MAY occur zero or more times in order to signal the set of supported languages in line with [RFC5646].
>
> The OPTIONAL `Description` element, if present, MAY occur zero or more times containing a sub-component, which provides additional information which describes the service. If present each instance MUST satisfy the requirements specified in [DSS-v2.0] for the `InternationalString` component, whereas it is RECOMMENDED to limit the size of the `value` component to 255 characters.
>
> The OPTIONAL `AuthInfo` element, if present, MUST satisfy the requirements specified in clause 3.1.10 for the `TypedLocator` component. This component MUST contain a URI, which points to the location where the metadata document for the authentication and authorization service can be retrieved and MAY in addition contain a `Type` component, which specifies the type of the provided metadata document. The present document defines the following values for the `Type` component:

- `urn:ietf:rfc:8414` – for OAuth 2.0 metadata according to [RFC8414]
- `urn:oasis:names:tc:SAML:2.0:metadata` – for SAML 2.0 metadata according to [SAML-MD]

> The `Protocol` element MUST occur 1 or more times containing a sub-component, which provides information about the supported protocols of the service. Each instance MUST satisfy the requirements specified in this document in section 3.1.2.
>
> The OPTIONAL `Extension` element, if present, MAY occur zero or more times containing a sub-component, which extends the semantic of the `Provider` component. If present each instance MUST satisfy the requirements specified in this document in section 3.1.9.

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.

Element	Implementing JSON member name
Name	name
Logo	logo
Region	region
SupportedLanguage	lang
Description	description
AuthInfo	authinfo
Protocol	protocol
Extension	ext

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

```
"info-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/info-ProtocolType"
      }
    },
    "ext": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-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.

```
<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="info:Description"/>
    <xs:element name="AuthInfo" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
    <xs:element name="Protocol" type="info:ProtocolType" maxOccurs="unbounded"
minOccurs="1"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info: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 `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.

Element	Implementing JSON member name
Server	srv
Specification	spec
Description	description
Profile	profile
Extension	ext

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

```
"info-ProtocolType": {
  "type": "object",
  "properties": {
    "srv": {
      "type": "string"
    },
    "spec": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "description": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/dsb-InternationalStringType"
      }
    },
    "profile": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-ProfileType"
      }
    },
    "ext": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-ExtensionType"
      }
    }
  },
  "required": ["profile"]
}
```

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.

```
<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 maxOccurs="unbounded" minOccurs="0" ref="info:Description"/>
    <xs:element name="Profile" type="info:ProfileType" maxOccurs="unbounded"
minOccurs="1"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info: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 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.

Element	Implementing JSON member name
`ProfileIdentifier`	`prfid`
`Specification`	`spec`
`Description`	`description`

Operation	op
Policy	pol
Extension	ext

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

```
"info-ProfileType": {
  "type": "object",
  "properties": {
    "prfid": {
      "type": "string"
    },
    "spec": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "description": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/dsb-InternationalStringType"
      }
    },
    "op": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-OperationType"
      }
    },
    "pol": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-PolicyType"
      }
    },
    "ext": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-ExtensionType"
      }
    }
  },
  "required": ["prfid", "op"]
}
```

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.

```
<xs:complexType name="ProfileType">
  <xs:sequence>
    <xs:element name="ProfileIdentifier" type="xs:anyURI"/>
```

```
    <xs:element name="Specification" type="xs:anyURI" maxOccurs="unbounded"
minOccurs="0"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info:Description"/>
    <xs:element name="Operation" type="info:OperationType"
maxOccurs="unbounded" minOccurs="1"/>
    <xs:element name="Policy" type="info:PolicyType" maxOccurs="unbounded"
minOccurs="0"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info:Extension"/>
  </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 `OperationIdentifier` element MUST contain one instance of a URI, 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 `OperationIdentifier` 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.

Element	Implementing JSON member name
Name	name
Specification	spec
Description	desc
Input	in
Option	opt
Output	out
Schema	schema
Extension	ext

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

```
"info-OperationType": {
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "spec": {
      "type": "string"
    },
    "desc": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/dsb-InternationalStringType"
      }
    },
    "in": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-ParameterType"
      }
    },
    "opt": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-ParameterType"
      }
    },
    "out": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-ParameterType"
      }
    },
    "schema": {
      "type": "string"
    },
    "ext": {
```

```
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-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.

```
<xs:complexType name="OperationType">
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Specification" type="xs:anyURI" maxOccurs="1"
minOccurs="0"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info:Description"/>
    <xs:element name="Input" type="info:ParameterType" maxOccurs="unbounded"
minOccurs="0"/>
    <xs:element name="Option" type="info:ParameterType" maxOccurs="unbounded"
minOccurs="0"/>
    <xs:element name="Output" type="info:ParameterType" maxOccurs="unbounded"
minOccurs="0"/>
    <xs:element name="Schema" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info:Extension"/>
  </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 semantic 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.

Element	Implementing JSON member name
`Name`	`name`
`Specification`	`spec`
`Description`	`desc`
`Format`	`form`
`Schema`	`schema`
`Extension`	`ext`

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

```
"info-ParameterType": {
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "spec": {
      "type": "string"
    },
    "desc": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/dsb-InternationalStringType"
```

```
      }
    },
    "form": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-FormatType"
      }
    },
    "schema": {
      "type": "string"
    },
    "ext": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-ExtensionType"
      }
    }
  },
  "required": ["name"]
}
```

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.

```
<xs:complexType name="ParameterType">
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Specification" type="xs:anyURI" maxOccurs="1"
minOccurs="0"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info:Description"/>
    <xs:element name="Format" type="info:FormatType" maxOccurs="unbounded"
minOccurs="0"/>
    <xs:element name="Schema" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info:Extension"/>
  </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:

The `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.

Element	Implementing JSON member name
FormatID	fid
Specification	spec
Description	desc
Parameter	format
Extension	ext
IsDefault	def

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

```
"info-FormatType": {
  "type": "object",
  "properties": {
    "fid": {
      "type": "string"
    },
    "spec": {
      "type": "string"
    },
    "desc": {
      "type": "array",
      "items": {
```

```
        "$ref": "#/definitions/dsb-InternationalStringType"
      }
    },
    "format": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-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"/>
    <xs:element name="Specification" type="xs:anyURI" maxOccurs="1"
minOccurs="0"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info:Description"/>
    <xs:element name="Parameter" type="info:ParameterType"
maxOccurs="unbounded" minOccurs="0"/>
    <xs:element name="Extension" type="dsb:AnyType" maxOccurs="unbounded"
minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="IsDefault" type="xs:boolean" default="false"
use="optional"/>
</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.

Element	Implementing JSON member name
PolicyByRef	pbref
PolicyByDef	pbdef
EarlierPolicy	ep
Extension	ext
Type	type

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

```
"info-PolicyType": {
  "type": "object",
  "properties": {
    "type": {
      "type": "string",
      "format": "uri"
    },
    "pbref": {
      "$ref": "#/definitions/info-PolicyByRefType"
    },
    "pbdef": {
      "$ref": "#/definitions/dsb-AnyType"
    },
    "ep": {
      "type": "array",
      "items": {
        "type": "string"
      }
```

```
    },
    "ext": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/info-ExtensionType"
      }
    }
  }
}
```

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.

```
<xs:complexType name="PolicyType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="PolicyByRef" type="info:PolicyByRefType"/>
      <xs:element name="PolicyByDef" type="dsb:AnyType"/>
    </xs:choice>
    <xs:element name="EarlierPolicy" type="xs:anyURI" maxOccurs="unbounded"
minOccurs="0"/>
    <xs:element maxOccurs="unbounded" minOccurs="0" ref="info:Extension"/>
  </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 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.

Element	Implementing JSON member name
`PolicyID`	`polid`
`PolicyLocation`	`polloc`

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

```
"info-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.

```
<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.

Element	Implementing JSON member name
`Name`	`name`
`Value`	`val`

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

```
"info-ExtensionType": {
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "val": {
      "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.

```
<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.

3.1.10 Component TypedLocator

The `TypedLocator` component defined in the present document is used for the definition of the `AuthInfo` component in clause 3.1.2 and MUST contain a URL, which points to a retrievable document and which MAY in addition contain a URI, which specifies the type of the provided document.

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

> The `value` element MUST contain one instance of a URI, which specifies the location of the document.
>
> The `Type` element MAY, if present, contain one instance of a URI, which specifies the type of the document.

3.1.10.1 TypedLocator – JSON Syntax

The `TypedLocatorType` JSON object SHALL implement in JSON syntax the requirements defined in the `TypedLocator` component.

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

Element	Implementing JSON member name
`value`	`value`
`Type`	`type`

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

```
"info-TypedLocator": {
  "type": "object",
  "properties": {
    "value": {
      "type": "string"
    },
    "type": {
      "type": "string"
    }
  },
  "required": ["value"]
}
```

3.1.10.2 TypedLocator – XML Syntax

The XML type `TypedLocatorType` SHALL implement the requirements defined in the `TypedLocator` component.

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

```
<xs:complexType name="TypedLocatorType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="Type" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Each child element of `TypedLocatorType` 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.

JSON member name	mapped from element name
authinfo	AuthInfo
def	IsDefault
ep	EarlierPolicy
ext	Extension
fid	FormatID
form	Format
format	Parameter
ID	Id
in	Input
lang	SupportedLanguage
logo	Logo
opid	OperationIdentifier
op	Operation
opt	Option
out	Output
pbdef	PolicyByDef
pbref	PolicyByRef
prfid	ProfileIdentifier
pol	Policy
polid	PolicyID
polloc	PolicyLocation
pre	NamespacePrefix
profile	Profile
protocol	Protocol
region	Region
spec	Specification
type	Type
uri	NamespaceURI
value	Value

xsd	Schema

The subsequent table allows to find the abbreviated JSON member names for a given element name.

Element	Implementing JSON member name
AuthInfo	authinfo
DigestMethod	alg
DigestValue	val
EarlierPolicy	ep
Extension	ext
Format	form
FormatID	fid
Id	ID
Input	in
IsDefault	def
Logo	logo
NamespacePrefix	pre
NamespaceURI	uri
Operation	op
OperationIdentifier	opid
Option	opt
Output	out
Parameter	format
Policy	pol
PolicyByDef	pbdef
PolicyByRef	pbref
PolicyID	polid
PolicyLocation	polloc
Profile	profile
ProfileIdentifier	prfid
Protocol	protocol
Region	region
Schema	schema
Specification	spec

SupportedLanguage	lang
Type	type
Value	value

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 the “canonical information URL” of the service provider, which is intended to provide information about the provided services.

The “TSP information URI” according to clause 5.4.4 of **[TS119612]** MAY be used as “canonical information URL” to provider the metadata for its digital signature related services.

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 9

Appendix C. Revision History

Revision	Date	Editor	Changes Made
WD01	2019-03-17	Detlef Hühnlein and Andreas Kuehne	Draft for discussion within DSS-X and potential ballot public review
CSD01	2019-03-18	Detlef Hühnlein and Andreas Kuehne	Version for public review