



REVIEW DRAFT

## FIDO Metadata Service

FIDO Alliance Review Draft 27 September 2017

**This version:**

<https://fidoalliance.org/specs/fido-v2.0-rd-20170927/fido-metadata-service-v2.0-rd-20170927.html>

**Previous version:**

<https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-metadata-service-v1.0-ps-20141208.html>

**Editor:**

[Rolf Lindemann, Nok Nok Labs, Inc.](#)

**Contributors:**

[Brad Hill, PayPal, Inc.](#)

[Davit Baghdasaryan, Nok Nok Labs, Inc.](#)

Copyright © 2013-2017 [FIDO Alliance](#) All Rights Reserved.

### Abstract

The FIDO Authenticator Metadata Specification defines so-called "Authenticator Metadata" statements. The metadata statements contain the "Trust Anchor" required to validate the attestation object, and they also describe several other important characteristics of the authenticator.

The metadata service described in this document defines a baseline method for relying parties to access the latest metadata statements.

### Status of This Document

*This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current FIDO Alliance publications and the latest revision of this technical report can be found in the [FIDO Alliance specifications index](https://www.fidoalliance.org/specifications/) at <https://www.fidoalliance.org/specifications/>.*

This document was published by the [FIDO Alliance](#) as a Review Draft. This document is intended to become a FIDO Alliance Proposed Standard. If you wish to make comments regarding this document, please [Contact Us](#). All comments are welcome.

**This is a Review Draft Specification and is not intended to be a basis for any implementations as the Specification may change.** Permission is hereby granted to use the Specification solely for the purpose of reviewing the Specification. No rights are granted to prepare derivative works of this Specification. Entities seeking permission to reproduce portions of this Specification for other uses must contact the FIDO Alliance to determine whether an appropriate license for such use is available.

Implementation of certain elements of this Specification may require licenses under third party intellectual property rights, including without limitation, patent rights. The FIDO Alliance, Inc. and its Members and any other contributors to the Specification are not, and shall not be held, responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

THIS FIDO ALLIANCE SPECIFICATION IS PROVIDED "AS IS" AND WITHOUT ANY WARRANTY OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY EXPRESS OR IMPLIED WARRANTY OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

### Table of Contents

- 1. [Notation](#)

- 1.1 [Key Words](#)
- 2. [Overview](#)
  - 2.1 [Scope](#)
  - 2.2 [Detailed Architecture](#)
- 3. [Metadata Service Details](#)
  - 3.1 [Metadata TOC Format](#)
    - 3.1.1 [Metadata TOC Payload Entry dictionary](#)
      - 3.1.1.1 [Dictionary `MetadataTOCPayloadEntry` Members](#)
    - 3.1.2 [StatusReport dictionary](#)
      - 3.1.2.1 [Dictionary `StatusReport` Members](#)
    - 3.1.3 [AuthenticatorStatus enum](#)
    - 3.1.4 [RogueListEntry dictionary](#)
      - 3.1.4.1 [Dictionary `RogueListEntry` Members](#)
    - 3.1.5 [Metadata TOC Payload dictionary](#)
      - 3.1.5.1 [Dictionary `MetadataTOCPayload` Members](#)
    - 3.1.6 [Metadata TOC](#)
      - 3.1.6.1 [Examples](#)
    - 3.1.7 [Metadata TOC object processing rules](#)
- 4. [Considerations](#)
- A. [References](#)
  - A.1 [Normative references](#)
  - A.2 [Informative references](#)

## 1. Notation

Type names, attribute names and element names are written as `code`.

String literals are enclosed in “”, e.g. “UAF-TLV”.

In formulas we use “|” to denote byte wise concatenation operations.

The notation `base64url(byte[8..64])` reads as 8-64 bytes of data encoded in base64url, "Base 64 Encoding with URL and Filename Safe Alphabet" [[RFC4648](#)] *without padding*.

Following [[WebIDL-ED](#)], dictionary members are optional unless they are explicitly marked as `required`.

WebIDL dictionary members **must not** have a value of null.

Unless otherwise specified, if a WebIDL dictionary member is DOMString, it **must not** be empty.

Unless otherwise specified, if a WebIDL dictionary member is a List, it **must not** be an empty list.

UAF specific terminology used in this document is defined in [[FIDOGlossary](#)].

All diagrams, examples, notes in this specification are non-normative.

### NOTE

Note: Certain dictionary members need to be present in order to comply with FIDO requirements. Such members are marked in the WebIDL definitions found in this document, as `required`. The keyword `required` has been introduced by [[WebIDL-ED](#)], which is a work-in-progress. If you are using a WebIDL parser which implements [[WebIDL](#)], then you may remove the keyword `required` from your WebIDL and use other means to ensure those fields are present.

## 1.1 Key Words

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](#)].

## 2. Overview

*This section is non-normative.*

[[FIDOMetadataStatement](#)] defines authenticator metadata statements.

These metadata statements contain the trust anchor required to verify the attestation object (more specifically the

KeyRegistrationData object), and they also describe several other important characteristics of the authenticator, including supported authentication and registration assertion schemes, and key protection flags.

These characteristics can be used when defining policies about which authenticators are acceptable for registration or authentication.

The metadata service described in this document defines a baseline method for relying parties to access the latest metadata statements.

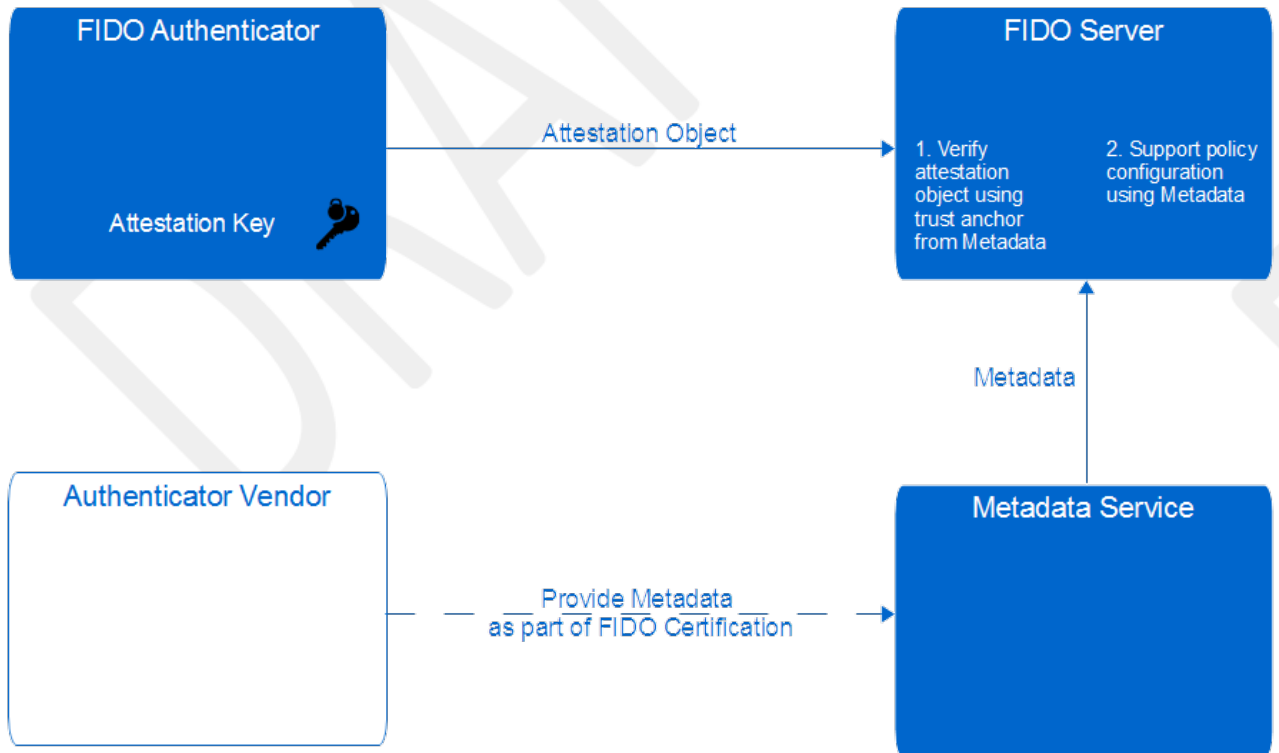


Fig. 1 FIDO Metadata Service Architecture Overview

## 2.1 Scope

This document describes the FIDO Metadata Service architecture in detail and it defines the structure and interface to access this service. It also defines the flow of the metadata related messages and presents the rationale behind the design choices.

## 2.2 Detailed Architecture

The metadata "table-of-contents" (TOC) file contains a list of metadata statements related to the authenticators known to the FIDO Alliance (FIDO Authenticators).

The FIDO Server downloads the metadata TOC file from a well-known FIDO URL and caches it locally.

The FIDO Server verifies the integrity and authenticity of this metadata TOC file using the digital signature. It then iterates through the individual entries and loads the metadata statements related to authenticator AIDs relevant to the relying party.

Individual metadata statements will be downloaded from the URL specified in the entry of the metadata TOC file, and may be cached by the FIDO Server as required.

The integrity of the metadata statements will be verified by the FIDO Server using the hash value included in the related entry of the metadata TOC file.

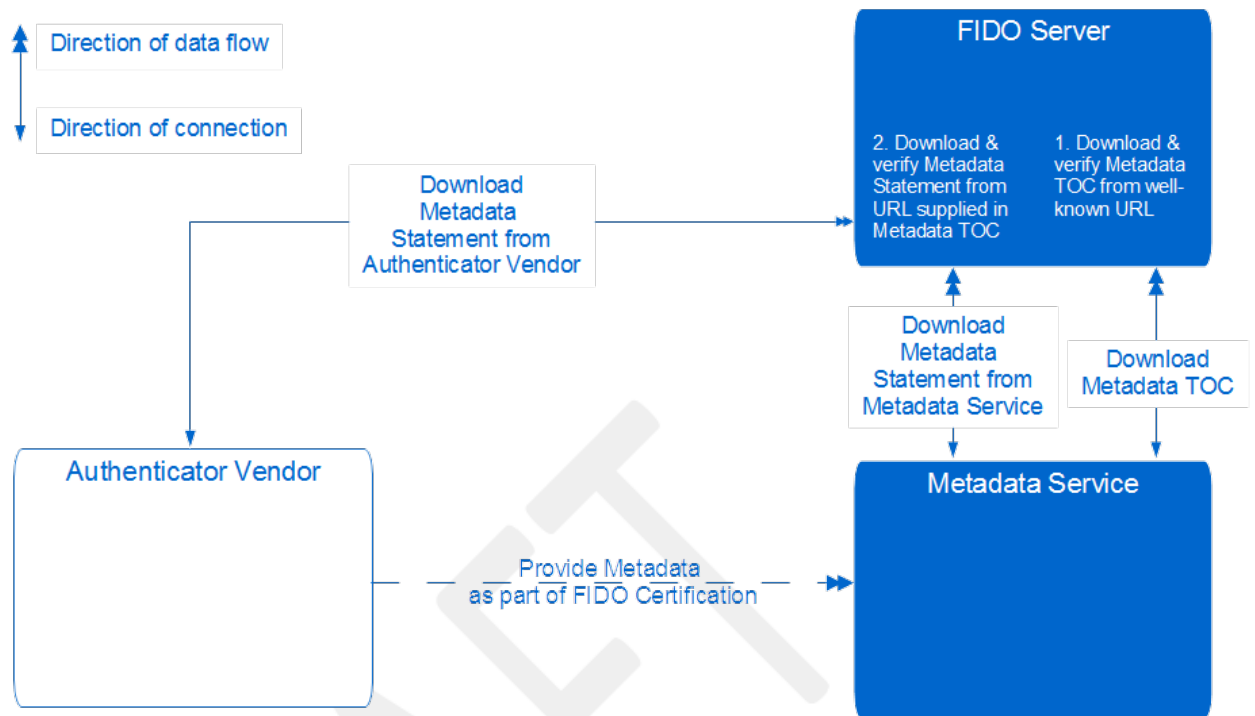


Fig. 2 FIDO Metadata Service Architecture

#### NOTE

The single arrow indicates the direction of the network connection, the double arrow indicates the direction of the data flow.

#### NOTE

The metadata TOC file is accessible at a well-known URL published by the FIDO Alliance.

#### NOTE

The relying party decides how frequently the metadata service is accessed to check for metadata TOC updates.

## 3. Metadata Service Details

*This section is normative.*

#### NOTE

The relying party can decide whether it wants to use the metadata service and whether or not it wants to accept certain authenticators for registration or authentication.

The relying party could also obtain metadata directly from authenticator vendors or other trusted sources.

### 3.1 Metadata TOC Format

#### NOTE

The metadata service makes the metadata TOC object (see [Metadata TOC](#)) accessible to FIDO Servers. This object is a "table-of-contents" for metadata, as it includes the AAID, the download URL and the hash value of the individual metadata statements. The TOC object contains one signature.

### 3.1.1 Metadata TOC Payload Entry dictionary

Represents the MetadataTOCPayloadEntry

#### WebIDL

```
dictionary MetadataTOCPayloadEntry {  
  AAID                aaid;  
  AAGUID              aaguid;  
  DOMString[]         attestationCertificateKeyIdentifiers;  
  DOMString           hash;  
  DOMString           url;  
  required StatusReport[] statusReports;  
  required DOMString  timeOfLastStatusChange;  
  DOMString           rogueListURL;  
  DOMString           rogueListHash;  
};
```

#### 3.1.1.1 Dictionary *MetadataTOCPayloadEntry* Members

##### **aaid** of type **AAID**

The AAID of the authenticator this metadata TOC payload entry relates to. See [UAFProtocol] for the definition of the AAID structure. This field **must** be set if the authenticator implements FIDO UAF.

##### NOTE

FIDO UAF authenticators support AAID, but they don't support AAGUID.

##### **aaguid** of type **AAGUID**

The Authenticator Attestation GUID. See [FIDOKeyAttestation] for the definition of the AAGUID structure. This field **must** be set if the authenticator implements FIDO 2.

##### NOTE

FIDO 2 authenticators support AAGUID, but they don't support AAID.

##### **attestationCertificateKeyIdentifiers** of type array of **DOMString**

A list of the attestation certificate public key identifiers encoded as hex string. This value **must** be calculated according to method 1 for computing the keyIdentifier as defined in [RFC5280] section 4.2.1.2. The hex string **must not** contain any non-hex characters (e.g. spaces). All hex letters **must** be lower case. This field **must** be set if neither **aaid** nor **aaguid** are set. Setting this field implies that the attestation certificate(s) are dedicated to a single authenticator model.

##### NOTE

FIDO U2F authenticators do not support AAID nor AAGUID, but they use attestation certificates dedicated to a single authenticator model.

##### **hash** of type **DOMString**

`base64url(string[1..512])`

The hash value computed over the base64url encoding of the UTF-8 representation of the JSON encoded metadata statement available at **url** and as defined in [FIDOMetadataStatement]. The hash algorithm related to the signature algorithm specified in the JWTHeader (see [Metadata TOC]) **must** be used.

If this field is missing, the metadata statement has not been published.

##### NOTE

This method of base64url encoding the UTF-8 representation is also used by JWT [JWT] to avoid encoding ambiguities.

##### **url** of type **DOMString**

Uniform resource locator (URL) of the encoded metadata statement for this authenticator model (identified by its AAID, AAGUID or attestationCertificateKeyIdentifier). This URL **must** point to the base64url encoding of the UTF-8 representation of the JSON encoded metadata statement as defined in [FIDOMetadataStatement].

If this field is missing, the metadata statement has not been published.

```
encodedMetadataStatement = base64url(utf8(JSONMetadataStatement))
```

#### NOTE

This method of the base64url encoding the UTF-8 representation is also used by JWT [JWT] to avoid encoding ambiguities.

**statusReports** of type array of [required StatusReport](#)

An array of status reports applicable to this authenticator.

**timeOfLastStatusChange** of type [required DOMString](#)

ISO-8601 formatted date since when the status report array was set to the current value.

**rogueListURL** of type [DOMString](#)

URL of a list of rogue (i.e. untrusted) individual authenticators.

**rogueListHash** of type [DOMString](#)

```
base64url(string[1..512])
```

The hash value computed over the Base64url encoding of the UTF-8 representation of the JSON encoded rogueList available at [rogueListURL](#) (with type [rogueListEntry\[\]](#)). The hash algorithm related to the signature algorithm specified in the JWTHeader (see [Metadata TOC](#)) **must** be used.

This hash value **must** be present and non-empty whenever [rogueListURL](#) is present.

#### NOTE

This method of base64url-encoding the UTF-8 representation is also used by JWT [JWT] to avoid encoding ambiguities.

### EXAMPLE 1: UAF Metadata TOC Payload

```
{ "no": 1234, "nextUpdate": "2014-03-31",
  "entries": [
    { "aaid": "1234#5678",
      "hash": "90da8da6de23248abb34da0d4861f4b30a793e198a8d5baa7f98f260db71acd4",
      "url": "https://fidoalliance.org/metadata/1234%x23abcd",
      "rogueListHash": "b5079cf40fd7ed174c645cc04df1e72b7f1229590585d16df62dd20b9541c6b5",
      "rogueListURL": "https://fidoalliance.org/metadata/1234%x23abcd.rl",
      "statusReports": [
        { status: "FIDO_CERTIFIED", effectiveDate: "2014-01-04" }
      ],
      "timeOfLastStatusChange": "2014-01-04"
    },
    { "attestationCertificateKeyIdentifiers": ["7c0903708b87115b0b422def3138c3c864e44573"],
      "hash": "785d16df640fd7b50ed174cb5645cc0f1e72b7f19cf22959052dd20b9541c64d",
      "url": "https://authnr-vendor-a.com/metadata/9876%x234321",
      "statusReports": [
        { status: "FIDO_CERTIFIED", effectiveDate: "2014-01-07" },
        { status: "UPDATE_AVAILABLE", effectiveDate: "2014-02-19",
          url: "https://example.com/update1234" }
      ],
      "timeOfLastStatusChange": "2014-02-19"
    }
  ]
}
```

#### NOTE

The character # is a reserved character and not allowed in URLs [RFC3986]. As a consequence it has been replaced by its hex value %x23.

The authenticator vendors can decide to let the metadata service publish its metadata statements or to publish metadata statements themselves. Authenticator vendors can restrict access to the metadata statements they publish themselves.

### 3.1.2 StatusReport dictionary

#### NOTE

Contains an [AuthenticatorStatus](#) and additional data associated with it, if any.

New `StatusReport` entries will be added to report known issues present in firmware updates.

The latest `StatusReport` entry **must** reflect the "current" status. For example, if the latest entry has status `USER_VERIFICATION_BYPASS`, then it is recommended assuming an increased risk associated with all authenticators of this AAID; if the latest entry has status `UPDATE_AVAILABLE`, then the update is intended to address at least all previous issues *reported* in this `StatusReport` dictionary.

#### WebIDL

```
dictionary StatusReport {  
  required AuthenticatorStatus status;  
  DOMString effectiveDate;  
  DOMString certificate;  
  DOMString url;  
  DOMString certificationDescriptor;  
  DOMString certificateNumber;  
  DOMString certificationPolicyVersion;  
  DOMString certificationRequirementsVersion;  
};
```

#### 3.1.2.1 Dictionary `StatusReport` Members

**status** of type `required AuthenticatorStatus`

Status of the authenticator. Additional fields **may** be set depending on this value.

**effectiveDate** of type `DOMString`

ISO-8601 formatted date since when the status code was set, if applicable. If no date is given, the status is assumed to be effective while present.

**certificate** of type `DOMString`

Base64-encoded [RFC4648] (not base64url!) DER [ITU-X690-2008] PKIX certificate value related to the current status, if applicable.

#### NOTE

As an example, this could be an Attestation Root Certificate (see [FIDOMetadataStatement]) related to a set of compromised authenticators (ATTESTATION\_KEY\_COMPROMISE).

**url** of type `DOMString`

HTTPS URL where additional information may be found related to the current status, if applicable.

#### NOTE

For example a link to a web page describing an available firmware update in the case of status `UPDATE_AVAILABLE`, or a link to a description of an identified issue in the case of status `USER_VERIFICATION_BYPASS`.

**certificationDescriptor** of type `DOMString`

Describes the externally visible aspects of the Authenticator Certification evaluation.

**certificateNumber** of type `DOMString`

The unique identifier for the issued Certification

**certificationPolicyVersion** of type `DOMString`

The version of the Authenticator Certification Policy the implementation is Certified to, e.g. "1.0.0".

**certificationRequirementsVersion** of type `DOMString`

The version of the Authenticator Security Requirements the implementation is Certified to, e.g. "1.0.0".

#### 3.1.3 AuthenticatorStatus enum

This enumeration describes the status of an authenticator model as identified by its AAID and potentially some additional information (such as a specific attestation key).

#### WebIDL

```
enum AuthenticatorStatus {  
  "NOT_FIDO_CERTIFIED",  
  "FIDO_CERTIFIED",  
  "USER_VERIFICATION_BYPASS",  
};
```

```

"ATTESTATION_KEY_COMPROMISE",
"USER_KEY_REMOTE_COMPROMISE",
"USER_KEY_PHYSICAL_COMPROMISE",
"UPDATE_AVAILABLE",
"REVOKED",
"SELF_ASSERTION_SUBMITTED",
"FIDO_CERTIFIED_L1",
"FIDO_CERTIFIED_L2",
"FIDO_CERTIFIED_L3",
"FIDO_CERTIFIED_L4",
"FIDO_CERTIFIED_L5"
};

```

## Enumeration description

<code>NOT_FIDO_CERTIFIED</code>	This authenticator is not FIDO certified.
<code>FIDO_CERTIFIED</code>	This authenticator has passed FIDO functional certification. This certification scheme is phased out and will be replaced by <code>FIDO_CERTIFIED_L1</code> .
<code>USER_VERIFICATION_BYPASS</code>	Indicates that malware is able to bypass the user verification. This means that the authenticator could be used without the user's consent and potentially even without the user's knowledge.
<code>ATTESTATION_KEY_COMPROMISE</code>	Indicates that an attestation key for this authenticator is known to be compromised. Additional data should be supplied, including the key identifier and the date of compromise, if known.
<code>USER_KEY_REMOTE_COMPROMISE</code>	This authenticator has identified weaknesses that allow registered keys to be compromised and should not be trusted. This would include both, e.g. weak entropy that causes predictable keys to be generated or side channels that allow keys or signatures to be forged, guessed or extracted.
<code>USER_KEY_PHYSICAL_COMPROMISE</code>	This authenticator has known weaknesses in its key protection mechanism(s) that allow user keys to be extracted by an adversary in physical possession of the device.
<code>UPDATE_AVAILABLE</code>	<p>A software or firmware update is available for the device. Additional data should be supplied including a URL where users can obtain an update and the date the update was published.</p> <p>When this code is used, then the field <code>authenticatorVersion</code> in the metadata Statement [<a href="#">FIDOMetadataStatement</a>] <b>must</b> be updated, if the update fixes severe security issues, e.g. the ones reported by preceding StatusReport entries with status code <code>USER_VERIFICATION_BYPASS</code>, <code>ATTESTATION_KEY_COMPROMISE</code>, <code>USER_KEY_REMOTE_COMPROMISE</code>, <code>USER_KEY_PHYSICAL_COMPROMISE</code>, <code>REVOKED</code>.</p> <div style="border-left: 3px solid green; padding-left: 10px; margin-top: 10px;"> <p><b>NOTE</b></p> <p>Relying parties might want to inform users about available firmware updates.</p> </div>
<code>REVOKED</code>	The FIDO Alliance has determined that this authenticator should not be trusted for any reason, for example if it is known to be a fraudulent product or contain a deliberate backdoor.
<code>SELF_ASSERTION_SUBMITTED</code>	The authenticator vendor has completed and submitted the self-certification checklist to the FIDO Alliance. If this completed checklist is publicly available, the URL will be specified in <code>StatusReport.url</code> .
<code>FIDO_CERTIFIED_L1</code>	The authenticator has passed FIDO Authenticator certification at level 1. This level is the more strict successor of <code>FIDO_CERTIFIED</code> .
<code>FIDO_CERTIFIED_L2</code>	The authenticator has passed FIDO Authenticator certification at level 2. This level is more strict than level 1.
<code>FIDO_CERTIFIED_L3</code>	The authenticator has passed FIDO Authenticator certification at level 3. This level is more strict than level 2.
<code>FIDO_CERTIFIED_L4</code>	The authenticator has passed FIDO Authenticator certification at level 4. This level is more strict than level 3.
<code>FIDO_CERTIFIED_L5</code>	The authenticator has passed FIDO Authenticator certification at level 5. This level is more strict than level 4.

More values might be added in the future. FIDO Servers **must** silently ignore all unknown `AuthenticatorStatus` values.

### 3.1.4 RogueListEntry dictionary



## NOTE

Contains a list of individual authenticators known to be rogue.

New `RogueListEntry` entries will be added to report new individual authenticators known to be rogue.

Old `RogueListEntry` entries will be removed if the individual authenticator is known to not be rogue any longer.

### WebIDL

```
dictionary RogueListEntry {  
  required DOMString sk;  
  required DOMString date;  
};
```

#### 3.1.4.1 Dictionary `RogueListEntry` Members

**sk** of type `required DOMString`

Base64url encoding of the rogue authenticator's secret key (sk value, see [FIDOEcdaaAlgorithm], section ECDA A Attestation).

## NOTE

In order to revoke an individual authenticator, its secret key (sk) must be known.

**date** of type `required DOMString`

ISO-8601 formatted date since when this entry is effective.

#### EXAMPLE 2: `RogueListEntry[]` example

```
[  
  { "sk": "30efa86aa6de25249acb35da0d4861f4b30a793e198a8d5baa7e96f240da51f3",  
    "date": "2016-06-07" },  
  { "sk": "93de8da6de23248abb34da0d4861f4b30a793e153a8d5bb27f98f260db71acd4",  
    "date": "2016-06-09" },  
]
```

#### 3.1.5 Metadata TOC Payload dictionary

Represents the `MetadataTOCPayload`

### WebIDL

```
dictionary MetadataTOCPayload {  
  DOMString legalHeader;  
  required Number no;  
  required DOMString nextUpdate;  
  required MetadataTOCPayloadEntry[] entries;  
};
```

##### 3.1.5.1 Dictionary `MetadataTOCPayload` Members

**legalHeader** of type `DOMString`

The `legalHeader`, if present, contains a legal guide for accessing and using metadata, which itself **may** contain URL(s) pointing to further information, such as a full Terms and Conditions statement.

**no** of type `required Number`

The serial number of this UAF Metadata TOC Payload. Serial numbers **must** be consecutive and strictly monotonic, i.e. the successor TOC will have a `no` value exactly incremented by one.

**nextUpdate** of type `required DOMString`

ISO-8601 formatted date when the next update will be provided at latest.

**entries** of type array of `required MetadataTOCPayloadEntry`

List of zero or more `MetadataTOCPayloadEntry` objects.

#### 3.1.6 Metadata TOC

The metadata table of contents (TOC) is a JSON Web Token (see [JWT](#) and [JWS](#)).

It consists of three elements:

- The base64url encoding, without padding, of the UTF-8 encoded JWT Header (see example below),
- the base64url encoding, without padding, of the UTF-8 encoded UAF Metadata TOC Payload (see example at the beginning of section [Metadata TOC Format](#)),
- and the base64url-encoded, also without padding, JWS Signature [[JWS](#)] computed over the to-be-signed payload using the Metadata TOC signing key, i.e.

```
tbsPayload = EncodedJWTHeader | "." | EncodedMetadataTOCPayload
```

All three elements of the TOC are concatenated by a period ("."):

```
MetadataTOC = EncodedJWTHeader | "." | EncodedMetadataTOCPayload | "." | EncodedJWSSignature
```

The hash algorithm related to the signing algorithm specified in the JWT Header (e.g. SHA256 in the case of "ES256") **must** also be used to compute the hash of the metadata statements (see section [Metadata TOC Payload Entry Dictionary](#)).

### 3.1.6.1 Examples

This section is non-normative.

#### EXAMPLE 3: Encoded Metadata Statement

```
eyAiOUFJRci6ICiXmJm0IzU2NzgiLAOKICaiQXR0ZXN0YXRpb25Sb290Q2VydGlmawNhdGUiOiAi
TULJQ1BUQ0NBZU9nQXdJQkFnSUpBT3VleHVM095MndNQW9HQ0Nxr1NND1CQU1DTUhzEElEQWVC
Z05WQkFNTQ0KRjFoAGJYQnNaU0JCZehSbGmZUmhkR2x2YmLCU2IyOTBNU1l3RkFZRFZRUUtEQTFH
U1VSUElFRnNiR2xoYm1ObA0KTvJfD0R3WURWUvFMREFoVlFVwWdWRmRITERFU01CQUdBMVVFQnd3
S1VHRnNieUJCYkhsdk1Rc3ddUV1EV1FRSQ0KREFKRFURUxNQWtHQTFVFRUJoTUNWVkl3SGhjTk1U
UXdoakU0TVRnek16TXlXaGNOTkRFeE1UQXpNVE16TXpNeQ0KV2pCN01TQXdiZ1lEV1FRFRERCZFRZ
VzF3YkdVZ1FYUjBawE4wVwHscGiYNGdVbT12ZERFV01CUUdBMVVFQ2d3Tg0KUmtsRVR5QkjiR3hw
wci1alpURVJNQThHQTFVRUN3d01WVUZHSUzSWFJ5d3hfakFRQmdOVkjbY01DVkJoYkc4Zw0KUVd4
MGJ6RUxNQWtHQTFVRUNBd0NRMEV4Q3pBSkInTlZCQVlUQWxwVE1Ga3dFd1lIS29aSXpqMENBUVlJ
S29aSQ0KemowREFRY0RRZ0FFSDhodjJEMehYyTU5L0JtcFE3U1plaEwvRk1HekZkMVFCzZ12QVvV
TlozYWpudVE5NFBSNw0KYU16SDMzblVTQnI4ZkhZRHjxt0JiNThweEdxSEpSeVgvNk5RTUu0d0hR
WURWUjBpQkZJRuzQb0hBM0NMaHhGyG0KQzBJdDd6RTR3OghrNUVkl01COEdBMVVKsXDRWU1CYUFG
UG9IQTNDTGh4RmJDE10N3pFNhc4aGs1RUovTUF3Rw0KQTFVZEV3UUZNU1CQWY4d0NnWU1Lb1pJ
emowRUF3SURTQUF3U1FJaEFKMDZRU1h0OWloSWJFS11LSWpZUGtyaQ0KvMRMSWd0ZnNiRfN1N0Yy
SmZ6cjRbaUJxbl1DWmYwK3pJNTVhUwVBSGpJekE5WG02M3JydUF4Ql05cHM5ejJYTg0KbFE9PSIs
DQogICJCEZXXjcmldwG1vbiI6ICJGSURPIEFsbG1hbmN1IFNhbXBsZSBVQUYgQXV0aGVudG1jYXRv
ciIsDQogICJvc2VyVmVyaWZpY2F0aW9uTWV0aG9kcyI6IDIsDQogICJwYXN0aGVudG1jYXRv
eXB1cyI6IDEsDQogICJlZlZlQm90ZWN0aW9uIjogNiwiNCiAgIklhdGNoZkZlQm90ZWN0aW9uIjog
MiwiNCiAgI1NlY3VyZURpc3BsYXkiOiA0LAOKICaiU2VjdXJlRGlzcgXheUNvbnR1bnRUeXB1cyI6
IFsiaW1hZ2UvcG5nI10sDQogICJlZlZlQm90ZWN0aW9uIjogNiwiNCiAgIklhdGNoZkZlQm90ZWN0
LDAsMSw2NCwWLDAsMSwYmJjQsMTYsMiwwLDAsMF1dLAOKICaiaXNTZWNvbmlRyYWN0b3JpYm90Ijog
ImZhbHN1IiwNCiAgIkljb24iOiA0LAOKICaiU2VjdXJlRGlzcgXheUNvbnR1bnRUeXB1cyI6
IzZDRFdHTkVtZEFzaWR3Z2dnSkJpUmlNaEzjLzR3eTgNCjg4NHp1OU5k
bG5HVGVZaSa1AjbjNuTysrODg5MzNmndmVCQng9UHfDekprVFV2QmJmXbXBRFd2Q1RjBjBjQ1Nad1hM
Q2RYOV1wNvNrMTkcmJiNWf0ZjU5OWZHkY1lcke1NDfXNDhdUDFMTFzHOVNJeVzOVWk4SWk4ZDVr
1RzaTmWtkZ2N2Fp0W43UVpQTxdizhLzmmVvVTJYTXENC1VkeTgrWmNhTm1Haw1FOH1YTjNSVWQz
YTE4bkYwZlVsb3ZaKzBDVHPXcGQyVmorZU9tMwJfexk2RHg0atVwVU1HV3ZlzbUwNnEyMjcnCmR0
dVdCSXVznI2b1dwVjBUE5MaG93MTc1MU5tMjFmDlBIM3JWfdqZno2NkxmcWw4dFg3R1JsoV1G
U1hzbVNZW15Y2VPRZJzazcNk1OVVNHUGe4WnNiTWU5cmZRVWFhV19KTVg5c3FkekRDU3ZwMwG4
SG1UWmc5eddiTEhJTW5UaGixNmVKK21WZlFxoH1hVvPRTkc2NGkNClhaKzAva3E2dU9aRk8wUXRr
dGRXS2ZyblJROTHlCajkvjVPSUZuazU0ak4wbWtVaXfStZNYRfcrTWwroThs0I2dfc3cldwWmNQ
YysncjB6Zr0THlZbFvJODZFNmVHRGpJTXv1VnBjdxNL1YXJmZ01Zr1JrNmJyaFpWci9KY0h6b29M
NzU1MGplZEExFeG9wV2NbcGkyW1VxaHUNCjdKTHZYvNRRVTgxemt6T1B1Zw1NU1L2VnVRC1g3UGJp
RFFZNUp2Wm9uZnRLKzFwwThIOXV0eDUzMGgwb2Iram1SWXfQNm91YV12RWUNCm5XL1dsWpWpOGN3
Yk1tNjgydFB3cVpXrJf0a18yU0gxM01SS11NG1vWnZyCglTcURYN2RYdfIeGeVEueszLytCV3NL
MWRUZ0h1N1YncjhoU0UozYndGa3dwrNjVT1E1MHMxcjNsXZzTOHpaY3ExNytcQmF3N0s4bEVLNFX6
a11lYXrOUe4cDdQM0d6REsrbmQzRFFvdys2VUMNcjhtVvk44Mml1djm4aw03TnrhWHRWmUNWctZS
Z3c0cGtzbbJkaTnidTJEZTdzZmFCQnhjcwZ2cVByVWpGUU5UUTIybGzKvVZWVdY4clQNCkpLRjve
blntVWpnZHFhNG1TUz1wbXNmRepSM0c2VG9IMGLXOWFWN0xXTEhZWEtsbFREdDBMVEF0a11JYwft
cDFRalZ2Kyt1eUdVeFYncKRMKEROVlhTbstiMXfSEhBsODRkZGZYMUxwM8vZDY5dHnvZDB2czVo
R3J10Xh10e8rznBmUjYfR2rhOVEQ2WjU3Qz1lTVdYzWZkZE8NC105NGJiOW9xzDFST25TN3JfVFR6
SG1tTXfPdmJPM2cWRGRWesZV1FCaEJ6dEsZNV1LTmRPbmM4TzNhY1M2ZkRaRmdLYVhMc0VKdDUN
CnJkcmxpQnFwOD1jSmNzL203VHZzMHJrakmdTjRiMGtQb1puM1VKDulPcm5aMjJ5UDFmbXZVeCtP
NwdTocviviVjFtK3PtdVl0VmhxN1QNCldiRklMvNzsanBsTGxvcdZDTFHqKzJxdHzHTE1MLZf2aW1J
U2RNQmd6U29Gwn11N1RxcZctqenhnc1BhVj1CQ3F1ZS90a11rNnY2bEsNCj1jd21VYy9TVhRmMUHe
ce0zYjU5Mnk3ADNUaHg1b3pLNj1ITHBV3Vbd2FuxUzvdjI2cTdjZWI4ZWZWWWFSZVAzaUZVOHPq
MwtuU3cMclpYSE1tbbkNwTPBZ2FsbzdVUWZTQ00zcVFRcjJIL1hgUddzc1h4NDVZbdkxQn1LQ2Vw
NG1vWm9IKzEmRzN4RDR0Vdd4OGt3eWo4bncNcmI5ZXyN1YwQjZkKzdINhpLdnVkuQUg1MzdGanF5
ek9IzEpuSeVU1em1YcS9XanhPYnZOTWJ2N25oeXdxWDJhVnNXdEM4KzQ4YUx1YXANCkU3cdV3S1pp
MEEYQVFSVjVd110Rst1SmMrYjYxa0FwcU1sdueEJnbWQvNFY1UVAvbXQxOeHQzdzUkhmdG1ldTjw
bwhWMHJuLOFMWD1NCjmyYnFkNEJGbkR4N1ZpMWNXUzJ1ZmYwSWJcNDdxZxh4bVvQOVF1dF1gdxKb
M3RZRDZHyldCQk1yaCthcE5i10tyTkYxK3VnQ2E0cmkNClhHZndNUFB0Vm1hdhmVm11NT0FBbnVY
i9SMDMHM1PU2PvYRFRWFBZbcHcNyrKdmZjMjwE5obEpnTTUxQ1U2dk45RkxpnbnB2SEJGVXkNCmlw
cmFlUG13SjUzREY1W1Rabm9tRU5nODVrT1VkmM9KaTJXcHI0T21ta2ZONHg0ekhmvZGyZhedjho
enVoTnFvYRpbEd2QTZER3UNCMvad083OEFBUW42Y21FazYrcnc1VmN2anzXtkRZUE9vsvV3YUtT
ahJ4QXVYTeGrSDRHwXVhZ1ZRGmXMFdGNvRhmZFoEUpZmNVAhINC1UvSmxJTkM2YzZ1bFJZzEJw
bzYrK1lmanq2MwXhTmZSBTRNRDVySjFqM0zVR0huakRTQk5hc11VZ01MeU1zektwYjd0WBHvSGZQ
czgNCmgzV3AxThpOzk5rNTRYeEMxd0RHVW1ze1hZzWz0NnovY0t0Vm00RUJ4YT1WUUDBe1lyM0xy
```



The FIDO Server **must** follow these processing rules:

1. The FIDO Server **must** be able to download the latest metadata TOC object from the well-known URL, when appropriate. The `nextUpdate` field of the [Metadata TOC](#) specifies a date when the download **should** occur at latest.
  2. If the `x5u` attribute is present in the JWT Header, then:
    1. The FIDO Server **must** verify that the URL specified by the `x5u` attribute has the same web-origin as the URL used to download the metadata TOC from. The FIDO Server **should** ignore the file if the web-origin differs (in order to prevent loading objects from arbitrary sites).
    2. The FIDO Server **must** download the certificate (chain) from the URL specified by the `x5u` attribute [JWS]. The certificate chain **must** be verified to properly chain to the metadata TOC signing trust anchor according to [RFC5280]. All certificates in the chain **must** be checked for revocation according to [RFC5280].
    3. The FIDO Server **should** ignore the file if the chain cannot be verified or if one of the chain certificates is revoked.
  3. If the `x5u` attribute is missing, the chain should be retrieved from the `x5c` attribute. If that attribute is missing as well, Metadata TOC signing trust anchor is considered the TOC signing certificate chain.
  4. Verify the signature of the Metadata TOC object using the TOC signing certificate chain (as determined by the steps above). The FIDO Server **should** ignore the file if the signature is invalid. It **should** also ignore the file if its number (`no`) is less or equal to the number of the last Metadata TOC object cached locally.
  5. Write the verified object to a local cache as required.
  6. Iterate through the individual entries (of type `MetadataTOCPayloadEntry`). For each entry:
    1. Ignore the entry if the AAID, AAGUID or attestationCertificateKeyIdentifiers is not relevant to the relying party (e.g. not acceptable by any policy)
    2. Download the metadata statement from the URL specified by the field `url`. Some authenticator vendors might require authentication in order to provide access to the data. Conforming FIDO Servers **should** support the HTTP Basic, and HTTP Digest authentication schemes, as defined in [RFC2617].
    3. Check whether the status report of the authenticator model has changed compared to the cached entry by looking at the fields `timeOfLastStatusChange` and `statusReport`. Update the status of the cached entry. It is up to the relying party to specify behavior for authenticators with status reports that indicate a lack of certification, or known security issues. However, the status `REVOKED` indicates significant security issues related to such authenticators.
4. Compute the hash value of the (base64url encoding without padding of the UTF-8 encoded) metadata statement downloaded from the URL and verify the hash value to the hash specified in the field `hash` of the metadata TOC object. Ignore the downloaded metadata statement if the hash value doesn't match.
5. Update the cached metadata statement according to the downloaded one.

#### NOTE

Authenticators with an unacceptable status should be marked accordingly. This information is required for building registration and authentication policies included in the registration request and the authentication request [UAFProtocol].

## 4. Considerations

*This section is non-normative.*

This section describes the key considerations for designing this metadata service.

**Need for Authenticator Metadata** When defining policies for acceptable authenticators, it is often better to describe the required authenticator characteristics in a generic way than to list individual authenticator AAIDs. The metadata statements provide such information. Authenticator metadata also provides the trust anchor required to verify attestation objects.

The metadata service provides a standardized method to access such metadata statements.

**Integrity and Authenticity** Metadata statements include information relevant for the security. Some business verticals might even have the need to document authenticator policies and trust anchors used for verifying attestation objects for auditing purposes.

It is important to have a strong method to verify and proof integrity and authenticity and the freshness of metadata statements. We are using a single digital signature to protect the integrity and authenticity of the Metadata TOC object and we protect the integrity and authenticity of the individual metadata statements by including their cryptographic hash values into the Metadata TOC object. This allows for flexible distribution of the metadata statements and the Metadata TOC object using standard content distribution networks.

**Organizational Impact** Authenticator vendors can delegate the publication of metadata statements to the metadata service in its entirety. Even if authenticator vendors choose to publish metadata statements themselves, the effort is very limited as the metadata statement can be published like a normal document on a website. The FIDO Alliance has control over the FIDO certification process and receives the metadata as part of that process anyway. With this

metadata service, the list of known authenticators needs to be updated, signed and published regularly. A single signature needs to be generated in order to protect the integrity and authenticity of the metadata TOC object.

**Performance Impact** Metadata TOC objects and metadata statements can be cached by the FIDO Server.

The update policy can be specified by the relying party.

The metadata TOC object includes a date for the next scheduled update. As a result there is *no additional impact* to the FIDO Server during FIDO Authentication or FIDO Registration operations.

Updating the Metadata TOC object and metadata statements can be performed asynchronously. This reduces the availability requirements for the metadata service and the load for the FIDO Server.

The metadata TOC object itself is relatively small as it does not contain the individual metadata statements. So downloading the metadata TOC object does not generate excessive data traffic.

Individual metadata statements are expected to change less frequently than the metadata TOC object. Only the modified metadata statements need be downloaded by the FIDO Server.

**Non-public Metadata Statements** Some authenticator vendors might want to provide access to metadata statements only to their subscribed customers.

They can publish the metadata statements on access protected URLs. The access URL and the cryptographic hash of the metadata statement is included in the metadata TOC object.

**High Security Environments** Some high security environments might only trust internal policy authorities. FIDO Servers in such environments could be restricted to use metadata TOC objects from a proprietary trusted source only. The metadata service is the baseline for most relying parties.

**Extended Authenticator Information** Some relying parties might want additional information about authenticators before accepting them. The policy configuration is under control of the relying party, so it is possible to only accept authenticators for which additional data is available and meets the requirements.

## A. References

### A.1 Normative references

#### [FIDOMetadataStatement]

B. Hill; D. Baghdasaryan; J. Kemp. *FIDO Metadata Statements v1.0*. Implementation Draft. URL: <https://fidoalliance.org/specs/fido-v2.0-rd-20170927/fido-metadata-statement-v2.0-rd-20170927.html>

#### [JWS]

M. Jones; J. Bradley; N. Sakimura. *JSON Web Signature (JWS)*. May 2015. RFC. URL: <https://tools.ietf.org/html/rfc7515>

#### [JWT]

M. Jones; J. Bradley; N. Sakimura. *JSON Web Token (JWT)*. May 2015. RFC. URL: <https://tools.ietf.org/html/rfc7519>

#### [RFC4648]

S. Josefsson. *The Base16, Base32, and Base64 Data Encodings (RFC 4648)*. October 2006. URL: <http://www.ietf.org/rfc/rfc4648.txt>

#### [RFC5280]

D. Cooper; S. Santesson; S. Farrell; S. Boeyen; R. Housley; W. Polk. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. May 2008. URL: <http://www.ietf.org/rfc/rfc5280.txt>

#### [WebIDL-ED]

Cameron McCormack. *Web IDL*. 13 November 2014. Editor's Draft. URL: <http://heycam.github.io/webidl/>

### A.2 Informative references

#### [FIDOEcdaaAlgorithm]

R. Lindemann; J. Camenisch; M. Drijvers; A. Edgington; A. Lehmann; R. Urian. *FIDO ECDA A Algorithm*. Implementation Draft. URL: <https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-ecdaa-v1.1-id-20170202.html>

#### [FIDOGlossary]

R. Lindemann; D. Baghdasaryan; B. Hill; J. Hodges. *FIDO Technical Glossary*. Implementation Draft. URL: <https://fidoalliance.org/specs/fido-v2.0-rd-20170927/fido-glossary-v2.0-rd-20170927.html>

#### [FIDOKeyAttestation]

. *FIDO 2.0: Key attestation format*. URL: <https://fidoalliance.org/specs/fido-v2.0-ps-20150904/fido-key-attestation-v2.0-ps-20150904.html>

#### [ITU-X690-2008]

. *X.690: Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*. (T-REC-X.690-200811). November 2008. URL: <http://www.itu.int/rec/T-REC-X.690-200811-I/en>

#### [RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels* March 1997. Best Current Practice. URL: <https://tools.ietf.org/html/rfc2119>

#### [RFC2617]

J. Franks; P. Hallam-Baker; J. Hostetler; S. Lawrence; P. Leach; A. Luotonen; L. Stewart. *HTTP Authentication: Basic and Digest Access Authentication*. June 1999. Draft Standard. URL:

<https://tools.ietf.org/html/rfc2617>

**[RFC3986]**

T. Berners-Lee; R. Fielding; L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax* January 2005. Internet Standard. URL: <https://tools.ietf.org/html/rfc3986>

**[UAFProtocol]**

R. Lindemann; D. Baghdasaryan; E. Tiffany; D. Balfanz; B. Hill; J. Hodges. *FIDO UAF Protocol Specification v1.0*. Proposed Standard. URL: <https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-protocol-v1.1-id-20170202.html>

**[WebIDL]**

Cameron McCormack; Boris Zbarsky; Tobie Langel. *Web IDL*. 15 December 2016. W3C Editor's Draft. URL: <https://heycam.github.io/webidl/>