FIDO UAF Authenticator Metadata Service v1.0

FIDO Alliance Proposed Standard 08 December 2014

This version:
https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-authnr-metadata-service-v1.0-ps-20141208.html
Previous version:
https://fidoalliance.org/specs/fido-uaf-metadata-service-v1.0-rd-20141008.pdf
Editor:
Rolf Lindemann, Nok Nok Labs, Inc.
Contributors:
Brad Hill, PayPal, Inc.
Davit Baghdasaryan, Nok Nok Labs, Inc.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

The FIDO UAF 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 at https://www.fidoalliance.org/specifications/.

This document was published by the FIDO Alliance as a Proposed Standard. If you wish to make comments regarding this document, please Contact Us. All comments are welcome.

This document has been reviewed by FIDO Aliance Members and is endorsed as a Proposed Standard. It is a stable document and may be used as reference material or cited from another document. FIDO Alliance's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment.

Table of Contents

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 MST 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.

The FIDO UAF specification defines Authenticator Metadata Statements [UAFAuthnrMetadata].

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 its AAID, supported authentication and registration assertion schemes, and key protection flags.

These characteristics can be used when defining policies [UAFProtocol] 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.

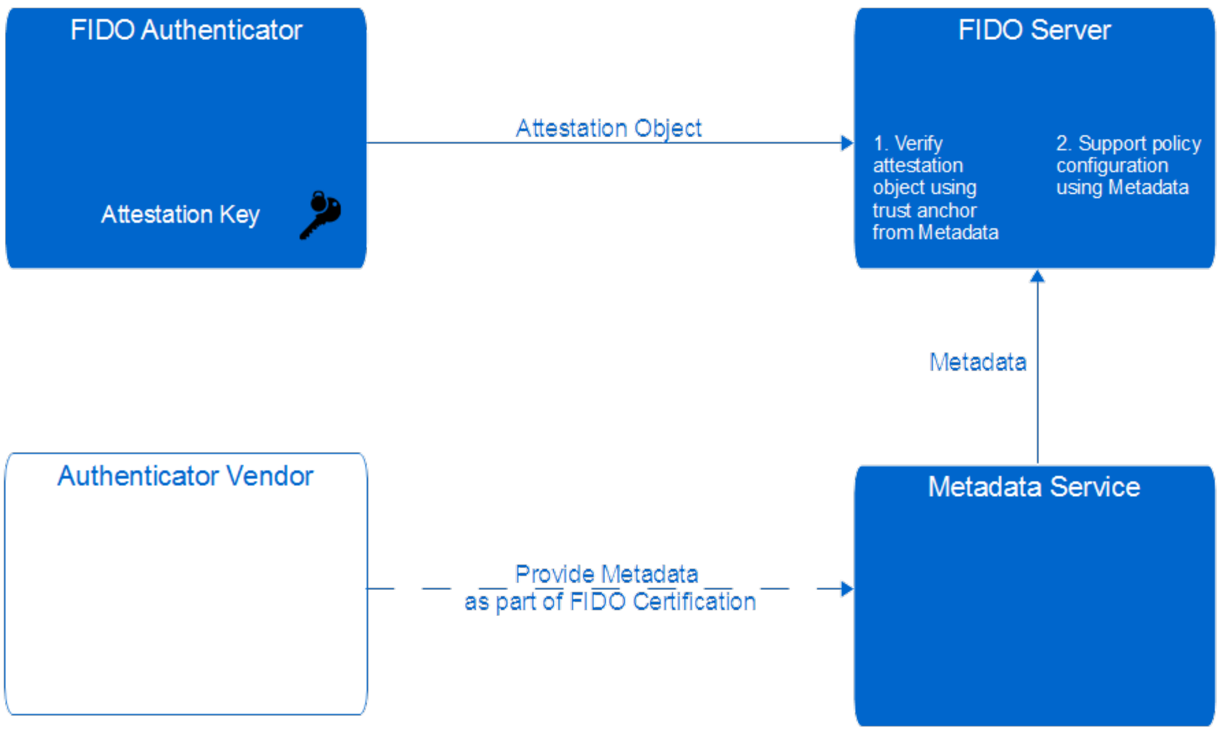
FIDO Authenticator
Attestation Key
Attestation Object
FIDO Server
1. Verify attestation object using trust anchor from Metadata
2. Support policy configuration using Metadata
Metadata
Authenticator Vendor
Provide Metadata as part of FIDO Certification
Metadata Service

Fig. 1 UAF 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 AAIDs relevant to the relying party.

Individual metadata statements will be downloaded from the URL specified in the entry of the 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.

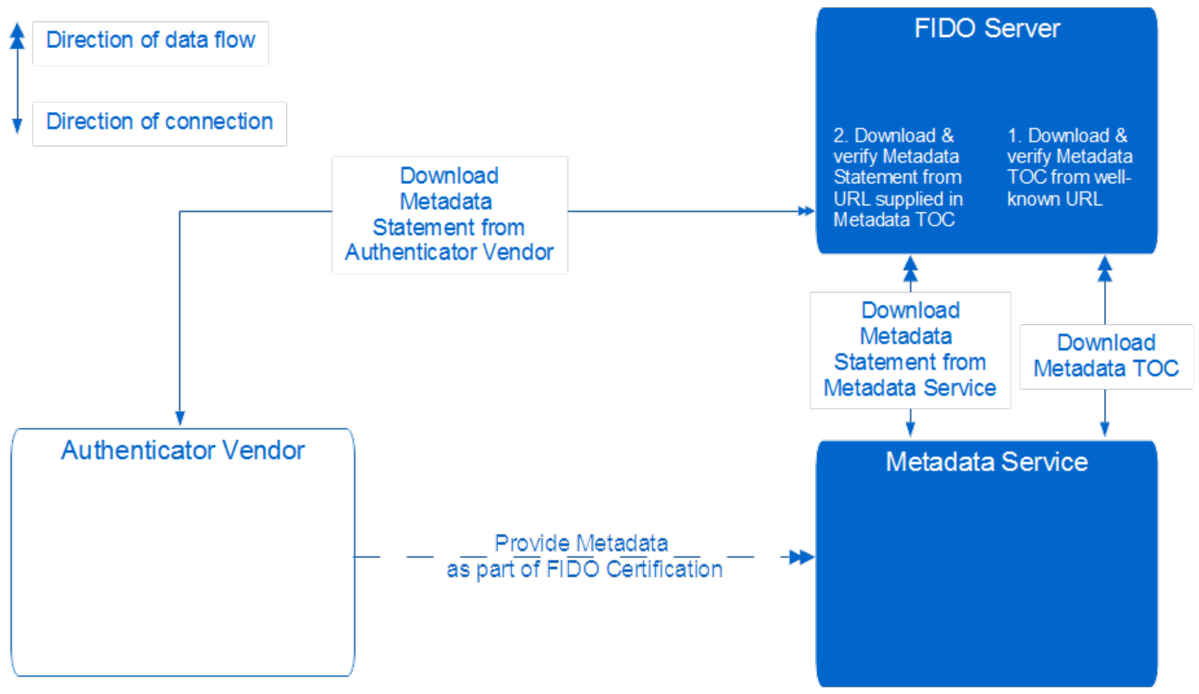
Direction of data flow
Direction of connection
FIDO Server
2. Download & verify Metadata Statement from URL supplied in Metadata TOC
1. Download & verify Metadata TOC from well-known URL
Download Metadata Statement from Authenticator Vendor
Download Metadata Statement from Metadata Service
Download Metadata TOC
Authenticator Vendor
Metadata Service
Provide Metadata as part of FIDO Certification

Fig. 2 UAF 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 freely accessible at a well-known URL published by the FIDO Alliance.

NOTE

The relying party decides how frequently the metadata rervice 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 {
    required AAID           aaid;
    required DOMString      hash;
    required DOMString      url;
    required StatusReport[] statusReports;
    required DOMString      timeOfLastStatusChange;
};
```

3.1.1.1 Dictionary `MetadataTOCPayloadEntry` Members

`aaid` of type required AAID
: The AAID of the authenticator this metadata TOC payload entry relates to. See [UAFProtocol] for the definition of the AAID structure.

`hash` of type required 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 [UAFAuthnrMetadata]. The hash algorithm related to the signature algorithm specified in the JWTHeader (see Metadata TOC) must be used.

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

`url` of type required DOMString
: Uniform resource locator (URL) of the encoded metadata statement for this authenticator model (identified by its AAID). This URL must point to the base64url encoding of the UTF-8 representation of the JSON encoded Metadata Statement as defined in [UAFAuthnrMetadata].

`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.

EXAMPLE 1: UAF Metadata TOC Payload

```
{ "no": 1234, "next-update": "2014-03-31",
  "entries": [
   { "aaid": "1234#5678",
     "hash": "90da8da6de23248abb34da0d4861f4b30a793e198a8d5baa7f98f260db71acd4",
     "url": "https://fidoalliance.org/metadata/1234%x23abcd",
     "statusReports": [
                         { status: "FIDO_CERTIFIED", effectiveDate: "2014-01-04"}
                       ],
     "timeOfLastStatusChange": "2014-01-04"
     },
   { "aaid": "9876#4321",
     "hash": "785d16df640fd7b50ed174cb5645cc0f1e72b7f19cf22959052dd20b9541c64d",
     "url": "https://authnr-vendor-a.com/metadata/9876%x234321",
     "statusReports": [
                         { status: "FIDO_CERTIFIED", effectiveDate: "2014-01-07"},
                         { status: "UPDATE_AVAILABLE", effectiveDate: "2014-03-08",
                           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;
};
```

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 [UAFAuthnrMetadata]) 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.

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 {
    "FIDO_CERTIFIED",
    "NOT_FIDO_CERTIFIED",
    "USER_VERIFICATION_BYPASS",
    "ATTESTATION_KEY_COMPROMISE",
    "USER_KEY_REMOTE_COMPROMISE",
    "USER_KEY_PHYSICAL_COMPROMISE",
    "UPDATE_AVAILABLE",
    "REVOKED"
};
```

Enumeration description	
`FIDO_CERTIFIED`	This authenticator is FIDO certified.
`NOT_FIDO_CERTIFIED`	This authenticator is not FIDO certified.
`USER_VERIFICATION_BYPASS`	Indicates that malware is able to bypass the user verification. This means that the authenticator could be used without user's consent and potentially even without user's knowledge.
`ATTESTATION_KEY_COMPROMISE`	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.
`USER_KEY_REMOTE_COMPROMISE`	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.
`USER_KEY_PHYSICAL_COMPROMISE`	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.
`UPDATE_AVAILABLE`	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. When this code is used, then the field `authenticatorVersion` in the metadata Statement [UAFAuthnrMetadata] must be updated, if the update fixes severe security issues, e.g. the ones reported by preceding StatusReport entries with status code `USER_VERIFICATION_BYPASS`, `ATTESTATION_KEY_COMPROMISE`, `USER_KEY_REMOTE_COMPROMISE`, `USER_KEY_PHYSICAL_COMPROMISE`, `REVOKED`. NOTE Relying parties might want to inform users about available firmware updates.
`REVOKED`	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.

3.1.4 Metadata TOC Payload Dictionary

Represents the MetadataTOCPayload

WebIDL

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

3.1.4.1 Dictionary `MetadataTOCPayload` Members

`no` of type required Number
: The serial number of this UAF Metadata TOC Payload. Serial numbers must be consecutive and stricly monotonical, 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.

3.1.5 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, 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).

3.1.5.1 Examples

This section is non-normative.

EXAMPLE 2: Encoded Metadata Statement

```
eyAiQUFJRCI6ICIxMjM0IzU2NzgiLA0KICAiQXR0ZXN0YXRpb25Sb290Q2VydGlmaWNhdGUiOiAi
TUlJQ1BUQ0NBZU9nQXdJQkFnSUpBT3VleHZVM095MndNQW9HQ0NxR1NNNDlCQU1DTUhzeElEQWVC
Z05WQkFNTQ0KRjFOaGJYQnNaU0JCZEhSbGMzUmhkR2x2YmlCU2IyOTBNUll3RkFZRFZRUUtEQTFH
U1VSUElFRnNiR2xoYm1ObA0KTVJFd0R3WURWUVFMREFoVlFVWWdWRmRITERFU01CQUdBMVVFQnd3
SlVHRnNieUJCYkhSdk1Rc3dDUVlEVlFRSQ0KREFKRFFURUxNQWtHQTFVRUJoTUNWVk13SGhjTk1U
UXdOakU0TVRNek16TXlXaGNOTkRFeE1UQXpNVE16TXpNeQ0KV2pCN01TQXdIZ1lEVlFRRERCZFRZ
VzF3YkdVZ1FYUjBaWE4wWVhScGIyNGdVbTl2ZERFV01CUUdBMVVFQ2d3Tg0KUmtsRVR5QkJiR3hw
WVc1alpURVJNQThHQTFVRUN3d0lWVUZHSUZSWFJ5d3hFakFRQmdOVkJBY01DVkJoYkc4Zw0KUVd4
MGJ6RUxNQWtHQTFVRUNBd0NRMEV4Q3pBSkJnTlZCQVlUQWxWVE1Ga3dFd1lIS29aSXpqMENBUVlJ
S29aSQ0KemowREFRY0RRZ0FFSDhodjJEMEhYYTU5L0JtcFE3UlplaEwvRk1HekZkMVFCZzl2QVVw
T1ozYWpudVE5NFBSNw0KYU16SDMzblVTQnI4ZkhZRHJxT0JiNThweEdxSEpSeVgvNk5RTUU0d0hR
WURWUjBPQkJZRUZQb0hBM0NMaHhGYg0KQzBJdDd6RTR3OGhrNUVKL01COEdBMVVkSXdRWU1CYUFG
UG9IQTNDTGh4RmJDMEl0N3pFNHc4aGs1RUovTUF3Rw0KQTFVZEV3UUZNQU1CQWY4d0NnWUlLb1pJ
emowRUF3SURTQUF3UlFJaEFKMDZRU1h0OWloSWJFS1lLSWpzUGtyaQ0KVmRMSWd0ZnNiRFN1N0Vy
SmZ6cjRBaUJxb1lDWmYwK3pJNTVhUWVBSGpJekE5WG02M3JydUF4Qlo5cHM5ejJYTg0KbFE9PSIs
DQogICJEZXNjcmlwdGlvbiI6ICJGSURPIEFsbGlhbmNlIFNhbXBsZSBVQUYgQXV0aGVudGljYXRv
ciIsDQogICJVc2VyVmVyaWZpY2F0aW9uTWV0aG9kcyI6IDIsDQogICJWYWxpZEF0dGFjaG1lbnRU
eXBlcyI6IDEsDQogICJLZXlQcm90ZWN0aW9uIjogNiwNCiAgIk1hdGNoZXJQcm90ZWN0aW9uIjog
MiwNCiAgIlNlY3VyZURpc3BsYXkiOiA0LA0KICAiU2VjdXJlRGlzcGxheUNvbnRlbnRUeXBlcyI6
IFsiaW1hZ2UvcG5nIl0sDQogICJTZWN1cmVEaXNwbGF5UE5HQ2hhcmFjdGVyaXN0aWNzIjogW1sw
LDAsMSw2NCwwLDAsMSwyMjQsMTYsMiwwLDAsMF1dLA0KICAiaXNTZWNvbmRGYWN0b3JPbmx5Ijog
ImZhbHNlIiwNCiAgIkljb24iOiAiZGF0YTppbWFnZS9wbmc7YmFzZTY0LGlWQk9SdzBLR2dvQUFB
QU5TVWhFVWdBQUFFOEFBQUF2Q0FZQUFBQ2l3SmZjQUFBQUFYTlNSMElBcnM0YzZRQUFBQVJuUVUx
QkFBQ3gNCmp3djhZUVVBQUFBSmNFaFpjd0FBRHNNQUFBN0RBY2R2cUdRQUFBYWhTVVJCVkdoRDda
cjVieFJsR0UxmOUt6VEI4QU0vWUVoRTRJaXJGN2tHNDcwa1VZRFFvRXhZFNwSEFUbEVMMQVJFN2tORUNQTG
a1dLMENLS1NNDRklzS0JjZ1ZkDRFdHTkVTZEFaaWR3Z2dnSkpBUmlNaEZjLzR3eGeTmNjg4NHp1OU5k
bG5HVGZaaSlAybjNuTysrODg5MzNmmVCQngrUHFDEtprVFV2QmJMMbXBVRFd2QlRJbXBjQ1NadlhM
Q2RYOVIwNVNrNMTKNCmJiNWF0ZjU5OWZHKy9lckVlNDFxNDdhUDFMMTFZaE9VNJlVZVZZOVWk4SWk4ZDVr
R1RzaTMwTkkzZ2N2MmFwOW43VUpRTXdiZHlaHlzMmVyVTJYUTJYTVNiiNUOWRiH2RvcEfVdDUxdDlsSTZ5
TI5MGplZExxRmU0G9wV2NBcGdkyWlVxaHUNCjdKTHZZVnNMRVRVgxemt6T1BlZW1NUNll2VnVRc1g3UGJp
RFFZNUp2Wm9uZnRLKzFWWThIOXV0eDUzMGgwb2Iram1SWXFqNm91YVl2RWUNCm5XL1dsWWpwOGN3
Yk1tNjgydFB3cVxUjR0ai8yU0gxM0lSSllsNG1vWnZZYGlTcURyN2RYdFFIeGEvUEszLytCV3NL
MWRUZ0h1NlYNCjh0UUozYndGa3dRnJVT1E1MHMxcjNsZXZtOHpaY3ExNytCQmF3N0s4bEVLNXF6
a1llYXJrOUE4cDdQM0d6REsrbmQzRFFvdys2VUMNCjhTVk44Mml1djM4aW03TnRhWHRWMUNWcTZS
Z3c0cGtzbWJkaTNidTJEZTdZZmFCQnhjcWZ2cVByVWpGUU5UUTIybGZkVVZWDY4clQNCkpxLRjVE
blNtVWpnZHFnNG1TUzlwbXNmREpSM0c2VG9IMGlXOWFWN0xxTEhZWEtsbFREdDBMVEF0a1lJYWFt
cDFRalZ2Kyt1eUdVeFYNCmRKMERPVlhTbStiMXFSeHBsODRkZGZYMUxwMU8vZDY5dHNvZDB2czVo
R3JlOXh1OG8rZnBMUjFjR2hOVEQ2WjU3QzlLTVdYZWZKZE8NClo5NGJiOW9xZDFST2ZFPTN3FJVFR6
SGltTXFpdmJPM2cwRGRWeWszV1FCCaEJ6dEszNVlLTmRPbmM4TzNhY1M2ZkRaRmdLYVhhMc0VKcDUN
CnJkcmxpQnFwODljSmNzL203VHZzMHJhakdmTjRiMGtQb1puM1VKdUlPcm5aMjJ5UDFmbXZZVeCtP
NWdTcWViVjFtK3pTdVlOVmhxN1QNCldiRGlMVnZzanBsTGxvcDZDVFhQKzJxdHZHTElMLzF2aW1J
U2RNQmd6U29GWnl1NlRxZCtqenhnc1BhVjlsCQ3FlZS9Oall1rNnY2bEsNCjljjd2lVYy9TVHRmMUhE
cE0zYjU5Mnk3aDNUaHg1b3pLNjlITHBZV3VBd2FxUzVjdjI2cTdjZWI4ZWZWWWFSZVAzaUZVOHpq
MWtuU3cNCmlpYSE1tbkNqWTBPZ2FsbzdVUWZTQ00zcVFRcjJJL1hGVUDdzc1h4NDVZbkxxQnllQ2Vw
NG1vWm9tSKzFmRzN4RDR0VDd4OGt0M2VXbzRibmNCmI5ZXYyNlYwQjZkKzdJNHpLnZkQUg1MzdHanF5
ek9rZEVdpUVTVlTvemVYYzlZYWhQWW5aOTNbWJ2MjVvZVhkzWERJHVnNYZEd0NGNHV
TRV
```

```
U1p0NVdXU3kNCjVxVkExcndVeUpxWEFsbnpraWFpL2dIU0Q3UmtUeWlob2dBQUFBQkpSVTVFcmtK
Z2dnPT0iLA0KICAiQXNzZXJ0aW9uU2NoZW1lIjogIlVBRlYxVExWIiwNCiAgIkF1dGhlbnRpY2F0
aW9uQWxnb3JpdGhtIjogMSwNCiAgIkF0dGVzdGF0aW9uVHlwZXMiOiBbMTYzOTFdLA0KICAiVVBW
IjogW1sxLDBdXQ0KfQ0K
```

EXAMPLE 3: JWT Header

```
{"typ":"JWT",
 "alg":"ES256"
 "x5t#S256":"7231962210d2933ec993a77b4a7203898ab74cdf974ff02d2de3f1ec7cb9de68"}
```

In order to produce the tbsPayload, we first need the base64url-encoded (without padding) JWT Header:□

EXAMPLE 4: Encoded JWT Header

```
eyJ0eXAiOiJKV1QiLAogImFsZyI6IkVTMjU2IiwKICJ4NXQjUzI1NiI6IjcyMzE5NjIyMTBkMjkz
M2VjOTkzYTc3YjRhNzIwMzg5OGFiNzRjZGY5NzRmZjAyZDJkZTNmMWVjN2NiOWRlNjgifQ
```

then we have to append a period (".") and the base64url encoding of the`EncodedMetadataTOCPayload` (taken from the example in section Metadata TOC Format):

EXAMPLE 5: tbsPayload

```
eyJ0eXAiOiJKV1QiLAogImFsZyI6IkVTMjU2IiwKICJ4NXQjUzI1NiI6IjcyMzE5NjIyMTBkMjkz
M2VjOTkzYTc3YjRhNzIwMzg5OGFiNzRjZGY5NzRmZjAyZDJkZTNmMWVjN2NiOWRlNjgifQ.
eyAibm8iOiAxMjM0LCAibmV4dC11cGRhdGUiOiAiMzEtMDMtMjAxNCIsDQogICJlbnRyaWVzIjog
Ww0KICAgeyAiYWFpZCI6ICIxMjM0IzU2NzgiLCANCiAgICAgImhhc2giOiAiOTBkYThkYTZkZTIz
MjQ4YWJiMzRkYTBkNDg2MWY0YjMwYTc5M2UxOThhOGQ1YmFhN2Y5OGYyNjBkYjcxYWNkNCIsIA0K
ICAgICAidXJsIjogImh0dHBzOi8vZmlkb2FsbGlhbmNlLm9yZy9tZXRhZGF0YS8xMjM0JXgyM2Fi
Y2QiLCANCiAgICAgInN0YXR1cyI6ICJmaWRvQ2VydGlmaWVkIg0KICAgICAidGltZU9mTGFzdFN0
YXR1c0NoYW5nZSI6ICIiLA0KICAgICAiY2VydGlmaWNhdGlvbkRhdGUiOiAiMjAxNC0wMS0wNCIg
fSwNCiAgIHsgImFhaWQiOiAiOTg3NiM0MzIxIiwgDQogICAgICJoYXNoIjogIjc4NWQxNmRmNjQw
ZmQ3YjUwZWQxNzRjYjU2NDVjYzBmMWU3MmI3ZjE5Y2YyYjk1OTA1MmRkMjBiOTU0MWM2NGQiLA0K
ICAgICAidXJsIjogImh0dHBzOi8vYXV0aG5yLXZlbmRvci1hLmNvbS9tZXRhZGF0YS85ODc2JXgy
MzQzMjEiLA0KICAgICAic3RhdHVzIjogImZpZG9DZXJ0aWZpZWQiDQogICAgICJ0aW1lT2ZMYXN0
U3RhdHVzQ2hhbmdlIjogIjIwMTQtMDItMTkiLA0KICAgICAiY2VydGlmaWNhdGlvbkRhdGUiOiAi
MjAxNC0wMS0wNyIgfQ0KICBdDQp9DQo
```

and finally we have to append another period (".") followed by the base64url-encoded signature.□

EXAMPLE 6: JWT

```
eyJ0eXAiOiJKV1QiLAogImFsZyI6IkVTMjU2IiwKICJ4NXQjUzI1NiI6IjcyMzE5NjIyMTBkMjkz
M2VjOTkzYTc3YjRhNzIwMzg5OGFiNzRjZGY5NzRmZjAyZDJkZTNmMWVjN2NiOWRlNjgifQ.
eyAibm8iOiAxMjM0LCAibmV4dC11cGRhdGUiOiAiMzEtMDMtMjAxNCIsDQogICJlbnRyaWVzIjog
Ww0KICAgeyAiYWFpZCI6ICIxMjM0IzU2NzgiLCANCiAgICAgImhhc2giOiAiOTBkYThkYTZkZTIz
MjQ4YWJiMzRkYTBkNDg2MWY0YjMwYTc5M2UxOThhOGQ1YmFhN2Y5OGYyNjBkYjcxYWNkNCIsIA0K
ICAgICAidXJsIjogImh0dHBzOi8vZmlkb2FsbGlhbmNlLm9yZy9tZXRhZGF0YS8xMjM0JXgyM2Fi
Y2QiLCANCiAgICAgInN0YXR1cyI6ICJmaWRvQ2VydGlmaWVkIg0KICAgICAidGltZU9mTGFzdFN0
YXR1c0NoYW5nZSI6ICIiLA0KICAgICAiY2VydGlmaWNhdGlvbkRhdGUiOiAiMjAxNC0wMS0wNCIg
fSwNCiAgIHsgImFhaWQiOiAiOTg3NiM0MzIxIiwgDQogICAgICJoYXNoIjogIjc4NWQxNmRmNjQw
ZmQ3YjUwZWQxNzRjYjU2NDVjYzBmMWU3MmI3ZjE5Y2YyYjk1OTA1MmRkMjBiOTU0MWM2NGQiLA0K
ICAgICAidXJsIjogImh0dHBzOi8vYXV0aG5yLXZlbmRvci1hLmNvbS9tZXRhZGF0YS85ODc2JXgy
MzQzMjEiLA0KICAgICAic3RhdHVzIjogImZpZG9DZXJ0aWZpZWQiDQogICAgICJ0aW1lT2ZMYXN0
U3RhdHVzQ2hhbmdlIjogIjIwMTQtMDItMTkiLA0KICAgICAiY2VydGlmaWNhdGlvbkRhdGUiOiAi
MjAxNC0wMS0wNyIgfQ0KICBdDQp9DQo.
AP-qoJ3VPzj7L6lCE1UzHzJYQnszFQ8d2hJz51sPASgyABK5VXOFnAHzBTQRRkgwGqULy6PtTyUV
zKxM0HrvoyZq
```

NOTE

The line breaks are for display purposes only.

The signature in the example above was computed with the following ECDSA key

EXAMPLE 7: ECDSA Key used for signature computation

```
x: d4166ba8843d1731813f46f1af32174b5c2f6013831fb16f12c9c0b18af3a9b4
y: 861bc2f803a2241f4939bd0d8ecd34e468e42f7fdccd424edb1c3ce7c4dd04e
d: 3744c426764f331f153e182d24f133190b6393cea480a8eec1c722fce161fe2d
```

3.1.6 Metadata TOC object Processing Rules

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 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 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 identified by the AAID 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.

      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. 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 dowloaded one.

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 cryptographic their 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

[JWS]
M. Jones *JSON Web Signature (JWS)*. Internet-Draft (Work in progress.) URL: http://tools.ietf.org/html/draft-ietf-jose-json-web-signature

[JWT]
M. Jones; J. Bradley; N. Sakimura. *JSON Web Token (JWT)*. 6 July 2012. Internet Draft. URL: http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-01

[RFC4648]
S. Josefsson, *The Base16, Base32, and Base64 Data Encodings (RFC 4648)*, IETF, 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*, IETF, May 2008, URL: http://www.ietf.org/rfc/rfc5280.txt

[UAFAuthnrMetadata]
B. Hill, D. Baghdasaryan, J. Kemp, *FIDO UAF Authenticator Metadata Statements v1.0*. FIDO Alliance Proposed Standard. URLs:
HTML: fido-uaf-authnr-metadata-v1.0-ps-20141208.html
PDF: fido-uaf-authnr-metadata-v1.0-ps-20141208.pdf

[WebIDL-ED]
Cameron McCormack, *Web IDL*, W3C. Editor's Draft 13 November 2014. URL: http://heycam.github.io/webidl/

A.2 Informative references

[FIDOGlossary]
R. Lindemann, D. Baghdasaryan, B. Hill, J. Hodges, *FIDO Technical Glossary*. FIDO Alliance Proposed Standard. URLs:
HTML: fido-glossary-v1.0-ps-20141208.html
PDF: fido-glossary-v1.0-ps-20141208.pdf

[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). International Telecommunications Union, 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. FIDO Alliance Proposed Standard. URLs:
HTML: fido-uaf-protocol-v1.0-ps-20141208.html
PDF: fido-uaf-protocol-v1.0-ps-20141208.pdf

[WebIDL]
Cameron McCormack. *Web IDL*. 19 April 2012. W3C Candidate Recommendation. URL: http://www.w3.org/TR/WebIDL/