FIDO Metadata Service
Proposed Standard, May 18, 2021

This version:  

Issue Tracking:  
GitHub

Editors:  
Billy Jack (Microsoft)  
Rolf Lindemann (Nok Nok Labs)  
Yuriy Ackermann (FIDO Alliance)

Copyright © 2021 FIDO Alliance. All Rights Reserved.

Abstract
The FIDO Authenticator Metadata Specification defines so-called “Authenticator Metadata” statements. The metadata statements contains 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.

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 NONINFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

This document has been reviewed by FIDO Alliance 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
1.1       Key Words
2       Overview
2.1       Scope
2.2 Detailed Architecture

3 Metadata Service Details
3.1 Metadata BLOB Format
3.1.1 Metadata BLOB Payload Entry dictionary
3.1.2 BiometricStatusReport dictionary
3.1.3 StatusReport dictionary
3.1.4 AuthenticatorStatus enum
3.1.4.1 Certification Related Statuses
3.1.4.2 Security Notification Statuses
3.1.4.3 Info Statuses
3.1.5 RogueListEntry dictionary
3.1.6 Metadata BLOB Payload dictionary
3.1.7 Metadata BLOB
3.1.7.1 Examples
3.2 Metadata BLOB object processing rules

4 Considerations

Index
Terms defined by this specification
Terms defined by reference

References
Normative References
Informative References

IDL Index

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.

For definitions of terms, please refer to the FIDO Glossary[FIDO glossary].

All diagrams, examples, notes in this specification are non-normative.
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 not 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.

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 BLOB file contains a list of metadata statements related to the authenticators known to the FIDO Alliance (FIDO Authenticators).

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

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

Individual metadata statements are included in the entry of the metadata BLOB file, and may be cached by the FIDO Server as required.

![Figure 2 FIDO Metadata Service Architecture](image)

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

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

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

3. Metadata Service Details

*This section is normative.*
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 BLOB Format

The metadata service makes the metadata BLOB object (see Metadata BLOB) accessible to FIDO Servers. This object contains all metadata for each authenticator including the metadata statements defined in [FIDOMetadataStatement]. The BLOB object contains one signature.

3.1.1. Metadata BLOB Payload Entry dictionary

Represents the MetadataBLOBPayloadEntry dictionary

```javascript
dictionary MetadataBLOBPayloadEntry {
    AAID aaid;
    AAGUID aaguid;
    DOMString[] attestationCertificateKeyIdentifiers;
    MetadataStatement metadataStatement;
    BiometricStatusReport[] biometricStatusReports;
    required StatusReport[] statusReports;
    required DOMString timeOfLastStatusChange;
    DOMString rogueListURL;
    DOMString rogueListHash;
};
```

**aaid**, of type **AAID**

The AAID of the authenticator this metadata BLOB 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 FIDO2.

Note: FIDO2 authenticators support AAGUID, but they don’t support AAID.

**attestationCertificateKeyIdentifiers**, of type **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.

FIDO U2F authenticators do not support AAID nor AAGUID, but they use attestation certificates dedicated to a single authenticator model.
**metadataStatement**, of type `MetadataStatement`

The `metadataStatement` JSON object as defined in [FIDO Metadata Statement](#).

**biometricStatusReports**, of type `BiometricStatusReport[]`

Status of the FIDO Biometric Certification of one or more biometric components of the Authenticator [FIDO Biometrics Requirements].

**statusReports**, of type `StatusReport[]`

An array of status reports applicable to this authenticator.

**timeOfLastStatusChange**, of type `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 BLOB) MUST be used.

This hash value MUST be present and non-empty whenever rogueListURL is present.

This method of base64url-encoding the UTF-8 representation is also used by JWT [JWT] to avoid encoding ambiguities.
Achieved level of the biometric certification of this biometric component of the authenticator [FIDOBiometrics Requirements].

A single USER_VERIFY short form case-sensitive string name constant, representing biometric...
modality. See section "User Verification Methods" in [FIDORegistry] (e.g. "fingerprint_internal"). This value MUST NOT be empty and this value MUST correspond to one or more entries in field userVerificationDetails in the related Metadata Statement[FIDOMetadataStatement]. This value MUST represent a biometric modality.

For example use USER_VERIFY_FINGERPRINT for the fingerprint based biometric component. In this case the related Metadata Statement must also claim fingerprint as one of the user verification methods.

effectiveDate, of type DOMString
ISO-8601 formatted date since when the certLevel achieved, if applicable. If no date is given, the status is assumed to be effective while present.

certificationDescriptor, of type DOMString
Describes the externally visible aspects of the Biometric Certification evaluation.

For example it could state that the "biometric component is implemented OnChip - keeping biometric data inside the chip only."

certificateNumber, of type DOMString
The unique identifier for the issued Biometric Certification.

certificationPolicyVersion, of type DOMString
The version of the Biometric Certification Policy the implementation is Certified to, e.g. "1.0.0".

certificationRequirementsVersion, of type DOMString
The version of the Biometric Requirements [FIDOBiometricsRequirements] the implementation is certified to, e.g. "1.0.0".

3.1.3. StatusReport dictionary

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.

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

status, of type 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.

authenticatorVersion, of type unsigned long
The authenticatorVersion that this status report relates to. In the case of FIDO_CERTIFIED* status values, the status applies to higher authenticatorVersions until there is a new statusReport.

For example, if the status would be USER_VERIFICATION_BYPASS, the authenticatorVersion indicates the vulnerable firmware version of the authenticator. Similarly, if the status would be UPDATE_AVAILABLE, the authenticatorVersion indicates the updated firmware version that is available now. If the status would be SELF_ASSERTION_SUBMITTED, the authenticatorVersion indicates the firmware version that the self assertion was based on.

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

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.

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.

For example it could state that the authenticator is a "SecurityKey based on a CC EAL 5 certified chip hardware".

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 Document Version of the Authenticator Security Requirements (DV) [FIDOAuthenticatorSecurityRequirements] the implementation is certified to, e.g. "1.2.0".

3.1.4. AuthenticatorStatus enum

This enumeration describes the status of an authenticator model as identified by its AAID/AAGUID or attestationCertificateKeyIdentifiers and potentially some additional information (such as a specific attestation key).
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_L1plus",
    "FIDO_CERTIFIED_L2",
    "FIDO_CERTIFIED_L2plus",
    "FIDO_CERTIFIED_L3",
    "FIDO_CERTIFIED_L3plus"
};

3.1.4.1. Certification Related Statuses

**NOT_FIDO_CERTIFIED**
This authenticator is not FIDO certified.

Applicable StatusReport fields are:

- effectiveDate - When status was achieved
- authenticatorVersion - The minimum applicable authenticator version.
- url - To the authenticator page or additional information about the authenticator

**SELF_ASSERTION_SUBMITTED**
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 url.

Applicable StatusReport fields are:

- effectiveDate - Date of incident being reported
- authenticatorVersion - New authenticator version that is

**FIDO_CERTIFIED**
This authenticator has passed FIDO functional certification. This certification scheme is phased out and will be replaced by FIDO_CERTIFIED_L1.

Applicable StatusReport fields are:

- effectiveDate - When certification was issued
- authenticatorVersion - The minimum version of the certified solution
- certificationDescriptor - Authenticator Description. I.e. "Munikey 7c Black Edition"
- certificateNumber - FIDO Alliance Certificate Number
- certificationPolicyVersion - Authenticator Certification Policy
- certificationRequirementsVersion - Security Requirements Version
- url - URL to the certificate, or the news article about achievement of the certification.

These fields are applicable to any of the FIDO_CERTIFIED_*.
The authenticator has passed FIDO Authenticator certification at level 1. This level is the more strict successor of FIDO_CERTIFIED.

FIDO_CERTIFIED_L1plus
The authenticator has passed FIDO Authenticator certification at level 1+. This level is the more than level 1.

FIDO_CERTIFIED_L2
The authenticator has passed FIDO Authenticator certification at level 2. This level is more strict than level 1+.

FIDO_CERTIFIED_L2plus
The authenticator has passed FIDO Authenticator certification at level 2+. This level is more strict than level 2.

FIDO_CERTIFIED_L3
The authenticator has passed FIDO Authenticator certification at level 3. This level is more strict than level 2+.

FIDO_CERTIFIED_L3plus
The authenticator has passed FIDO Authenticator certification at level 3+. This level is more strict than level 3.

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. Relying parties SHOULD reject any future registration of this authenticator model.

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.

Applicable StatusReport fields are:
- effectiveDate - Date of incident being reported
- authenticatorVersion - New authenticator version that is
- url - URL to the news/corporate article explaining the reason for revocation

ATTESTATION_KEY_COMPROMISE
Indicates that an attestation key for this authenticator is known to be compromised. The relying party SHOULD check the certificate field and use it to identify the compromised authenticator batch. If the certificate field is not set, the relying party should reject all new registrations of the compromised authenticator. The Authenticator manufacturer should set the date to the date when compromise has occurred.

Applicable StatusReport fields are:
- effectiveDate - Date of incident being reported
- authenticatorVersion - Minimum affected authenticator version
• certificate - Base64 DER-encoded PKIX certificate identifying compromised attestation root. If missing, then assume all authenticators of this model are compromised.
• url - URL to the news/corporate article explaining the incident

**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.

Applicable StatusReport fields are:
- effectiveDate - Date of incident being reported
- authenticatorVersion - Minimum affected authenticator version
- url - URL to the news/corporate article explaining the incident

**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.

Applicable StatusReport fields are:
- effectiveDate - Date of incident being reported
- authenticatorVersion - Minimum affected authenticator version
- url - URL to the news/corporate article explaining the incident

### 3.1.4.3. Info Statuses

**UPDATE_AVAILABLE**
A software or firmware update is available for the device. The Authenticator manufacturer should set the url to the URL where users can obtain an update and the date the update was published. When this status code is used, then the field authenticatorVersion in the authenticator Metadata Statement [FIDOMetadataStatement] 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. The Relying party MUST reject the Metadata Statement if the authenticatorVersion has not increased

Applicable StatusReport fields are:
- effectiveDate - Date of incident being reported
- authenticatorVersion - New authenticator version that is available. MUST match authenticatorVersion in the metadata statement.
- url - URL to the page with the update info

Relying parties might want to inform users about available firmware updates.

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

### 3.1.5. RogueListEntry dictionary
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.

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

**sk**, of type **DOMString**
Base64url encoding of the rogue authenticator's secret key (sk value, see [FIDOEcdaaAlgorithm], section ECDAA Attestation).

**date**, of type **DOMString**
ISO-8601 formatted date since when this entry is effective.

**EXAMPLE: ROGUELISTENTRY**
```
[{
  "sk": "MO-oaqbeJSSayzXaDuuhh9LMKeT4Zio1bqn6W8k0aUfM",
  "date": "2016-06-07"},
{
  "sk": "k96Npt4jI1q7NNoNSGH0swp5PhU6jVyf5jyYntxrNQ",
  "date": "2016-06-09"},
]
```

3.1.6. Metadata BLOB Payload dictionary

Represents the MetadataBLOBPayload dictionary.

```
dictionary MetadataBLOBPayload {
  DOMString legalHeader;
  required Number no;
  required DOMString nextUpdate;
  required MetadataBLOBPayloadEntry[] entries;
};
```

**legalHeader**, of type **DOMString**
The legalHeader, which MUST be in each BLOB, is an indication of the acceptance of the relevant legal agreement for using the MDS. The FIDO Alliance’s Blob will contain this legal header: "legalHeader": "Retrieval and use of this BLOB indicates acceptance of the appropriate agreement located at https://fidoalliance.org/metadata/metadata-legal-terms/"

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

**nextUpdate**, of type **DOMString**
ISO-8601 formatted date when the next update will be provided at latest.
3.1.7. Metadata BLOB

The metadata BLOB 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 Metadata BLOB Payload (see example below), and
- the base64url-encoded, also without padding, JWS Signature computed over the to-be-signed payload using the Metadata BLOB signing key, i.e. tbsPayload = EncodedJWTHeader | "." | EncodedMetadataBLOPPayload

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

EncodedMetadataBLOPPayload = EncodedJWTHeader | "." | EncodedMetadataBLOBPayload | "." | EncodedJWTSignature

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 BLOB Payload Format).

This section is not normative.

3.1.7.1. Examples

This section is not normative.
null
In order to produce the tbsPayload, we first need the base64url-encoded (without padding) JWT Header:

```json
{
  "alg": "ES256",
  "typ": "JWT",
  "x5c": ["MIICZTCCagAwIBAgIBATAKAggghkjQ0QQDAjCB0zEnMCUGA1EwweRvBLTVBM
RSBNRFRMz1FRFU0IQSU3RVRJQRVFM5IWyAIYJKoZIhvcNAQkFhNhLeGtcGxl
QGV4YlwbGGuY29fMRQeGyjDQQKDAQFeGtCgxlIE9SRzEgYMA4GA1UECwwHRXhh
bXkS2TELMAkGAEwBMIvMvY2cZaJbBvRYBAgMak1ZMRlWE8AYDVQQHQDAfLXytL
ZmllbGQwHhNfMVhNDE3MTExZ3whcNHzWhNDE3MTExZ3whcMjBpTepMccGAgUEwaw
RVhBiTVBMRBSNRMzwFInJR05JTkcgO6VrSEVLGSENVBEUejJAgBkgqhkigQpwBCQEW
E2V4YlwbGvAZXhbxBSZSijb2ot5FADSgBNVBAcMC0Fv4VYlwbGgUT1JHMRoAqYVYD
VQQDADFeGgCgXMswCYDQVQQgJwJyVkJELMAkGAlUEAcmCVkxeJAgBqNBVACm
CVdha2VmaWVsZDB2M2BMBgYqGSM49AgEGCCGCqGSM49AHEA0rABNQJcxs6W7qix
+SVDAajFlNn10KgkE5jCwDm6q9S0DAMZybb4HRivs+PSR5rHR5UPdfX+UzoqEbd
Wg5319pujLDAqAgMA1UeEQMCAwHqYDVQR0dBBYEFQSapcVX4zoVAHAnRipZqwq7
Y2oMAcGACqGSM49BAMCA8MEUCIGIQQG7za8E1uyRikgNDXIPis1aLz3jH9WXvF
xL4bJ+gCspGifG/tvBu0JUU+vvoHIo/otAUAch5bNf3PuizISd+PTUC=",
"MIIEZSCCVgAwIBAgIBAJANBkgkqkhjIG9w9BA0sFeCDBmzFMB0GAgwJrWVb
TVBMRBSNRMzwFInJR05JTkcgO6VrSEVLGSENVBEUejJAgBkgqhkigQpwBCQEW
E2V4YlwbGvAZXhbxBSZSijb2ot5FADSgBNVBAcMC0Fv4VYlwbGgUT1JHMRoAqYVYD
VQQDADFeGgCgXMswCYDQVQQgJwJyVkJELMAkGAlUEAcmCVkxeJAgBqNBVACm
CVdha2VmaWVsZDB2M2BMBgYqGSM49AgEGCCGCqGSM49AHEA0rABNQJcxs6W7qix
+SVDAajFlNn10KgkE5jCwDm6q9S0DAMZybb4HRivs+PSR5rHR5UPdfX+UzoqEbd
Wg5319pujLDAqAgMA1UeEQMCAwHqYDVQR0dBBYEFQSapcVX4zoVAHAnRipZqwq7
Y2oMAcGACqGSM49BAMCA8MEUCIGIQQG7za8E1uyRikgNDXIPis1aLz3jH9WXvF
xL4bJ+gCspGifG/tvBu0JUU+vvoHIo/otAUAch5bNf3PuizISd+PTUC="
]"
```

EXAMPLE: JWT HEADER

```
{
  "alg": "ES256",
  "typ": "JWT",
  "x5c": [
```

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

**EXAMPLE: ENCODED JWT HEADER**

```
ewogICJhbGciOiAiRVMyNTYiLAogICJ0eXAiOiAiSldUIiwKICAieDVjIjogWwogICAgIk1JSUNaVENDQWd1Z0F3SUJBZ0lCQVRBS0JnZ3Foa2pPUFFRREFqQ0JvekVuTUNVR0ExVUVBd3dlUlZoQlRWQk1SU0JO
UkZNeklGUkZVMVFnU1U1VVJWSk5SVVJKUVZSRk1TSXdJQVlKS29aSWh2Y05BUWtCRmhObGVHRnRjR3hsUUdWNFlXMXdiR1V1WTI5dE1SUXdFZ1lEVlFRS0RBdEZlR0Z0Y0d4bElFOVNSekVRTUE0R0ExVUVDd3dIUlhoaGJYQnNaVEVMTUE0Y0d4bElFOVNSekVRTUE0R0ExVUVDd3dIUlhoaGJYQnNaVEVMTUE0Y0d4bElFOVNSekVRTUE0R0ExVUVDd3dIUlhoaGJYQnNaVEVMTUE0Y0d4bElFOVNSekVRTUE0R0ExVUVDd3dI
```

**EXAMPLE: TBSPAYLOAD**

```
eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCIsIng1YyI6WyJNSUlDWlRDQ0FndWdBd0lCQWdJQkFUQUtCZ2dxaGtpT1BRUURBakNCb3pFbk1DVUdBMVVFQXd3ZVJWaEJUVkJNUlNCTlJGTXpJRlJGVTFRZ1NVNVVSVkpOUlVSSlFWUkZNU0l3SUFzakNCcFRFcE1DY0dBMVVFQXd3Z1JWaEJUVkJNUlNCTlJGTXpJRk5KUjA1SlRrY2dRMFZTVkVsR1NVTkJWRVV4SWpBZ0Jna3Foa2lHOXcwQkNRRVdF
```

(taken from the example in section Metadata BLOB Payload)
and finally we have to append another period (".") followed by the base64url-encoded signature.

EXAMPLE: JWT

```json
eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.

.......

MMSISImVmZmVjdGl2ZURhdGUiOiIyMDIwLTExLTE5IiwiY2VydGlmaWNhdGlvbkRlc2NyaXB0b3IiOiJGSURPIEFsbGlhbmNlfHNlbmhXb2JjZS5BSGSURPImBdBXRoZW50aWNhdGlyY2YgdGlnaWNhdGV0dGVdIiwiIjoiI10iJG SURPMjEwMDAyMDE1MTY5MTA0MjE5NjU4NjQ2MjIzLTI3LTI1LDE3LTExMjE2LTExMjE2MzExLjI1LjI0IiwiIiwiY2VydGlmaWNhdGlvbkZlcnNpb24iOiJGSURPIEFsbGlhbmNlfHNI...```

```css
...```
3dCV3S1ppMEEYQVFSvJvdui10RSt1SmMrYjxYa0FwCuUeJnbeWQvNYFU1VAvbXQoEh0q2dkuHmg1dLTVbVshVmmJl0UF4MFD2zMm2zIzDRCn5eE0dWdTFv1IvydXwzMEzl0j3cV4e4Gvaj1ldRxZanVwZDN0U0qY2WYJkxJncmcrgYBY9KJcK5GSt1Z0NH3WEdm010uUHRwawF2afuZwu1PQFudVvILIv0WweU9T9UZEH0U4FwE1g2RZm5ZmSbsfmNMDNFTDFVTZTj1FmducH3t0QkZvFwelCmFGU13j3zUeREY1lWrb1am99u50NdVRtVUmKmVatLJcxH1tO0t2a22Z0HgbeKmavGZVYrzhEdj0hN0vTPnFap0PbdQZTjER3VlWnDnZ6lBvFumNpws2z3J33Nvjdomp2cEU5EwBp0Vd2L2HyueEF1Wesa0g9Y1lR2NWNURmtJBX8yJvU6Ym4aCBkFTZT2jJvWhy659Bk60aTzJvMv5vSultkOnqUvriwZQeDyBdx0eD1zJJ1EINEXJkMwozRm9HSG5q6RFCNrhFwNvVTuTUX5TX6S3Bi1N3RyCG9I1Zb0Ggg3V3A3H0p05NTtTeYeMmd09RHWlzIzh2ZwoNo7V0t0v0mR0U3y4yjTjwJ0U6yYU1sakhFS2tr3rnp3hKatZUUEyaeDwRTF6kzg1kT2KxcHfEcm63h3M4MTbhCXRKv362z1b31rJnW44avzuZ3s3aCtmRwzhXnrUjrJbEMU5V5ZvJz25kRnUXFLYxh6dUNkRTBp6sJMOJUO40AhtBthWkMxhMz2wzj1a3wS3mDTRLNWS84nk10zbqSm5a7TmxTzr2aBqV2FudVucE5K50aRwztZbe1mN1TeW1vdEpDx TenwDpcnWpRAwpFp2l2duU0e3umbTholbo2BOQBPv3SFTCmZ2ndT01CIjC3dbxzB3j28ZrFeR荷bnPbs25ljjpegy1zC16lnhMwjv2cvmJoVI1iizmFpFb9z191btmb3duijmYmWxxZ0Xosey2jZC16inwyvZqrC0m92z0Wn0i1izmFpFb9p29191bmtub3dulJpYmWxxZ1sdLCjhdXroZW50awNhhd9gy2V05WS5my6Ie6y67Z2JXz9aWuc166yWVKMzVfjv1LICJG5URPX2jMCJdLCjehRnlbnNp25zlJipbnMvWy0rC0m90Z0WN0i1iawaghYy1zZWyNyXzSXiWxEdfnWkldljoimeDmEemMmxMTBJzjRlNdIOGE0MDNHjYrnmWxyMmVmZTUICjvRhp25zIj7pInBsXYQ0i1JmzWyzZSI snJrjIjoidHJ1ZSISImnsawVUDfFbi61InRydWuiLCJ1ClC16InRydWuiLCjClCd16InRydWuiLCjClCd16r va2VujjOizMsfcuiLjC3jbf5maMcl01IjymWxxZ53LCYJtXhc2dtaTpaIjoxjMawjLc3waSwDvdkFId6h Qcm90b2Vnbmi00s1xsvF4WQ3J1LZVgdGIhbevEw05SW5SMaX0joxNi1vBwFq43J1LZVgdGhbeLb kTGvUz3R0joxMjgsInRyWy5Czg9sChnHl0i0sIdXNnIvIi1wBmJzL8sIfsZx2yYaKobxM0l71rN5cu i0Ijw3dWsa2M05vIyVXnnIjotN30s3ey0eB1li0i1cHiBgiBglvilt63isImfsZY16L1N310CLjcy TyxNhBrdxR0zw50awNhNld9gy2Q90zmLmTGVUz3R0joxMDlCjKjCwzHdxO3QlZFBy3RlY0q01is zincpcm13yXjCnyv21ib16nX19LCjizdG0xN52XbcvRZjpebyejz2d70FDxM0i10GJGSRPN0FwNRJrl kFCRis1rmVmjgd12ZURhgdGU10iyMDE5LTA0in0seyjzdGfd0M10i1GJGSRPNGXNFUJRkJFkRF9FM S1Ms1iMvmvJjg12ZURhGDG10u1iyLDTe1EL5tIy12yDgldMawNh0GvlbrRef1cyNa0B0i10J iGSURPIEfs6GbhhbINIFNHbXbSZBSGSRPM18BDXR0zw50awNhNd9gy1iYi2YVdlGmawNh0Gvd0iwi1Xi ioj3JSURPMj1EmdYa1MDE1MT1yMTAuM1SisM1NclmRzpMzljYXlyb2p2b0x3ypF1XWZjx2aw9uji0m5V4Lje i1lcjJXr3awp2Yfa80uwVv1XWmyZLN1bnRzVmc21bl61JeuxC4in1LDCjawa22YTM0XU3RHdHVZ2ohbdmljoiMjAxOs155wN5wC9JXX0.--kclwrorJAla16bxLXXzeeDkFED0scbKAY2WDezoCA-Eyj_N0BWI0AmhP6xSa3C8mgFwAugy2330Eq_BHTO_RshsA

The line breaks are for display purposes only.

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

--- BEGIN CERTIFICATE ---

MIICZTCCAgugAwIBAgIBATAKBggqhkjOPQDAjCBozEnMCUGA1UEAwweRVhBTVBMRSNBRMzIFRlf90U5iQSU5jUNRjQVRmVSMVJyZkoI3hvcNAA0KBrFhNleGFtcGxlOQY4YW1wbgU6u2Y9TmR0eQyDVQKDAtFeGtcGxiIE95R9zEM04AIGilCwvHRXhxbXZTELMaK1EBUlcVCMmxCAjBMIkZh2QAYDQw0DAgEGCCsGAQEBMMA0GCSqGSIb3DQEJARYwcm9jaW1zZm9yZmFjZWxBXzcmYXJpdHRvb2xvZ2xlY2hvb2wuY29tZ29yZ2hvb2wuY29t

--- END CERTIFICATE ---

--- BEGIN EC PRIVATE KEY ---

MIICZTCCAgugAwIBAgIBATAKBggqhkjOPQDAjCBozEnMCUGA1UEAwweRVhBTVBMRSNBRMzIFRlf90U5iQSU5jUNRjQVRmVSMVJyZkoI3hvcNAA0KBrFhNleGFtcGxlOQY4YW1wbgU6u2Y9TmR0eQyDVQKDAtFeGtcGxiIE95R9zEM04AIGilCwvHRXhxbXZTELMaK1EBUlcVCMmxCAjBMIkZh2QAYDQw0DAgEGCCsGAQEBMMA0GCSqGSIb3DQEJARYwcm9jaW1zZm9yZmFjZWxBXzcmYXJpdHRvb2xvZ2xlY2hvb2wuY29tZ29yZ2hvb2wuY29t

--- END EC PRIVATE KEY ---

The root certificate to validate certificate path in the X5C is:

-----BEGIN CERTIFICATE-----
MIICZTCCAgugAwIBAgIBATAKBggqhkjOPQDAjCBozEnMCUGA1UEAwweRVhBTVBMRSNBRMzIFRlf90U5iQSU5jUNRjQVRmVSMVJyZkoI3hvcNAA0KBrFhNleGFtcGxlOQY4YW1wbgU6u2Y9TmR0eQyDVQKDAtFeGtcGxiIE95R9zEM04AIGilCwvHRXhxbXZTELMaK1EBUlcVCMmxCAjBMIkZh2QAYDQw0DAgEGCCsGAQEBMMA0GCSqGSIb3DQEJARYwcm9jaW1zZm9yZmFjZWxBXzcmYXJpdHRvb2xvZ2xlY2hvb2wuY29tZ29yZ2hvb2wuY29t

-----END CERTIFICATE-----

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

The ECDSA key used for signature computation was:

--- BEGIN CERTIFICATE ---
MIICZTCCAgugAwIBAgIBATAKBggqhkjOPQDAjCBozEnMCUGA1UEAwweRVhBTVBMRSNBRMzIFRlf90U5iQSU5jUNRjQVRmVSMVJyZkoI3hvcNAA0KBrFhNleGFtcGxlOQY4YW1wbgU6u2Y9TmR0eQyDVQKDAtFeGtcGxiIE95R9zEM04AIGilCwvHRXhxbXZTELMaK1EBUlcVCMmxCAjBMIkZh2QAYDQw0DAgEGCCsGAQEBMMA0GCSqGSIb3DQEJARYwcm9jaW1zZm9yZmFjZWxBXzcmYXJpdHRvb2xvZ2xlY2hvb2wuY29tZ29yZ2hvb2wuY29t

--- END CERTIFICATE ---

The root certificate to validate certificate path in the X5C is:
3.2. Metadata BLOB object processing rules§

The FIDO Server MUST follow these processing rules:

1. Download and cache the root trust anchor from the respective MDS root location e.g. More information can be found at https://fidoalliance.org/metadata/

2. To validate the digital certificates used in the digital signature, the certificate revocation information MUST be available in the form of CRLs at the respective MDS CRL location e.g. More information can be found at https://fidoalliance.org/metadata/

3. The FIDO Server MUST be able to download the latest metadata BLOB object from the well-known URL when appropriate, e.g. https://mds.fidoalliance.org/ The Metadata BLOB specifies a date when the download SHOULD occur at latest.

4. 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 BLOB 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 BLOB signing trust.
This section is not 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 BLOB object and all metadata statements.
Organizational Impact

The FIDO Alliance has control over the FIDO certification process and authentication vendors provide the metadata as part of that process. With this metadata service, the list of known authenticators and their metadata statements need 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 BLOB object and all embedded metadata statements.

Performance Impact

Metadata BLOB objects and metadata statements can be cached by the FIDO Server.

The update policy can be specified by the relying party.

The metadata BLOB 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.

High Security Environments

Some high security environments might only trust internal policy authorities. FIDO Servers in such environments could be restricted to use metadata BLOB 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.

Index§

Terms defined by this specification§

- aaguid
- aaid
- attestationCertificateKeyIdentifiers
- "ATTESTATION_KEY_COMPromise"
- AuthenticatorStatus
- authenticatorVersion
- BiometricStatusReport
- biometricStatusReports
- certificate
- certificateNumber
- dict-member for BiometricStatusReport
- dict-member for StatusReport
- certificationDescriptor
- dict-member for BiometricStatusReport
- dict-member for StatusReport
- certificationPolicyVersion
- dict-member for BiometricStatusReport
- dict-member for StatusReport
- certificationRequirementsVersion
- dict-member for BiometricStatusReport
dict-member for StatusReport
certLevel
date
effectiveDate
dict-member for BiometricStatusReport
dict-member for StatusReport
entries
"FIDO_CERTIFIED"
"FIDO_CERTIFIED_L1"
"FIDO_CERTIFIED_L1plus"
"FIDO_CERTIFIED_L2"
"FIDO_CERTIFIED_L2plus"
"FIDO_CERTIFIED_L3"
"FIDO_CERTIFIED_L3plus"
legalHeader
MetadataBLOBPayload
MetadataBLOBPayloadEntry
metadataStatement
modality
nextUpdate
no
"NOT_FIDO_CERTIFIED"
"REVOKED"
RogueListEntry
rogueListHash
rogueListURL
"SELF_ASSERTION_SUBMITTED"
sk
status
StatusReport
statusReports
timeOfLastStatusChange
"UPDATE_AVAILABLE"
url
"USER_KEY_PHYSICAL_COMPROMISE"
"USER_KEY_REMOTE_COMPROMISE"
"USER_VERIFICATION_BYPASS"

Terms defined by reference

[webauthn-1] defines the following terms:
  AAGUID
[WebIDL] defines the following terms:

- DOMString
- unsigned long
- unsigned short

References

Normative References

[FIDOAuthenticatorSecurityRequirements]

[FIDOBiometricsRequirements]
Stephanie Schuckers; et al. FIDO Biometrics Requirements. October 2020. URL: https://fidoalliance.org/specs/biometric/requirements/Biometrics-Requirements-v2.0-fd-20201006.html

[FIDOMetadataStatement]

[JWS]

[JWT]

[RFC4648]

[RFC5280]

[WEBAUTHN-1]
Dirk Balfanz; et al. Web Authentication: An API for accessing Public Key Credentials Level 1. 4 March 2019. REC. URL: https://www.w3.org/TR/webauthn-1/

[WebIDL]

[WebIDL-ED]

Informative References

[FIDOEcdaaAlgorithm]
R. Lindemann; et al. FIDO ECDAA Algorithm. Implementation Draft. URL: https://fidoalliance.org/specs/fido-v2.0-id-20180227/fido-ecdaa-algorithm-v2.0-id-20180227.html

[FIDOGlossary]

[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]
dictionary MetaDataBLOBPayloadEntry {
    AAID aaid;
    AAGUID aaguid;
    DOMString[] attestationCertificateKeyIdentifiers;
    MetadataStatement metadataStatement;
    BiometricStatusReport[] biometricStatusReports;
    required StatusReport[] statusReports;
    required DOMString timeOfLastStatusChange;
    DOMString rogueListURL;
    DOMString rogueListHash;
};

dictionary BiometricStatusReport {
    required unsigned short certLevel;
    required DOMString modality;
    DOMString effectiveDate;
    DOMString certificationDescriptor;
    DOMString certificateNumber;
    DOMString certificationPolicyVersion;
    DOMString certificationRequirementsVersion;
};

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

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 L1plus",
    "FIDO CERTIFIED L2",
    "FIDO CERTIFIED L2plus",
};
"FIDO CERTIFIED L3",
"FIDO CERTIFIED L3plus"
};

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

dictionary MetadataBLOBPayload {
    DOMString legalHeader;
    required Number no;
    required DOMString nextUpdate;
    required MetadataBLOBPayloadEntry[] entries;
};