REVIEW DRAFT

FIDO 2.0: Requirements for Native Platforms

FIDO Alliance Review Draft 04 October 2017

This version:
https://fidoalliance.org/specs/fido-undefined-v2.0-rd-20171004/fido-platform-api-reqs-v2.0-rd-20171004.html
Editors:
Hubert Le Van Gong, PayPal
Rolf lindemann, Nok Nok Labs, Inc.
Contributors:
Alexei Czeskis, Google
Jeff Hodges, PayPal

Abstract

Platform APIs are specific to the individual platform. Instead of mandating a specific FIDO API (e.g. function names, parameter lists, etc.), FIDO specifies a set of functional requirements for a platform. Platform vendors can decide how to best provide such functionality in their respective platform API. Platforms implementing all FIDO functional requirements can be FIDO certified and hence indicate that they provide native FIDO support.

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 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

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

1.1 Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

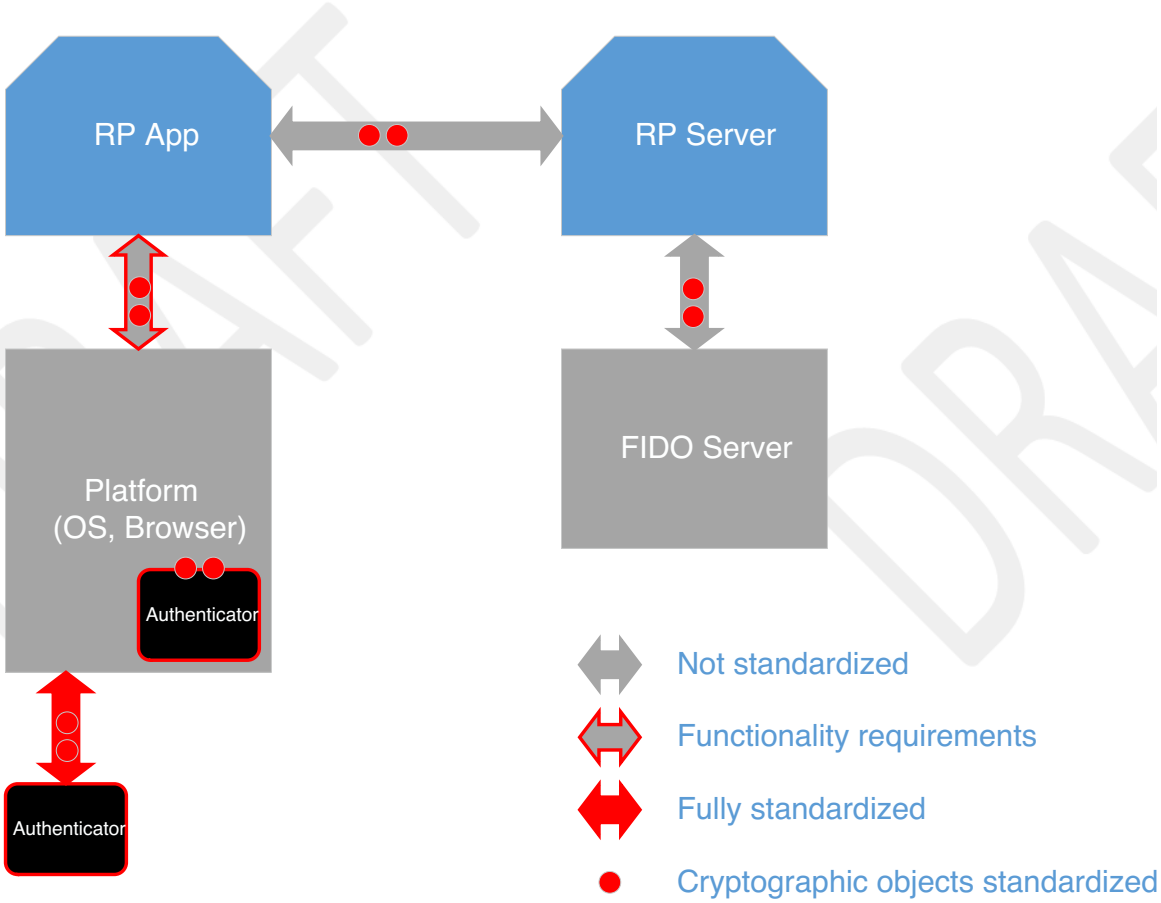
The key words must, must not, required, should, should not, recommended, may, and optional in this specification are to be interpreted as described in [RFC2119].

2. Overview

This section is non-normative.

FIDO defines an authenticator to maintain **FIDO Credentials** (see [FIDOGlossary]) and protect the use of such FIDO Credentials by some **User**

Gesture. FIDO 2 specifies the format of the cryptographic objects (attestationStatement [FIDOKeyAttestation] and FIDOSignature [FIDOSignatureFormat]).



3. The Relying Party ID (RP ID) Concept

Authenticators use different credentials for different FIDO Relying Parties (RPs). An RP comprises one or more application *facets*. Examples of facets are: a web application, or a native app (e.g., Android App, iOS App, or Windows App). Facets have identifiers -- for example, an Android app can be identified by its package signature, and a web application can be identified by its *web origin* [RFC6454]. As a full example, an RP named "Example" may include various subdomains under example.com ("www.example.com", "login.example.com", etc.), as well as fielding various Android, iOS, and Windows Apps. A user's FIDO Authenticator uses the **same** credential for a user's given RP account across an RP's application facets, and uses a **different** credential when interacting with other RPs.

We achieve this through the following mechanism: Each RP is associated with an *RP ID*, which is the "public suffix + 1" or "PS+1" (which is also referred to as the "Effective Top-Level Domain plus One" or "eTLD+1"), that the RP uses as the basis for its DNS names [PUBLICSUFFIXWEB]. To continue the above example, "www.example.com" and "login.example.com" share the PS+1 of "example.com", which is thus the RP ID of the "Example" RP.

When generating, or using, FIDO credentials on the web platform, i.e., via browser-implemented JavaScript APIs, the FIDO Client derives the RP ID from the web origin of the caller of the FIDO API. For example, if the calling web origin is "https://login.example.com", the FIDO Client determines the RP ID of that caller to be "example.com".

When generating, or using, credentials via "native" platform APIs (e.g., Android, Window, iOS), it is the platform's responsibility to associate a native app with an RP ID. For example, the Android OS platform must provide a mechanism for associating Example's Android app with the eTLD+1 "example.com". One way it may accomplish this, for example, is by having the same *Android OS developer account* prove ownership of both the Example Android app and the "example.com" domain name, or through some other mechanism determined by the platform provider.

Requirements incorporating the RP ID concept in platform APIs are expressed in the following section.

4. Platform API Requirements

In order to natively support FIDO, a platform must provide the following functionality:

1. **Credential Generation**

   There must be some API functionality allowing an application to generate a new FIDO Credential.
   The purpose of this functionality is to use asymmetric keys protected by a user gesture in a challenge response protocol for authentication.

   1.1. Authenticator Discovery

   At a minimum, the discovery API must provide the ability for an application to query for all available authenticators on the device and their associated authenticator data (including at a minimum authenticator ID, protocol version supported, and supported protocol extensions).
   The purpose of this functionality is to allow the relying party to select the most appropriate authenticator without user input.

   1.2. Authenticator Selection

   There must be some API functionality allowing an application to influence the authenticator being used for generating a new FIDO Credential.
   The purpose of this functionality is to allow the relying party to select the most appropriate authenticator without user input.

   1.3. Authenticator Behavior

There must be some API functionality allowing an application to influence the behavior (e.g. details of user verification, specifics of the crypto, etc.) of the authenticator being used for generating a new FIDO Credential - to the extent this is supported by the authenticator.
The purpose of this functionality is to enable an ecosystem of authenticators.

1.4. Attestation

There must be some API functionality allowing an application to request an AttestationStatement [FIDOKeyAttestation] for a FIDO Credential.
The purpose of this functionality is to provide a cryptographic proof of the authenticator model protecting the FIDO Credential to the relying party.

The `attData` field in the attestation object must include the public key (i.e. FIDO Credential) or a cryptographic hash of it.

1.5. Application Context Binding

There must be some API functionality allowing an application to bind an application context to the AttestationStatement.
The purpose of this functionality is to provide a general means for providing the server challenge and any kind of channel binding. This is required in order to protect against replay attacks as well as man-in-the-middle attacks.

1.6. RP ID Binding

There must be some API functionality binding the FIDO Credential being generated to the RP ID of the calling application, the RP ID must be in the form of a PS+1 domain name, and the platform provider must have a means of associating native applications to the RP ID
The purpose of this functionality is to restrict the access of a FIDO Credential to applications implemented by the same relying party.

2. **Credential Usage**

There must be some API functionality allowing an application to use a FIDO Credential for creating a FIDOSignature object [FIDOSignatureFormat].
The purpose of this functionality is to use asymmetric keys protected by a user gesture in a challenge response protocol for authentication.

2.1. Get Credential

There must be some API functionality allowing an application to regain access to a previously generated FIDO Credential.
The purpose of this functionality is to allow an application to generate a FIDO Credential once and use it multiple times.

2.2. RP ID Binding

There must be some API functionality preventing applications with different RP IDs from accessing FIDO Credentials bound to other RP IDs.
The purpose of this functionality is to protect the user's privacy.

2.3. Application Context Binding

There must be some API functionality allowing an application to bind an application context to the AttestationStatement.
The purpose of this functionality is to have a mean to provide the server challenge and any kind of channel binding. This is required in order to protect against replay attacks and against man-in-the-middle attacks.

2.4. Authenticator Behavior

There must be some API functionality allowing an application to influence the behavior (e.g. details of user verification, specifics of the crypto, etc.) of the authenticator being used - to the extent this is supported by the authenticator.
The purpose of this functionality is enable an ecosystem of authenticators.

2.5. Pluggable Authenticators

The platform shall support external authenticators plugged into the platform using the FIDO Client To Authenticator Protocol [FIDOCTAP]. If the platform doesn't support external authenticators as described above, it must have at least one embedded authenticator.

3. **Digital Signature**

Platforms must support the generation of a FIDOSignature object [FIDOSignatureFormat].

3.1. Client Data

The Client Data [FIDOSignatureFormat] is composed of several data elements (e.g. server challenge, facetId etc.), some of which may not be directly available to the platform (e.g. the server challenge). The platform signature API must support a way for its invokers to provide relevant Client Data elements as input.

3.2. Hashed Client Data to Authenticators

The Platform FIDO signature implementation must provide the hashed Client Data to the authenticator for inclusion in the authenticator signature.

3.3. Supporting Key Determination at the Authenticator

The Platform signature implementation must enable the authenticator to determine which key pair should be used to generate the authenticator signature.

3.4. Extra Authenticator Data

The Platform signature implementation must preserve any extra data passed by the authenticator in the authenticator signature and include such data in the FIDOSignature object.

3.5. **Miscellaneous**

3.5.1. Platforms must provide an API to delete keys from the platform.

An example on how to provide the required functionality can be found in [FIDOWebApi].

A. References

A.1 Normative references

[FIDOCTAP]
. *FIDO 2.0: Client To Authenticator Protocol*. URL: https://fidoalliance.org/specs/fido-v2.0-rd-20170927/fido-client-to-authenticator-protocol.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

[FIDOSignatureFormat]
. *FIDO 2.0: Signature format*. URL: https://fidoalliance.org/specs/fido-v2.0-ps-20150904/fido-signature-format-v2.0-ps-20150904.html

[FIDOWebApi]
. *FIDO 2.0: Web API for accessing FIDO 2.0 credentials*. URL: https://fidoalliance.org/specs/fido-v2.0-ps-20150904/fido-web-api-v2.0-ps-20150904.html

[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

A.2 Informative references

[PUBLICSUFFIXWEB]
. *Public Suffix List*. URL: https://publicsuffix.org/

[RFC6454]
A. Barth. *The Web Origin Concept (RFC 6454)*. June 2011. URL: http://www.ietf.org/rfc/rfc6454.txt