WORKING DRAFT

Credential Exchange Protocol

Working Draft, October 03, 2024

This version:
https://fidoalliance.org/specs/cx/cxp-v1.0-wd-20241003.html

Issue Tracking:
GitHub

Editor:
Nick Steele (1Password)

Contributors:
Rew Islam (Dashlane)
Anders Åberg (Bitwarden)
René Léveillé (1Password)
Oscar Hinton (Bitwarden)
Jonathan Salamon (Dashlane)
Ayman Bedair (NordPass)
Lee Campbell (Google)
Reema Bajwa (Google)

Copyright © 2024 FIDO Alliance. All Rights Reserved.

Abstract

This document defines a protocol to securely move one or more credentials between two credential providing applications same or separate devices.

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://fidoalliance.org/specifications/.

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

This is a Working Draft Specification and is not intended to be a basis for any implementations as the Specification may change. This document is merely a FIDO Alliance working group internal and **member-confidential** document. It has no official standing of any kind and does not represent consensus of the FIDO Alliance. 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. Introduction§

Individuals and organizations use credential providers to create and manage credentials on their behalf as a means to use stronger authentication factors. These credential providers can be used in browsers, on network servers, and on mobile and desktop platforms, and often sharing or synchronizing credentials between different instances of the same provider is an easy and common task.

However, the transfer of credentials between two different providers has traditionally been an infrequent occurrence, such as when a user or organization is attempting to migrate credentials from one provider to another. As it becomes more common for users to have multiple credential providers that they use to create a manage credentials, it becomes important to address some of the security concerns with regard to migration currently:

- Credential provider applications often export credentials to be imported in an insecure format, such as CSV, that undermines the security of the provider and potentially opens the credential owner to vulnerability.
- Credential providers have no standard structure for the exported credential CSV, which can sometimes result in failure to properly migrate one or more credentials into a new provider.
- Some credentials might be unallowed to be migrated, due to device policy or lack of algorithmic capability by the importing credential provider.

- Because organizations lack a secure means of migrating user credentials, often they will apply device policy that prevents the export of credentials to a new provider under any circumstances, opting to create multiple credentials for a service.

In order to support credential provider interoperability and provide a more secure means of credential transfer between providers, this document outlines a protocol for the import and export of one or more credentials between two credential providers on behalf of a user or organization in both an offline or online context. Using Diffie-Hellman key exchange, this protocol allows the creation of a secure channel or data payload between two providers.

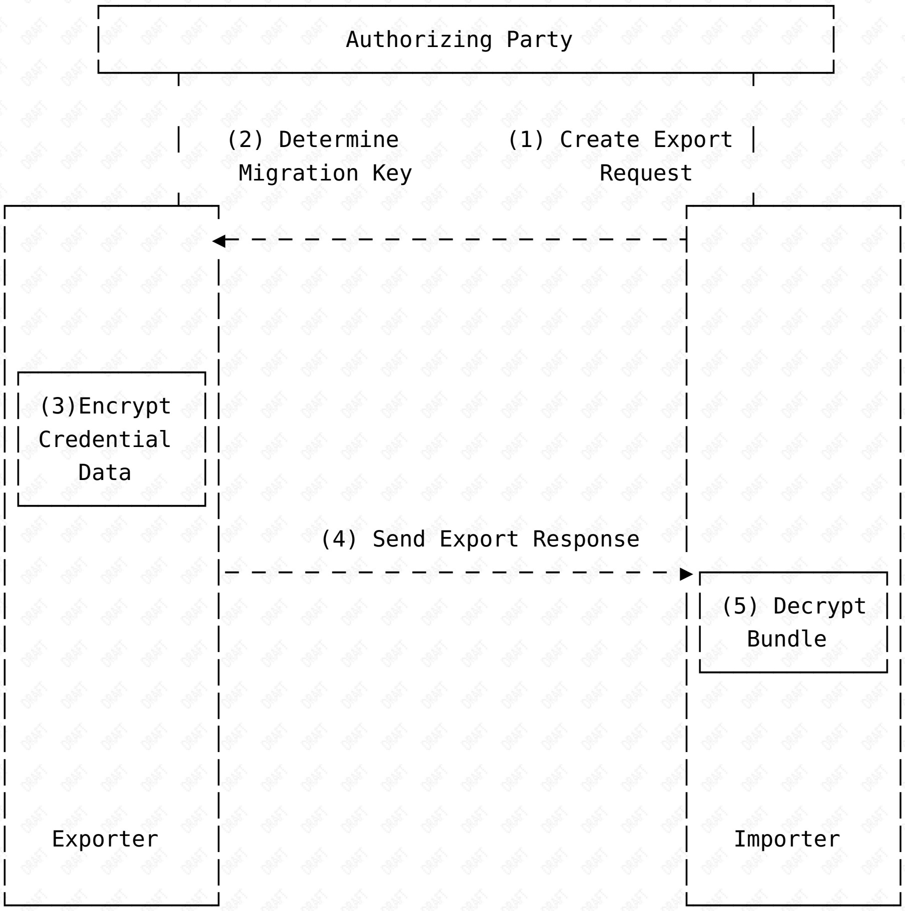
1.1. Scope§

This protocol describes the secure transmission of one or more credentials between two credential providers on the same or different devices managed by the same credential owner, capable of function in both online and offline contexts. This protocol does not make any assumptions about the channels in which credential data is passed from the source provider to the destination provider. The destruction of credentials after migration by the credential provider source is out of scope as well.

1.2. Terminology§

{::boilerplate bcp14-tagged} Certain security-related terms are to be understood in the sense defined in[RFC4949]. These terms include, but are not limited to, "attack", "authentication" "authorization", "certificate", "credential", "encryption", "identity", "sign", "signature", "trust", "validate", and "verify".

2. Protocol Overview§


Authorizing Party
(2) Determine Migration Key
(1) Create Export Request
(3)Encrypt Credential Data
(4) Send Export Response
(5) Decrypt Bundle
Exporter
Importer

The flow illustrated above shows the following:

DRAFT

1. The importing credential provider initiates the flow by creating an export request for the exporting provider. The import request includes a challenge, the scope of the export request, and a declaration of the type of encryption scheme to be used. In most cases, this will use Diffie-Hellman and the importer will provide a public key in the export request.
2. If an end-user and/or authorizing party approves the request, the exporter uses the export request key to generate or retrieve the migration key used to encrypt the credential data.
3. The source collects and encrypts the requested credential data for export and signs the challenge provided by the importer.
4. The export response is sent to the destination provider and includes the encrypted credential data, the signed challenge response, and the public key of the source credential provider’s migration key.
5. The importing provider validates the challenge and retreives the migration key, decrypting the credential data which is formatted normatively with CXF, and then stores the result.

2.1. Participants§

Credential Owner

An entity that is able to access and authenticate with the credentials stored within acredential provider. The credential owner is in charge of authorizing or delegating authorization of the migration between the Exporter and the Importer In the case of a credential owner being an individual or is authorized by an organization to manage these credentials, they can be referred to as the end-user.

Credential Provider

Hardware or software capable of storing and managing credentials on behalf of aCredential Owner. While there can be assumptions in this document that the credential providers are distinct participants, a single provider MAY operate as both the Importer and Exporter in an exchange.

Importing Provider
Importer

The importing provider, or importer, initiates the export request and is the final storage destination for the exported credentials.

Exporting Provider
Exporter

The exporting provider, or exporter, encrypts and transfers the credential data to theimporting provider.

Authorizing Party

An OPTIONAL authority that can grant and attest certificates on behalf of aCredential Owner. These certificates are used to authenticate the credential agent and MAY be used to create the migration key used on behalf of the source and importing provider.

2.2. Supporting Different Network Conditions§

This protocol can work in both online and offline scenarios, as well as in air-gapped networks where one or both devices might not have access to the internet. Different network conditions might result in participants like an authorizing party or another outside service from being included, but the core exchange protocol should not be affected.

3. Protocol API§

3.1. Credential Types§

Credential Types are defined through the Credential Exchange Format [CXF] as the `CredentialType` enum. The exported credentials MUST be formatted using [CXF] in order to have interoporability.

3.2. Export Request§

The export request initiates the protocol which contains a set of encryption parameters. These encryption parameters MUST have an associated public key if it is necessary for that instance of given parameters. This request or elements of it MAY be created by the authorizing party if one is present, or created by the destination credential provider with or without input from the credential owner.

```
ExportRequest = {
    version: uint .size 2 .default 0,
    hpke: [ + HpkeParameters ],
    archive: [ + ArchiveAlgorithm / tstr ],
    mode: tstr,
    importer: tstr,
    ? credentialTypes: [ + CredentialType / tstr ],
    ? knownExtensions: [ + tstr ],
  }
```

version

The protocol version that the Importing Provider wants to use in the exchange. The version MUST correspond to a published level of the CXP standard.

hpke

This member defines a list of HPKE Parameters that the Importing Provider supports in order of preference. It is up to the Exporting Provider to select a matching set of parameters that both support.

archive

This member defines a list of archiving algorithms that theImporting Provider supports in order of preference. It is up to the Exporting Provider to select an algorithm that both support. The values of this list SHOULD be members of Archive Algorithm and the Exporting Provider MUST ignore any unknown values.

mode

Defines the Response Mode of how the Exporting Provider should respond.

importer

This member is the Relying Party Identifier of the Importing Provider.

credentialTypes

This OPTIONAL member lists the types of credentials that theImporting Provider understands and requests from the Exporting Provider. This list SHOULD be validated by the user before initiating the exchange. The values in the list SHOULD be members of `CredentialType` and the Exporting Provider MUST ignore any unknown values.

If this member is not present then it is understood that theImporting Provider is requesting all credential types. If this member is present but the list is empty, the Exporting Provider MUST send only the `Account` object without any `Collection` information.

knownExtensions

This OPTIONAL member lists the extensions that theImporting Provider understands. This list SHOULD be members of name defined in [CXF] and the Exporting Provider MUST ignore all unknown values.

If this member is not present, then it is understood that theImporting Provider is requesting all extensions that the Exporting Provider wishes to include. If this member is present but the list is empty, theExporting Provider MUST NOT include any extensions in the resulting export.

3.2.1. Export Request File§

In response modes where the Importer cannot directly request an exchange from an Exporter, the Export Request SHALL be stored as a JSON-encoded document to be supplied to theExporter by the Credential Owner.

3.2.2. Response Modes§

Response modes allow providers to negotiate on how they wish to store or receive the Export Response. They do not change the response payload, but allow for different usage scenarios.

```
ResponseMode =
    "direct" /
    "indirect"/
    "self"
```

direct
In the case that a platform or Exporter provides an interface or transport over which the exchange ceremony can take place, the Importer may submit the Export Request using this layer and if indirect is requested, the Exporter MUST return the Export Response over the same transport or return an appropriate error.

indirect
For a platform or Exporter that does not provide a direct means of exchange, or at the request of the Credential Owner, an Importer can request indirect mode from the Exporter, which MUST output the Export Response to the filesystem. The Importer MAY request this mode in the case that a direct means of exchange is possible, or produce an Export Request as file that can be provided to the Exporter by the Credential Owner.

self
This mode SHOULD be used in cases where the Exporter and Importer are the same entity in the exchange. If self is requested, the Credential Provider may want use a symmetric key provided by the Credential Owner or Authorizing Party to derive the {ExportResponse} encryption key. This mode is meant to facilitate the secure storage and backup of credentials for a Credential Owner outside of the credential providers.

3.3. Export Response§

Includes both the credential payload and any metadata necessary to decrypt and marshall the credentials into the importer’s storage. The response can be received directly by the Importer. If it cannot be received directly, the Export Response SHALL be stored as a JSON-encoded document to be supplied to the Importer by the Credential Owner.

```
ExportResponse = {
    version: uint .size 2 .default 0,
    hpke: HPKEParameters,
    archive: ArchiveAlgorithm / tstr,
    exporter: tstr,
    payload: b64url,
}
```

version
The protocol version that the Exporting Provider understands. The value SHOULD be the same as version however there is the possibility of the Exporting Provider having a previous version of the protocol implemented and therefore responding with a lower version. The Importing Provider MAY refuse this version downgrade.

hpke
This member defines the encryption parameters selected by the Exporting Provider. The value MUST correspond to an entry in hpke.

archive
This member defines the archiving algorithm selected by the Exporting Provider. The value MUST correspond to an entry in archive.

exporter
This member is the Relying Party Identifier or the Exporting Provider.

payload

DRAFT

This contains the base64url encoded Credential Payload.

3.4. Credential Payload§

One or more normatively formatted credentials that are passed inside the export response. The format MUST follow the zip archive format as defined in [CXF] where each file is separately encrypted using the key defined by the selected HPKE Parameters. The file names are replaced with the anonymous identifier in the export request. All files are stored as JSON Web Encryption files.

```
CXP-Export/
├─ index.jwe
├─ documents/
│  ├─ 1b3.jwe
│  ├─ d5f.jwe
│  ├─ 7h9.jwe
```

3.5. Supporting Types§

3.5.1. HPKE Parameters ## {#sctn-hpke-parameters}§

```
HPKEParameters = {
    mode: HPKEMode / tstr,
    kem: uint .size 2,
    kdf: uint .size 2,
    aead: uint .size 2,
    key: JWK
}
```

mode

The encryption mode as defined in [RFC9180] and SHOULD be a member of HPKE Mode. The Exporting Provider SHOULD ignore any HPKE Parameters where this value is unknown.

kem

The encryption key encapsulation method as defined in [RFC9180]. The value SHOULD be from the HPKE KEM Identifiers IANA table, the Exporting Provider SHOULD ignore any HPKE Parameters where this value is unknown.

kdf

The encryption key derivation function as defined in [RFC9180]. The value SHOULD be from the HPKE KDF Identifiers IANA table, the Exporting Provider SHOULD ignore any HPKE Parameters where this value is unknown.

aead

The authenticated encryption with additional data algorithm as defined in [RFC9180]. The value SHOULD be from the HPKE AEAD Identifiers IANA table, the Exporting Provider SHOULD ignore any HPKE Parameters where this value is unknown.

key

This member is only present in the case that the option is not using a pre-shared key. It is a `JSON Web Key` representing that provider’s key necessary for the other provider’s generation of the AEAD key.

3.5.2. HPKE Modes§

DRAFT

DRAFT

```
HPKEMode =
    "base" /
    "psk" /
    "auth" /
    "auth-psk"
```

base
Base mode of encrypting a public key.

psk
Authenticates the possession of a pre-shared key.

auth
Authenticates the possession of a KEM private key.

auth-psk
Authenticates possession of both a pre-shared key and a KEM private key.

3.5.3. Archive Algorithms§

```
ArchiveAlgorithm =
    "deflate"
```

deflate
Archiving through the use of the DEFLATE algorithm defined in[RFC1951].

4. IANA Considerations§

This document has no IANA actions.

5. Implementation Requirements§

This section defines which algorithms and features of this specification are mandatory to implement. Applications using this specification can impose additional requirements upon implementations that they use.

6. Security Considerations§

This document outlines a secure means of credential exchange through the encryption of credentials but does not make any considerations for the security and threats inroduced by Credential Providers themselves. This section outlines considerations to make when implementing CXP and potential areas of vulnerability in objects used.

6.1. Key Compromise§

If one of the Credential Providers is compromised, or the encrypting credentials used by theCredential Owner or Authorizing Party to create the Export Response are compromised, the integrity of the the protocol is no longer ensured. It is important that the key material used provides a strong guarantee of security and privacy for the user, and that in the event of a breach to the Credential Providers, the Credential Owner is informed and can take remediation steps.

Index§

Terms defined by this specification§

DRAFT

dfn for ExportResponse

Terms defined by reference§

[CXF] defines the following terms:

- Account
- Collection
- CredentialType
- cxf
- name

[RFC7517] defines the following terms:

- JWK

[WebAuthn] defines the following terms:

- relying party identifier

References§

Normative References§

[CXF]
R. Léveillé. *Credential Exchange Format*. June 20, 2024. FIDO Alliance Working Draft. URL: https://drafts.fidoalliance.org/fido-2/stable-links-to-latest/cxf.html

[RFC4949]
R. Shirey. *Internet Security Glossary, Version 2*. August 2007. Informational. URL:https://www.rfc-editor.org/rfc/rfc4949

[RFC7517]
M. Jones. *JSON Web Key (JWK)*. May 2015. Proposed Standard. URL:https://www.rfc-editor.org/rfc/rfc7517

[RFC9180]
R. Barnes; et al. *Hybrid Public Key Encryption*. February 2022. Informational. URL:https://www.rfc-editor.org/rfc/rfc9180

[WebAuthn]
Dirk Balfanz (Google); et al. *Web Authentication: An API for accessing Public Key Credentials Level 2*. 8 April 2021. TR. URL: https://www.w3.org/TR/webauthn-2/

Informative References§

[RFC1951]
P. Deutsch. *DEFLATE Compressed Data Format Specification version 1.3*. May 1996. Informational. URL: https://www.rfc-editor.org/rfc/rfc1951