# Credential Exchange Format

## Working Draft, May 22, 2024

**This version:**
https://fidoalliance.org/specs/cx/cxf-v1.0-wd-20240522.html

**Issue Tracking:**
Github
GitHub

**Editor:**
René Léveillé (1Password)

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



## Abstract

This document defines the data structures and format of credentials being passed or referenced between two applications during credential exchange.

## 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§

NOTE: The name of this specification is subject to change.

Credential migration has traditionally been an infrequent occurrence, when a user is attempting to migrate credentials from one credential provider to a new one, such as moving to a new password manager or mobile device. This has historically been a very manual process for credential providers, as there exists no normative structure to the credentials being exported by a credential provider. The goal of CXF is to define those normative data structures to allow for interoperability and control by resource owners over credentials that need to be migrated or referenced by one or more providers.

## 1.1. Motivation§

Historically, there is no normative structure for passing credentials between credential providers, leading to a lack of interoperability and in some cases, the loss of credentials during transfer. While the Credential Exchange Protocol aims to define the standard protocol for the import and export of credentials, there additionally needs to be a standard format for the credential data being exchanged. The Credential Exchange Format aims to solve non-normative credential transfer for this protocol and other forms of credential exchange between providers to help make the process easier for users and organizations to securely handle exchange events.

## 1.2. Scope§

This document outlines the data structures and format needed to exchange credentials and does not make any assumptions about the protocol used for the transfer, such as the protocol outlined by CXP.

## 1.3. Terminology§

[Define any key terms and concepts used throughout this document.]

# 2. Format Overview§

CXF defines a schema around an account owner and all of its associated secrets. These secrets are defined in a way where the most common attributes have dedicated fields, all the while allowing extra fields to be added as extensions.

## 2.1. Format Design Principles§

Everything in a zip archive, each part is encrypted using the keys defined in CXP.

```
CXF-Export/
├─ index.json
├─ documents/
│  ├─ foo.docx
│  ├─ vault.ico
│  ├─ bar.pdf
```

[Detail the key principles that guided the design of the CXF format.]

## 2.2. Data Structures§

[Explain the overall structure of CXF, including its main sections and their purposes.]

## 2.3. Encoding Considerations§

[Discuss considerations related to encoding and data representation within the CXF format.]

# 3. Data Structure Specification§

[Provide detailed specifications for each section of the CXF data structure.]

## 3.1. Header Section§

[Describe the contents and purpose of the header section within the CXF data structure.]

### 3.1.1. `Header`§

```
dictionary Header {
    required unsigned short version;
    required DOMString exporter;
    required unsigned long long timestamp;
    required sequence<Account> accounts;
};
```

**`version`, of type unsigned short**
: The version of the format definition, The current version is `0`.

**`exporter`, of type DOMString**
: The name of the exporting app (should this be an rpid?)

**`timestamp`, of type unsigned long long**
: The UNIX timestamp during at which the export document was completed.

**`accounts`, of type sequence<Account>**
: The list of `Account`s being exported.

### 3.1.2. `Account` Dictionary§

```
dictionary Account {
    required Base64URLString id;
    required DOMString userName;
    required DOMString email;
    DOMString fullName;
    DOMString icon;
    required sequence<Collection> collections = [];
    required sequence<Item> items = [];
    sequence<Extension> extensions;
};
```

**`id`, of type Base64URLString**
A unique identifier for the `Account` which is machine-generated and an opaque byte sequence with a maximum size of 64 bytes. It is not meant to be displayed to the user.

**`userName`, of type DOMString**
A pseudonym defined by the user to name their account. If none is set, this should be an empty string.

**`email`, of type DOMString**
The email used to register the account in the previous provider.

**`fullName`, of type DOMString**
This OPTIONAL field holds the user's full name.

**`icon`, of type DOMString**
This OPTIONAL field defines if the user has set an icon as the account's avatar.

**`collections`, of type sequence<Collection>, defaulting to `[]`**
All the collections this account owns. If the user has collections that were shared with them by another account, it MUST not be present in this list.

**`items`, of type sequence<Item>, defaulting to `[]`**
All items that this account owns and that are not stored in a collection, or are a part of many collections. If the user has access to items that were shared with them by another account, it MUST not be present in this list.

**`extensions`, of type sequence<Extension>**
This OPTIONAL field contains all the extensions to the `Account`'s attributes.

#### 3.1.3. `Collection` Dictionary

```
dictionary Collection {
    required Base64URLString id;
    required DOMString title;
    DOMString subtitle;
    DOMString icon;
    required sequence<Item> items = [];
    sequence<Collection> subCollections;
    sequence<Extension> extensions;
};
```

**`id`, of type Base64URLString**
A unique identifier for the `Collection` which is machine-generated and an opaque byte sequence with a maximum size of 64 bytes. It is not meant to be displayed to the user.

**`title`, of type DOMString**
The display name of the `Collection`.

**`subtitle`, of type DOMString**
This OPTIONAL field is a subtitle or a description of the `Collection`.

**`icon`, of type DOMString**
This OPTIONAL field is a relative path from this file to the icon file acting as this `Collection`'s avatar.

**`items`, of type sequence<Item>, defaulting to `[]`**

Enumerates all the items in this `Collection`.

**`subCollections`, of type sequence<Collection>**
Enumerates any sub-collections if the provider supports recursive organization.

**`extensions`, of type sequence<Extension>**
This enumeration contains all the extensions to the `Collection`'s attributes.

## 3.2. Credential Section

[Explain the components and fields of the credential section, detailing how credentials are represented.]

### 3.2.1. `Item` Dictionary

```
dictionary Item {
    required Base64URLString id;
    required unsigned long long creationAt;
    required unsigned long long modifiedAt;
    required DOMString type;
    required DOMString title;
    DOMString subtitle;
    required sequence<Credential> credentials;
    sequence<DOMString> tags;
    sequence<Extension> extensions;
};
```

**`id`, of type Base64URLString**
A unique identifier for the `Item` which is machine-generated and an opaque byte sequence with a maximum size of 64 bytes. It is not meant to be displayed to the user.

**`creationAt`, of type unsigned long long**
The UNIX timestamp at which this item was originally created.

**`modifiedAt`, of type unsigned long long**
The UNIX timestamp of the last modification brought to this `Item`.

**`type`, of type DOMString**
This member contains a hint to the objects in the `credentials` array. It SHOULD be a member of `ItemType`.

**`title`, of type DOMString**
This member's value is the user-defined name or title of the item.

**`subtitle`, of type DOMString**
This OPTIONAL member is a subtitle or description for the `Item`.

**`credentials`, of type sequence<Credential>**
This member contains a set of `Credential`s that SHOULD be associated to the `type`.

**`tags`, of type sequence<DOMString>**
This OPTIONAL member contains user-defined tags that they may use to organize the item.

**`extensions`, of type sequence<Extension>**
This member contains all the extensions the exporter MAY have to define the `Item` type that is being exported to be as complete of an export as possible.

## 3.3. Credential Data Types

### 3.3.1. `Credential` Base Dictionary

```
dictionary Credential {
    required DOMString type;
};
```

**type, of type DOMString**

This member contains a **string representation of the credential type**. The value SHOULD be a member of `CredentialType` but importers MAY attempt to store unknown item types in their own way as a best effort.

> NOTE: The `type` value will be the same for all items implementing a particular credential which means that developers can rely on `obj.type` returning a string that unambiguously represents the specific kind of `Credential` they are dealing with.

#### 3.3.2. `BasicAuth`§

```
dictionary BasicAuth: Credential {
    required CredentialType type = "basic-auth";
    required sequence<DOMString> urls;
    EditableField username;
    EditableField password;
};
```

#### 3.3.3. `Passkey` Dictionary§

```
dictionary Passkey: Credential {
    required CredentialType type = "passkey";
    required Base64URLString credentialId;
    required DOMString rpId;
    required DOMString userName;
    required DOMString userDisplayName;
    required DOMString userHandle;
    // JWK, CoseKey, pkcs#8 ?
    required object key;
    Fido2Extensions fido2Extensions;
};
```

#### 3.3.4. `CreditCard`§

```
dictionary CreditCard: Credential{
    required CredentialType type = "credit-card";
    required DOMString number;
    required DOMString fullName;
    DOMString cardType;
    DOMString verificationNumber;
    DOMString expiryDate;
    DOMString validFrom;
};
```

### 3.4. Metadata Section§

[Detail the metadata section's role in providing additional information about the credential data.]

## 3.5. Supporting Data Structures

### 3.5.1. `ItemType` Enumeration

```
enum ItemType {
    "login",
    "document",
    "identity"
};
```

**`login`**
: An `Item` that SHOULD contain any of the following `Credential` types:

- `BasicAuth`,
- `Passkey`,
- `Totp`,
- `CryptographicKey`.

**`document`**
: An `Item` that SHOULD contain any of the following `Credential` types:

- `Note`,
- `File`.

**`identity`**
: An `Item` that SHOULD contain any of the following `Credential` types:

- `CreditCard`
- `Address`
- `DriverLicense`
- `SocialSecurityNumber`

### 3.5.2. `CredentialType` Enumeration

```
enum CredentialType {
    "basic-auth",
    "passkey",
    "totp",
    "cryptographic-key",
    "note",
    "file",
    "address",
    "credit-card",
    "social-security-number"
};
```

### 3.5.3. `EditabelField` Dictionary

```
dictionary EditableField {
    required Base64URLString id;
    required DOMString fieldType;
    required DOMString value;
    DOMString label;
    DOMString designation;
};
```

#### 3.5.4. `Fido2Extensions` dictionary§

```
dictionary Fido2Extensions {
    Fido2HmacSecret hmacSecret;
    Base64URLString credBlob;
    Fido2LargeBlob largeBlob;
    boolean payments;
    Fido2SupplementalKeys supplementalKeys;
};
```

#### 3.5.5. `Fido2HmacSecret`§

```
dictionary Fido2HmacSecret {
    required DOMString algorithm;
    required Base64URLString secret;
};
```

#### 3.5.6. `Fido2LargeBlob`§

```
dictionary Fido2LargeBlob {
    required unsigned long long size;
    required DOMString alg;
    required Base64URLString data;
};
```

#### 3.5.7. `Fido2SupplementalKeys`§

```
dictionary Fido2SupplementalKeys {
    boolean device;
    boolean provider;
};
```

### 3.6. Defined Extensions§

```
dictionary Extension {
    required DOMString name;
    // Should there be an included schema? or use a URI to define the schema?
};
```

**name, of type DOMString**

The name of the extension which will define the contents associated. If the extension is defined in this

document then the value will directly use that name. If this is a custom extension defined by the exporter, then the value MUST take the following format: `EXPORTER_RP_ID/EXTENSION_NAME`. As an example `1password.com/VaultType`.

#### 3.6.1. Sharing an Entity (`Sharing`)

```
dictionary Shared: Extension {
    required DOMString name = "shared";
    required sequence<SharingAccessor> accessors;
};
```

##### *3.6.1.1. `SharingAccessor`*

```
dictionary SharingAccessor {
    required DOMString type;
    required Base64URLString accountId;
    required DOMString name;
    required sequence<DOMString> permissions;
};
```

**`type`, of type DOMString**
: This member specifies the type of access that the user by the `accountId` has to this entity. The value SHOULD be a member of `SharingAccessorType` but importers MUST ignore any `SharingAccessor` entries that are unknown values for this member.

**`accountId`, of type Base64URLString**
: This member points to an `Account`'s `id` that has been given access to this collection by the current `Account`.

**`name`, of type DOMString**
: This member contains the `userName` if `type` is of value `user`. If `type` is of value `group` this member then contains the group's name.

**`permissions`, of type sequence<DOMString>**
: This member lists the permissions that this `accountId` has to the associated `Collection`. The values SHOULD be members of `SharingAccessorPermission` but importers MUST ignore unknown values, ignoring any unknown values in `permissions`. The importer MUST ignore any `SharingAccessor`s that have an empty `permissions` list, whether it's been exported as empty, or the result of ignoring all unknown values.

##### *3.6.1.2. `SharingAccessorType` Enumeration*

```
enum SharingAccessorType {
    "user",
    "group"
};
```

**`user`**
: Indicates the respective `SharingAccessor` is describing a user's permissions on the `Collection`.

**`group`**
: Indicates the respective `SharingAccessor` is describing a group of users' permissions on the `Collection`.

#### 3.6.1.3. `SharingAccessorPermission` Enumeration§

```
enum SharingAccessorPermission {
    "read",
    "update",
    "create",
    "delete",
    "share",
    "manage"
};
```

**`read`**
: Indicates that the respective `SharingAccessor` has read permissions on all `Item`s in the associated `Collection`.

**`update`**
: Indicates that the respective `SharingAccessor` has update permissions on all `Item`s in the associated `Collection`.

**`create`**
: Indicates that the respective `SharingAccessor` has the permission to create new `Item`s in the associated `Collection`.

**`delete`**
: Indicates that the respective `SharingAccessor` has the permission to delete any `Item` in the associated `Collection`

**`share`**
: Indicates that the respective `SharingAccessor` can share any `Item` from the associated `Collection` with users or groups if they so choose.

**`manage`**
: Indicates that the respective `SharingAccessor` can manage this `Collection`, meaning they can edit the collection's attributes, share it with others, etc.

## 4. Usage Guidelines§

[Offer guidelines for using the CXF format to import and export credentials securely.]

### 4.1. Importing Credentials§

[Explain the steps and considerations for importing credentials using the CXF format.]

### 4.2. Exporting Credentials§

[Provide instructions for exporting credentials to the CXF format.]

### 4.3. Security Considerations§

[Highlight the security considerations that should be taken into account when using the CXF format.]

## 5. Examples§

[Present practical examples of importing and exporting credentials using the CXF format.]

### 5.1. Importing a Credential Set§

[Walk through the process of importing a set of credentials using CXF.]

### 5.2. Exporting a Credential Set§

[Provide an example of exporting a credential set to the CXF format.]

## 6. IANA Considerations§

[Outline considerations related to IANA registrations, including the CXF media type.]

### 6.1. CXF Media Type§

[Specify the media type for CXF and its registration details.]

## 7. Security Considerations§

[Provide an in-depth analysis of the security aspects of the CXF format and its use.]

## Conformance§

Conformance requirements are expressed with a combination of descriptive assertions and RFC 2119 terminology. The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in the normative parts of this document are to be interpreted as described in RFC 2119. However, for readability, these words do not appear in all uppercase letters in this specification.

All of the text of this specification is normative except sections explicitly marked as non-normative, examples, and notes. [RFC2119]

Examples in this specification are introduced with the words “for example” or are set apart from the normative text with `class="example"`, like this:

> EXAMPLE 1 ¶
> This is an example of an informative example.

Informative notes begin with the word “Note” and are set apart from the normative text with `class="note"`, like this:

> Note, this is an informative note.

## Index§

### Terms defined by this specification§

"address"

alg

algorithm

"basic-auth"

BasicAuth

cardType

Collection

collections

"create"

create

creationAt

credBlob

Credential

credentialId

credentials

CredentialType

"credit-card"

CreditCard

"cryptographic-key"

data

"delete"

delete

designation

device

"document"

document

EditabelField

EditableField

email

expiryDate

exporter

Extension

extensions

- dict-member for Account
- dict-member for Collection
- dict-member for Item

Fido2Extensions

fido2Extensions

Fido2HmacSecret

Fido2LargeBlob

Fido2SupplementalKeys

fieldType

"file"

fullName

- dict-member for Account
- dict-member for CreditCard

"group"

group

Header

hmacSecret

icon

- dict-member for Account
- dict-member for Collection

id

- dict-member for Account
- dict-member for Collection
- dict-member for EditableField
- dict-member for Item

"identity"

identity

Item

items

- dict-member for Account
- dict-member for Collection

ItemType

key

label

largeBlob

"login"

login

"manage"

manage

modifiedAt

name

- dict-member for Extension
- dict-member for Shared
- dict-member for SharingAccessor

"note"

number

"passkey"

Passkey

password

payments

permissions

provider

"read"

version

### Terms defined by reference§

[FileAPI] defines the following terms:

File

[WEBAUTHN-3] defines the following terms:

Base64URLString

[WEBIDL] defines the following terms:

DOMString

boolean

object

sequence

unsigned long long

unsigned short

## References§

### Normative References§


**[FileAPI]**
Marijn Kruisselbrink. *File API*. URL: https://w3c.github.io/FileAPI/

**[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

**[WEBAUTHN-3]**
Michael Jones; Akshay Kumar; Emil Lundberg. *Web Authentication: An API for accessing Public Key Credentials - Level 3*. URL: https://w3c.github.io/webauthn/

**[WEBIDL]**
Edgar Chen; Timothy Gu. *Web IDL Standard*. Living Standard. URL: https://webidl.spec.whatwg.org/


## IDL Index§

```
dictionary Header {
    required unsigned short version;
    required DOMString exporter;
    required unsigned long long timestamp;
    required sequence<Account> accounts;
};

dictionary Account {
    required Base64URLString id;
    required DOMString userName;
    required DOMString email;
    DOMString fullName;
    DOMString icon;
    required sequence<Collection> collections = [];
    required sequence<Item> items = [];
    sequence<Extension> extensions;
};

dictionary Collection {
    required Base64URLString id;
    required DOMString title;
```

```
    DOMString subtitle;
    DOMString icon;
    required sequence<Item> items = [];
    sequence<Collection> subCollections;
    sequence<Extension> extensions;
};

dictionary Item {
    required Base64URLString id;
    required unsigned long long creationAt;
    required unsigned long long modifiedAt;
    required DOMString type;
    required DOMString title;
    DOMString subtitle;
    required sequence<Credential> credentials;
    sequence<DOMString> tags;
    sequence<Extension> extensions;
};

dictionary Credential {
    required DOMString type;
};

dictionary BasicAuth: Credential {
    required CredentialType type = "basic-auth";
    required sequence<DOMString> urls;
    EditableField username;
    EditableField password;
};

dictionary Passkey: Credential {
    required CredentialType type = "passkey";
    required Base64URLString credentialId;
    required DOMString rpId;
    required DOMString userName;
    required DOMString userDisplayName;
    required DOMString userHandle;
    // JWK, CoseKey, pkcs#8 ?
    required object key;
    Fido2Extensions fido2Extensions;
};

dictionary CreditCard: Credential{
    required CredentialType type = "credit-card";
    required DOMString number;
    required DOMString fullName;
    DOMString cardType;
    DOMString verificationNumber;
    DOMString expiryDate;
    DOMString validFrom;
};

enum ItemType {
    "login",
    "document",
    "identity"
};

enum CredentialType {
    "basic-auth",
    "passkey",
    "totp",
    "cryptographic-key",
```

```
    "note",
    "file",
    "address",
    "credit-card",
    "social-security-number"
};

dictionary EditableField {
    required Base64URLString id;
    required DOMString fieldType;
    required DOMString value;
    DOMString label;
    DOMString designation;
};

dictionary Fido2Extensions {
    Fido2HmacSecret hmacSecret;
    Base64URLString credBlob;
    Fido2LargeBlob largeBlob;
    boolean payments;
    Fido2SupplementalKeys supplementalKeys;
};

dictionary Fido2HmacSecret {
    required DOMString algorithm;
    required Base64URLString secret;
};

dictionary Fido2LargeBlob {
    required unsigned long long size;
    required DOMString alg;
    required Base64URLString data;
};

dictionary Fido2SupplementalKeys {
    boolean device;
    boolean provider;
};

dictionary Extension {
    required DOMString name;
    // Should there be an included schema? or use a URI to define the schema?
};

dictionary Shared: Extension {
    required DOMString name = "shared";
    required sequence<SharingAccessor> accessors;
};

dictionary SharingAccessor {
    required DOMString type;
    required Base64URLString accountId;
    required DOMString name;
    required sequence<DOMString> permissions;
};

enum SharingAccessorType {
    "user",
    "group"
};

enum SharingAccessorPermission {
    "read",
```

```
    "update",
    "create",
    "delete",
    "share",
    "manage"
};
```