

PASSWORD MANAGER > IMPORT & EXPORT

Condition a Bitwarden .csv or .json

View in the help center:

<https://bitwarden.com/help/condition-bitwarden-import/>

Condition a Bitwarden .csv or .json

This article defines the format you should use when manually conditioning a **.csv** or **.json** for import into Bitwarden. This format is identical to that used by **.csv** or **.json** files created by [exporting your Bitwarden vault](#).

Make sure that you are using the right format depending on whether you are importing data to your vault or to an organization vault.

Condition a .csv



Tip

Bitwarden **.csv** files will only handle logins and secure notes. If you need to import or export identities and cards as well, [use JSON](#).

.csv for individual vault

[Download sample csv](#)

Create a UTF-8 encoded plaintext file with the following header as the first line in the file:

Bash

```
folder,favorite,type,name,notes,fields,reprompt,login_uri,login_username,login_password,login_totp
```

For example:

Bash

```
folder,favorite,type,name,notes,fields,reprompt,login_uri,login_username,login_password,login_totp
Social,1,login,Twitter,,0,twitter.com,me@example.com,password123,
,,login,EVGA,,,https://www.evga.com/support/login.asp,hello@bitwarden.com,fakepassword,TOTPSEED123
,,login,My Bank,Bank PIN is 1234,"PIN: 1234",,https://www.wellsfargo.com/home.jhtml,john.smith,pass
word123456,
,,note,My Note,"This is a secure note.",,,,,
```

When importing this file, select **Bitwarden (csv)** as your file format.

.csv for organization

[Download sample csv](#)

Create a UTF-8 encoded plaintext file with the following header as the first line in the file:

```
Bash

collections,type,name,notes,fields,reprompt,login_uri,login_username,login_password,login_totp
```

For example,

```
Bash

collections,type,name,notes,fields,reprompt,login_uri,login_username,login_password,login_totp
"Social,Marketing",login,Twitter,,,0,twitter.com,me@example.com,password123,
"Finance",login,My Bank,"Bank PIN is 1234","PIN: 1234",0,https://www.wellsfargo.com/home.jhtml,john.smith,password123456,
"Finance",login,EVGA,,,0,https://www.evga.com/support/login.asp,hello@bitwarden.com,fakepassword,TPSEED123
"Finance",note,My Note,"This is a secure note.",,,0,,,
```



Tip

If you're conditioning a **.csv** with nested collections, create dedicated entries for **each collection that does not have an item in it**, for example:

```
Bash

collections,type,name,notes,fields,reprompt,login_uri,login_username,login_password,login_totp
Parent Collection,,,,,,,,,
Parent Collection/First Child Collection,,,,,,,,,
Parent Collection/First Child Collection/Second Child Collection,login,Shared Credential,,,https://website.com,username,password,,
```

When importing this file, select **Bitwarden (csv)** as your file format.

Minimum required values

You may not have data for all the values shown in the above formats, however most are optional. In order for the Bitwarden **.csv** importer to function properly, you are only required to have the following values for any given object:

Bash

```
folder,favorite,type,name,notes,fields,reprompt,login_uri,login_username,login_password,login_totp
,,login,Login Name,,,,,
,,note,Secure Note Name,,,,,
```

Condition a .json

[↓ Download sample json](#)

.json for individual vault

Create a UTF-8 encoded plaintext file in the following format:

Bash

```
{
  "folders": [
    {
      "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
      "name": "Folder Name"
    },
    ...
  ],
  "items": [
    {
      "passwordHistory": [
        {
          "lastUsedDate": "YYYY-MM-00T00:00:00.000Z",
          "password": "passwordValue"
        }
      ],
      "revisionDate": "YYYY-MM-00T00:00:00.000Z",
      "creationDate": "YYYY-MM-00T00:00:00.000Z",
      "deletedDate": null,
      "id": "yyyyyyyy-yyy-yyy-yyy-yyyyyyyyyyyy",
      "organizationId": null,
      "folderId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
      "type": 1,
      "reprompt": 0,
      "name": "My Gmail Login",
      "notes": "This is my gmail login for import.",
      "favorite": false,
      "fields": [
        {
          "name": "custom-field-1",
          "value": "custom-field-value",
          "type": 0
        },
        ...
      ],
    },
  ],
}
```

```
"login": {
  "uris": [
    {
      "match": null,
      "uri": "https://mail.google.com"
    }
  ],
  "username": "myaccount@gmail.com",
  "password": "myaccountpassword",
  "totp": otpauth://totp/my-secret-key
},
"collectionIds": null
},
...
]
```

When importing this file, select **Bitwarden (json)** as your file format.

.json for organization

[↓ Download sample json](#)

Create a UTF-8 encoded plaintext file in the following format:

Bash

```
...
],
"login": {
  "uris": [
    {
      "match": null,
      "uri": "https://mail.google.com"
    }
  ],
  "username": "myaccount@gmail.com",
  "password": "myaccountpassword",
  "totp": otpauth://totp/my-secret-key
},
"collectionIds": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
},
...
]
}
```

When importing this file, select **Bitwarden (json)** as your file format.

Import to existing collections

By conditioning your organization **.json** file appropriately, you can import new login items to pre-existing [collections](#).

The following example demonstrates the proper format for importing a single item into a pre-existing collection. Note that you will need to:

- Obtain organization and collection IDs. These can be obtained by navigating to the collection in your web app and pulling them from the address bar (e.g. <https://vault.bitwarden.com/#/organizations/<organizationId>/vault?collectionId=<collectionId>>).
- Define a **"collections": []** array that contains data for the pre-existing collection, including organization and collection IDs (see above) as well as its name. As long as these 3 data points match, a new collection will not be created on import and instead items in the file will be imported to the pre-existing collection.

Bash

```
{
  "encrypted": false,
  "collections": [
    {
      "id": "b8e6df17-5143-495e-92b2-aff700f48ecd",
      "organizationId": "55d8fa8c-32bb-47d7-a789-af8710f5eb99",
      "name": "My Existing Collection",
      "externalId": null
    }
  ],
  "folders": [],
  "items": [
    {
      "id": "2f27f8f8-c980-47f4-829a-aff801415845",
      "organizationId": "55d8fa8c-32bb-47d7-a789-af8710f5eb99",
      "folderId": null,
      "type": 1,
      "reprompt": 0,
      "name": "Item to Import",
      "notes": "A login item for sharing.",
      "favorite": false,
      "login": {
        "uris": [
          {
            "match": null,
            "uri": "https://mail.google.com"
          }
        ],
        "username": "my_username",
        "password": "my_password",
        "totp": null
      },
      "collectionIds": ["b8e6df17-5143-495e-92b2-aff700f48ecd"]
    }
  ]
}
```

```
]
}
```

Minimum required key-value pairs

You may not have data for all the key-value pairs shown in the above formats, however most are optional. In order for the Bitwarden `.json` importer to function properly, you are only required to have the following key-value pairs for each object:

Bash

```
{
  "items": [
    {
      "type": 1,
      "name": "Login Item's Name",
      "login": {}
    },
    {
      "type": 2,
      "name": "Secure Note Item's Name",
      "secureNote": {}
    },
    {
      "type": 3,
      "name": "Card Item's Name",
      "card": {}
    },
    {
      "type": 4,
      "name": "Identity Item's Name",
      "identity": {}
    }
  ]
}
```

The `"login":`, `"secureNote":`, `"card":`, and `"identity":` objects can be imported as empty objects, however we recommend conditioning files with as much data as you are able.