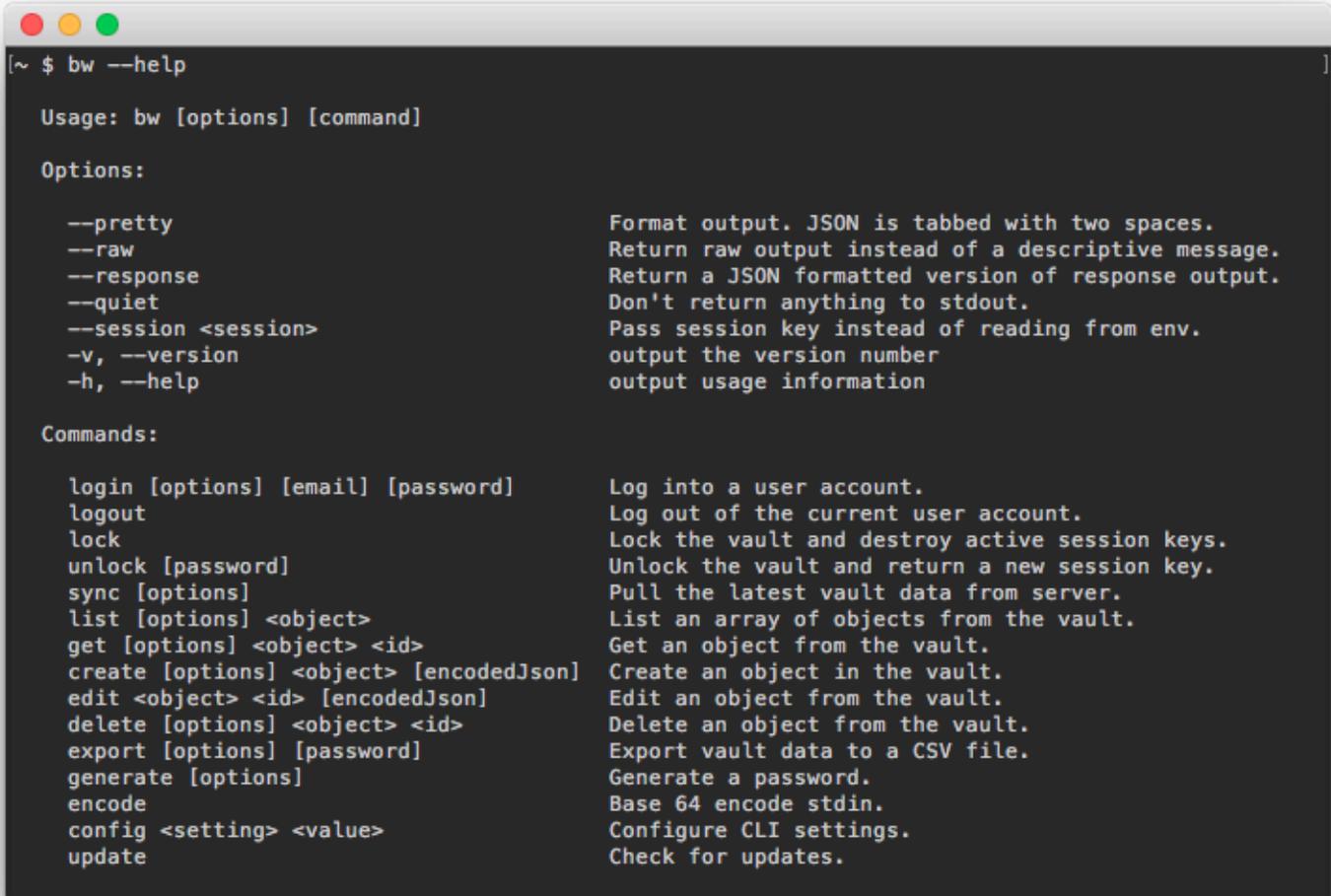


PASSWORD MANAGER > ENTWICKLERTOOLS

# Bitwarden-Kommandozeile

## Bitwarden-Kommandozeile

Die Bitwarden Kommandozeilen-Schnittstelle (CLI) ist ein leistungsstarkes, voll ausgestattetes Werkzeug zum Zugreifen auf und Verwalten Ihres Tresors. Die meisten Funktionen, die Sie in anderen Bitwarden Client-Anwendungen (Desktop, Browser-Erweiterung usw.) finden, sind auch über die CLI verfügbar.



```
[~ $ bw --help]

Usage: bw [options] [command]

Options:

  --pretty          Format output. JSON is tabbed with two spaces.
  --raw            Return raw output instead of a descriptive message.
  --response       Return a JSON formatted version of response output.
  --quiet          Don't return anything to stdout.
  --session <session> Pass session key instead of reading from env.
  -v, --version    output the version number
  -h, --help      output usage information

Commands:

  login [options] [email] [password]  Log into a user account.
  logout                               Log out of the current user account.
  lock                                 Lock the vault and destroy active session keys.
  unlock [password]                   Unlock the vault and return a new session key.
  sync [options]                       Pull the latest vault data from server.
  list [options] <object>             List an array of objects from the vault.
  get [options] <object> <id>         Get an object from the vault.
  create [options] <object> [encodedJson] Create an object in the vault.
  edit <object> <id> [encodedJson]    Edit an object from the vault.
  delete [options] <object> <id>     Delete an object from the vault.
  export [options] [password]         Export vault data to a CSV file.
  generate [options]                  Generate a password.
  encode                               Base 64 encode stdin.
  config <setting> <value>           Configure CLI settings.
  update                              Check for updates.
```

Bitwarden-Kommandozeile

Die Bitwarden CLI ist selbst dokumentiert. Vom Befehlszeileninterface aus, erfahren Sie mehr über die verfügbaren Befehle mit:

*Bash*

```
bw --help
```

Oder geben Sie `--help` als Option bei jedem `bw` Befehl ein, um verfügbare Optionen und Beispiele zu sehen:

*Bash*

```
bw list --help
```

```
bw move --help
```

Die meisten Informationen, die Sie benötigen, können Sie mit `--help` abrufen, jedoch repliziert dieser Artikel all diese Informationen und geht bei einigen Themen mehr ins Detail.

## Herunterladen und installieren

Die CLI kann plattformübergreifend auf Windows, macOS und Linux-Distributionen verwendet werden. Um das Bitwarden CLI herunterzuladen und zu installieren:

### Note

Für arm64 Geräte, installieren Sie die CLI mit `npm`.

## ⇒ Nativer ausführbarer Code

In jeder Plattform sind nativ verpackte Versionen der CLI verfügbar und haben keine Abhängigkeiten. Laden Sie herunter, indem Sie einen dieser Links verwenden:

- [Windows x64](#)
- [macOS x64](#)
- [Linux x64](#)

Beachten Sie, dass Sie, wenn Sie die heruntergeladene native ausführbare Datei verwenden, die ausführbare Datei zu Ihrem PATH hinzufügen müssen oder sonst Befehle aus dem Verzeichnis ausführen müssen, in das die Datei heruntergeladen wurde.

### Tip

Auf UNIX-Systemen erhalten Sie möglicherweise die Meldung `Permission denied`. Wenn dies der Fall ist, erteilen Sie die Erlaubnis, indem Sie folgenden Befehl ausführen:

*Bash*

```
chmod +x </path/to/executable>
```

## ⇒ NPM

Wenn Sie Node.js auf Ihrem System installiert haben, können Sie die CLI mit NPM installieren. Die Installation mit NPM ist die einfachste Möglichkeit, Ihre Installation auf dem neuesten Stand zu halten und sollte die **bevorzugte Methode für diejenigen sein, die bereits mit NPM vertraut sind**:

*Bash*

```
npm install -g @bitwarden/cli
```

Betrachten Sie das Paket auf [npmjs.org](https://www.npmjs.org).

#### **Note**

Die Installation des Bitwarden CLI auf Linux-Systemen mit **npm** kann erfordern, dass zuerst die Abhängigkeit **build-essential** (oder das entsprechende Äquivalent der Distribution) installiert wird. Zum Beispiel:

*Plain Text*

```
apt install build-essential
```

### ⇒Schokoladig

Zur Installation mit Chocolatey:

*Bash*

```
choco install bitwarden-cli
```

Betrachten Sie das Paket auf [community.chocolatey.org](https://community.chocolatey.org).

### ⇒Schnapp

Zur Installation mit Snap:

*Bash*

```
sudo snap install bw
```

Betrachten Sie das Paket auf [snapcraft.io](https://snapcraft.io).

## Anmelden

Bevor Sie sich anmelden, stellen Sie sicher, dass Ihre CLI mit dem richtigen Server verbunden ist (zum Beispiel **EU-Cloud** oder selbst gehostet) mit dem **config**-Befehl ([mehr erfahren](#)). Es gibt drei Methoden, um sich mit dem **Zugangsdaten** Befehl in der Bitwarden CLI anzumelden, von denen jede für verschiedene Situationen geeignet ist. Bitte überprüfen Sie die folgenden Optionen, um zu bestimmen, welche Methode Sie verwenden sollten:

- [Verwendung von E-Mail-Adresse und Master-Passwort](#)
- [Verwendung eines API-Schlüssels](#)
- [Verwendung von SSO](#)

Unabhängig davon, welche Option Sie verwenden, stellen Sie immer sicher, dass Sie die Befehle `bw sperren` oder `bw abmelden` verwenden, wenn Sie fertig sind.

### 💡 Tip

Bei der [Anmeldung mit E-Mail-Adresse und Master-Passwort](#) wird Ihr Master-Passwort verwendet, sodass die Befehle `login` und `unlock` zum Anmelden und Entsperren miteinander verknüpft werden können, um Ihre Identität zu authentifizieren und Ihren Tresor gleichzeitig zu entschlüsseln. Bei der Verwendung [eines API-Schlüssels](#) oder von [SSO](#) müssen Sie dem `login`-Befehl einen expliziten `bw unlock`-Befehl folgen lassen, wenn Sie direkt mit Tresordaten arbeiten wollen.

Das liegt daran, dass Ihr Master-Passwort die Quelle des Schlüssels ist, der zur Entschlüsselung der Tresordaten benötigt wird. Es gibt jedoch einige Befehle, für die Ihr Tresor nicht entschlüsselt werden muss, darunter `config`, `encode`, `generate`, `update` und `status`.

## Verwendung von E-Mail-Adresse und Passwort

Für interaktive Sitzungen wird die [Anmeldung mit E-Mail und Passwort empfohlen](#). Um sich mit E-Mail-Adresse und Passwort anzumelden:

### Bash

```
bw login
```

Dies wird eine Aufforderung für Ihre **E-Mail-Adresse**, **Master-Passwort** und (falls aktiviert) bei **Zwei-Schritt-Login-Code** initiieren. Die CLI unterstützt derzeit die zweistufige Anmeldung über [Authentifizierung](#), [E-Mail-Adresse](#) oder [YubiKey](#).

Sie können diese Faktoren zu einem einzigen Befehl zusammenfügen, wie im folgenden Beispiel, jedoch wird dies aus Sicherheitsgründen nicht empfohlen:

### Bash

```
bw login [email] [password] --method <method> --code <code>
```

Siehe [Enums](#) für zweistufige Zugangsdaten Werte.

### 💡 Tip

Sie werden zur zusätzlichen Authentifizierung aufgefordert oder erhalten eine Fehlermeldung, wonach Ihre Authentifizierungsanfrage von einem Bot zu kommen scheint? Verwenden Sie `client_secret` für Ihren API-Schlüssel, um die Authentifizierungsanforderung zu beantworten. [Mehr erfahren](#).

## Verwendung eines API-Schlüssels

Es wird empfohlen, sich mit dem [persönlichen API-Schlüssel](#) anzumelden für automatisierte Arbeitsabläufe, um einem externen Anwendung Zugang zu gewähren, oder wenn Ihr Konto eine 2FA-Methode verwendet, die von der CLI nicht unterstützt wird (FIDO2 oder Duo). Um sich mit dem API-Schlüssel anzumelden:

### Bash

```
bw login --apikey
```

Dies wird eine Aufforderung für Ihre persönliche `client_id` und `client_secret` initiieren. Sobald Ihre Sitzung mit diesen Werten authentifiziert ist, können Sie den Befehl `entsperren` verwenden. [Erfahren Sie mehr](#).

#### 💡 Tip

Wenn Ihre Organisation `SSO` erfordert, können Sie sich trotzdem mit `--apikey` über die Kommandozeile anmelden.

## Verwendung von API-Schlüssel-Umgebungsvariablen

In Szenarien, in denen automatisierte Arbeiten mit der Bitwarden CLI durchgeführt werden, können Sie Umgebungsvariablen speichern, um die Notwendigkeit einer manuellen Eingriff bei der Authentifizierung zu vermeiden.

Name der Umgebungsvariable	Erforderlicher Wert
BW_CLIENTID	Kunden-ID
BW_CLIENTGEHEIMNIS	Kundengeheimnis

## Verwendung von SSO

Es wird empfohlen, sich mit `SSO` anzumelden, wenn eine Organisation eine `SSO`-Authentifizierung erfordert. Um sich mit `SSO` anzumelden:

### Bash

```
bw login --sso
```

Dies wird den `SSO`-Authentifizierungsfluss in Ihrem Web-Browser initiieren. Sobald Ihre Sitzung authentifiziert ist, können Sie den Befehl `entsperren` verwenden. [Erfahren Sie mehr](#).

#### 💡 Tip

Wenn Ihre Organisation `SSO` erfordert, können Sie alternativ `--apikey` verwenden, um sich über die Kommandozeile anzumelden.

## Melden Sie sich bei mehreren Konten an

Wie bei der Verwendung von `Kontowechsel` auf anderen Bitwarden-Apps hat die CLI die Möglichkeit, sich gleichzeitig bei mehreren Konten anzumelden, indem sie die `BITWARDENCLI_APPDATA_DIR` Umgebungsvariable verwendet, die auf den Speicherort einer `bw` Konfigurationsdatei zeigt, normalerweise benannt als `data.json`. Sie können beispielsweise Aliase in einem `.bashrc` Profil für zwei separate Einstellungen setzen:

### Bash

```
alias bw-personal="BITWARDENCLI_APPDATA_DIR=~/.config/Bitwarden\ CLI\ Personal /path/to/bw $@"
alias bw-work="BITWARDENCLI_APPDATA_DIR=~/.config/Bitwarden\ CLI\ Work /path/to/bw $@"
```

Mit diesem Beispiel könnten Sie dann die Zugangsdaten für zwei Konten verwenden, indem Sie zuerst `source /path/to/.bashrc` ausführen, gefolgt von `bw-personal` Zugangsdaten und `bw-work` Zugangsdaten.

## Entsperren

Die Verwendung eines `API-Schlüssels` oder `SSO` zum Anmelden erfordert, dass Sie den `Zugangsdaten`-Befehl mit einem expliziten `bw entsperren` befolgen, wenn Sie direkt mit Tresor-Daten arbeiten werden.

Das Entsperren Ihres Tresors generiert einen **Sitzungsschlüssel**, der als Entschlüsselungsschlüssel dient, um mit Daten in Ihrem Tresor zu interagieren. Der **Sitzungsschlüssel muss verwendet werden**, um einen Befehl auszuführen, der die Tresor Daten berührt (zum Beispiel, **auf listen, erhalten, bearbeiten**). Sitzungsschlüssel sind gültig, bis sie mit `bw sperren` oder `bw abmelden` ungültig gemacht werden, sie bleiben jedoch nicht bestehen, wenn Sie ein neues Terminalfenster öffnen. Generieren Sie jederzeit einen neuen Sitzungsschlüssel mit:

### Bash

```
bw unlock
```

Wenn Sie fertig sind, beenden Sie immer Ihre Sitzung mit dem Befehl `bw sperren`.

## Entsperroptionen

Sie können die Optionen `--passwordenv` oder `--passwordfile` zusammen mit `bw entsperren` verwenden, um Ihr Master-Passwort abzurufen, anstatt es manuell einzugeben, zum Beispiel:

1. Das Folgende wird nach einer Umgebungsvariable `BW_PASSWORD` suchen. Wenn `BW_PASSWORD` nicht leer ist und korrekte Werte hat, wird die CLI erfolgreich entsperren und einen Sitzungsschlüssel zurückgeben:

### Bash

```
bw unlock --passwordenv BW_PASSWORD
```

2. Das Folgende sucht nach der Datei `~/Users/Me/Documents/mp.txt` (die Ihr Master-Passwort als erste Zeile haben muss). Wenn die Datei nicht leer ist und einen korrekten Wert hat, wird die CLI erfolgreich entsperren und einen Sitzungsschlüssel zurückgeben:

### Bash

```
bw unlock --passwordfile ~/Users/Me/Documents/mp.txt
```

### ⚠ Warning

Wenn Sie die Option `--passwordfile` verwenden, schützen Sie Ihre Passwortdatei, indem Sie den Zugriff ausschließlich auf den Benutzer beschränken, der `bw unlock` ausführen muss, und nur diesem Benutzer Lesezugriff gewähren.

## Verwendung eines Sitzungsschlüssels

Wenn Sie Ihren Tresor mit `bw Zugangsdaten` mit `E-Mail-Adresse` und `Passwort` oder `bw entsperren` entsperren, gibt die CLI sowohl einen `Export BW_SITZUNG` (Bash) als auch einen `env:BW_SITZUNG` (PowerShell) Befehl zurück, einschließlich Ihres Sitzungsschlüssels. Kopieren und einfügen Sie den relevanten Eintrag, um die erforderliche Umgebungsvariable zu speichern.

Mit der gesetzten `BW_SESSION` Umgebungsvariable, werden `bw` Befehle auf diese Variable verweisen und können sauber ausgeführt werden, zum Beispiel:

### Bash

```
export BW_SESSION="5PBYGU+5yt3RHcCjoeJKx/wByU34vokGRZjXpSH7Ylo8w=="
```

```
bw list items
```

Alternativ, wenn Sie die Umgebungsvariable nicht setzen, können Sie den Sitzungsschlüssel als Option mit jedem `bw` Befehl übergeben:

### Bash

```
bw list items --session "5PBYGU+5yt3RHcCjoeJKx/wByU34vokGRZjXpSH7Ylo8w=="
```

### 💡 Tip

Wenn Sie fertig sind, beenden Sie Ihre Sitzung immer mit den Befehlen `bw lock` oder `bw logout`. Dadurch wird der aktive Sitzungsschlüssel ungültig.

## Kernbefehle erschaffen

Der `erstellen` Befehl erstellt ein neues Objekt (`Eintrag`, `Anhang`, und mehr) in Ihrem Tresor:

### Bash

```
bw create (item|attachment|folder|org-collection) <encodedJson> [options]
```

Der Befehl `erstellen` nimmt kodiertes JSON entgegen. Ein typischer Arbeitsablauf zur Erstellung eines Objekts könnte etwa so aussehen:

1. Verwenden Sie den Befehl `Vorlage holen` (siehe `Kernbefehle holen` für Details), um die entsprechende JSON-Vorlage für den Objekt-Typ auszugeben.

2. Verwenden Sie einen [Befehlszeilen-JSON-Prozessor wie jq](#), um die ausgegebene Vorlage nach Bedarf zu manipulieren.
3. Verwenden Sie den `encode` Befehl (siehe [Details](#)), um das manipulierte JSON zu kodieren.
4. Verwenden Sie den `create` Befehl, um ein Objekt aus dem codierten JSON zu erstellen.

Zum Beispiel:

*Bash*

```
bw get template folder | jq '.name="My First Folder"' | bw encode | bw create folder
```

oder

*Bash*

```
bw get template item | jq ".name=\"My Login Item\" | .login=$(bw get template item.login | jq '.use  
rname=\"jdoe\" | .password=\"myp@ssword123\"')\" | bw encode | bw create item
```

Nach erfolgreicher Erstellung wird das neu erstellte Objekt als JSON zurückgegeben.

## Erstellen Sie andere Eintragstypen

Der Befehl "Erstellen" erstellt standardmäßig einen Eintrag für Zugangsdaten, aber Sie können einen [Befehlszeilen-JSON-Prozessor wie jq](#) verwenden, um ein `.Typ=` Attribut zu ändern und andere [Eintragstypen](#) zu erstellen:

Name	Wert
Zugangsdaten	<code>.Typ=1</code>
Sichere Notiz	<code>Typ=2</code>
Karte	<code>Typ=3</code>
Identität	<code>Typ=4</code>

Zum Beispiel wird der folgende Befehl eine sichere Notiz erstellen:

### Bash

```
bw get template item | jq '.type = 2 | .secureNote.type = 0 | .notes = "Contents of my Secure Note." | .name = "My Secure Note"' | bw encode | bw create item
```

### Note

Beachten Sie im obigen Beispiel, dass sichere Notizen eine untergeordnete Vorlage (`.secureNote.type`) erfordern. Sie können die untergeordneten Vorlagen des Elementtyps mit `bw get template` anzeigen (siehe [hier](#) für Details).

## Anhang erstellen

Der Befehl `Dateianhang erstellen` hängt eine Datei an einen **vorhandenen** Eintrag an.

Im Gegensatz zu anderen **Erstellungs**-Operationen, müssen Sie keinen JSON-Prozessor verwenden oder **kodieren**, um einen Anhang zu erstellen. Verwenden Sie stattdessen die Option `--file`, um die anzuhängende Datei anzugeben, und die Option `--itemid`, um den Eintrag anzugeben, an den sie angehängt werden soll. Zum Beispiel:

### Bash

```
bw create attachment --file ./path/to/file --itemid 16b15b89-65b3-4639-ad2a-95052a6d8f66
```

### Tip

Wenn Sie die genaue `itemid`, die Sie verwenden möchten, nicht kennen, verwenden Sie `bw get item`, um das Element (siehe [Details](#)) einschließlich seiner `id` auszugeben.

## bekommen

Der `get` Befehl ruft ein einzelnes Objekt (**Eintrag**, **Benutzername**, **Passwort** und mehr) aus Ihrem Tresor ab:

### Bash

```
bw get (item|username|password|uri|totp|exposed|attachment|folder|collection|organization|org-collection|template|fingerprint) <id> [options]
```

Der `get` Befehl nimmt einen Eintrag `id` oder eine Zeichenkette als Argument. Wenn Sie eine Zeichenkette verwenden (zum Beispiel alles andere als eine genaue `id`), wird `get` Ihre Tresorobjekte durchsuchen, um eines mit einem übereinstimmenden Wert zu suchen. Zum Beispiel würde der folgende Befehl ein Github-Passwort zurückgeben:

### Bash

```
bw get password Github
```

### Note

Der `get`-Befehl kann **nur ein Ergebnis ausgeben**, Sie sollten also konkrete Suchbegriffe verwenden. Wenn mehrere Ergebnisse gefunden werden, gibt die Kommandozeile einen Fehler aus.

## Anhang erhalten

Der Befehl `Anhang holen` lädt einen Dateianhang herunter:

### Bash

```
bw get attachment <filename> --itemid <id>
```

Der Befehl `Anhang holen` nimmt einen **Dateinamen** und eine **genaue ID**. Standardmäßig lädt `Anhang holen` den Anhang in das aktuelle Arbeitsverzeichnis herunter. Sie können die Option `--output` verwenden, um ein anderes Ausgabeverzeichnis anzugeben, zum Beispiel:

### Bash

```
bw get attachment photo.png --itemid 99ee88d2-6046-4ea7-92c2-acac464b1412 --output /Users/myaccount/Pictures/
```

### Note

Bei der Verwendung von `--output` **muss** der Pfad mit einem Schrägstrich (`/`) enden, um ein Verzeichnis oder einen Dateinamen anzugeben (`/Users/myaccount/Pictures/photo.png`).

## Notizen bekommen

Der Befehl `Notizen abrufen` ruft die Notiz für jeden Tresor-Eintrag ab:

### Bash

```
bw get notes <id>
```

Der Befehl `Notizen holen` nimmt eine genaue Eintrag `id` oder Zeichenkette. Wenn Sie eine Zeichenkette verwenden (zum Beispiel alles andere als eine genaue `id`), wird `Notizen holen` Ihre Tresorobjekte durchsuchen, um eines mit einem übereinstimmenden Wert zu suchen. Zum Beispiel würde der folgende Befehl eine Github Notiz zurückgeben:

### Bash

```
bw get notes Github
```

## Vorlage bekommen

Der Befehl `get template` gibt das erwartete JSON-Format für ein Objekt zurück (`Eintrag`, `Eintrag.Feld`, `Eintrag.Anmeldung` und mehr):

### Bash

```
bw get template (item|item.field|item.login|item.login.uri|item.card|item.identity|item.securenote|
folder|collection|item-collections|org-collection)
```

Während Sie `get template` verwenden können, um das Format auf Ihrem Bildschirm auszugeben, ist der häufigste Anwendungsfall, die Ausgabe in eine `bw create` Operation zu leiten, unter Verwendung eines [Befehlszeilen-JSON-Prozessors](#) wie `jq` und `bw encode`, um die aus der Vorlage abgerufenen Werte zu manipulieren, zum Beispiel:

### Bash

```
bw get template folder | jq '.name="My First Folder"' | bw encode | bw create folder
```

### Note

Jede `item.xxx`-Vorlage sollte als untergeordnetes Objekt zu einer `item`-Vorlage verwendet werden, z. B.:

### Bash

```
bw get template item | jq ".name=\"My Login Item\" | .login=$(bw get template item.login | jq
'.username=\"jdoe\" | .password=\"myp@ssword123\"')\" | bw encode | bw create item
```

## bearbeiten

Der `bearbeiten` Befehl bearbeitet ein Objekt (`Eintrag`, `Eintrag-Sammlungen`, usw.) in Ihrem Tresor:

### Bash

```
bw edit (item|item-collections|folder|org-collection) <id> [encodedJson] [options]
```

Der `bearbeiten` Befehl nimmt eine **genaue id** (das zu bearbeitende Objekt) und codiertes JSON (zu machende Änderungen) entgegen. Ein typischer Arbeitsablauf könnte etwa so aussehen:

1. Verwenden Sie den `get` Befehl (siehe [Details](#)), um das Objekt auszugeben, das zu bearbeiten ist.
2. Verwenden Sie einen [Befehlszeilen-JSON-Prozessor](#) wie `jq`, um das ausgegebene Objekt nach Bedarf zu manipulieren.
3. Verwenden Sie den `encode` Befehl (siehe [Details](#)), um das manipulierte JSON zu kodieren.

4. Verwenden Sie den Befehl **bearbeiten** (einschließlich des Objekts **id**), um das Objekt zu bearbeiten.

Zum Beispiel, um das Passwort eines Zugangsdaten-Eintrags zu bearbeiten:

#### Bash

```
bw get item 7ac9cae8-5067-4faf-b6ab-acfd00e2c328 | jq '.login.password="newp@ssw0rd"' | bw encode |  
bw edit item 7ac9cae8-5067-4faf-b6ab-acfd00e2c328
```

Oder, um die Sammlung(en), in der ein Eintrag enthalten ist, zu bearbeiten:

#### Bash

```
echo '["5c926f4f-de9c-449b-8d5f-aec1011c48f6"]' | bw encode | bw edit item-collections 28399a57-73a  
0-45a3-80f8-aec1011c48f6 --organizationid 4016326f-98b6-42ff-b9fc-ac63014988f5
```

Oder, eine Sammlung zu bearbeiten:

#### Bash

```
bw get org-collection ee9f9dc2-ec29-4b7f-9afb-aac8010631a1 --organizationid 4016326f-98b6-42ff-b9fc  
-ac63014988f5 | jq '.name="My Collection"' | bw encode | bw edit org-collection ee9f9dc2-ec29-4b7f-  
9afb-aac8010631a1 --organizationid 4016326f-98b6-42ff-b9fc-ac63014988f5
```

Der Befehl **bearbeiten** führt eine **ersetzen** Operation auf dem Objekt aus. Sobald abgeschlossen, wird das aktualisierte Objekt als JSON zurückgegeben.

## Liste

Der Befehl **Liste** ruft ein Array von Objekten (**Einträge**, **Ordner**, **Sammlungen** und mehr) aus Ihrem Tresor ab:

#### Bash

```
bw list (items|folders|collections|organizations|org-collections|org-members) [options]
```

Optionen für den **list** Befehl sind **Filter**, die bestimmen, was zurückgegeben wird, einschließlich **--url**, **--folderid**, **--collectionid**, **--organizationid** und **--trash**. Jeder Filter akzeptiert **null** oder **nichtnull**. Die Kombination mehrerer Filter in einem Befehl führt eine ODER-Operation aus, zum Beispiel:

#### Bash

```
bw list items --folderid null --collectionid null
```

Dieser Befehl gibt Einträge zurück, die sich nicht in einem Ordner oder einer Sammlung befinden.

Zusätzlich können Sie nach spezifischen Objekten **suchen**, indem Sie **--suchen** verwenden. Die Kombination von Filter und Suche in einem Befehl führt eine UND-Operation aus, zum Beispiel:

*Bash*

```
bw list items --search github --folderid 9742101e-68b8-4a07-b5b1-9578b5f88e6f
```

Dieser Befehl wird nach Einträgen mit der Zeichenkette **github** im angegebenen Ordner suchen.

## löschen

Der **löschen** Befehl löscht ein Objekt aus Ihrem Tresor. **löschen** nimmt **nur eine genaue id** als Argument.

*Bash*

```
bw delete (item|attachment|folder|org-collection) <id> [options]
```

Standardmäßig wird **löschen** einen Eintrag an den **Papierkorb** senden, wo er 30 Tage verbleiben wird. Sie können einen Eintrag dauerhaft löschen, indem Sie die Option **-p**, **--permanent** verwenden.

*Bash*

```
bw delete item 7063feab-4b10-472e-b64c-785e2b870b92 --permanent
```

Um eine **org-Sammlung** zu löschen, müssen Sie auch **--organizationid** angeben. Siehe **Organisations-IDs**.

### Warning

Während Elemente, die mit **delete** gelöscht werden, bis zu 30 Tage lang mit dem Befehl **restore** wiederhergestellt werden können (siehe **Details**), werden Elemente, die mit **delete --permanent** gelöscht werden, **vollständig entfernt und können nicht wiederhergestellt werden**.

## wiederherstellen

Der **wiederherstellen** Befehl stellt ein gelöscht Objekt aus Ihrem Papierkorb wieder her. **wiederherstellen** benötigt **nur eine genaue ID** als Argument.

*Bash*

```
bw restore (item) <id> [options]
```

Zum Beispiel:

*Bash*

```
bw restore item 7063feab-4b10-472e-b64c-785e2b870b92
```

## senden

Der **senden** Befehl erstellt ein Bitwarden **Send** Objekt für ephemeres Teilen. Dieser Abschnitt wird einfache **senden** Operationen im Detail darstellen, jedoch ist **senden** ein äußerst flexibles Werkzeug und wir empfehlen, den speziellen Artikel über **Senden von CLI** zu konsultieren.

Um einen einfachen Text zu senden:

*Bash*

```
bw send -n "My First Send" -d 7 --hidden "The contents of my first text Send."
```

Um eine einfache Datei zu erstellen, senden Sie:

*Bash*

```
bw send -n "A Sensitive File" -d 14 -f /Users/my_account/Documents/sensitive_file.pdf
```

## empfangen

Der **empfangen** Befehl greift auf ein Bitwarden **senden** Objekt zu. Ein **Send**-Objekt empfangen:

*Bash*

```
bw receive --password passwordforaccess https://vault.bitwarden.com/#/send/yawoill8rk6VM6zCATXv2A/9WN8wD-hzsDJjfnXLeNc2Q
```

## Befehle der Organisation

### Organisations-IDs

Der Zugriff auf eine Organisation über die CLI erfordert die Kenntnis einer ID für Ihre Organisation, sowie IDs für einzelne **Mitglieder** und **Sammlungen**.

Rufen Sie diese Informationen direkt über die CLI mit **bw list** ab, zum Beispiel:

*Bash*

```
bw list organizations
bw list org-members --organizationid 4016326f-98b6-42ff-b9fc-ac63014988f5
bw list org-collections --organizationid 4016326f-98b6-42ff-b9fc-ac63014988f5
```

 **Tip**

Sie können sowohl `bw list` als auch `Sammlungen` und `org-Sammlungen` auflisten. Der Befehl `bw list collections` listet alle Sammlungen auf, unabhängig davon, zu welcher Organisation sie gehören. `bw list org-collections` listet nur Sammlungen auf, die zur mit `--organizationid` angegebenen Organisation gehören.

## bewegen

 **Note**

**August 2021:** Der Befehl `share` wurde in `move` geändert. [Erfahren Sie mehr](#).

Der `move`-Befehl überträgt einen Tresor-Eintrag [an eine Organisation](#):

**Bash**

```
bw move <itemid> <organizationid> [encodedJson]
```

Der `Bewegen`-Befehl erfordert, dass Sie eine Sammlungs-ID `kodieren`, und nimmt eine **genaue ID** (das zu teilende Objekt) und eine **genaue Organisations-ID** (die Organisation, mit der das Objekt geteilt werden soll) an. Zum Beispiel:

**Bash**

```
echo '["bq209461-4129-4b8d-b760-acd401474va2"]' | bw encode | bw move ed42f44c-f81f-48de-a123-ad01013132ca dfgbhc921-04eb-43a7-84b1-ac74013bqb2e
```

Sobald abgeschlossen, wird der aktualisierte Eintrag zurückgegeben.

## bestätigen

Der `bestätigen` Befehl bestätigt [eingeladene Mitglieder](#) in Ihrer Organisation, die ihre Einladung angenommen haben:

**Bash**

```
bw confirm org-member <id> --organizationid <orgid>
```

Der `bestätigen` Befehl benötigt eine **genaue Mitglied ID** und eine **genaue OrganisationsID**, zum Beispiel:

**Bash**

```
bw confirm org-member 7063feab-4b10-472e-b64c-785e2b870b92 --organizationid 310d5ffd-e9a2-4451-af87-ea054dce0f78
```

## Andere Befehle

### Konfiguration

Der `config` Befehl gibt die Einstellungen an, die Bitwarden CLI verwenden soll:

*Bash*

```
bw config server <setting> [value]
```

Ein Hauptverwendungszweck von `bw config` besteht darin, Ihre CLI mit einem selbst gehosteten Bitwarden-Server zu verbinden:

*Bash*

```
bw config server https://your.bw.domain.com
```

#### Tip

Verbinden Sie sich mit dem Bitwarden EU-Server, indem Sie den folgenden Befehl ausführen:

*Bash*

```
bw config server https://vault.bitwarden.eu
```

Geben Sie `bw config server` ohne einen Wert ein, um den Server zu lesen, mit dem Sie verbunden sind.

Benutzer mit einzigartigen Setups können sich dafür entscheiden, die URL jedes Dienstes unabhängig anzugeben. Beachten Sie, dass jede nachfolgende Verwendung des Config-Befehls alle vorherigen Spezifikationen überschreibt, daher muss dies jedes Mal als einzelner Befehl ausgeführt werden, wenn Sie eine Änderung vornehmen:

*Bash*

```
bw config server --web-vault <url> \  
  --api <url> \  
  --identity <url> \  
  --icons <url> \  
  --notifications <url> \  
  --events <url> \  
  --key-connector <url>
```

### Note

Der Befehl `bw config server --key-connector` ist erforderlich, wenn Ihre Organisation [Key Connector](#) verwendet und Sie sich mit der Option `--apikey` anmelden, nachdem Sie Ihr Master-Passwort entfernt haben.

Wenden Sie sich an den Besitzer der Organisation, um die erforderliche URL zu erhalten.

## Synchronisation

Der **Synchronisation**-Befehl lädt Ihren verschlüsselten Tresor vom Bitwarden-Server herunter. Dieser Befehl ist am nützlichsten, wenn Sie etwas in Ihrem Bitwarden-Tresor auf einer anderen Client-Anwendung (zum Beispiel Web-Tresor, Browser-Erweiterung, mobile App) geändert haben, seit Sie sich [anmelden](#) auf dem CLI.

### Bash

```
bw sync
```

Sie können die Option `--last` übergeben, um nur den Zeitstempel (ISO 8601) der letzten durchgeführten Synchronisation zurückzugeben.

### Tip

Es ist wichtig zu wissen, dass `sync` **nur einen Pull-Vorgang** am Server durchführt. Die Daten werden automatisch auf den Server übertragen, sobald Sie eine Änderung an Ihrem Tresor vornehmen (z. B. [create](#), [edit](#), [delete](#)).

## kodieren

Der **encode** Befehl kodiert stdin in Base 64. Dieser Befehl wird typischerweise in Kombination mit einem [Befehlszeilen-JJSON-Prozessor](#) wie `jq` verwendet, wenn **Erstellungs-** und **Bearbeitungs-**operationen durchgeführt werden, zum Beispiel:

### Bash

```
bw get template folder | jq '.name="My First Folder"' | bw encode | bw create folder

bw get item 7ac9cae8-5067-4faf-b6ab-acfd00e2c328 | jq '.login.password="newp@ssw0rd"' | bw encode |
bw edit item 7ac9cae8-5067-4faf-b6ab-acfd00e2c328
```

## Import

Der **Import**-Befehl importiert Daten aus einem Bitwarden-Export oder einer anderen [unterstützten Passwort-Manager-Anwendung](#). Der Befehl muss auf eine Datei zeigen und die folgenden Argumente enthalten:

### Bash

```
bw import <format> <path>
```

Zum Beispiel:

### Bash

```
bw import lastpasscsv /Users/myaccount/Documents/mydata.csv
```

### 💡 Tip

Bitwarden unterstützt viele Formate für den Import – zu viele, um sie hier aufzulisten! Verwenden Sie `bw import --formats`, um die Liste in Ihrer Kommandozeile anzuzeigen, oder [lesen Sie hier mehr](#).

Wenn Sie eine [verschlüsselte .json-Datei importieren](#), die Sie mit einem Passwort erstellt haben, werden Sie aufgefordert, das Passwort einzugeben, bevor der Import abgeschlossen ist.

## Export

Der `Export`-Befehl exportiert TresorDaten als eine `.json` oder `.csv`, oder verschlüsselte `.json` Datei:

### Bash

```
bw export [--output <filePath>] [--format <format>] [--password <password>] [--organizationid <orgid>]
```

Standardmäßig wird der `Export`-Befehl eine `.csv` Datei generieren (entspricht der Angabe von `--format csv`) im aktuellen Arbeitsverzeichnis, jedoch können Sie spezifizieren:

- `--format json` zum Export einer `.json` Datei
- `--format encrypted_json` zum Export einer verschlüsselten `.json` Datei
  - `--Passwort` um ein Passwort anzugeben, das zur Verschlüsselung von verschlüsselten `.json` Exporten anstelle Ihres Konto-Verschlüsselungsschlüssels verwendet wird.
- `--output` zum Export an einen bestimmten Ort
- `--raw`, um den Export an stdout zurückzugeben, anstatt an eine Datei

## Export aus einem Organisationstresor

Mit dem `Export` Befehl und der Option `--organizationid` können Sie einen Organisationstresor exportieren:

### Bash

```
bw export --organizationid 7063feab-4b10-472e-b64c-785e2b870b92 --format json --output /Users/myaccount/Downloads/
```

## generieren

Der `generieren` Befehl generiert ein starkes Passwort oder eine [Passphrase](#):

**Bash**

```
bw generate [--lowercase --uppercase --number --special --length <length> --passphrase --separator  
<separator> --words <words>]
```

Standardmäßig wird der Befehl **generieren** ein 14-Zeichen-Passwort mit Großbuchstaben, Kleinbuchstaben und Nummern generieren. Dies entspricht dem Bestehen:

**Bash**

```
bw generate -uln --length 14
```

Sie können komplexere Passwörter generieren, indem Sie die dem Befehl zur Verfügung stehenden Optionen nutzen, einschließlich:

- **--Großbuchstaben, -G** (Großbuchstaben einbeziehen)
- **--kleinschreibung, -k** (Kleinschreibung einbeziehen)
- **--Nummer, -n** (Nummern einbeziehen)
- **--speziell, -s** (beinhaltet Sonderzeichen)
- **--Länge** (Länge des Passworts, mindestens 5)

**Generiere ein Passwort**

Mit dem **generieren** Befehl und der **--passphrase** Option können Sie eine Passphrase anstelle eines Passworts generieren:

**Bash**

```
bw generate --passphrase --words <words> --separator <separator>
```

Standardmäßig wird **bw generieren --passphrase** ein dreiwörtiges Passwort generieren, das durch einen Bindestrich (-) getrennt ist. Dies entspricht dem Bestehen:

**Bash**

```
bw generate --passphrase --words 3 --separator -
```

Sie können ein komplexes Passwort generieren, indem Sie die dem Befehl zur Verfügung stehenden Optionen nutzen, einschließlich:

- **--Wörter** (Anzahl der Wörter)
- **--Trennzeichen** (Trennzeichen)

- `--kapitalisieren, -c` (einschließen, um das Passwort in den Titelkasten zu setzen)
- `--includeNumber` (Nummern in der Passphrase einbeziehen)

## Aktualisierung

Der Befehl **Aktualisierung** überprüft, ob Ihre Bitwarden CLI die neueste Version ausführt. **Aktualisierung aktualisiert die CLI nicht automatisch für Sie.**

*Bash*

```
bw update
```

Wenn eine neue Version erkannt wird, müssen Sie die neue Version der CLI mit der gedruckten URL für die ausführbare Datei herunterladen oder die für den Paketmanager, den Sie zum [Herunterladen der CLI](#) verwendet haben, verfügbaren Tools verwenden (zum Beispiel, `npm install -g @bitwarden/cli`).

## Status

Der Befehl **Status** gibt Statusinformationen über die Bitwarden CLI zurück, einschließlich der [konfigurierten](#) Server-URL, des Zeitstempels für die letzte Synchronisation ([ISO 8601](#)), der E-Mail-Adresse und ID des Benutzers sowie des Status des Tresors.

*Bash*

```
bw status
```

Der Status wird Informationen als JSON-Objekt zurückgeben, zum Beispiel:

*Bash*

```
{
  "serverUrl": "https://bitwarden.example.com",
  "lastSync": "2020-06-16T06:33:51.419Z",
  "userEmail": "user@example.com",
  "userId": "00000000-0000-0000-0000-000000000000",
  "status": "unlocked"
}
```

**Status** kann einer der folgenden sein:

- „**unlocked**“ bedeutet, dass Sie angemeldet sind und Ihr Tresor entsperrt ist (eine **BW\_SESSION**- Schlüsselumgebungsvariable wird mit einem [aktiven Sitzungsschlüssel](#) gespeichert).
- „**locked**“ bedeutet, dass Sie angemeldet sind, aber Ihr Tresor gesperrt ist ( **keine** **BW\_SESSION**- Schlüsselumgebungsvariable wird mit einem [aktiven Sitzungsschlüssel](#) gespeichert).

- „unauthenticated“ bedeutet, dass Sie nicht angemeldet sind

## 💡 Tip

Wenn gilt "status": "unauthenticated", geben `lastSync`, `userEmail` und `userID` immer `null` aus.

## dienen

Der `serve`-Befehl startet einen lokalen Express-Webserver, der verwendet werden kann, um alle Aktionen, die über die CLI zugänglich sind, in Form von RESTful-API-Aufrufen über eine HTTP-Schnittstelle auszuführen.

### Bash

```
bw serve --port <port> --hostname <hostname>
```

Standardmäßig startet `serve` den Webserver auf Port 8087, jedoch können Sie mit der Option `--port` einen alternativen Port angeben.

Standardmäßig bindet `serve` Ihren API-Webserver an `localhost`, jedoch können Sie mit der Option `--hostname` einen alternativen Hostnamen angeben. API-Anfragen können nur vom gebundenen Hostnamen gemacht werden.

Standardmäßig blockiert `serve` jede Anfrage mit einem `Origin`-Header. Sie können diesen Schutz mit der Option `--disable-origin-protection` umgehen, jedoch wird **dies nicht empfohlen**.

## ⚠ Warning

Sie können `--hostname all` angeben, um keine Hostnamenbindung zu haben, dies wird jedoch jeder Maschine im Netzwerk erlauben, API-Anfragen zu stellen.

Sehen Sie sich die [API-Spezifikation an](#), um Hilfe beim Tätigen von Anrufen mit `serve` zu erhalten.

## Anhänge

### Globale Optionen

Die folgenden Optionen stehen weltweit zur Verfügung:

Option	Beschreibung
<code>--hübsch</code>	Formatausgabe. JSON ist mit zwei Leerzeichen getabt.
<code>--roh</code>	Geben Sie rohe Ausgabe zurück, anstatt einer beschreibenden Nachricht.
<code>--Antwort</code>	Geben Sie eine JSON-formatierte Version der Antwortausgabe zurück.

Option	Beschreibung
<code>--leise</code>	Geben Sie nichts an stdout zurück. Sie könnten diese Option beispielsweise verwenden, wenn Sie einen Anmeldeinformati <sup>o</sup> nswert in eine Datei oder Anwendung leiten.
<code>--keineinteraktion</code>	Fordern Sie keine interaktive Benutzereingabe auf.
<code>--Sitzung</code>	Übergeben Sie den Sitzungsschlüssel anstelle des Lesens aus einer Umgebungsvariable.
<code>-v, --version</code>	Geben Sie die Bitwarden CLI-Version Nummer aus.
<code>-h, --hilfe</code>	Zeige Hilfetext für den Befehl an.

## ZSH Shell-Vervollständigung

Die Bitwarden CLI beinhaltet Unterstützung für ZSH Shell-Vervollständigung. Um die Shell-Vervollständigung einzurichten, verwenden Sie eine der folgenden Methoden:

1. **Vanilla ZSH:** Fügen Sie die folgende Zeile zu Ihrer `.zshrc` Datei hinzu:

*Bash*

```
eval "$(bw completion --shell zsh); compdef _bw bw;"
```

2. **Vanilla (Anbieter-Vervollständigungen):** Führen Sie den folgenden Befehl aus:

*Bash*

```
bw completion --shell zsh | sudo tee /usr/share/zsh/vendor-completions/_bw
```

3. **zinit:** Führen Sie die folgenden Befehle aus:

Bash

```
bw completion --shell zsh > ~/.local/share/zsh/completions/_bw  
zinit creinstall ~/.local/share/zsh/completions
```

## Verwendung von selbstsignierten Zertifikaten

Wenn Ihr selbst gehosteter Bitwarden-Server ein selbstsigniertes TLS-Zertifikat kompromittiert, geben Sie die Node.js-Umgebungsvariable `NODE_EXTRA_CA_CERTS` an:

Bash

Bash

```
export NODE_EXTRA_CA_CERTS="absolute/path/to/your/certificates.pem"
```

PowerShell

Bash

```
$env:NODE_EXTRA_CA_CERTS="absolute/path/to/your/certificates.pem"
```

## Aufzählungen

Die folgenden Tabellen listen die Werte auf, die in dokumentierten Szenarien benötigt werden:

### Zwei-Stufen-Zugangsdaten-Methoden

Wird verwendet, um anzugeben, welche [Zwei-Schritt-Zugangsdaten-Methode](#) beim [Anmelden](#) verwendet werden soll:

Name	Wert
Authentifikator	0
E-Mail	1
YubiKey	3

**Note**

FIDO2 und Duo werden nicht von der Kommandozeile unterstützt.

**Eintragstypen**

Wird mit dem **create**-Befehl verwendet, um einen **Tresor Eintrag Typ** zu spezifizieren:

Name	Wert
Zugangsdaten	1
Sichere Notiz	2
Karte	3
Identität	4

**Zugangsdaten URI Übereinstimmungs-Typen**

Wird mit dem **Erstellen** und **Bearbeiten** Befehl verwendet, um das **URI Übereinstimmungsverhalten** für einen Zugangsdaten-Eintrag zu spezifizieren:

Name	Wert
Domain	0
Server	1
Beginnt mit	2
Exakt	3

Name	Wert
Regulärer Ausdruck	4
Niemals	5

### Feldtypen

Wird mit den Befehlen **erstellen** und **bearbeiten** verwendet, um **benutzerdefinierte Felder** zu konfigurieren:

Name	Wert
Text	0
Versteckt	1
Ja/Nein	2

### Organisation Benutzertypen

Zeigt einen **Typ des Benutzers** an:

Name	Wert
Besitzer	0
Administrator	1
Benutzer	2
Verwalter	3

Name	Wert
Benutzerdefiniert	4

### Organisation Benutzerstatus

Zeigt den [Status](#) eines Benutzers innerhalb der Organisation an:

Name	Wert
Eingeladen	0
Akzeptiert	1
Bestätigt	2
Widerrufen	-1