

“CyBruArk”

# CYBERARK® BRUNO REST API COLLECTION

# USER GUIDE



# CONTENTS

The Basics

Inside the REST Request

Authentication

Credential Providers

Next Steps





CyberArk® Bruno REST API Collection

# THE BASICS

# PREREQUISITES

Download the relevant Collection and Environment from GitHub

<https://github.com/IAM-Jah/CyberArk-REST-API-Bruno>

The screenshot shows the GitHub repository page for 'CyberArk-REST-API-Bruno'. The repository is owned by 'IAM-Jah' and has 3 stars. The main content area displays the 'Bruno Collections for CyberArk Identity Security REST API'. It includes a description of the Bruno Collection and Environment files, a list of optimizations, and a preview of the Bruno IDE interface showing the 'Privilege Cloud and Shared Services REST API' collection.

**Bruno Collections for CyberArk Identity Security REST API**

Bruno Collection and Environment files for **CyberArk Identity Security** REST API testing and automation. 945 REST API requests are included in the Collections, which cover Self-Hosted PAM, Privilege Cloud - Standard, Privilege Cloud on Identity Security Platform for Shared Services (ISPSS/Shared Services), and many shared services from the SaaS platform.

Each API request includes small optimizations to help get you up and running quickly, for example:

- Authentication requests test for and store the resulting tokens as Environment variables.
- Authentication token is automatically populated to request headers where necessary.
- Documentation links and important notes are stored inside each request's 'Docs' section.
- Mandatory URI parameters are selected by default and vice-versa.
- Central Credential Provider 'GetPassword' request included with every Collection.

Collection Edit View Window Help

bruno

Collections

Privilege Cloud and Shared Services REST API

POST OAuth2 ... POST OIDC Au...

POST https://[[IdentityTenantID]].cyberark.cloud/oauth2/platformtoken

Params Body Headers<sup>1</sup> Auth Vars Script Assert Tests Docs

Form URL Encoded

Key Value

👁️ for updates

Give me a ★!

Download desired **Environment**

The screenshot shows the file list for the 'CyberArk-REST-API-Bruno' repository. The files are listed in a table with columns for 'Name' and 'Last commit message'. Two files are highlighted with red boxes: 'Privilege Cloud Shared Services Environment.json' and 'Privilege Cloud and Shared Services REST API.json'.

**CyberArk-REST-API-Bruno / Privilege Cloud and Shared Services REST API**

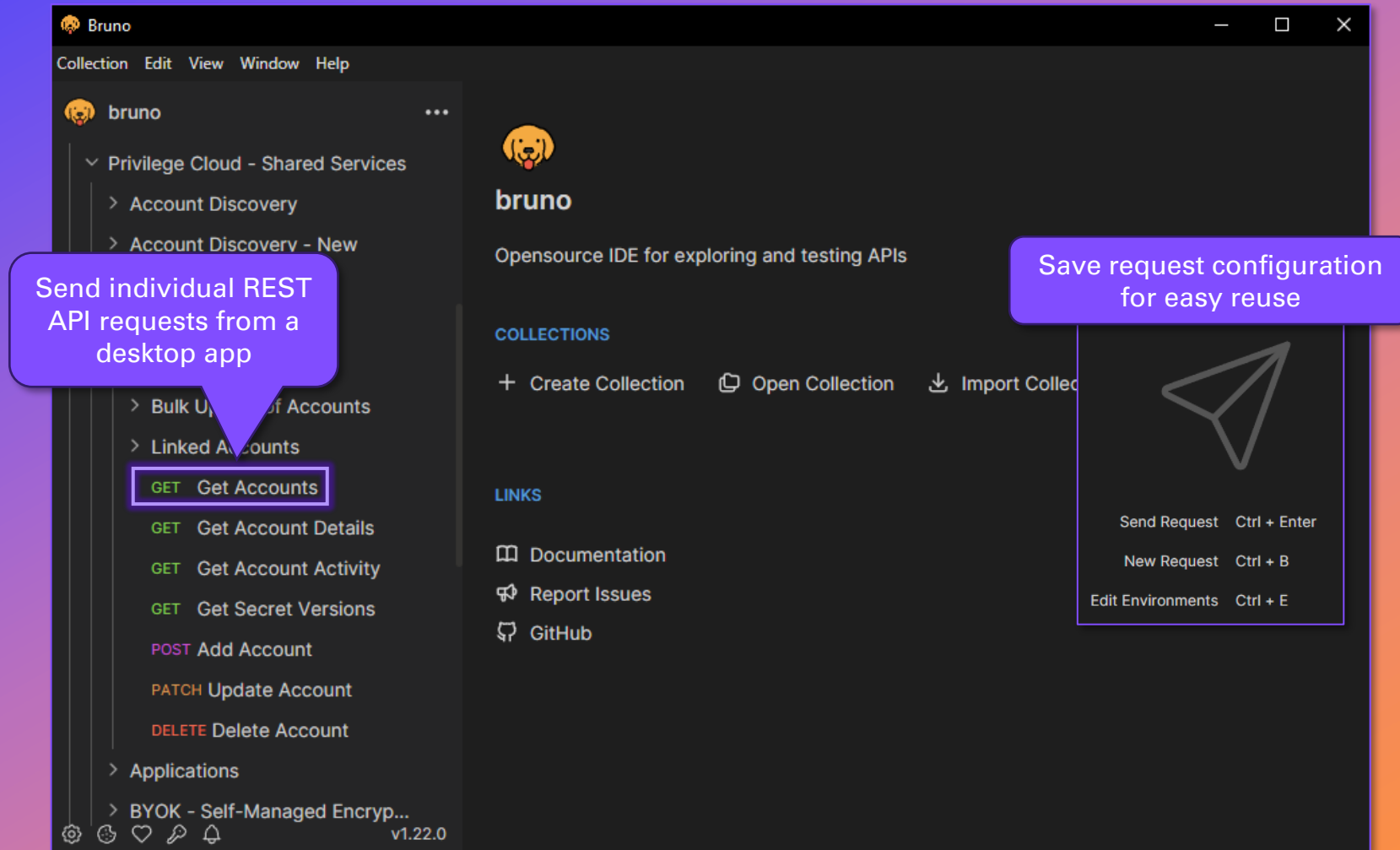
IAM-Jah Collection consolidated and improved. 📄

Name	Last commit message
..	
Privilege Cloud Shared Services Environment.json	Collection consolidated
Privilege Cloud and Shared Services REST API.json	Collection consolidated
README.md	Collection consolidated

Download desired **Collection**

# BRUNO

Bruno is an open-source IDE for developing, testing and exploring APIs  
**You can use it to send single API calls**



# COLLECTIONS

'Collections' contain template API requests

The image shows the Bruno API client interface. On the left, a sidebar lists various collections under the 'bruno' workspace. The 'Privilege Cloud and Shared Services REST API' collection is expanded, showing sub-collections like 'Central Credential Provider', 'CM - Connector Management', 'Conjur Cloud', 'DPA - Dynamic Privileged Access', 'EPM - Endpoint Privilege Manager SaaS', 'Identity - Shared Services', 'Privilege Cloud - Shared Services', 'Privilege Cloud - Standard', 'RA - Remote Access', 'SCA - Secure Cloud Access', 'Secrets Hub', and 'CyberArk Self-Hosted REST API'. A callout bubble points to the 'Privilege Cloud and Shared Services REST API' collection, stating: 'Each CyberArk solution has an associated Collection – 945 API calls altogether!'. The main panel shows the configuration for a specific API call: 'POST https://{{identityTenantID}}.id.cyberark.cloud/oauth2/platformtoken'. The 'Body' tab is selected, showing a table of parameters:

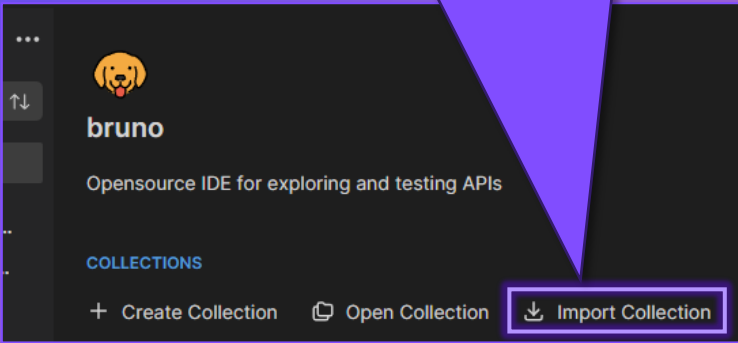
Key	Value	
grant_type	client_credentials	<input checked="" type="checkbox"/>
client_id	{{identityAPIUsername}}	<input checked="" type="checkbox"/>
client_secret	{{identityAPIPassword}}	<input checked="" type="checkbox"/>

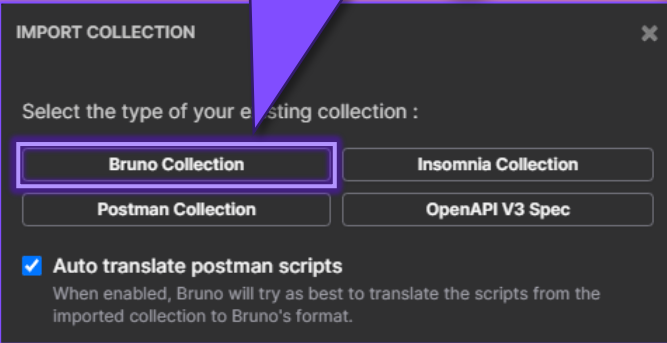
A callout bubble points to the 'POST OIDC Authentication' call in the sidebar, stating: 'Each individual API call has its own configuration'.

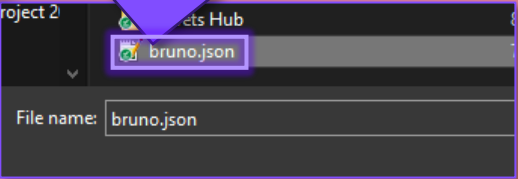
# IMPORT A COLLECTION

Collections can be imported in multiple formats or managed via Git client (via Open Collection)

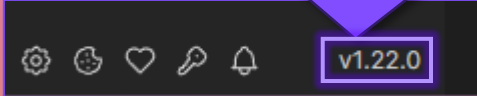
- 1 On the home page, select 'Import Collection'


- 2 Select 'Bruno Collection'


- 3 Select the JSON file for import



V1.19+ required for JSON import





# ENVIRONMENTS

‘Environments’ contain frequently used data stored as variables

Enable or disable variables as needed for your environment

Fill in default values for static variables

ENVIRONMENTS

Privilege Cloud Shared Services Env

+ Create

Enabled	Name	Value	Secret	
<input checked="" type="checkbox"/>	accountID	21	<input type="checkbox"/>	
<input checked="" type="checkbox"/>	accountName	accountObjectNameFromDetailsPage	<input type="checkbox"/>	
<input checked="" type="checkbox"/>	activeSessionId	sessionId	<input type="checkbox"/>	
<input checked="" type="checkbox"/>	appAuthMethodID	appAuthMethodID	<input type="checkbox"/>	

<input checked="" type="checkbox"/>	identityAPIPassword	*****	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	identityPassword	*****	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	identityToken	*****...	<input checked="" type="checkbox"/>

Select the ‘Secret’ box to mask the value and prevent it from being exported

# IMPORT AN ENVIRONMENT

Review and update newly imported Environments

**1** Press 'ctrl + e' with any API selected

**2** Select 'Import'

**3** Click inside the box

**4** Select the JSON file for import

Privilege Cloud Shared Services Environment.json



CyberArk® Bruno REST API Collection

# INSIDE THE REST REQUEST

# CRAFT A REQUEST

The image shows the Bruno API client interface with five numbered callouts illustrating the steps to craft a request:

- 1 Select an API call from the Collection**: Points to the 'Login' sub-collection under 'Identity - Shared Services' in the left sidebar. The 'POST Start Authentication' item is highlighted.
- 2 Load an appropriate Environment**: Points to the 'Privilege Cloud Shared Services Env' dropdown menu in the top right corner.
- 3 Review the HTTPS URI and Method**: Points to the request method 'POST' and the URL 'https://{{{IdentityTenantID}}}.id.cyberark.cloud/Security/StartAuthentication' in the request editor.
- 4 Review and update all relevant data**: Points to the 'Params' tab in the request editor, which contains a table for query parameters.
- 5 Send the request**: Points to the 'Send Request' button in the bottom right corner, which has the keyboard shortcut 'Ctrl + Enter'.

The interface also shows a list of API calls in the left sidebar, including 'POST Advance Authentication - User Passw...', 'POST Advance Authentication - Security Qu...', 'POST Advance Authentication - Security Qu...', 'POST Get Challenge State', 'POST Start Step-up Challenge', 'POST Answer Out-of-band Challenge', and 'POST Submit OATH OTP Code'.

# VARIABLES

Frequently used data may be stored as **variables** in a Bruno **Environment**

Variables with a value assigned appear in **{{greenBrackets}}**

The screenshot shows a REST client interface for a "Privilege Cloud and Services REST API". A POST request is defined with the URL `https://{{identityTenantID}}.id.cyberark.cloud/Security/StartAuthentication`. The "Body" tab is selected, showing a JSON payload: 

```
{
  "TenantID": "{{identityTenantID}}",
  "User": "{{identityUsername}}",
  "Version": "1.0"
}
```

. At the bottom, a small table lists variables: 

Name	Value
AppID	{{appID}}
Safe	{{safeName}}

. The variable `{{safeName}}` is highlighted with a green box.

No value assigned = **{{redBrackets}}**

**{{values}}** are stored per Environment

Privilege Cloud Shared Services Env		
<input checked="" type="checkbox"/>	identityAPIUsername	APIUser@cyberark.cloud.1234
<input checked="" type="checkbox"/>	identityMechanismID	4VieaKZqTcFJdzjSju65YleIXmMRroRCKYC
<input checked="" type="checkbox"/>	identityMechanismID	-Q23_cimi-x_UgzeHUDIFlz8R4s0cgFHnYr
<input checked="" type="checkbox"/>	identitySessionID	OFoq6C02mAO7IE7KiJM3h6vNjxOewyxLI
<input checked="" type="checkbox"/>	identityTenantID	abi4731
<input checked="" type="checkbox"/>	identityUsername	elijah.hopkins@cyberark.cloud.20878

```
"TenantID": "abi4738",
"User": "elijah.hopkins@cyberark.cloud.20885",
"Version": "1.0"
```

Typing values in directly is also supported, but not convenient!

# PARAMETERS

Variables that are part of the REST API URI are represented as 'Params'

Enabled parameters are added to the URI with proper syntax

GET GetPassw...

GET https://{{CCPAddress}}/AIMWebService/api/Accounts?AppID={{appID}}&Safe={{safeName}}

Params<sup>2</sup> Body Headers<sup>1</sup> Auth Vars Script Assert Tests Docs

Query

Name	Value	
AppID	{{appID}}	<input checked="" type="checkbox"/>
Safe	{{safeName}}	<input checked="" type="checkbox"/>
Object	{{accountName}}	<input type="checkbox"/>
UserName	string	<input type="checkbox"/>
Address	my.host	<input type="checkbox"/>
Database	string	<input type="checkbox"/>

Disabled parameters are ignored

# BODY

The Body contains data sent in the request when required by the API

For the 'Add Account' request, the body contains all the data you would typically provide to the GUI

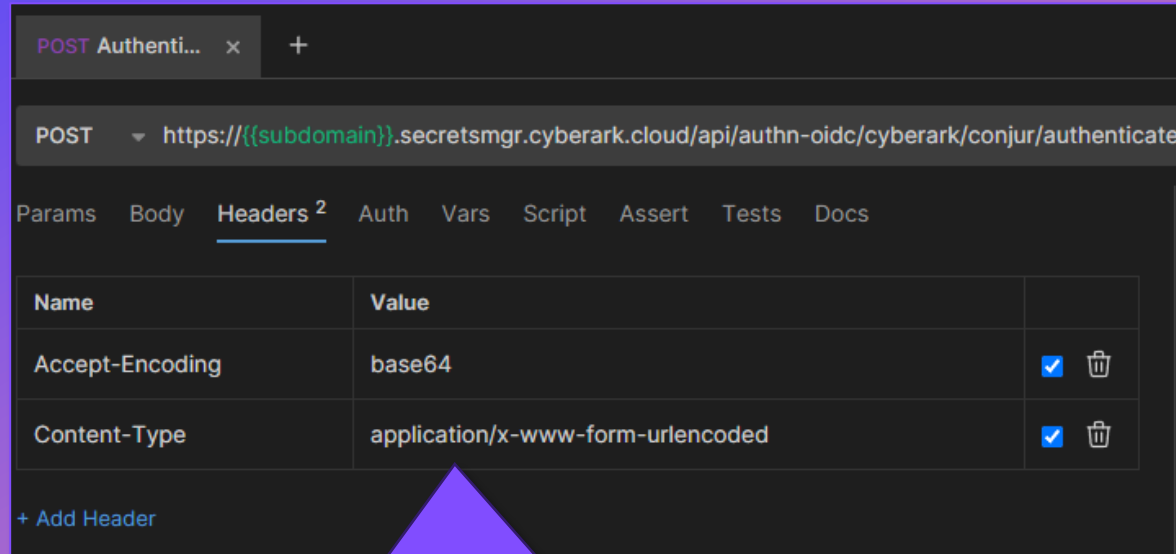
**Update all data** in the body for your use case. Optional parameters may be *//commented out*

JSON is most common format for this Collection but URL-encoded form is also used (e.g., for SAML)

```
POST Add Acc... x +
POST https://{subdomain}.privilegecloud.cyberark.cloud/PasswordVault/API/Accounts/
Params Body Headers 1 Auth Vars Script Assert Tests Docs JSON Prettify
1 {
2   "name": "accountName",
3   "address": "accountAddress",
4   "userName": "accountUsername",
5   "platformId": "{platformID}",
6   "safeName": "{safeName}",
7   "secretType": "password/private SSH key",
8   "secret": "secretValue-WillNotBeReturnedInResult",
9   "platformAccountProperties": {
10    "LogonDomain": "string"
11    // "Port": 1,
12    // "CustomFileCategory": "value"
13  },
14  "secretManagement": {
15    "automaticManagementEnabled": true,
16    "manualManagementReason": "string",
17    // "status": "inProcess, succeeded, failed, partiallySucceeded",
18    // "lastModifiedDateTime": "stringWithDate-Time",
19    // "lastReconciledDateTime": "stringWithDate-Time",
20    // "lastVerifiedDateTime": "stringWithDate-Time"
21  },
22  "remoteMachinesAccess": {
23    "remoteMachines": "server1.domain.com;server2.domain.com",
24    "accessRestrictedToRemoteMachines": true
25  }
26 }
```

# HEADERS

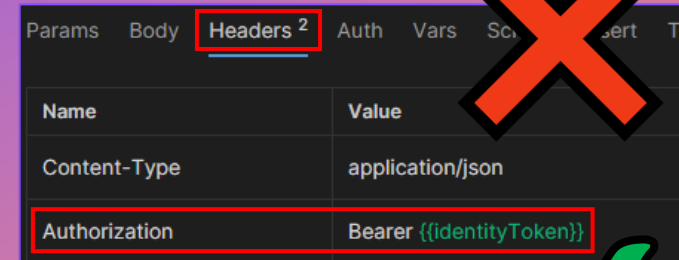
HTTP Headers contain metadata sent in the request (when required by the API)



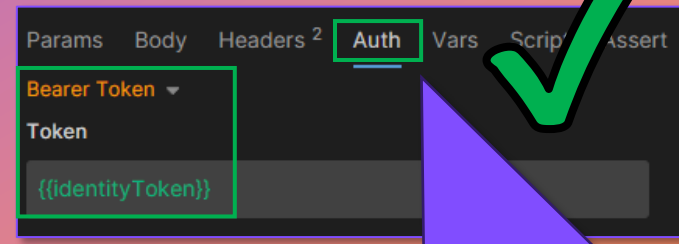
Name	Value	
Accept-Encoding	base64	<input checked="" type="checkbox"/>
Content-Type	application/x-www-form-urlencoded	<input checked="" type="checkbox"/>

+ Add Header

Update all data in the headers for your use case. Refer to documentation!



Name	Value
Content-Type	application/json
Authorization	Bearer {{identityToken}}



Bearer Token ▾

Token

{{identityToken}}

While 'Authorization' *could* be sent in the header, this Collection is being updated to use the 'Auth' section instead



# AUTH

The 'Auth' tab handles authentication credentials for the whole Collection

If AuthN is handled here, you do not need to send the token in the headers

Other AuthN methods

GET Get Applic... x +

GET https://{{subdomain}}.privilegecloud.cyberark.cloud/Pass

Params Body Headers 1 Auth Vars Script Assert Tests

Docs

Bearer Token ▾

Token

{{identityToken}}

{{identityToken}} is used for Identity Security Platform (ISP)

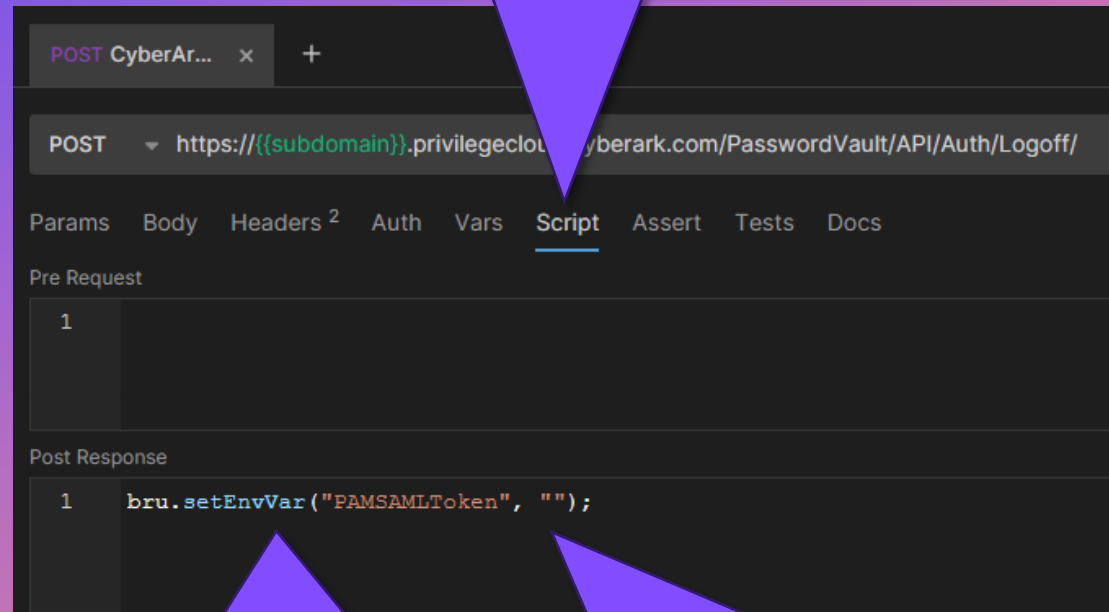
Bearer Token ▾

- AWS Sig v4
- Basic Auth
- Bearer Token
- Digest Auth
- OAuth 2.0
- Inherit
- No Auth

# SCRIPT

Scripts contain logic that can run before or after an API call

This Collection rarely uses 'Scripts'



Scripts are written in Bruno markup language

This script clears the `{{PAMSAMLTOKEN}}` environment variable upon logoff

# TESTS

Tests are scripts that are executed at runtime and report the result back to you

Tests are written in JavaScript using assertion libraries like Chai.js

Test results can be reviewed here

Privilege Cloud and Shared Services REST API

POST Start Aut... POST Advance... POST Advance... POST Advance... POST Advance...

POST https://{{{identityTenantID}}}.id.cyberark.cloud/Security/AdvanceAuthentication

Params Body Headers<sup>1</sup> Auth Vars Script Assert **Tests** Docs

```
1 const jsonData = res.getBody().Result
2
3 test("Identity Token Exists", function() {
4   expect(jsonData).to.have.property('Token');
5   bru.setEnvVar("identityToken", jsonData.Token);
6 });
```

Response Headers Timeline **Tests<sup>1</sup>**

Tests (1/1), Passed: 1, Failed: 0

✓ Identity Token Exists

Assertions (0/0), Passed: 0, Failed: 0

This test checks for a returned token, then stores it in the `{{{identityToken}}}` environment variable

## Common Chai Assertions

- Equality Check: `expect(value).to.equal(10)`
- Deep Equality (for comparing objects): `expect(obj1).to.deep.equal(obj2)`
- Property Existence: `expect(obj).to.have.property('key')`
- Array/Length Check: `expect(arr).to.have.lengthOf(3)`
- Type Checking: `expect(value).to.be.a('string')`

# DOCS

'Docs' references are included for each API call

POST Start Aut... x +

POST https://({{identityTenantID}}).id.cyberark.cloud/Security/StartAuthentication

Params Body Headers<sup>1</sup> Auth Vars Script Assert Tests Docs

Edit

Official documentation: <https://api-docs.cyberark.com/docs/identity/authentication-and-authorization/operations/create-a-security-start-authentication>

Start authentication:

CyberArk Identity uses OAuth 2.0 access tokens to authenticate API requests. You acquire access tokens on behalf of CyberArk Identity users by calling /Security/StartAuthentication. Clients requiring access to resources outside of a specific user's context should use the client credentials flow instead. To identify the user being authenticated, CyberArk Identity needs their User identifier and TenantId.

Advance Authentication:

/Security/StartAuthentication initiates the user's authentication process. If the user exists, the response contains a Summary of either LoginSuccess or NewPackage. You receive LoginSuccess when the request includes an .ASPXAUTH cookie from prior successful authentication. Otherwise, NewPackage responses contain an authentication package consisting of one or two challenges the user must answer. Each challenge must be completed using an MFA mechanism. Mechanisms can be either in-band or out-of-band (OOB). You collect in-band mechanism responses, such as passwords, directly from the user. Users must interact with out-of-band mechanisms, such as SMS codes, from outside of your app. You should complete authentication by calling /Security/AdvanceAuthentication sequentially for each mechanism the user chooses to answer.

Federated users:

When you specify a User associated with a SAML or OIDC-based federation, then the response contains:

Official documentation is linked

Copy/paste of the Docs page



CyberArk® Bruno REST API Collection

# AUTHENTICATION

# OVERVIEW

AuthN to Identity Security Platform (SaaS)

Privilege Cloud and Shared Services REST API
> Central Credential Provider
> CM - Connector Management
> Conjur Cloud
> DPA - Dynamic Privileged Access
> EPM - Endpoint Privilege Manager SaaS
< Identity - Shared Services
> Application Management
Authentication
Login
< POST Start Authentication
POST Advance Authentication - Start OC
POST Advance Authentication - Poll OO
POST Advance Authentication - Answer
POST Advance Authentication - User Pa
POST Advance Authentication - Security
POST Advance Authentication - Security

Token used in 'Auth' tab

Params <sup>1</sup>	Body	Headers <sup>1</sup>	Auth
Bearer Token ▾			
Token			
{{identityToken}}			

```
test("Identity Token Exists", function() {  
  if (jsonData.Token) {  
    bru.setEnvVar("identityToken", jsonData.Token);  
  }  
})
```

Token stored as {{identityToken}}

AuthN to Vault or PVWA for Privilege Cloud Standalone

Privilege Cloud and Shared Services REST API
> Central Credential Provider
> CM - Connector Management
> Conjur Cloud
> DPA - Dynamic Privileged Access
> EPM - Endpoint Privilege Manager SaaS
> Identity - Shared Services
> Privilege Cloud - Shared Services
< Privilege Cloud - Standard
> Accounts
> Applications
Authentication
> Logoff
Login
POST CyberArk, LDAP, RADIUS
POST SAML Logon

```
const jsonData = JSON.parse(responseBody);  
bru.setEnvVar("PAMSessionToken", jsonData);
```

Token stored as {{PAMSessionToken}}

AuthN to Vault or PVWA for PAM Self-Hosted

CyberArk Self-Hosted REST API
> Central Credential Provider
> EPM - Endpoint Privilege Manager On-Prem
> RA - Remote Access
< Self-Hosted PAM
> Accounts
> Applications
Authentication
> Authentication Methods Config
> Logoff
Login
POST CyberArk, LDAP, RADIUS, Windows - Logon
POST SAML - Logon
POST PTA Authentication
POST Shared Logon Authentication - Logon

Params	Body	Headers <sup>2</sup>	Auth	Vars	Script	Ass
Name		Value				
Content-Type		application/json				
Authorization		{{PAMSessionToken}}				

Token sent in Headers

# ISP AUTHENTICATION

## START AUTHENTICATION 1/2

The screenshot shows the Bruno API client interface. On the left, a sidebar lists collections, with 'Privilege Cloud and Shared Services REST API' expanded to show 'Authentication' > 'Login' > 'POST Start Authentication'. A callout '1' points to this item. The main panel shows the 'POST Start Aut...' endpoint with a URL containing a variable. A callout '3' points to the tabs (Params, Body, Headers, Auth, Vars, Script, Assert, Tests, Docs), with a note 'Review and update all tabs and variables'. The 'Body' tab is active, showing a JSON payload with variables for 'TenantID', 'User', and 'Version'. A callout points to the 'TenantID' and 'User' fields with the text 'Only 'tenant ID' and 'username' are sent'. At the bottom right, a keyboard shortcuts menu is open, with a callout '4' pointing to the 'Send Request' option (Ctrl + Enter).

1 Select 'Start Authentication'

2 Load an Environment with 'ctrl + e'

3 Review and update all tabs and variables

Only 'tenant ID' and 'username' are sent

4 Send the request

Send Request Ctrl + Enter

New Request Ctrl + R

Edit Environments Ctrl + S



# ISP AUTHENTICATION

## START AUTHENTICATION 2/2

After providing username and tenant ID, Identity returns the available MFA challenges

Response Headers Timeline Tests 4

200 OK 3.41s 1.4

```
1 {
2   "success": true,
3   "Result": {
4     "ClientHints": {
5       "PersistDefault": false,
6       "AllowPersist": true,
7       "AllowForgotPassword": true,
8       "EndpointAuthenticationEnabled": false
9     },
10    "Version": "1.0",
11    "SessionId": "JdURMegInpm-SijLKlao0UHDkUbOvrFAgEnIycxOCMsl",
12    "EventDescription": null,
13    "RetryWaitingTime": 0,
14    "SecurityImageName": null,
15    "AllowLoginMfaCache": false,
16    "Challenges": [
17      {
18        "Mechanisms": [
19          {
20            "AnswerType": "Text",
21            "Name": "UP",
22            "PromptMechChosen": "Enter Password",
23            "PromptSelectMech": "Password",
24            "MechanismId": "ItaunBmSFuG3wdsykDWQUwBsmRRwt54vFY8NLdhjQA1",
25            "Enrolled": true
26          }
27        ]
28      },
29      {
30        "Mechanisms": [
31          {
32            "MaskedEmailAddress": "****@cyberark.com",
33            "AnswerType": "StartTextOob",
34            "Name": "EMAIL",
35            "PromptMechChosen": "Email sent to xxxx@cyberark.com. Click the link or manually enter the code to authenticate.",
36            "PromptSelectMech": "Email... @cyberark.com",
37            "PartialAddress": "cyberark.com",
38            "MechanismId": "_K8JkE0LUbSQIPTxF2jSX-xpUQmEs35Jt1UdT6ZEVG01",
39            "Enrolled": true
40          }
41        ]
42      }
43    ]
44  }
45 }
```

'Tests' identify and store the data you need for the next API call(s)

Response Headers Timeline Tests 4

Tests (4/4), Passed: 4, Failed: 0

✓ Identity Session ID Exists

✓ Identity Mechanism ID Exists

✓ Identity Mechanism ID 2 Exists

✓ Identity Token Exists

Assertions (0/0), Passed: 0, Failed: 0

Environment gets updated

✓	identityMechanismId	_K8JkE0LUbSQIPTxF2jSX-
✓	identityMechanismId	ItaunBmSFuG3wdsykDW
✓	identitySessionID	JdURMegInpm-SijLKlao0U

We must answer each MFA challenge with the corresponding API call



# ISP AUTHENTICATION

## ADVANCE AUTHENTICATION – USER PASSWORD

The screenshot shows the Bruno API client interface. On the left, the sidebar lists the project structure: **bruno** > **Conjur Cloud** > **DPA - Dynamic Privileged Access** > **EPM - Endpoint Privilege Manager SaaS** > **Identity - Shared Services** > **Application Management** > **Authentication** > **Login**. Under **Login**, several POST requests are listed, with **POST Advance Authentication - User Password** highlighted.

The main panel shows the details for the selected request: **POST** `https://{{identityTenantID}}.id.cyberark.cloud/Security/AdvanceAuthentication`. The **Body** tab is active, displaying a JSON payload:

```
1 {
2   "SessionId": "{{identitySessionID}}",
3   "MechanismId": "{{identityMechanismID}}",
4   "Action": "Answer",
5   "Answer": "{{identityPassword}}",
6   "PersistentLogin": "true"
7 }
```

Callout 1 points to the selected request in the sidebar. Callout 2 points to the JSON body. Callout 3 points to the **Send Request** button (labeled **Ctrl + Enter**). Callout 4 points to the **Tests** tab, which contains a test script:

```
1 const jsonData = res.getBody().Result
2
3 test("Identity Token Exists", function() {
4   expect(jsonData).to.have.property('Token');
5   bru.setEnvVar("identityToken", jsonData.Token);
6 });
```

Callout 4 also includes the text: "A test checks if token is returned and stores as {{identityToken}}".

Callout 5 points to the **Response** tab, which shows the following JSON response:

```
1 {
2   "success": true,
3   "Result": {
4     "Summary": "StartNextChallenge"
5   },
6   "Message": null,
7   "MessageID": null,
8   "Exception": null,
9   "ErrorID": null,
10  "ErrorCode": null,
11  "IsSoftError": false,
12  "InnerExceptions": null
13 }
```

Callout 5 includes the text: "'StartNextChallenge' means more MFA is required (no token yet)".

- 1 Select 'Advance Authentication – User Password'
- 2 Review and update the 'Body' variables
- 3 Send the request
- 4 A test checks if token is returned and stores as {{identityToken}}
- 5 'StartNextChallenge' means more MFA is required (no token yet)

# ISP AUTHENTICATION

## ADVANCE AUTHENTICATION – START OOB

'Start OOB' is used to initiate 'out-of-band' MFA challenges like email or SMS confirmation

**1** Select 'Advance Authentication – Start OOB'

**2** Review and update the 'Body' variables

**3** Send the request

**4** 'OobPending' means the challenge was initiated

POST Advance Authentication - Start OOB

POST https://{{identityTenantID}}.id.cyberark.cloud/Security/AdvanceAuthentication

Body (JSON)

```
{
  "TenantId": "{{identityTenantID}}",
  "SessionId": "{{identitySessionID}}",
  "MechanismId": "{{identityMechanismID2}}",
  "Action": "StartOOB"
}
```

Send Request Ctrl + Enter

Response (JSON)

```
{
  "success": true,
  "Result": {
    "Summary": "OobPending"
  },
  "Message": null,
  "MessageID": null,
  "Exception": null,
  "ErrorID": null,
  "ErrorCode": null,
  "IsSoftError": false,
  "InnerExceptions": null
}
```

# ISP PLATFORM AUTHENTICATION

## ADVANCE AUTHENTICATION – POLL OOB

'Poll OOB' is used to check if OOB MFA has been completed yet

The screenshot shows the Bruno REST client interface. On the left, the sidebar lists the project structure, with 'POST Advance Authentication - Poll OOB' selected under the 'Authentication' section. The main panel shows the request configuration for 'POST Advance...'. The URL is `https://{{identityTenantID}}.id.cyberark.cloud/Security/AdvanceAuthentication`. The 'Body' tab is active, showing a JSON payload with the following fields:

```
1 {
2   "TenantId": "{{identityTenantID}}",
3   "SessionId": "{{identitySessionID}}",
4   "MechanismId": "{{identityMechanismID2}}",
5   "Action": "Poll"
6 }
```

The 'Action' field is highlighted with a red box. The 'Send Request' button is also visible at the bottom right of the interface.

'OobPending' means the challenge is incomplete

4

The screenshot shows the 'Response' tab of the Bruno REST client. The JSON response is as follows:

```
1 {
2   "success": true,
3   "Result": {
4     "Summary": "OobPending"
5   },
6   "Message": null,
7   "MessageID": null,
8   "Exception": null,
9   "ErrorID": null,
10  "ErrorCode": null,
11  "IsSoftError": false,
12  "InnerExceptions": null
13 }
```

The 'Summary' field is highlighted with a red box, indicating the 'OobPending' status.

1

Select 'Advance Authentication – Poll OOB'

2

All data is the same except the 'Action'

Send the request

3

# ISP AUTHENTICATION

## ADVANCE AUTHENTICATION – ANSWER OOB

'Answer OOB' is used to provide the confirmation code from OOB MFA

**1** Select 'Advance Authentication – Answer OOB'

**2** Update 'Answer' with your MFA code

**3** Send the request

**4** A test checks if token is returned and stores as `{{identityToken}}`

bruno

- > Conjur Cloud
- > DPA - Dynamic Privileged Access
- > EPM - Endpoint Privilege Manager SaaS
- > Identity - Shared Services
  - > Application Management
  - > Authentication
    - > Login
      - POST Start Authentication
      - POST Advance Authentication - Start OOB
      - POST Advance Authentication - Poll OOB
      - POST Advance Authentication - Answer OOB
      - POST Advance Authentication - User Passwo...

Privilege Cloud and Shared Services REST API

POST Advance... POST Advance... POST Advance... x +

POST https://{{identityTenantID}}.id.cyberark.cloud/Security/AdvanceAuthentication

Params Body Headers 1 Auth Vars Script Assert Tests Docs JSON

```
1 {
2   "TenantId": "{{identityTenantID}}",
3   "SessionId": "{{identitySessionID}}",
4   "MechanismId": "{{identityMechanismID2}}",
5   "Action": "Answer",
6   "Answer": "123456"
7 }
```

Send Request Ctrl + Enter

New Request Ctrl + N

Edit Environments Ctrl + O

Params Body Headers 1 Auth Vars Script Assert Tests Docs

```
1 const jsonData = res.getBody().Result
2
3 test("Identity Token Exists", function() {
4   expect(jsonData).to.have.property('Token');
5   bru.setEnvVar("identityToken", jsonData.Token);
6 });
```

# ISP AUTHENTICATION

## ADVANCE AUTHENTICATION – SECURITY QUESTION

This API is used to provide the answer to 'security question' MFA

1 Select 'Advance Authentication – Security Question'

2 Update 'Answer' with your answer

3 Send the request

4 A test checks if token is returned and stores as `{{identityToken}}`

```
POST https://{{identityTenantID}}.id.cyberark.cloud/Security/AdvanceAuthentication

{
  "TenantId": "{{identityTenantID}}",
  "SessionId": "{{identitySessionID}}",
  "MechanismId": "{{identityMechanismID2}}",
  "Action": "Answer",
  "Answer": "{{securityQuestionAnswer}}"
}
```

```
const jsonData = res.getBody().Result
test("Identity Token Exists", function() {
  expect(jsonData).to.have.property('Token');
  bru.setEnvVar("identityToken", jsonData.Token);
});
```

## PRIVILEGE CLOUD - STANDALONE AUTHENTICATION

The screenshot shows the Bruno API client interface. The sidebar on the left displays a tree view of API collections. The main panel shows a POST request to the CyberArk Logon API. The 'Params' tab is active, showing a table with columns 'Name' and 'Value'. The 'authType' parameter is highlighted with a red box, and its value is {{authType}}. A red callout box points to the 'authType' parameter, stating: {{authType}} = Cyberark | LDAP | RADIUS. Another red callout box points to the 'Send Request' button, stating: Send the request. A third red callout box points to the 'const jsonData = JSON.parse(responseBody);' line in the script, stating: A test checks if token is returned and stores as {{PAMSessionToken}}. The bottom panel shows the request body with a JSON object containing 'token' and 'tokenType' fields.

- 1 Select the Logon API
- 2 Load an Environment with 'ctrl + e'
- 3 Review and update all tabs and variables
- 4 Send the request
- 5 A test checks if token is returned and stores as {{PAMSessionToken}}



# PAM SELF-HOSTED AUTHENTICATION

**1** Select the Logon API

**2** Load an Environment with 'ctrl + e'

**3** Review and update all tabs and variables

**4** Send the request

**5** A script checks if token is returned and stores as `{{PAMSessionToken}}`

The screenshot shows the Bruno REST client interface. The sidebar on the left displays a tree view of API collections, with 'CyberArk Self-Hosted REST API' expanded and 'Authentication' > 'Logon' selected. The main panel shows a POST request configuration for the endpoint `https://{{PVWAAddress}}/PasswordVault/API/auth/:authType/Logon`. The 'Params' tab is active, showing a table with one parameter:

Name	Value
authType	Cyberark   Windows   LDAP   RADIUS

The 'Script' tab at the bottom contains the following code:

```
const jsonData = JSON.parse(responseBody);
bru.setEnvVar("PAMSessionToken", jsonData);
```



CyberArk® Bruno REST API Collection

# CREDENTIAL PROVIDERS



# CENTRAL CREDENTIAL PROVIDER

## GETPASSWORD

The GetPassword request is included in all Collections

Central Credential Provider FQDN

Privilege Cloud and Shared Services REST API

GET GetPassw...

GET https://{{CCPAddress}}/AIMWebService/api/Accounts?AppID={{appID}}&Safe={{safeName}}

Params 2 Body Headers 1 Auth Vars Script Assert Tests Docs

Query

Name	Value	
AppID	{{appID}}	<input checked="" type="checkbox"/>
Safe	{{safeName}}	<input checked="" type="checkbox"/>
Object	{{accountName}}	<input type="checkbox"/>
UserNam	string	
Addres	string	

Include AppID and safe name in all requests

```
const data = res.getBody().Result

test("Store secret as environment variable", function() {
  if (data.Content) {
    bru.setEnvVar("CCPRetrievedSecret", data.Content);
  }
});
```

A test checks if secret is returned and stores as {{CCPRetrievedSecret}}



CyberArk® Bruno REST API Collection

# NEXT STEPS

# NEXT STEPS

Now you can start automating!

Test familiar tasks through the API

The screenshot displays a REST client interface for the 'Privilege Cloud and Shared Services REST API'. The selected endpoint is a POST request to `https://{{identityTenantID}}.id.cyberark.cloud/Roles/StoreRole`. The request body is a JSON object:

```
1 {
2   "Name": "CreatedViaBruno",
3   "Description": "Role created via Bruno API test",
4   "RoleType": "PrincipalList"
5 }
```

The 'Auth' tab is active, showing a 'Bearer Token' configuration with the token value `{{identityToken}}`. The response tab shows a successful JSON response:

```
1 {
2   "success": true,
3   "Result": {
4     "_RowKey": "0c62b343_1a15_4f72_b10b_db8990445a8e"
5   },
6   "Message": null,
7   "MessageID": null,
8   "Exception": null,
9   "ErrorID": null,
10  "ErrorCode": null,
11  "IsSoftError": false,
12  "InnerExceptions": null
13 }
```

Two callout boxes provide additional context:

- A box pointing to the token field states: "The token is valid for the max session duration".
- A box at the bottom right, accompanied by a lightbulb icon, states: "Don't forget to use the 'logout' API to end your session".

# HAPPY AUTOMATING!



Eli Hopkins

Security Architect

/in/ewhopkins

@IAM-Jah

CyBruArk@gmail.com

