# Password Safe

# PSRUN User Guide 6.10



TC:2/7/2020

# Table of Contents

# Overview

PSRUN is an API client designed to allow the execution of BeyondInsight and Password Safe API calls, optionally sending a set of environmental factors to the server to verify the client's identity.

## Supported Platforms

The following platforms are supported:

- Windows 10 and higher
- Linux 64-bit (Red Hat and Debian variants)
- AIX 5.2 and higher
- HPUX ia64
- Solaris

## Prerequisites

**Windows:**

- Microsoft Visual C++
- Redistributable for Visual Studio 2017, x86

**Linux:**

- glibc-2.3.4 RPM for i686

# Usage

PSRUN can issue API calls directly or by using short commands.

> For more details, please see the *BeyondInsight and Password Safe API Guide*

**Usage:**

```
psrun2 [options] host key user method endpoint [payload]

psrun2 [options] host key user short-command [payload]
```

## Parameters

- **Host**: The BeyondInsight host
- **Key**: The API registration key
- **User**: The BeyondInsight user that is granted permission to use the API key
  - If using a domain account, escape it with a backslash (for example, **domain\\user**)
  - If BeyondInsight requires a user password, append it to the value (for example, **"user;pwd=[my-password]"**)
- **Method**: The API action, must be one of **GET**, **PUT**, **POST**, or **DELETE**
- **Endpoint**: An API endpoint (for example, **Assets**, **Credentials**, **Imports**, etc.)
- **Payload**: The request body, specified in key=value format if calling the API directly, or as a list of values if using short commands.

## Options

PSRUN behavior can be controlled by options, which must be specified before the address parameter:

| | |
|---|---|
| **-v** | Verbose, logs all communication as well as the factors sent to the server. |
| **-sf** | Skips factors if they are not required for authentication. |
| **-i** | Allows insecure communication when the server certificate cannot be verified. |
| **-quote** | Wrap column output in double quotes. |
| **-separator <separator>** | Delimit column output (default is TAB). |
| **-noheaders** | Hides column names in the output. |
| **-filter** | Shows only specified columns. |
| **-cert "path"** | Specifies the path to the client certificate file. |
| **-certpass "password"** | Specifies the password for the certificate file. |
| **-e "command"** | Executes a system command instead of displaying result . |
| **--help** | Display this usage and short command help. |

## Error Code

PSRUN returns an error code of **0** for success, or **1** for failure.

## Examples

```
psrun2 -quote -separator "," address …
```

```
psrun2 -filter "SystemId,SystemName,AccountId,AccountName" … ListAccounts
```

```
psrun2 -cert CertificateFileName address …
```

```
psrun2 -cert "CertificateFileName" -certpass "CertificatePassword" address …
```

```
psrun2 -e "echo '[Password]'" address RetrievePassword …
```

```
psrun2 … POST UserGroups/1/Permissions [ PermissionID=1 AccessLevelID=1 PermissionID=1
AccessLevelID=2 ]
```

```
psrun2 $(cat params.txt) ListSystems
```

# Short Commands

Short Commands simplify API workflows by reducing command-line input and chaining successive calls in a single command, instead of calling each endpoint directly.

Short command parameters are ordered, not named; they do not need to be prefixed with the parameter name and need only be in the correct order. For example, the syntax for the command **RetrievePassword** is:

```
psrun2 -i $host $key $user RetrievePassword $MANAGEDSYSTEM $MANAGEDACCOUNT $REASON
```

## RetrievePassword (alias: RetrievePasswordByName)

APIs: GET ManagedAccounts, POST Requests, GET Credentials, PUT Requests/{id}/Checkin

Or: POST ISARequests (for ISA-based access)

Finds an account by name (if necessary), creates a request, then retrieves a password. After printing the password, the request is released (see **DoNotRelease** parameter).

### Parameters

- **SystemName**: The managed system name. Use **DatabaseName\InstanceName** for databases.
- **AccountName**: The managed account name. Can use IDs instead of names (but do not mix both).
- **Reason**: The reason to retrieve a password.
- **DurationMinutes** (optional): The request duration (in minutes). Default request duration is 10 minutes.
- **Type** (optional, default: **password**): The type of credentials to retrieve (password, dsskey).
- **DoNotRelease** (optional): Do not release created request. Allowed values are **DoNotRelease** or **-p**.

**Examples:**

```
psrun2 $(cat conn) RetrievePassword SystemName AccountName "your reason"
```

```
psrun2 $(cat conn) RetrievePassword 1 2 "your reason"
```

```
psrun2 $(cat conn) RetrievePassword 1 2 "your reason" 25 password DoNotRelease
```

```
psrun2 $(cat conn) RetrievePassword 1 2 "your reason" 25 -p
```

> ***Note: RetrievePassword** is affected by the number of approvers. This command works only with **Auto Approve** enabled in the Access Policy.*

## ListAssets

API: GET Workgroups/{workgroupID}/Assets or Workgroups/{workgroupName}/Assets

### Parameters

- **Workgroup**: ID or name of the workgroup.
- **Limit** (optional): Number of records to return.
- **Offset** (optional): Number of records to skip before returning <limit> records (works only with **limit**).

**Examples:**

```
psrun2 $(cat conn) ListAssets 1
```

```
psrun2 $(cat conn) ListAssets PasswordSafe
```

```
psrun2 $(cat conn) ListAssetts PasswordSafe 2 2
```

## ListWorkgroups

API: GET Workgroups

**Example:**

```
psrun2 $(cat conn) ListWorkgroups
```

## ListSystems

API: GET ManagedSystems or GET ManagedSystems/{id}

### Parameters

- **id (optional)**: ID of the managed system.

**Examples:**

```
psrun2 $(cat conn) ListSystems
```

```
psrun2 $(cat conn) ListSystems 123
```

## ListAccounts

API: GET ManagedAccounts?systemName={system}&accountName={account}&workgroupName={workgroup}

### Parameters

- **SystemName** (optional): Managed system name (must be used with **AccountName**).
- **AccountName** (optional): Managed account name (must be used with **SystemName**).

- **WorkgroupName** (optional): Workgroup name.
- **Type** (optional): Type of managed accounts to return.
  - **System**: Returns local accounts.
  - **Domainlinked**: Returns domain accounts linked to systems.
  - **Database**: Returns database accounts.
  - **Cloud**: Returns cloud system accounts.
  - **Application**: Returns application accounts.

**Examples:**

```
psrun2 -separator "," -filter "SystemId,SystemName,AccountId,AccountName" $(cat conn) ListAccounts
TestSystemName TestAccountName "BeyondTrust Workgroup"
```

```
psrun2 $(cat conn) ListAccounts
```

```
psrun2 $(cat conn) ListAccounts database
```

## ListAliases

API: GET Aliases or GET Aliases/{name}

### Parameters

- **Name**: Name of the managed account alias.

**Examples:**

```
psrun2 $(cat conn) ListAliases
```

```
psrun2 $(cat conn) ListAliases AliasName
```

## ListGroups

API: GET UserGroups or GET <base>/UserGroups/{id} or GET <base>/UserGroups/{name}

### Parameters

- **Group** (optional): ID or name of the user group.

**Examples:**

```
psrun2 $(cat conn) ListGroups
```

```
psrun2 $(cat conn) ListGroups 1
```

```
psrun2 $(cat conn) ListGroups Administrators
```

## ListGroupMembership

API: GET UserGroups/{userGroupId}/Users

### Parameters

- **UserGroupId:** User group ID.

**Example:**

```
psrun2 $(cat conn) ListGroupMembership 1
```

## ListRequest

API: GET Requests

### Parameters

- **Status** (optional, default: **all**): The status of requests to return (**all**, **active**, **pending**).
- **Queue** (optional, default: **req**): The type of request queue to return (**req**, **app**).

**Examples:**

```
psrun2 $(cat conn) ListRequests active
```

```
psrun2 $(cat conn) ListRequests all req
```

## ListRoles

API: GET Roles

**Example:**

```
psrun2 $(cat conn) ListRoles
```

## ListSmartRules

API: GET SmartRules

### Parameters

- **Type** (optional, default: **all**): The type of smart rules to return (**all**, **ManagedAccount**, **Asset**, **Vulnerabilities**)

**Examples:**

```
psrun2 $(cat conn) ListSmartRules
```

```
psrun2 $(cat conn) ListSmartRules Asset
```

## Request

API: POST Requests

### Parameters

- **AccessType** (optional, default: **View**): The type of access requested (**View**, **RDP**, **SSH**).
- **SystemId**: ID of the managed system to request.
- **AccountId**: ID of the managed account to request.
- **DurationMinutes**: The request duration (in minutes).
- **Reason** (optional): The reason for the request.
- **AccessPolicyScheduleID** (optional): The schedule ID of an access policy to use for the request. If omitted, automatically selects the best schedule.
- **ConflictOption** (optional, default: **renew**): The conflict resolution option to use if an existing request is found for the same user, system, and account (**reuse**, **renew**). If omitted and a conflicting request is found, returns a 409.
    - **Reuse**: Return an existing, approved request ID for the same user / system / account / access type (if one exists). If the request does not already exist, create a new request using the request body details.
    - **Renew**: Cancel any existing approved requests for the same user / system / account and create a new request using the request body details.

**Example:**

```
psrun2 $(cat conn) Request 1 1 120 "Request reason"
```

## ISARequests

API: POST ISARequests

### Parameters

- **Type** (optional, default: **password**): the type of credentials to retrieve (**password**, **dsskey**)
- **SystemID** (required): ID of the managed system to request.
- **AccountID** (required): ID of the managed account to request.

- **DurationMinutes** (optional): The request duration (in minutes).
- **Reason** (optional): The reason for the request.

**Examples:**

```
psrun2 $(cat conn) ISARequests 1 1 15 "Reason"
```

```
psrun2 $(cat conn) ISARequests 1 1
```

## Retrieve

API: GET Credentials/{requestId}

### Parameters

- **RequestId**: ID of the request.
- **Type** (optional, default value: **password**): the type of credentials to retrieve (**password**, **dsskey**).

**Example:**

```
psrun2 $(cat conn) Retrieve 12 dsskey
```

## Release

API: PUT Requests/{requestId}/Checkin

### Parameters

- **ID**: ID of the request to release.
- **Reason** (optional): A reason or comment why the request is being released.

**Example:**

```
psrun2 $(cat conn) Release 123 "reason for release"
```

## ImportFile

API: POST Imports (Base64FileContents option)

### Parameters

- **WorkgroupName**: Name of the workgroup
- **ImportType** (case-sensitive, default: **PASSWORDSAFE**) – Type of import being queued:
  - **PASSWORDSAFE**: Password Safe import file. Expected file extension: .xml
  - **RETINARTD**: Retina© RTD import file. Expected file extension: .rtd
  - **NESSUS**: Nessus© import file. Expected file extension: .csv
  - **NESSUSSECCEN**: NessusSecurityCenter© import file. Expected file extension: .csv
  - **NEXPOSE**: Nexpose© import file. Expected file extension: .csv or .xml
  - **QUALYSGUARD**: QualysGuard© import file. Expected file extension: .csv or .xml
  - **METASPLOIT**: **METASPLOIT**© import file. Expected file extension: .xml
  - **MCAFEEVM**: McAfee Vulnerability Management© import file. Expected file extension: .csv
  - **TRIPWIRE**: Tripwire© import file. Expected file extension: .csv
- **FileName**: Name of the file to be imported
- **Filter** (optional, case-sensitive, default: **All Assets**): Asset selection filter
  - **All Assets**: No filter, import all
  - **Single IPv4 address** (example, 10.0.0.1)
  - **IPv4 range** (example, 10.0.0.1 - 10.0.0.5)
  - **CIDR** (example, 10.0.0.0 / 24)

**Example:**

```
psrun2 $(cat conn) ImportFile "PasswordSafe" PASSWORDSAFE data.xml
```

## ForceReset

API: GET ManagedAccounts?systemName={system}&accountName={account}, PUT ManagedAccounts/{accountId}/Credentials

> ***Note: ForceReset** updates a managed account password, public and private key. This command can also be used without parameters, with a password parameter (optionally with **UpdateSystem**), or with all parameters.*

### Parameters

- **SystemName**: Managed system name.
- **AccountName**: Managed account name.
- **Password**: New password, use empty quotes to auto-generate a value
- **UpdateSystem** (optional, default **1**): Whether to update the credentials on the referenced system
- **PublicKey**: The new public key to set on the host (could be a value or a name of the file)
- **PrivateKey**: The private key to set (provide passphrase if encrypted, could be a value or a name of the file)
- **Passphrase** (optional): The passphrase to use for an encrypted private key

**Examples:**

Generates random password (and keys, depending on account configuration):

```
psrun2 $(cat conn) ForceReset SystemName AccountName
```

Updates password on system and in BeyondInsight:

```
psrun2 $(cat conn) ForceReset SystemName AccountName Password
```

Updates password in BeyondInsight but does not try to change password on system:

```
psrun2 $(cat conn) ForceReset SystemName AccountName Password 0
```

Updates password and keys on system and in BeyondInsight:

```
psrun2 $(cat conn) ForceReset SystemName AccountName Password 1 "publicFile" "privateFile"
```

# Examples

*From the Password Safe API Guide:*

## GET AccessLevels

### Purpose

Returns a list of access levels for permissions, for example, **None**, **Read**, and **Read/Write**

### Required Permissions

- User Accounts Management (Read)

### Request Body

None

### Response Body

Content-Type: application/json

```
[
{
AccessLevelID:int,
Name: string,
},
…
]
```

### Response Codes

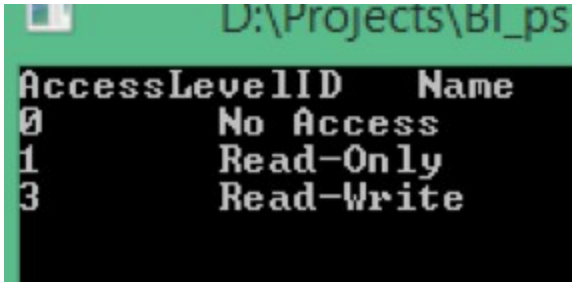
- 200 - Request successful. Access Levels in the response body.

> For more information, please see Common Response Codes.

Here's how to issue this API via PSRUN:

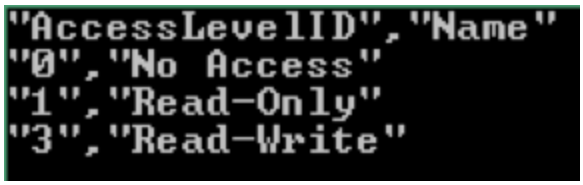
```
psrun2 127.0.0.1 3ea6..acb5acc cli GET AccessLevels
```

By default, results are tab-separated.



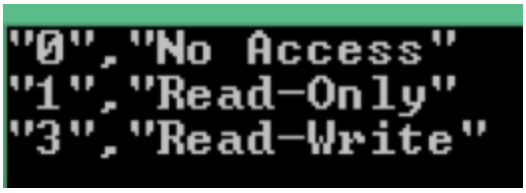
You can change this behavior to get CSV output:

```
psrun2 -quote -separator "," 127.0.0.1 3ea6..acb5acc
cli GET AccessLevels
```




It is also possible to remove the headers:

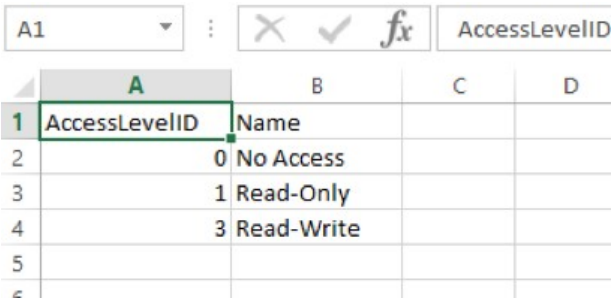
```
psrun2 -quote -separator "," -noheaders 127.0.0.1
3ea6..acb5acc cli GET AccessLevels
```




You can export the results through redirection:

```
psrun2 127.0.0.1 3ea6..acb5acc cli GET AccessLevels > results.xls
psrun2 -quote -separator "," 127.0.0.1 3ea6..acb5acc cli GET AccessLevels > results.csv
```

The results can be viewed in Excel.




## POST Requests

### Purpose

Creates a new release request.

### Required Roles

- Requestor or Requestor/Approver Role to Managed Account referenced by ID
- For Information Systems Administrator (ISA) role access see ISARequests and ISASessions.

### Request Body

Content-Type: application/json

```
{
AccessType: string,
SystemID: int,
AccountID: int,
ApplicationID: int, // can be null
```

```
DurationMinutes : int,
Reason : string,
AccessPolicyScheduleID : int, // can be null
ConflictOption : string,
Reason : string,
TicketSystemID : int,
TicketNumber : string,
RotateOnCheckin: bool
}
```

### Request Body Details

- **AccessType:** (optional, default: View) The type of access requested (View, RDP, SSH, App)
  - **View:** 'View Password' access.
  - **RDP:** RDP access (corresponds to POST Sesssions SessionType 'RDP' or 'rdpfile').
  - **SSH:** SSH access (corresponds to POST Sesssions SessionType 'SSH').
  - **App:** Application access (corresponds to POST Sesssions SessionType 'App' or 'appfile').
- **SystemID:** (required) ID of the Managed System to request.
- **AccountID:** (required) ID of the Managed Account to request.
- **ApplicationID:** (required when AccessType="App"): ID of the Application for an Application-based request.
- **DurationMinutes:** (required: 1-525600) The request duration (in minutes).
- **Reason:** (optional) The reason for the request.
- **AccessPolicyScheduleID:** (optional) The Schedule ID of an Access Policy to use for the request. If omitted, automatically selects the best schedule.
- **ConflictOption:** (optional) The conflict resolution option to use if an existing request is found for the same user, system, and account (reuse, renew). If omitted and a conflicting request is found, returns a 409 code (see below).
  - **reuse:** Returns an existing, approved request ID for the same user/system/account/access type (if one exists). If the request does not already exist, creates a new request using the request body details.
  - **renew:** Cancesl any existing approved requests for the same user/system/account and creates a new request using the request body details.
- **TicketSystemID:** ID of the ticket system. If omitted, then default ticket system will be used.
- **TicketNumber:** Number of associated ticket. Can be required if ticket system is marked as required in the global options.
- **RotateOnCheckin:** (optional, default: true) True to rotate the credentials on check-in/expiry, otherwise false. This property can only be used if the Access Policy (either auto-selected or given in **AccessPolicyScheduleID**) supports it.

For more information, please see the 'Allow API Rotation Override' Access Policy setting under 'View' access.

***Note:*** *In reference to* ***RotateOnCheckin****, If the Managed Account given in* ***AccountID*** *does not rotate the credentials after check-in/expiry, this setting is ignored.*

### Response Body

```
{
RequestID: int
}
```

### Response Codes

- 200 – Existing request is being reused. Existing request ID in the response body.
- 201 – Request successful. Request ID in the response body.
- 403 – User does not have permissions to request the indicated account or the account does not have API access enabled. Response body contains a status code indicating the reason for this forbidden access:
  - 4031 – User does not have permission to request the account or the account is not valid for the system.
  - 4032 – Requestor Only API or account. Only Requestors can access this API or account.
  - 4033 – Approver Only API or account. Only Approvers can access this API or account.
  - 4035 - Not enough approvers configured to approve a request.
- 409 – Conflicting request exists. This user or another user has already requested a password for the specified account within the next <durationMinutes> window.

For more information, please see Common Response Codes.

PSRUN command:

```
psrun2 127.0.0.1 3ea6..acb5acc "cli;pwd=[Password1]" POST Requests SystemId=1 AccountId=12

DurationMinutes=30 Reason="Just to test request"
```

# Authentication Factors

In addition to executing API calls, PSRUN also provides authentication factors to the server. These factors assist in verifying the client's identity.

When PSRUN executes an API call, it sends these factors as part of the header. On the server, the received factors are verified via user-configured PSRUN rules. If there are no rules, no validation takes place, and the server sends back the requested API response.

For each rule, the received factors are checked against the expected rule values. If a rule fails, the next rule is attempted. If the rule passes, the factors are considered valid.

Additionally, a unique signature is sent by PSRUN. If the factors pass the rule and signature verification is enabled, the server recomputes the signature and attempts to match it with the one sent by the client. If the signatures match, the signature is considered verified. Signature verification is an extra check to ensure the client and server are in sync so that out-of-date clients will not be authenticated.

The list of accepted PSRUN factors can be specified in BeyondInsight:

- IP address
- MAC address
- System name
- FQDN
- Domain name
- User ID
- Root volume ID
- OS version