

Tenable and Keeper Commander Integration Guide

Last Revised: October 01, 2026



Table of Contents

Welcome to the Tenable Integration with Keeper Commander	4
Overview	4
What Information Does the Keeper Commander Integration Collect?	4
Keeper Commander REST API Versions	5
API Requests per Target	6
Credentialed Scans	6
What the Keeper Commander Integration Does Not Collect	7
Keeper Commander Integration Limitations	7
Configure the Keeper Commander Integration for Tenable One Vulnerability Management ..	8
Required Permissions	8
Scan Configuration – SSH, Windows, and Database	8
Configure the Keeper Commander Integration for Tenable Nessus	14
Required Permissions	14
Scan Configuration – SSH, Windows, and Database	14
Scan Results Review	19
Plugin Families and Plugins	19
Debug Log Reporting	20
What the Logs Contain	21
Common Reasons for Credentialed Checks Showing "no" Status	21



Additional Information	22
Helpful tips	22
Testing integration connectivity	23
Optional Features	25
Privilege Escalation	25



Welcome to the Tenable Integration with Keeper Commander

This document provides information and steps for integrating Tenable One Vulnerability Management and Tenable Nessus with Keeper Commander. For more information, refer to the following product documentation:

- [Tenable One Vulnerability Management](#)
- [Tenable Nessus](#)

Overview

The Tenable integration with Keeper Commander provides a credentialed-scanning solution that lets security teams store privileged secrets in Keeper Security's vault and retrieve them automatically at scan time. Keeper centrally manages passwords and SSH private keys – Tenable never stores them in the scan policy, and they rotate without any manual policy update..

You can integrate Keeper Commander with Tenable One Vulnerability Management or Tenable Nessus to perform credentialed scanning of SSH, Windows (SMB), and Database targets.

The benefits of integrating Tenable with Keeper Commander include:

- Centralized management of privileged credentials for Windows, SSH, and Database targets.
- API key authentication to the Keeper Commander service, so no target passwords or SSH keys need to be stored in Tenable.
- Support for both password-based and SSH private key-based target authentication by selecting the appropriate Keeper Commander record type.

What Information Does the Keeper Commander Integration Collect?

The Keeper Commander PAM integration collects credentials from the Keeper Commander Service Mode REST API. The Keeper Commander PAM integration collects only the credential material



required to authenticate to a scan target. The integration retrieves a Keeper Commander record and reads the following fields from it:

- **login**: The target account username (required for all record types).
- **password**: The target account password, read from `pamUser` records.
- **keyPair.privateKey**: A PEM-encoded SSH private key, read from `sshKeys` records.
- **passphrase**: The passphrase for an encrypted SSH private key. Stored as a separate field on the `sshKeys` record (not nested inside `keyPair`). Extracted automatically when present – no additional configuration is required.

Note: password vs. SSH private key – The integration infers the credential type from the Keeper record itself. For `pamUser` records it extracts `login` and `password`. For `sshKeys` records it extracts `login` and `keyPair.privateKey`, plus `passphrase` if the record includes one. If the expected field is absent, credential retrieval fails and an error is written to the debug log

Keeper Commander REST API Versions

The Keeper Commander Service Mode REST API exposes two API generations. The current Tenable integration requires the v2 async API.

Note: If your Keeper Commander deployment only exposes the v1 endpoint, upgrade Keeper Commander to a version that includes the v2 async API (17.1.7 and later) before configuring this integration.

The v2 API uses a three-step async pattern:

1. **POST /api/v2/executecommand-async** – submits the command and returns a **request_id** immediately.
2. **GET /api/v2/status/<request_id>** – poll for completion. The integration polls once per second for up to 30 seconds. The **status** field in the response is one of **pending**, **completed**, **failed**, or **expired**.
3. **GET /api/v2/result/<request_id>** – fetches the final result once the status is **completed**.



The Engine URL field in the scan credential is an optional base prefix for the Commander service (for example, if the Keeper Commander `executecommand-async` endpoint is found at `/commander/api/v2/executecommand-async`, then the appropriate Engine URL value would be `/commander`. The integration appends the correct API path for each step.

API Requests per Target

For each distinct Keeper Commander record referenced in the scan policy, the integration makes the following API calls to the Keeper Commander service:

- Three requests to retrieve the primary credential: one POST to submit the **get <uid> --unmask --format json** command, up to 30 GET polls for status, and one GET to fetch the result.
- If **Query Mode** is set to **Search string**, one additional set of three requests is made first to execute the **search <term>** command and resolve the record UID before the record retrieval above. If **Search scan target** is also enabled, the search term includes the target's IP address or hostname, so each distinct target that resolves to a different record produces a separate set of requests (results are still cached per unique search term).
- If privilege escalation is configured with a separate **Escalation Credential ID** or **Escalation Search Text**, one extra set of three requests is made to retrieve the escalation record. If **Escalation Query Mode** is **Search String** and **Escalation search scan target** is enabled, the same per-target expansion applies.

Retrieved credentials are cached for the duration of the scan, keyed by lookup mode and lookup value (record UID or search term), to avoid redundant requests for the same record. The cache is not shared between scan chunks, so each chunk produces a minimum of one credential retrieval sequence.

Credentialed Scans

Credentialed scans allow the Tenable scanner to log in to the target system directly and perform a deeper assessment than is possible without credentials. This includes checking installed software versions, configuration settings, patch levels, and compliance status.



What the Keeper Commander Integration Does Not Collect

- The integration does not collect vulnerability data, software inventory, device inventory, or configuration details from target systems – it only retrieves credentials that Tenable then uses to authenticate to those targets.
- The integration does not parse other record types (such as login, bankAccount, or custom types), and attempting to use them causes credential retrieval to fail.
- The integration does not enumerate vault contents or discover scan targets. Specify targets explicitly in the scan.

Keeper Commander Integration Limitations

- The integration requires the Keeper Commander Service Mode REST API with API key authentication.
- The integration requires the v2 version of the API, which is present in Keeper Commander versions 17.1.7 and later.
- The Keeper Commander service endpoint specified in **Engine URL** must be reachable from every Tenable Nessus scanner used in the scan.
- When **Query Mode** is **Search string**, the integration uses the first matching record returned by the Keeper Commander search command. If multiple records match the search term, only the first result is used. Use **Record UID** mode to retrieve a specific record unambiguously.
- SSH Private Key (**sshKeys** record type) is supported only within the SSH credential type. Windows and Database credentials always use password authentication.
- There is no Auto-Discovery / dynamic-scanning variant for this integration. Targets must be enumerated explicitly in the scan.
- By default, each scan credential resolves a single fixed record UID or search term for all targets covered by that credential. Enable **Search scan target** (or **Escalation search scan target**) to append the scan target's IP address or hostname to the lookup at scan time, causing the integration to retrieve a target-specific record for each host.



Configure the Keeper Commander Integration for Tenable One Vulnerability Management

The Tenable integration with Keeper Commander exists as an authentication method within the SSH, Windows, and Database credential types in Tenable One Vulnerability Management scan policies.

Required Permissions

- A Tenable One Vulnerability Management user account with a minimum role of Standard.
- A Keeper Commander service instance reachable from the Tenable Nessus scanner, with the v2 async API enabled.
- An API key configured in the Keeper Commander service. The API key must have read access to every Keeper record referenced by the scan policy.
- The Keeper Commander record must be of type **pamUser** (for password authentication) or **sshKeys** (for SSH key authentication), and must contain a valid **login** field.

Scan Configuration – SSH, Windows, and Database

Complete the following steps to configure Tenable One Vulnerability Management with the Keeper Commander integration using SSH, Windows, or Database credentials.

1. Log in to your Tenable One Vulnerability Management user interface.
2. In the left navigation pane, click **Scans**.

The **Scans** page appears.

3. In the upper-right corner of the page, click **Create a Scan**.

The **Select a Scan Template** page appears.

4. Select a scan template.



The scan configuration page appears.

5. In the **Name** box, type a name for the scan.
6. In the **Targets** box, type an IP address, hostname, or range of IP addresses.
7. (Optional) Add a description, folder location, and scanner location, and specify target groups.
8. Click the **Credentials** tab.

The **Credentials** pane appears.

9. In the **Select a Credential** menu, select **SSH**, **Windows**, or **Database**.

The **Settings** pane for the selected credential appears.

10. In the **Auth Type** drop-down, click **Keeper Commander**. The Keeper Commander options appear.
11. Configure each option for the selected SSH, Windows, or Database credential using the following values.

Option	Description	Required
Host	The IP address or hostname of the Keeper Commander service.	Yes
Port	The port the Keeper Commander service listens on. For example: 8080.	Yes
Engine URL	Optional base path prefix for the Keeper Commander API (for example, /commander). Leave this blank to use the root path. The integration appends the v2 async API paths (/api/v2/executecommand-async , /api/v2/status/<id> , /api/v2/result/<id>) to this prefix.	No



API Key	The API key configured in the Keeper Commander service. Sent in the api-key request header for all Keeper Commander API calls.	Yes
Query Mode	How the integration identifies the Keeper Commander record. Select Record UID to retrieve a specific record by its unique identifier, or Search string to find the first record whose title matches a keyword. Default: Record UID.	Yes
Credential ID	The Keeper Commander record UID to retrieve. Only visible when Query Mode is set to Record UID .	Yes (if Query Mode is Record UID)
Search Text	The keyword matched against Keeper Commander record titles. The first matching record is used. Only visible when Query Mode is set to Search string .	Yes (if Query Mode is Search string)
Domain (SSH and Windows only)	Auto-populated from the Keeper record. If the record contains a custom field with the label <code>domain</code> (case-insensitive), its value is used as the Windows or SSH domain for authentication. No user interface entry is required – the value is read directly from the Keeper record at scan time.	No



Search scan target	When enabled, the integration appends the scan target's IP address or hostname to the Credential ID value at scan time and performs a keyword search instead of a UID lookup. Use this when record titles include the target address (for example, a Credential ID of root SSH password 192.168.1.100 resolves to root SSH password 192.168.1.100 for target 192.168.1.100). Only visible when Query Mode is set to Record UID . Default: No.	No
Use SSL	When enabled, the integration connects to the Keeper Commander service over HTTPS. Default: Yes.	Yes
Verify SSL Certificate	When enabled, Tenable verifies the TLS certificate of the Keeper Commander service. Disable only if the Keeper Commander instance uses a self-signed certificate not in the scanner's trust store. Default: Yes.	No
Elevate privileges with (SSH only)	The privilege escalation method used after initial authentication. Options: Nothing (default), .k5login, Cisco 'enable', dzdo, pbrun, su, su+sudo, sudo. Selecting a method reveals the escalation fields described in the Optional Features section.	No



Escalation Query Mode (SSH only)	How the integration identifies the escalation record. Select Record UID to retrieve a specific record by UID, or Search string to find the first record whose title matches a keyword. Default: Record UID.	No
Escalation Credential ID (SSH only)	The Keeper Commander record UID of the record whose password field is used as the privilege escalation password. Only visible when Escalation Query Mode is set to Record UID . If left blank while an escalation method is selected, the primary record UID is reused and its password serves as the escalation password.	No
Escalation Search Text (SSH only)	The keyword matched against Keeper Commander record titles to find the escalation record. The first matching record is used. Only visible when Escalation Query Mode is set to Search string .	No
Escalation search scan target (SSH only)	When enabled, the integration appends the scan target's IP address or hostname to the Escalation Credential ID value at scan time and performs a keyword search instead of a UID lookup. Only visible when Escalation Query Mode is set to	No



	Record UID. Default: No.	
Kerberos Target Authentication (SSH) / Use Kerberos KDC (Windows)	When enabled, Kerberos authentication is used to log in to SSH or Windows targets. Selecting Yes reveals the Key Distribution Center (KDC), KDC Port (default 88), KDC Transport (tcp / udp), and Kerberos Domain fields.	No

12. Do one of the following:

- To save without launching the scan, click **Save**.
- To save and launch the scan immediately, click **Save & Launch**.

Note: If you scheduled the scan to run at a later time, the **Save & Launch** option is not available.



Configure the Keeper Commander Integration for Tenable Nessus

The Tenable integration with Keeper Commander exists as an authentication method within the SSH, Windows, and Database credential types in Tenable Nessus scan policies.

Required Permissions

- A Tenable Nessus user account with permission to create and configure scans.
- The same Keeper Commander service instance, API key, and record permissions described in the Tenable One Vulnerability Management section above.

Scan Configuration – SSH, Windows, and Database

Complete the following steps to configure Tenable Nessus with the Keeper Commander integration using SSH, Windows, or Database credentials.

1. Log in to your Tenable Nessus user interface.
2. In the left navigation pane, click **Scans**.

The **Scans** page appears.

3. In the upper-right corner of the page, click **Create a Scan**.

The **Select a Scan Template** page appears.

4. Select a scan template.

The scan configuration page appears.

5. In the **Name** box, type a name for the scan.
6. In the **Targets** box, type an IP address, hostname, or range of IP addresses.
7. (Optional) Add a description, folder location, and scanner location, and specify target groups.
8. Click the **Credentials** tab.



The **Credentials** pane appears.

9. In the **Select a Credential** menu, select **SSH**, **Windows**, or **Database**.

The Settings pane for the selected credential appears.

10. In the **Auth Type** drop-down, click **Keeper Commander**.

The **Keeper Commander** options appear.

11. Configure each option for the selected SSH, Windows, or Database credential using the following values.

Option	Description	Required
Host	The IP address or hostname of the Keeper Commander service.	Yes
Port	The port the Keeper Commander service listens on. For example: 8080.	Yes
Engine URL	Optional base path prefix for the Keeper Commander API (for example, /commander). Leave this blank to use the root path. The integration appends the v2 async API paths (/api/v2/executecommand-async , /api/v2/status/<id> , /api/v2/result/<id>) to this prefix.	No
API Key	The API key configured in the Keeper Commander service. Sent in the api-key request header for all Keeper Commander API calls.	Yes
Query Mode	How the integration identifies the Keeper	Yes



	Commander record. Select Record UID to retrieve a specific record by its unique identifier, or Search string to find the first record whose title matches a keyword. Default: Record UID.	
Credential ID	The Keeper Commander record UID to retrieve. Only visible when Query Mode is set to Record UID .	Yes (if Query Mode is Record UID)
Search Text	The keyword matched against Keeper Commander record titles. The first matching record is used. Only visible when Query Mode is set to Search string .	Yes (if Query Mode is Search string)
Domain (SSH and Windows only)	Auto-populated from the Keeper record. If the record contains a custom field with the label domain (case-insensitive), its value is used as the Windows or SSH domain for authentication. No user interface entry is required – the value is read directly from the Keeper record at scan time.	No
Use SSL	When enabled, the integration connects to the Keeper Commander service over HTTPS. Default: Yes.	Yes
Verify SSL Certificate	When enabled, Tenable Nessus verifies the TLS certificate of the Keeper	No



	Commander service. Disable only if the Keeper Commander instance uses a self-signed certificate not in the scanner's trust store. Default: Yes.	
Elevate privileges with (SSH only)	The privilege escalation method used after initial authentication. Options: Nothing (default), .k5login, Cisco 'enable', dzdo, pbrun, su, su+sudo, sudo. Selecting a method reveals the escalation fields described in the Optional Features section.	No
Escalation Query Mode (SSH only)	How the integration identifies the escalation record. Select Record UID to retrieve a specific record by UID, or Search string to find the first record whose title matches a keyword. Default: Record UID.	No
Escalation Credential ID (SSH only)	The Keeper Commander record UID of the record whose password field is used as the privilege escalation password. Only visible when Escalation Query Mode is set to Record UID . If left blank while an escalation method is selected, the primary record UID is reused and its password serves as the escalation password.	No
Escalation Search Text	The keyword matched against Keeper	No



(SSH only)	Commander record titles to find the escalation record. The first matching record is used. Only visible when Escalation Query Mode is set to Search string .	
Escalation search scan target (SSH only)	When enabled, the integration appends the scan target's IP address or hostname to the Escalation Credential ID value at scan time and performs a keyword search instead of a UID lookup. Only visible when Escalation Query Mode is set to Record UID . Default: No.	No
Kerberos Target Authentication (SSH) / Use Kerberos KDC (Windows)	When enabled, Kerberos authentication is used to log in to SSH or Windows targets. Selecting Yes reveals the Key Distribution Center (KDC), KDC Port (default 88), KDC Transport (tcp / udp), and Kerberos Domain fields.	No

12. Do one of the following:

- To save without launching the scan, click **Save**.
- To save and launch the scan immediately, click **Save & Launch**.

Note: If you scheduled the scan to run at a later time, the **Save & Launch** option is not available.



Scan Results Review

Plugin Families and Plugins

You can use the Keeper Commander integration to gather credentials for target authentication during credentialed scans through SSH, Windows (SMB), or Database credential types. The following Tenable plugins are relevant when reviewing scan results.

Misc. plugin family:

- **Plugin #204872: Integration Status:** When using any Tenable PAM integration, the Integration Status plugin confirms whether credential retrieval was a success or failure.

Settings plugin family:

- **Plugin #14273: SSH settings:** Reads the SSH PAM credential configuration and calls the Keeper Commander integration to retrieve credentials for SSH targets. This plugin is not indicative of authentication issues by itself; authentication failures are reported by the credential-status plugins below.
- **Plugin #10870: Login configurations:** Reads the Windows (SMB) credential configuration and calls the Keeper Commander integration to retrieve credentials for Windows targets.
- **Plugin #33815: Database settings:** Reads the database credential configuration and calls the Keeper Commander integration to retrieve credentials for database targets.
- **Plugin #19506: Nessus Scan Information:** Reports scan metadata including whether credentialed checks succeeded. Check this plugin first when investigating authentication issues.
- **Plugin #91822: Database Authentication Failure(s) for Provided Credentials:** Indicates that the database credentials retrieved from Keeper Commander could not authenticate to the target database. Check that the record UID or search term resolves to a record with a valid username and password for the target database.



- **Plugin #141118: Target Credential Status by Authentication Protocol – Valid Credentials Provided:** Confirms credential validity by successfully authenticating to the remote target. Look for "Proto: SMB" or "Proto: SSH" in the output.
- **Plugin #110095: Target Credential Issues by Authentication Protocol – No Issues Found:** Indicates that credentials were provided and authentication succeeded for all targeted protocols.
- **Plugin #104410: Target Credential Status by Authentication Protocol – Failure for Provided Credentials:** Indicates that credentials were provided but authentication to the target failed – the values retrieved from Keeper Commander may be incorrect or may lack sufficient permission on the target.
- **Plugin #110723: Target Credential Status by Authentication Protocol – No Credentials Provided:** Indicates that no credentials were available for the targeted protocol. If this appears when a Keeper Commander credential is configured, the integration likely failed to retrieve the record.
- **Plugin #117885: Target Credential Issues by Authentication Protocol – Intermittent Authentication Failure:** Indicates that authentication succeeded for some targets but not others – may point to per-target credential differences or intermittent Keeper Commander service connectivity.

Debug Log Reporting

Logs generated by other plugins are reported by Plugin #84239: Debugging Log Report. Debug logs for the Keeper Commander integration are written by the Tenable Nessus scanner during the scan. Debugging log report output is controlled by the debug logging settings in Tenable Nessus.

The integration writes to log files named after the credential plugin that invoked it, with "Keeper Commander" appended so the PAM-integration log lines can be distinguished from the target-authentication log lines:



- **SSH:** `ssh_settings.nasl~Keeper Commander`
- **Windows (SMB):** `logins.nasl~Keeper Commander`
- **Database:** `database_settings.nasl~Keeper Commander`

What the Logs Contain

- Configuration settings loaded from the scan policy, with sensitive values (API key, passwords, and private key material) automatically masked.
- The query mode in use (UID or search) and the lookup value.
- Whether a cached credential was found before any API call was made.
- Each Keeper Commander API request – command submitted (with sensitive details hidden), HTTP response status, and for v2 async requests, the poll attempt count and each intermediate status value.
- Whether the record was successfully parsed and which credential type (**pamUser** or **sshKeys**) was extracted.
- Escalation credential retrieval details when privilege escalation is configured, including the escalation record UID used.
- Any errors returned by the Keeper Commander service, including HTTP error codes and Keeper Commander-level error messages.

Common Reasons for Credentialed Checks Showing "no" Status

- The Keeper Commander service is unreachable from the scanner because of a firewall, an incorrect Engine URL, or an incorrect port.
- The API Key is invalid, expired, or does not have read access to the target Keeper Commander record.
- The Keeper Commander service has lost its authenticated session to the Keeper Commander cloud. When this happens the service returns a 200 response with **"data":null** and a message such as **"Logging in to Keeper as <email>"** – the API call appears to succeed but contains no credential data. Re-authenticate on the Keeper Commander host by opening an interactive



Keeper Commander shell (**keeper shell**) and running **login <email>** to re-establish the session, then restart the Keeper Commander service.

- The record UID supplied in **Credential ID** does not exist in the vault, or the search term in **Search Text** matches no records.
- The Keeper Commander record type does not match the selected **SSH Private Key** setting – for example, an **sshKeys** record is referenced but SSH Private Key is set to No, causing the integration to look for a password field that is not present in an **sshKeys** record.
- The Keeper Commander record is missing a **login** field, which is required regardless of record type.
- The Keeper Commander service exposes only the v1 API (**/api/v1/executecommand**). The integration requires the v2 async API endpoints.
- SSL certificate verification fails because the Keeper Commander instance uses a self-signed certificate and **Verify SSL Certificate** is enabled. Either add the certificate to the scanner's trust store or disable Verify SSL Certificate.
- The v2 async request timed out – the integration polls for up to 30 seconds. If the Keeper Commander service does not complete the command within that window, credential retrieval fails.

Additional Information

Helpful tips

- Tenable One Vulnerability Management or Tenable Security Center passes the policy and credential values down to Tenable Nessus. These values include the Keeper Commander Engine URL, API Key, Query Mode, and Credential ID or Search Text.
- The Tenable Nessus scanner communicates directly with the Keeper Commander service API, and the service returns the username and password (or SSH private key) required for target authentication.
- The scanner then uses those values for target authentication.



- When using **Search** string mode, make record titles specific enough that the first search result is always the intended record. If titles are ambiguous, switch to **Record UID** mode to retrieve the record unambiguously.
- The integration caches credentials per record UID (or search term) for the duration of the scan. If a record is rotated mid-scan, the cached pre-rotation credential is used for the remainder of that scan.

Testing integration connectivity

You can use curl from the Tenable Nessus scanner host to verify connectivity to the Keeper Commander service and test the v2 async API flow before running a scan. The following commands mirror the three API calls the integration makes for each credential retrieval:

1. Submit the command:

```
ENGINE_URL="https://commander.internal:8080"
API_KEY="<your-api-key>"
RECORD_UID="<keeper-record-uid>"

curl -sS -X POST "${ENGINE_URL}/api/v2/executecommand-async" \
  -H "Content-Type: application/json" \
  -H "api-key: ${API_KEY}" \
  -d "{\"command\": \"get ${RECORD_UID} --unmask --format json\"}"
```

A successful response contains a **request_id** field. A 401 response indicates an invalid API key.

2. Step 2 – poll for completion:

```
REQUEST_ID="<request_id from step 1>"

curl -sS "${ENGINE_URL}/api/v2/status/${REQUEST_ID}" \
  -H "api-key: ${API_KEY}"
```

Repeat until the **status** field is **completed**, **failed**, or **expired**. The integration polls once per second for up to 30 seconds.

3. Step 3 – fetch the result:



```
curl -sS "${ENGINE_URL}/api/v2/result/${REQUEST_ID}" \
-H "api-key: ${API_KEY}"
```

A successful response contains a **data** object with **record_uid** and a **fields** array. Verify that a **login** field and either a **password** or **keyPair** field are present and non-empty.

To test the v1 synchronous endpoint on an older Keeper Commander deployment, use a single POST:

```
curl -sS -X POST "${ENGINE_URL}/api/v1/executecommand" \
-H "Content-Type: application/json" \
-H "api-key: ${API_KEY}" \
-d "{\"command\": \"get ${RECORD_UID} --unmask --format json\"}"
```

If this returns a result directly (without a **request_id**), the deployment supports v1 only and the Tenable integration does not work until the Keeper Commander service is upgraded to support the v2 async API.



Optional Features

Privilege Escalation

You can configure privilege escalation to work with the Keeper Commander integration only within the SSH credential type. You can add privilege escalation while configuring an SSH credentialed scan with the Keeper Commander integration using the **Elevate privileges with** field option, which enables you to select the privilege account type used to gain elevated access to the target machine.

Note: The field is labeled "Elevate privileges with" in the Keeper Commander integration for both Tenable One Vulnerability Management and Tenable Nessus.

The following privilege account types are available for selection:

- Nothing (default)
- .k5login
- Cisco 'enable'
- Dzdo
- Pbrun
- Su
- su+sudo
- sudo
- Checkpoint Gaia 'expert'

When an escalation method other than **Nothing** is selected and an **Escalation Credential ID** is supplied, the integration retrieves a second Keeper record and uses its password field as the escalation (sudo/su) password. If **Escalation Credential ID** is left blank, the integration reuses the primary record UID and the primary record's password serves as the escalation password.

For the **sudo** escalation type specifically, the standard sudo prompt expects the invoking user's own password rather than a separate root account password. In that case, leave **Escalation Credential**



ID blank so the integration reuses the primary record's password – no additional Keeper Commander API call is made.

Each escalation type can be configured with the following additional fields.

Option	Description	Required
Escalation Query Mode	How the integration identifies the escalation record. Select Record UID to retrieve a specific record by UID, or Search string to find the first record whose title matches a keyword. Default: Record UID. Configurable within: .k5login, Cisco 'enable', dzdo, pbrun, su, su+sudo, sudo.	No
Escalation Credential ID	The Keeper Commander record UID of the record that contains the escalation account credentials. The password field of that record is used as the sudo/su password. Only visible when Escalation Query Mode is Record UID . Configurable within: .k5login, Cisco 'enable', dzdo, pbrun, su, su+sudo, sudo.	No
Escalation Search Text	The keyword matched against Keeper Commander record titles to find the escalation record. The first matching record is used. Only visible when Escalation Query Mode is Search string . Configurable within: .k5login, Cisco 'enable', dzdo, pbrun, su, su+sudo, sudo.	No
Escalation search scan target	When enabled, appends the scan target's IP address or hostname to the Escalation Credential ID value at scan time and performs a keyword search instead of a UID lookup. Only	No



	visible when Escalation Query Mode is Record UID . Default: No. Configurable within: .k5login, Cisco 'enable', dzdo, pbrun, su, su+sudo, sudo.	
Location of [account type] (directory)	The directory path for the escalation binary (for example /usr/bin for sudo). Configurable within: dzdo, pbrun, su, su+sudo, sudo.	No