



Linux Pluggable Authentication Module (PAM)

SurePassID Authentication Server

Version 26.06.03+g605aeac

© 2013-2026 SurePassID, Corp. All rights reserved. Protected by patents pending. SurePassID, the SurePassID logo and design, and Secure SSO are registered trademarks or trademarks of SurePassID, Corp. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

SurePassID, Corp.

360 Central Avenue
First Central Tower
Suite 800
St. Petersburg, FL 33701
USA

+1 (888) 200-8144

<https://www.surepassid.com>

Contents

1. About this guide	4
Release history	4
2. Functionality	4
Supported PAM services	5
Application key requirements	5
3. Supported platforms	5
Related (untested) distros	6
Package format and architecture	6
Dependencies	6
4. PAM service configuration	6
Application key	6
Authentication server connection	7
Proxy server	7
Authentication mode	8
First-factor password mode	8
Bypass group	8
Offline codes	9
Push authentication	9
OTP code visibility	9
Second-factor methods	10
Deprecated method options	10
Method-selector values	10
Method-selector messages	11
Custom prompts	11
5. Installation	11
Unattended installation	12
Air-gapped installation	12
Installing on a related (untested) distro	13
Configuring sshd for MFA	13
When is it called automatically?	13
Manual invocation	14
Removal and sshd revert behavior	14
5a. Package Repository Installation	14
RHEL / Rocky Linux / CentOS Stream (dnf)	14
Debian / Ubuntu (apt)	15
5b. Ansible Installation	15
Install the collection	15
Online deployment	15
Air-gapped deployment	16
Customizing the role	16

Verifying the deployment	16
Enterprise mirrors	16
Red Hat Satellite / Foreman+Katello	16
Canonical Landscape	17
Uyuni / SUSE Manager / Spacewalk	18
Using enterprise mirrors with Ansible	18
6. Removing PAM service configurations	18
Runtime dependencies	19
RHEL-family (RHEL, Rocky, CentOS Stream)	19
Debian-family (Debian, Ubuntu)	19
Verifying the closure	19

1. About this guide

This document is a comprehensive guide for integrating SurePassID multi-factor authentication (MFA) into Linux systems using the Pluggable Authentication Module (PAM). It is intended as a reference for system administrators looking to meet their organization's security requirements.

The guide covers:

- **Functionality** — authentication methods and features supported by the PAM module.
- **Supported platforms** — Linux distributions and versions.
- **Configuration** — the configuration file and its options.
- **Installation** — interactive, unattended, air-gapped, and package-repository installation methods.
- **Removal** — disabling and uninstalling the module.
- **Runtime dependencies** — libraries required on the target host.

Release history

Release notes for each version of the PAM module are published at <https://downloads.surepassid.com/PAM/linux/CHANGELOG.md>.

2. Functionality

The SurePassID PAM module supports the following SurePassID multi-factor authentication features:

- Hardware OTP tokens (TOTP or HOTP).
- Software OTP apps (TOTP or HOTP).
- OTP codes via SMS, email, or voice.
- Offline OTP (using HOTP).
- Mobile push authentication.
- Mobile push authentication with FIDO.
- SMS push authentication confirmation.
- Voice push authentication confirmation.
- MFA bypass group — allows designated users to skip MFA (e.g., service accounts). See the `bypass_group_name` option in the configuration file.
- Offline codes — cached HOTP codes for use when the MFA server is unreachable. See the `offline_codes` option in the configuration file.

Supported PAM services

These PAM services in `/etc/pam.d` are supported:

- `sshd`
- `sudo`
- `su`
- `login` (console login)
- `gdm-password` (desktop login and screensaver)
- `lightdm`, `sddm`, `kdm` (alternative desktop managers — wired up by the install script when present)

Application key requirements

The SurePassID PAM module uses the following SurePassID MFA Server application-key access rights. The configuration file indicates which rights are needed for each enabled feature.

Access right	Required by
ValidateOtp	Required for all OTP-based authentication.
ValidateUser	Required when using first-factor authentication (<code>auth_mode_1fa</code> or <code>auth_mode_1fa_2fa</code>).
SendOtpEmail	Required when <code>otp_email</code> or <code>auto_otp_email</code> is enabled.
SendOtpSms	Required when <code>otp_sms</code> or <code>auto_otp_sms</code> is enabled.
SendOtpVoice	Required when <code>otp_voice</code> or <code>auto_otp_voice</code> is enabled.
SendPushApp	Required when <code>push_app</code> or <code>auto_push_app</code> is enabled.
SendPushSms	Required when <code>push_sms</code> or <code>auto_push_sms</code> is enabled.
SendPushVoice	Required when <code>push_voice</code> or <code>auto_push_voice</code> is enabled.

3. Supported platforms

Family	Releases
Debian	Trixie (13.x), Bookworm (12.x), Bullseye (11.x)
Ubuntu	Resolute Raccoon (26.04 LTS), Noble Numbat (24.04 LTS), Jammy Jellyfish (22.04 LTS), Focal Fossa (20.04 ESM)
Red Hat	RHEL 10, RHEL 9, RHEL 8
Enterprise Linux	
Rocky Linux	Rocky 10 (Red Quartz), Rocky 9 (Blue Onyx), Rocky 8 (Green Obsidian)
Oracle Linux	Oracle Linux 10, Oracle Linux 9, Oracle Linux 8
CentOS Stream	CentOS Stream 10, CentOS Stream 9

Both **x86_64 (amd64)** and **aarch64 (arm64)** architectures are supported across every release listed above.

End-of-life notice: CentOS Linux 8 (EOL December 2021) and CentOS Stream 8 (EOL May 2024) are no longer receiving security patches from their upstream vendors. The PAM module may still function on these releases, but they are not actively tested and support is best-effort.

Related (untested) distros

The installer also accepts distributions that declare themselves related to a supported family via `/etc/os-release`'s `ID_LIKE` field (for example AlmaLinux, Oracle Linux, or Linux Mint). Installation is permitted but not part of SurePassID's tested matrix, and the installer will display a notice and require explicit acceptance.

Paid support is available for installs on related distros — please contact SurePassID sales for details. If a customer issue on a related distro can be reproduced on a supported distro, it will be addressed at no extra charge.

Package format and architecture

Each supported release ships as either an RPM (Enterprise Linux families: RHEL, Rocky, Oracle, CentOS Stream) or a DEB (Debian/Ubuntu), built for both **x86_64 (amd64)** and **aarch64 (arm64)**. The package manager selects the correct architecture automatically for the host. Related distributions install via `ID_LIKE` detection using the matching family's package format (see *Related (untested) distros* above).

Dependencies

The PAM module is dynamically linked against a small set of system libraries. They are installed automatically by the package manager when the customer host has internet access; for air-gapped installs the dependencies must be pre-staged.

See the **Runtime dependencies** section for the full per-distro runtime-dependency manifest.

The two functional dependencies are:

- **OpenSSL** — TLS connectivity to the SurePassID MFA server.
- **Jansson** — JSON encode/decode for the API requests.

4. PAM service configuration

The distribution tarball includes a template configuration file named `pam_surepassid.conf_TEMPLATE`. All available options are documented inline in that file. Make a copy, update it for your deployment, and keep it on the target host — the install script will prompt for its path.

After running the install script, a copy of the configuration is placed at `/etc/surepassid.d/pam_surepassid.conf` with `root:root 0640` permissions. The directory `/etc/surepassid.d/` is created with `root:root 0750`.

Inline overrides. Every directive documented below is normally set in the `.conf` file, but it may also be supplied as an inline module argument in `/etc/pam.d/<service>`. Inline arguments are applied *after* the configuration file and override it, allowing per-service overrides (for example, adding `debug` to a single service). See `pam_surepassid(8)`.

Application key

To authenticate to the SurePassID MFA server, the module requires an application key created in the SurePassID Administration Portal.

`application_key_id`

The application key identifier.

(Legacy alias: `sp_account`.)

Required

`application_key`

The application key secret.

(Legacy alias: `sp_account_key`.)

Required

The access rights the key must be granted depend on which features you enable; `ValidateOtp` is always required for second-factor verification. The per-method access rights are listed in the **Second-factor methods** table below.

Authentication server connection

`host`

Hostname of the SurePassID Authentication Server.

Required

`port`

Server port.

Default: 443

`path`

REST API path.

Default: `/AuthServer/REST/OATH/OathServer.aspx`

`connection_timeout_secs`

Connection timeout in seconds; 0 means no timeout. A value of 5 is recommended when offline codes are used.

Default: 0

`ca_cert_file`

Path to a CA-certificates PEM bundle.

Default: The system CA bundle is located automatically.

Proxy server

`proxy_server`

Hostname or IP address of an outbound HTTP proxy.

Optional

`proxy_port`

Proxy port.

Default: 8080

`proxy_username`

Proxy username.

Required if `proxy_password` is set.

proxy_password

Proxy password.

Required if proxy_username is set.

SELinux note. The bundled SELinux policy permits outbound connections on standard HTTP/HTTPS ports (80, 443, 8443) and common proxy ports (3128, 8080, 8118). If your proxy uses a non-standard port and SELinux is enabled, label it so SELinux allows the connection (replace `PORT` with your proxy port):

```
sudo semanage port -a -t squid_port_t -p tcp PORT
```

Use `-m` instead of `-a` if the port is already assigned to another type. `semanage` is provided by `pol` `icycoreutils-python-utils`.

Authentication mode

Selects which factors the module enforces. Default: `auth_mode_2fa`

`auth_mode_1fa`

First-factor (password) authentication only. Requires the `ValidateUser` access right.

`auth_mode_2fa`

Second-factor (OTP/push) authentication only — the default. Requires `ValidateOtp`.

`auth_mode_1fa_2fa`

Both first- and second-factor authentication. Requires `ValidateUser` and `ValidateOtp`.

First-factor password mode

When first-factor authentication is enabled, one password mode may be selected.

Default: `prompt_pass`

`prompt_pass`

Always prompt the user for the password; ignore any password from a previous stacked module.

`try_first_pass`

Try the previous stacked module's password first, then prompt if it does not satisfy this module.

`use_first_pass`

Use the previous stacked module's password and never prompt; if no suitable password is available, access is denied.

`password_otp_delimiter_char`

When both factors are enabled, the character that separates the password and OTP in a single input.

Default: `,`

Bypass group

`bypass_group_name`

Name of an `/etc/group` whose members bypass MFA.

Default: `surepassid-bypass`

Create the group and add users to it (replace `USERNAME` with the account to exempt):

```
sudo groupadd surepassid-bypass
sudo usermod -a -G surepassid-bypass USERNAME
```

Offline codes

Offline codes let users authenticate when the SurePassID server is unreachable. By default, offline codes are **not** used.

`offline_codes`

Enable cached offline codes. The number available equals the token's OTP window size (default 30) minus the reserve.

`offline_codes_reserve`

Number of codes reserved for online use to refresh the cache.

Default: 5

`offline_codes_show_remaining`

Always show the number of remaining offline codes. By default the count is shown only when five or fewer remain.

`allow_single_factor` (**alias** `allow_bypass_mfa_server`)

Disable MFA entirely if the server connection fails. **Not recommended** — use `offline_codes` instead. Disabled by default.

Push authentication

`push_app_name`

Application name shown on the push notification.

Default: Linux

`push_reason`

Reason shown on the push notification.

Default: Login

`max_push_wait_secs`

Maximum seconds to wait for a push response.

Default: 300

The standalone `allow_push_authentication` directive and the `prompt_push` prompt are **deprecated**; configure push via `allowed_2fa_methods` (`push_app / auto_push_app`) instead.

OTP code visibility

By default, OTP verification codes are displayed as the user types them (controlled by the `echo_verification_code` option in the configuration file). To hide codes as they are entered, comment out or remove the `echo_verification_code` line from your configuration file.

Second-factor methods

The `allowed_2fa_methods` option specifies the permitted second-factor authentication methods from the list below. The methods are displayed in the order listed. Required application-key access rights are shown in the right-hand column.

Method	Description	Required access rights
<code>otp_token</code>	OTP from a hard or soft token.	<code>ValidateOtp</code>
<code>otp_sms</code>	OTP sent via SMS.	<code>ValidateOtp</code> , <code>SendOtpSms</code>
<code>otp_email</code>	OTP sent via email.	<code>ValidateOtp</code> , <code>SendOtpEmail</code>
<code>otp_voice</code>	OTP sent via voice call.	<code>ValidateOtp</code> , <code>SendOtpVoice</code>
<code>push_app</code>	Push request sent to the SurePassID Mobile Authenticator app.	<code>SendPushApp</code>
<code>push_sms</code>	SMS confirmation.	<code>SendPushSms</code>
<code>push_voice</code>	Voice confirmation.	<code>SendPushVoice</code>
<code>auto_otp_sms</code>	OTP sent via SMS (automatically, no method prompt).	<code>ValidateOtp</code> , <code>SendOtpSms</code>
<code>auto_otp_email</code>	OTP sent via email (automatically, no method prompt).	<code>ValidateOtp</code> , <code>SendOtpEmail</code>
<code>auto_otp_voice</code>	OTP sent via voice call (automatically, no method prompt).	<code>ValidateOtp</code> , <code>SendOtpVoice</code>
<code>auto_push_app</code>	Push request sent to the SurePassID Mobile Authenticator app.	<code>SendPushApp</code>
<code>auto_push_sms</code>	SMS confirmation (automatic).	<code>SendPushSms</code>
<code>auto_push_voice</code>	Voice confirmation (automatic).	<code>SendPushVoice</code>

Deprecated method options

The following options are deprecated in favor of `allowed_2fa_methods`:

`send_otp_method`

Use the `allowed_2fa_methods` values `otp_sms`, `auto_otp_sms`, `otp_email`, `auto_otp_email`, `otp_voice`, or `auto_otp_voice`.

`allow_push_authentication`

Use the `allowed_2fa_methods` values `push_app` or `auto_push_app`.

Method-selector values

Set the text users type to indicate their chosen authentication method. Values are case-insensitive. The values shown are the standard defaults.

```
otp_sms_value=os
otp_email_value=oe
otp_voice_value=ov
push_app_value=pa
push_sms_value=ps
push_voice_value=pv
```

Method-selector messages

Personalize the message shown when users are asked to choose an authentication method. The defaults are:

```
otp_sms_message="OTP via SMS"
otp_email_message="OTP via Email"
otp_voice_message="OTP via Voice Call"
push_app_message="Push Application Confirmation"
push_sms_message="Push SMS Confirmation"
push_voice_message="Push Voice Call Confirmation"
```

Custom prompts

Custom prompts for authentication methods. A prompt is **not** displayed when an `auto_push_*` authentication method is used.

`prompt_otp_and_method`

Presented when the user is asked to enter an OTP code **or** select an authentication method.

Default: "Enter a SurePassID verification code or select an authentication method
: "

`prompt_method`

Presented when the user is only asked to select an authentication method.

Default: "Select an authentication method: "

`prompt_otp`

Presented when the user is only asked to enter an OTP code.

Default: "SurePassID verification code: "

5. Installation

The installation is performed by running the script `pam_surepassid-install.sh` (shipped inside the distribution tarball) and following the prompts. The script requires superuser privileges via `sudo` and will prompt for your password automatically. You must create a configuration file before running the script (see **PAM service configuration**, above).

Example interactive run:

```
$ ./pam_surepassid-install.sh
Superuser privileges are required.
[sudo] password for your-username:
Installing pam_surepassid.so ...
<<<<< OUTPUT FROM THE INSTALL PROCESS >>>>>
Installation of pam_surepassid.so complete!
Enter path of SurePassID PAM configuration file. [./pam_surepassid.conf]:
Add SurePassID MFA PAM for sshd (recommended)? [y]/n:

!!! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! !!!
!!! WARNING WARNING WARNING WARNING WARNING !!!
!!! The following should only be configured !!!
!!! after testing that sshd works properly. !!!
!!! WARNING WARNING WARNING WARNING WARNING !!!
!!! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! !!!

Add SurePassID MFA PAM for the gdm-password desktop login (optional)? y/[n]:
Add SurePassID MFA PAM for sudo (optional)? y/[n]:
Add SurePassID MFA PAM for su (optional)? y/[n]:
```

After the installation completes, you can test it via SSH:

```
$ ssh user@localhost echo "It worked!"
Password:
Enter a SurePassID verification code or select an authentication method: 123456
It worked!
```

Note: The exact prompt and available authentication methods depend on the `allowed_2fa_methods` setting in your configuration file.

Unattended installation

For automated provisioning, supply the configuration file path and a list of services on the command line. The install script configures only the services you list and exits without prompting:

```
sudo ./pam_surepassid-install.sh \
    /path/to/pam_surepassid.conf \
    sshd sudo
```

You can also pass `--services-file <path>` to read the service list from a file (one service per line).

Air-gapped installation

The install script never reaches out to a generic third-party host to test for internet connectivity. It optionally probes the SurePassID host parsed from your `.conf` (with a 2-second timeout) only to decide whether to *offer* an online dependency refresh — it never blocks the install on the result.

On an air-gapped host, the script will:

1. Skip the optional `dnf upgrade / apt-get update` prompts.
2. List exactly which required packages are missing if any.
3. Proceed to install the bundled `.rpm` or `.deb` once the listed dependencies are satisfied via your local mirror.

See the **Runtime dependencies** section for the dependency manifest you need to pre-stage on the local mirror.

Installing on a related (untested) distro

If your host runs a distribution that is not officially supported but declares itself related to a supported family (via `ID_LIKE` in `/etc/os-release`), the installer will display a notice and require explicit acceptance before proceeding.

To accept non-interactively, use one of:

```
# CLI flag
sudo ./pam_surepassid-install.sh --accept-unsupported my.conf sshd

# Environment variable
export PAM_SUREPASSID_ACCEPT_UNSUPPORTED=1
sudo ./pam_surepassid-install.sh my.conf sshd
```

In non-interactive mode (`PAM_SUREPASSID_NONINTERACTIVE=1` or `stdin` is not a TTY), acceptance is implicit.

See [Supported platforms](#) for the full policy on related distros.

Configuring sshd for MFA

The package ships a utility, `/usr/sbin/pam_surepassid-configure-sshd`, that configures the OpenSSH server for SurePassID MFA. It ensures:

- **KbdInteractiveAuthentication** (or **ChallengeResponseAuthentication** on OpenSSH < 8.7) is set to `yes`.
- If **AuthenticationMethods** is explicitly configured and does not already include `keyboard-interactive`, it appends `, keyboard-interactive` to each method that lacks it.

The script is **idempotent** — safe to run multiple times.

When is it called automatically?

Scenario	Debian/Ubuntu	RHEL/Rocky/CentOS
Package install/upgrade	Automatic (postinst)	Automatic (%post)
openssh-server installed later	Automatic (dpkg trigger)	Manual

Manual invocation

If openssh-server is installed after the PAM module on RHEL-family systems — or any time you need to re-apply the sshd configuration — run:

```
sudo pam_surepassid-configure-sshd
```

The script creates timestamped backups of any files it modifies (e.g., /etc/ssh/sshd_config.ORG_2026-05-04_1430).

Removal and sshd revert behavior

Removing the SurePassID MFA configuration (via `pam_surepassid-remove.sh`, or `pam_surepassid-configure remove-service`) strips the SurePassID `pam_surepassid.so` include from each `/etc/pam.d/<service>` file and deletes `/etc/pam.d/surepassid`.

Removal does **not** revert the changes `pam_surepassid-configure-sshd` made to `/etc/ssh/sshd_config` (such as `KbdInteractiveAuthentication yes` or appended `AuthenticationMethods`). Those edits are harmless without the PAM include, but if you want to restore the original sshd configuration, do so manually from the timestamped backup the configure step created:

```
sudo cp /etc/ssh/sshd_config.ORG_<timestamp> /etc/ssh/sshd_config
sudo systemctl restart sshd
```

5a. Package Repository Installation

As an alternative to the tarball installer, you can install the PAM module directly from the SurePassID package repository using your distribution's package manager. This also enables automatic updates.

RHEL / Rocky Linux / CentOS Stream (dnf)

Import the GPG signing key and create the repo file:

```
sudo rpm --import \
    https://downloads.surepassid.com/PAM/linux/repo/RPM-GPG-KEY-surepassid

sudo tee /etc/yum.repos.d/surepassid.repo > /dev/null <<'EOF'
[surepassid]
name=SurePassID PAM Module - EL$releasever - $basearch
baseurl=https://downloads.surepassid.com/PAM/linux/repo/rpm/el$releasever/$basearch/
enabled=1
gpgcheck=1
gpgkey=https://downloads.surepassid.com/PAM/linux/repo/RPM-GPG-KEY-surepassid
EOF
```

Install the module:

```
sudo dnf install pam_surepassid
```

Debian / Ubuntu (apt)

Download and install the signing keyring:

```
curl -fsSL \
  https://downloads.surepassid.com/PAM/linux/repo/deb/surepassid-archive-keyring.gpg \
  | sudo tee /usr/share/keyrings/surepassid-archive-keyring.gpg > /dev/null
```

Add the repository source (replace <codename> with your release):

Distribution	Codename
Ubuntu 20.04 LTS	focal
Ubuntu 22.04 LTS	jammy
Ubuntu 24.04 LTS	noble
Debian 12	bookworm
Debian 13	trixie

```
echo "deb [signed-by=/usr/share/keyrings/surepassid-archive-keyring.gpg] \
  https://downloads.surepassid.com/PAM/linux/repo/deb <codename> main" \
  | sudo tee /etc/apt/sources.list.d/surepassid.list > /dev/null
```

Install the module:

```
sudo apt-get update
sudo apt-get install libpam-surepassid
```

5b. Ansible Installation

For organizations using Ansible for configuration management, the SurePassID PAM module is available as an Ansible collection on [Ansible Galaxy](#).

The role (`surepassid.linux_pam`) is verified with Molecule across five scenarios: EL8, EL9, Ubuntu, Debian (bookworm), and an air-gapped install.

Install the collection

```
ansible-galaxy collection install surepassid.linux_pam
```

Online deployment

Create an inventory file listing your target hosts, then run the included playbook:

```
ansible-playbook surepassid.linux_pam.deploy \
  -i inventory.yml \
  -e target=all \
  -e pam_surepassid_conf=/path/to/pam_surepassid.conf
```

The role configures the appropriate package repository (yum on EL, apt on Debian/Ubuntu), installs the PAM module, deploys your configuration file, and wires sshd for MFA.

Air-gapped deployment

For hosts without internet access, download the collection tarball and the air-gap ISO mirror from the [GitHub Release](#) page. Transfer both files to your Ansible control node, then:

```
# Install the collection from the local tarball.
ansible-galaxy collection install \
    surepassid-linux_pam-<version>.tar.gz

# Run the playbook with ISO source.
ansible-playbook surepassid.linux_pam.deploy \
    -i inventory.yml \
    -e target=all \
    -e pam_surepassid_conf=/path/to/pam_surepassid.conf \
    -e pam_surepassid_source=iso \
    -e pam_surepassid_iso_path=/path/to/pam_surepassid-airgap.iso
```

The role copies the ISO to each target host, mounts it, and configures a local file-based package repository so no internet access is required.

Customizing the role

The role supports these variables (set via `-e` or in your playbook `vars:`):

Variable	Default	Description
<code>pam_surepassid_conf</code>	(none)	Required. Path to your <code>.conf</code> file.
<code>pam_surepassid_services</code>	[sshd]	PAM services to wire for MFA.
<code>pam_surepassid_source</code>	repo	Install source: <code>repo</code> , <code>tarball</code> , or <code>iso</code> .
<code>pam_surepassid_tarball_path</code>	""	Path to tarball (when source is <code>tarball</code>).
<code>pam_surepassid_iso_path</code>	""	Path to ISO (when source is <code>iso</code>).
<code>pam_surepassid_gpg_verify</code>	true	Verify GPG signatures on packages.
<code>pam_surepassid_repo_base_url</code>	(auto)	Base URL for the SurePassID package repository.

Verifying the deployment

After the playbook completes, verify from the control node:

```
ansible -i inventory.yml all -m command \
    -a "pam_surepassid-configure status" --become
```

Each host should show `sshd` (and any other services you specified) as wired for SurePassID MFA.

Enterprise mirrors

If your organization uses Red Hat Satellite, Canonical Landscape, Foreman+Katello, Uyuni, or Spacewalk, the SurePassID signed package repository is fully compatible with these tools. No SurePassID-specific plugins or content views are required — our repository uses standard `repomd.xml` (RPM) and `Release + Packages.gz` (DEB) metadata signed with a standard GPG key.

Red Hat Satellite / Foreman+Katello

1. Import the SurePassID GPG key:

Navigate to **Content** → **Content Credentials** → **Create Content Credential**. Upload the key from <https://downloads.surepassid.com/PAM/linux/repo/RPM-GPG-KEY-surepassid>.

2. Create a custom product and repository:

- Go to **Content** → **Products** → **Create Product**.
- **Name:** SurePassID PAM Module.
- Add a repository for each EL version you support:

Name	Type	URL
EL 8 x86_64	yum	https://downloads.surepassid.com/PAM/linux/repo/rpm/el8/x86_64/
EL 8 aarch64	yum	https://downloads.surepassid.com/PAM/linux/repo/rpm/el8/aarch64/
EL 9 x86_64	yum	https://downloads.surepassid.com/PAM/linux/repo/rpm/el9/x86_64/
EL 9 aarch64	yum	https://downloads.surepassid.com/PAM/linux/repo/rpm/el9/aarch64/
EL 10 x86_64	yum	https://downloads.surepassid.com/PAM/linux/repo/rpm/el10/x86_64/
EL 10 aarch64	yum	https://downloads.surepassid.com/PAM/linux/repo/rpm/el10/aarch64/

- Set the GPG key to the credential imported above.

3. Sync the product: **Content** → **Products** → **SurePassID PAM Module** → **Sync Now**.

4. Create or update a Content View to include the new repositories, then publish and promote to your lifecycle environments.

5. Hosts in those environments can now install via:

```
sudo dnf install pam_surepassid
```

Canonical Landscape

1. In the Landscape dashboard, navigate to **Repositories** → **Manage Mirrors**.

2. Add a new mirror:

- **URI:** <https://downloads.surepassid.com/PAM/linux/repo/deb>
- **Suite:** your codename (e.g. jammy, noble, bookworm)
- **Components:** main
- **Signing key:** import from <https://downloads.surepassid.com/PAM/linux/repo/deb/surepassid-archive-keyring.gpg>

3. Sync the mirror and assign it to your managed computers.

4. Hosts can now install via:

```
sudo apt-get install libpam-surepassid
```

Uyuni / SUSE Manager / Spacewalk

1. Go to **Software** → **Manage** → **Channels** → **Create Channel**.
2. Add a custom software channel for each EL or DEB repository URL listed in the Satellite table above.
3. Sync the channel from the SurePassID repository URL.
4. Assign the channel to your registered systems.

Using enterprise mirrors with Ansible

When your organization mirrors the SurePassID repository internally, override the repository base URL in the Ansible role so hosts install packages from your mirror instead of the public repository:

```
ansible-playbook surepassid.linux_pam.deploy \
  -i inventory.yml \
  -e target=all \
  -e pam_surepassid_conf=/path/to/pam_surepassid.conf \
  -e
  pam_surepassid_repo_base_url=https://satellite.example.com/pulp/content/MyOrg/Library/custom/SurePassID
```

The role accepts the `pam_surepassid_repo_base_url` variable. When set, it replaces the default `downloads.surepassid.com` URLs in the generated repo files on each target host.

6. Removing PAM service configurations

To remove the SurePassID PAM service configurations, run the `pam_surepassid-remove.sh` script (shipped inside the distribution tarball) and follow the prompts.

Note: This does **not** remove the PAM plugin itself (`pam_surepassid.so`). It only disables it by stripping the include lines from every `/etc/pam.d/<service>` file that currently references it, and by deleting `/etc/pam.d/surepassid`. To remove the package itself, use the host's package manager (`dnf remove pam_surepassid` or `apt remove libpam-surepassid`).

Example:

```
$ ./pam_surepassid-remove.sh
Superuser privileges are required.
[sudo] password for your-username:
=====
/etc/pam.d/sshd
/etc/pam.d/surepassid
=====
Are you sure that you would like to remove the SurePassID MFA PAM configuration? y/[n]:
y
Removing SurePassID MFA PAM configuration from sshd
Removing SurePassID MFA PAM configuration /etc/pam.d/surepassid
SurePassID MFA PAM configuration removed!
```

Runtime dependencies

The SurePassID PAM module is dynamically linked against a small set of system libraries. Customer hosts must have these installed before `pam_surepassid-install.sh` will succeed.

For online installs the customer-facing install script offers to fetch missing packages via `dnf` / `apt`. For air-gapped installs the dependencies must be pre-staged using the host's package manager and a local mirror.

RHEL-family (RHEL, Rocky, CentOS Stream)

Package	Purpose
<code>jansson</code>	JSON parsing for the SurePassID API requests
<code>openssl-libs</code>	TLS for the SurePassID API requests

Installed transitively (already present on a base install):

- `pam`, `glibc`

Note — Related distros. Distributions that declare `ID_LIKE=rhel` or `ID_LIKE=fedora` (e.g. AlmaLinux, Oracle Linux) have identical runtime-dependency requirements. The installer allows these hosts but displays an untested-distro notice.

Debian-family (Debian, Ubuntu)

The published `.deb` declares its dependencies via the package's `Depends:` field, so `apt-get install -f -y` after `dpkg -i` resolves them automatically when online. For reference, the expected runtime closure is:

Package	Purpose
<code>libssl3</code>	TLS — Debian 12, Ubuntu 22.04+
<code>libssl1.1</code>	TLS — Ubuntu 20.04 only
<code>libjansson4</code>	JSON parsing
<code>libpam0g</code>	PAM runtime
<code>libc6</code>	<code>glibc</code>
<code>openssh-server</code>	The most common consumer of the PAM stack

Verifying the closure

```
# RHEL-family
ldd /lib64/security/pam_surepassid.so

# Debian-family
ldd "/lib/${(dpkg-architecture -qDEB_HOST_MULTIARCH)}/security/pam_surepassid.so"
```

Any `not found` in the output means the corresponding package above is missing on this host.