



## ***Red Hat Enterprise Linux Release 9.2 Manual Pages on 'pam\_systemd.8' command***

***\$ man pam\_systemd.8***

PAM\_SYSTEMD(8)            pam\_systemd            PAM\_SYSTEMD(8)

NAME

pam\_systemd - Register user sessions in the systemd login manager

SYNOPSIS

pam\_systemd.so

DESCRIPTION

pam\_systemd registers user sessions with the systemd login manager systemd-logind.service(8), and hence the systemd control group hierarchy.

The module also applies various resource management and runtime parameters to the new session, as configured in the JSON User Records[1] of the user, when one is defined.

On login, this module ? in conjunction with systemd-logind.service ? ensures the following:

1. If it does not exist yet, the user runtime directory /run/user/\$UID is either created or mounted as new "tmpfs" file system with quota applied, and its ownership changed to the user that is logging in.
2. The \$XDG\_SESSION\_ID environment variable is initialized. If auditing is available and pam\_loginuid.so was run before this module (which is highly recommended), the variable is initialized from the auditing session id (/proc/self/sessionid). Otherwise, an independent session counter is used.
3. A new systemd scope unit is created for the session. If this is the

first concurrent session of the user, an implicit per-user slice unit below `user.slice` is automatically created and the scope placed into it. An instance of the system service `user@.service`, which runs the `systemd` user manager instance, is started.

4. The `"$TZ"`, `"$EMAIL"` and `"$LANG"` environment variables are configured for the user, based on the respective data from the user's JSON record (if it is defined). Moreover, any environment variables explicitly configured in the user record are imported, and the `umask`, nice level, and resource limits initialized.

On logout, this module ensures the following:

1. If enabled in `logind.conf(5)` (`KillUserProcesses=`), all processes of the session are terminated. If the last concurrent session of a user ends, the user's `systemd` instance will be terminated too, and so will the user's slice unit.
2. If the last concurrent session of a user ends, the user runtime directory `/run/user/$UID` and all its contents are removed, too.

If the system was not booted up with `systemd` as init system, this module does nothing and immediately returns `PAM_SUCCESS`.

## OPTIONS

The following options are understood:

`class=`

Takes a string argument which sets the session class. The `XDG_SESSION_CLASS` environment variable (see below) takes precedence. One of `"user"`, `"greeter"`, `"lock-screen"` or `"background"`. See `sd_session_get_class(3)` for details about the session class.

`type=`

Takes a string argument which sets the session type. The `XDG_SESSION_TYPE` environment variable (see below) takes precedence. One of `"unspecified"`, `"tty"`, `"x11"`, `"wayland"` or `"mir"`. See `sd_session_get_type(3)` for details about the session type.

`desktop=`

Takes a single, short identifier string for the desktop

environment. The `XDG_SESSION_DESKTOP` environment variable (see below) takes precedence. This may be used to indicate the session desktop used, where this applies and if this information is available. For example: "GNOME", or "KDE". It is recommended to use the same identifiers and capitalization as for `$XDG_CURRENT_DESKTOP`, as defined by the Desktop Entry Specification[2]. (However, note that the option only takes a single item, and not a colon-separated list like `$XDG_CURRENT_DESKTOP`.) See `sd_session_get_desktop(3)` for further details.

`debug[=]`

Takes an optional boolean argument. If yes or without the argument, the module will log debugging information as it operates.

## MODULE TYPES PROVIDED

Only session is provided.

## ENVIRONMENT

The following environment variables are initialized by the module and available to the processes of the user's session:

### `$XDG_SESSION_ID`

A short session identifier, suitable to be used in filenames. The string itself should be considered opaque, although often it is just the audit session ID as reported by `/proc/self/sessionid`. Each ID will be assigned only once during machine uptime. It may hence be used to uniquely label files or other resources of this session.

Combine this ID with the boot identifier, as returned by `sd_id128_get_boot(3)`, for a globally unique identifier.

### `$XDG_RUNTIME_DIR`

Path to a user-private user-writable directory that is bound to the user login time on the machine. It is automatically created the first time a user logs in and removed on the user's final logout. If a user logs in twice at the same time, both sessions will see the same `$XDG_RUNTIME_DIR` and the same contents. If a user logs in once, then logs out again, and logs in again, the directory

contents will have been lost in between, but applications should not rely on this behavior and must be able to deal with stale files. To store session-private data in this directory, the user should include the value of `$XDG_SESSION_ID` in the filename. This directory shall be used for runtime file system objects such as `AF_UNIX` sockets, FIFOs, PID files and similar. It is guaranteed that this directory is local and offers the greatest possible file system feature set the operating system provides. For further details, see the XDG Base Directory Specification[3].

`$XDG_RUNTIME_DIR` is not set if the current user is not the original user of the session.

`$TZ`, `$EMAIL`, `$LANG`

If a JSON user record is known for the user logging in these variables are initialized from the respective data in the record.

The following environment variables are read by the module and may be used by the PAM service to pass metadata to the module. If these variables are not set when the PAM module is invoked but can be determined otherwise they are set by the module, so that these variables are initialized for the session and applications if known at all.

`$XDG_SESSION_TYPE`

The session type. This may be used instead of `type=` on the module parameter line, and is usually preferred.

`$XDG_SESSION_CLASS`

The session class. This may be used instead of `class=` on the module parameter line, and is usually preferred.

`$XDG_SESSION_DESKTOP`

The desktop identifier. This may be used instead of `desktop=` on the module parameter line, and is usually preferred.

`$XDG_SEAT`

The seat name the session shall be registered for, if any.

`$XDG_VTNR`

The VT number the session shall be registered for, if any. (Only

applies to seats with a VT available, such as "seat0")

If not set, pam\_systemd will initialize \$XDG\_SEAT and \$XDG\_VTNR based on the \$DISPLAY variable (if the latter is set).

## SESSION LIMITS

PAM modules earlier in the stack, that is those that come before pam\_systemd.so, can set session scope limits using the PAM context objects. The data for these objects is provided as NUL-terminated C strings and maps directly to the respective unit resource control directives. Note that these limits apply to individual sessions of the user, they do not apply to all user processes as a combined whole. In particular, the per-user user@.service unit instance, which runs the systemd --user manager process and its children, and is tracked outside of any session, being shared by all the user's sessions, is not covered by these limits.

See systemd.resource-control(5) for more information about the resources. Also, see pam\_set\_data(3) for additional information about how to set the context objects.

systemd.memory\_max=

Sets unit MemoryMax=.

systemd.tasks\_max=

Sets unit TasksMax=.

systemd.cpu\_weight=

Sets unit CPUWeight=.

systemd.io\_weight=

Sets unit IOWeight=.

systemd.runtime\_max\_sec=

Sets unit RuntimeMaxSec=.

Example data as can be provided from an another PAM module:

```
pam_set_data(handle, "systemd.memory_max", (void *)"200M", cleanup);
```

```
pam_set_data(handle, "systemd.tasks_max", (void *)"50", cleanup);
```

```
pam_set_data(handle, "systemd.cpu_weight", (void *)"100", cleanup);
```

```
pam_set_data(handle, "systemd.io_weight", (void *)"340", cleanup);
```

```
pam_set_data(handle, "systemd.runtime_max_sec", (void *)"3600", cleanup);
```

## EXAMPLE

Here's an example PAM configuration fragment that allows users sessions to be managed by systemd-logind.service:

```
##%PAM-1.0

auth    sufficient pam_unix.so
-auth   sufficient pam_systemd_home.so

auth    required pam_deny.so

account required pam_nologin.so

-account sufficient pam_systemd_home.so

account sufficient pam_unix.so

account required pam_permit.so

-password sufficient pam_systemd_home.so

password sufficient pam_unix.so sha512 shadow try_first_pass use_authtok

password required pam_deny.so

-session optional pam_keyinit.so revoke

-session optional pam_loginuid.so

-session optional pam_systemd_home.so

-session optional pam_systemd.so

session required pam_unix.so
```

## SEE ALSO

systemd(1), systemd-logind.service(8), logind.conf(5), loginctl(1),  
pam\_systemd\_home(8), pam.conf(5), pam.d(5), pam(8), pam\_loginuid(8),  
systemd.scope(5), systemd.slice(5), systemd.service(5)

## NOTES

### 1. JSON User Records

[https://systemd.io/USER\\_RECORD](https://systemd.io/USER_RECORD)

### 2. Desktop Entry Specification

<https://standards.freedesktop.org/desktop-entry-spec/latest/>

### 3. XDG Base Directory Specification

<https://standards.freedesktop.org/basedir-spec/basedir-spec-latest.html>