homectl

- Description
- Options
- General User Record Properties
- Authentication User Record Properties
- Blob Directory User Record Properties
- Account Management User Record Properties
- Resource Management User Record Properties
- Storage User Record Properties
- LUKS Storage User Record Properties
- Mounting User Record Properties
- CIFS User Record Properties
- Login Behaviour Record Properties
- Commands
- Credentials
- Kernel Command Line
- Exit status
- Examples
- See Also

systemd

homectl

1

homectl

systemd-homed-firstboot.service

Create, remove, change or inspect home directories

homectl

OPTIONS

COMMAND

NAME

Description

`homectl` may be used to create, remove, change or inspect a user's home directory. It's primarily a command interfacing with `systemd-homed.service(8)` which manages home directories of users.

Home directories managed by `systemd-homed.service` are self-contained, and thus include the user's full metadata record in the home's data storage itself, making them easy to migrate between machines. In particular, a home directory describes a matching user record, and every user record managed by `systemd-homed.service` also implies existence and encapsulation of a home directory. The user account and home directory become the same concept.

The following backing storage mechanisms are supported:

- An individual LUKS2 encrypted loopback file for a user, stored in `/home/*.home`. At login the file system contained in this files is mounted, after the LUKS2 encrypted volume has been attached. The user's password is identical to the encryption passphrase of the LUKS2 volume. Access to data without preceding user authentication is thus not possible, even for the system administrator. This storage mechanism provides the strongest data security and is thus recommended.
- Similar, but the LUKS2 encrypted file system is located on regular block device, such as a USB storage stick. In this mode home directories and all data they include are nicely migratable between machines, simply by plugging the USB stick into different systems at different times.
- An encrypted directory using `fscrypt` on file systems that support it (at the moment this is primarily `ext4`), located in `/home/*.homedir`. This mechanism also provides encryption, but substantially weaker than LUKS2, as most file system metadata is unprotected. Moreover it currently does not support changing user passwords once the home directory has been created.
- A `btrfs` subvolume for each user, also located in `/home/*.homedir`. This provides no encryption, but good quota support.
- A regular directory for each user, also located in `/home/*.homedir`. This provides no encryption, but is a suitable fallback available on all machines, even where LUKS2, `fscrypt` or `btrfs` support is not available.
- An individual Windows file share (CIFS) for each user.

Note that `systemd-homed.service` and `homectl` will not manage "classic" UNIX user accounts as created with `useradd(8)` or similar tools. In particular, this functionality is not suitable for managing system users (i.e. users with a UID below 1000) but is exclusive to regular ("human") users.

Note that users/home directories managed via `systemd-homed.service` do not show up in `/etc/passwd` and similar files, they are synthesized via glibc NSS during runtime. They are thus resolvable and may be enumerated via the `getent(1)` tool.

This tool interfaces directly with `systemd-homed.service`, and may execute specific commands on the home directories it manages. Since every home directory managed that way also defines a JSON user and group record these home directories may also be inspected and enumerated via `userdbctl(1)`.

Home directories managed by `systemd-homed.service` are usually in one of two states, or in a transition state between them: when `active` they are unlocked and mounted, and thus accessible to the system and its programs; when `inactive` they are not mounted and thus not accessible. Activation happens automatically at login of the user and usually can only complete after a password (or other authentication token) has been supplied. Deactivation happens after the user fully logged out. A home directory remains active as long as the user is logged in at least once, i.e. has at least one login session. When the user logs in a second time simultaneously the home directory remains active. It is deactivated only after the last of the user's sessions ends.

Options

The following general options are understood (further options that control the various properties of user records managed by `systemd-homed.service` are documented further down):

`--identity=FILE`
Read the user's JSON record from the specified file. If passed as `-` read the user record from standard input. The supplied JSON object must follow the structure documented in JSON User Records. This option may be used in conjunction with the `create` and `update` commands (see below), where it allows configuring the user record in JSON as-is, instead of setting the individual user record properties (see below).

`--json=FORMAT`; `-j`
Controls whether to output the user record in JSON format, if the `inspect` command (see below) is used. Takes one of `pretty`, `short` or `off`. If `pretty` human-friendly whitespace and newlines are inserted in the output to make the JSON data more readable. If `short` all superfluous whitespace is suppressed. If `off` (the default) the user information is not shown in JSON format but in a friendly human-readable formatting instead. The `-j` option picks `pretty` when run interactively and `short` otherwise.

`--export-format=FORMAT`; `-E`; `-EE`
When used with the `inspect` verb in JSON mode (see above) may be used to suppress certain aspects of the JSON user record on output. Specifically, if `stripped` format is used the binding and runtime fields of the record are removed. If `minimal` format is used the cryptographic signature is removed too. If `full` format is used the full JSON record is shown (this is the default). This option is

useful for copying an existing user record to a different system in order to create a similar user there with the same settings. Specifically: `homectl inspect -EE | ssh root@othersystem homectl create -i-` may be used as simple command line for replicating a user on another host. `-E` is equivalent to `-j --export-format=stripped`, `-EE` to `-j --export-format=minimal`. Note that when replicating user accounts user records acquired in `stripped` mode will retain the original cryptographic signatures and thus may only be modified when the private key to update them is available on the destination machine. When replicating users in `minimal` mode, the signature is removed during the replication and thus the record will be implicitly signed with the key of the destination machine and may be updated there without any private key replication.

`--offline`
Do not attempt to update the copy of the user record and blob directory that is embedded inside of the home area. This allows for operation on home areas that are absent, or without needing to authenticate as the user being modified.

`--key-name=`
When used with the `add-signing-key` command, specify or override the name under which to store the public key being added. The specified name can be chosen freely, but must be suffixed with `.public`. If this option is not used the name is derived from the specified filename. If a key is read from standard input this option is mandatory in order to provide a suitable name for the key being added.

`--seize=`
Takes a boolean argument. When used with `create` or `register`, controls whether to strip cryptographic signatures from the provided JSON user records, which has the effect of signing them with the local signing key (`local.public`) instead. If this switch is set to true, added user records hence become locally managed (and thus can be modified locally), while if it is set to false the user records remain managed and owned by its origin (and thus cannot be modified locally). This switch defaults to true for `create` and false for `register`.

`--prompt-new-user`
If used in conjunction with `firstboot` and no regular user account exists on the system so far the tool will interactively query for user information and create an account.

`--prompt-shell=`
Takes a boolean argument. Controls whether to interactively query for a login shell, if `firstboot --prompt-new-user` is used. Defaults to true.

`--prompt-groups=`
Takes a boolean argument. Controls whether to interactively query for auxiliary groups to add the new user to, if `firstboot --prompt-new-user` is used. Defaults to true.

`--chrome=`
Takes a boolean argument. By default the interactive user creation via `firstboot --prompt-new-user` screen will show reverse color "chrome" bars at the top and the bottom of the terminal screen, which may be disabled by setting this option to false.

`--mute-console=`
Takes a boolean argument. If true kernel log output and service manager status output to the system console is temporarily disabled while `firstboot --prompt-new-user` is running, so that its own output is not interrupted. Defaults to false.

`--match=this|other|any|auto`; `-A`; `-N`; `-T`
Option `--match=` takes one of `this`, `other`, `any` or `auto`. Some user record settings can be defined to match only specific machines, or all machines but one, or all machines. With this switch it is possibly to control to which machines to apply the settings appearing on the command line after it. If `this` is specified the setting will only apply to the local system (positive match), if `other` it will apply to all but the local system (negative match), if `any` it will apply to all systems (unless there's a matching positive or negative per-machine setting). If `auto` returns to the default logic: whether a setting applies by default to the local system or all systems depends on the option in question.

Note that only some user record settings can be conditioned like this. This option has no effect on the others and is ignored there. This option may appear multiple times in a single command line to apply settings conditioned by different matches to the same user record. See JSON User Records for details on which settings may be used with such per-machine matching and which ones may not.

`-A` is a shortcut for `--match=any`, `-T` is short for `--match=this` and `-N` is short for `--match=other`.

Here's an example call that sets the storage field to `luks` on the local system, but to `cifs` on all others:

```
# homectl update lennart -T --storage=luks -N --storage=cifs
```

General User Record Properties

The following options control various properties of the user records/home directories that `systemd-homed.service` manages. These switches may be used in conjunction with the `create` and `update` commands for configuring various aspects of the home directory and the user account:

`--real-name=NAME`; `-c` <NAME>
The real name for the user. This corresponds with the GECOS field on classic UNIX NSS records.

`--realm=REALM`
The realm for the user. The realm associates a user with a specific organization or installation, and allows distinguishing users of the same name defined in different contexts. The realm can be any

string that also qualifies as valid DNS domain name, and it is recommended to use the organization's or installation's domain name for this purpose, but this is not enforced nor required. On each system only a single user of the same name may exist, and if a user with the same name and realm is seen it is assumed to refer to the same user while a user with the same name but different realm is considered a different user. Note that this means that two users sharing the same name but with distinct realms are not allowed on the same system. Assigning a realm to a user is optional.

`--alias=NAME,NAME…`
Additional names for the user. Takes one or more valid UNIX user names, separated by commas. May be used multiple times to define multiple aliases. An alias username may be specified wherever the primary user name may be specified, and resolves to the same user record.

`--email-address=EMAIL`
Takes an electronic mail address to associate with the user. On log-in the `$EMAIL` environment variable is initialized from this value.

`--location=TEXT`
Takes location specification for this user. This is free-form text, which might or might not be usable by geo-location applications. Example: `--location="Berlin, Germany"` or `--location="Basement, Room 3a"`

`--birth-date=DATE`
Takes a birth date for the user in ISO 8601 calendar date format (`YYYY-MM-DD`). The earliest representable year is 1900. If an empty string is passed the birth date is reset to unset.

`--icon-name=ICON`
Takes an icon name to associate with the user, following the scheme defined by the Icon Naming Specification.

`--home-dir=PATH`; `-dPATH`
Takes a path to use as home directory for the user. Note that this is the directory the user's home directory is mounted to while the user is logged in. This is not where the user's data is actually stored, see `--image-path=` for that. If not specified, defaults to `/home/$USER`.

`--uid=UID`
Takes a preferred numeric UNIX UID to assign this user. If a user is to be created with the specified UID and it is already taken by a different user on the local system then creation of the home directory is refused. Note though, if after creating the home directory it is used on a different system and the configured UID is taken by another user there, then `systemd-homed` may assign the user a different UID on that system. The specified UID must be outside of the system user range. It is recommended to use the 60001…60513 UID range for this purpose. If not specified, the UID is automatically picked. If the home directory is found to be owned by a different UID when logging in, the home directory and everything underneath it will have its ownership changed automatically before login completes.

Note that changing this option for existing home directories generally has no effect on home directories that already have been registered locally (have a local *binding*), as the UID used for an account on the local system is determined when the home directory is first activated on it, and then remains in effect until the home directory is removed.

Note that users managed by `systemd-homed` always have a matching group associated with the same name as well as a GID matching the UID of the user. Thus, configuring the GID separately is not permitted.

`--member-of=GROUP`; `-G` <GROUP>
Takes a comma-separated list of auxiliary UNIX groups this user shall belong to. Example: `--member-of=wheel` to provide the user with administrator privileges. Note that `systemd-homed` does not manage any groups besides a group matching the user in name and numeric UID/GID. Thus any groups listed here must be registered independently, for example with `groupadd(8)`. Any non-existent groups are ignored. This option may be used more than once, in which case all specified group lists are combined. If the user is currently a member of a group which is not listed, the user will be removed from the group.

`--capability-bounding-set=CAPABILITIES`; `--capability-ambient-set=CAPABILITIES`
These options take a space separated list of process capabilities (e.g. `CAP_WAKE_ALARM`, `CAP_BLOCK_SUSPEND`, …) that shall be set in the capability bounding and ambient sets for all the user's sessions. See `capabilities(7)` for details on the capabilities concept. These options may be used more than once, in which case the specified lists are combined. If the parameter begins with a `~` character the effect is inverted: the specified capability is dropped from the specific set.

`--access-mode=MODE`
Takes a UNIX file access mode written in octal. Configures the access mode of the home directory itself. Note that this is only used when the directory is first created, and the user may change this any time afterwards. Example: `--access-mode=0700`

`--umask=MASK`
Takes the access mode mask (in octal syntax) to apply to newly created files and directories of the user ("umask"). If set this controls the initial umask set for all login sessions of the user, possibly overriding the system's defaults.

`--skel=PATH`
Takes a file system path to a directory. Specifies the skeleton directory to initialize the home directory with. All files and directories in the specified path are copied into any newly create home directory. If not specified, defaults to `/etc/skel/`.

`--shell=SHELL`
Takes a file system path. Specifies the shell binary to execute on terminal logins. If not specified, defaults to `/bin/bash`.

`--setenv=VARIABLE[=VALUE]`
Takes an environment variable assignment to set for all user processes. May be used multiple times to set multiple environment variables. When `=` and <VALUE> are omitted, the value of the

variable with the same name in the program environment will be used.

Note that a number of other settings also result in environment variables to be set for the user, including `--email=`, `--timezone=` and `--language=`.

`--timezone=TIMEZONE`
Takes a time zone location name that sets the timezone for the specified user. When the user logs in the `$TZ` environment variable is initialized from this setting. Example: `--timezone=Europe/Amsterdam` will result in the environment variable `TZ=:Europe/Amsterdam`. (`:` is used intentionally as part of the timezone specification, see `tzset(3)`.)

`--language=LANG`
Takes a comma- or colon-separated list of languages preferred by the user, ordered by descending priority. The `$LANG` and `$LANGUAGE` environment variables are initialized from this value on login, and thus values suitable for these environment variables are accepted here, for example `--language=de_DE.UTF-8`. This option may be used more than once, in which case the language lists are concatenated.

`--default-area=AREA`
Takes a string identifying a home directory "area" to use as default. Areas are secondary home directories within the primary home directory of a user. When logging in a user can specify the area they wish to log into, which ensures that the `$HOME` environment variable is set to `~/Areas/` suffixed by the area name.

For details on the area concept see `pam_systemd_home(8)`. Note that this option just defines the default, which can be overridden at login time.

When this option is specified with an empty string as value any previously declared default area is removed from the user record.

Authentication User Record Properties

`--ssh-authorized-keys=KEYS`
Either takes a SSH authorized key line to associate with the user record or a `@` character followed by a path to a file to read one or more such lines from. SSH keys configured this way are made available to SSH to permit access to this home directory and user record. This option may be used more than once to configure multiple SSH keys.

`--pkcs11-token-uri=URI`
Takes an RFC 7512 PKCS#11 URI referencing a security token (e.g. YubiKey or PIV smartcard) that

shall be able to unlock the user account. The security token URI should reference a security token with exactly one pair of X.509 certificate and private key. A random secret key is then generated, encrypted with the public key of the X.509 certificate, and stored as part of the user record. At login time it is decrypted with the PKCS#11 module and then used to unlock the account and associated resources. See below for an example how to set up authentication with a security token.

Instead of a valid PKCS#11 URI, the special strings `list` and `auto` may be specified. If `list` is passed, a brief table of suitable, currently plugged in PKCS#11 hardware tokens is shown, along with their URIs. If `auto` is passed, a suitable PKCS#11 hardware token is automatically selected (this operation will fail if there isn't exactly one suitable token discovered). The latter is a useful shortcut for the most common case where a single PKCS#11 hardware token is plugged in.

Note that many hardware security tokens implement both PKCS#11/PIV and FIDO2 with the `hmac-secret` extension (for example: the YubiKey 5 series), as supported with the `--fido2-device=` option below. Both mechanisms are similarly powerful, though FIDO2 is the more modern technology. PKCS#11/PIV tokens have the benefit of being recognizable before authentication and hence can be used for implying the user identity to use for logging in, which FIDO2 does not allow. PKCS#11/PIV devices generally require initialization (i.e. storing a private/public key pair on them, see example below) before they can be used; FIDO2 security tokens generally do not required that, and work out of the box.

`--fido2-credential-algorithm=STRING`
Specify COSE algorithm used in credential generation. The default value is `es256`. Supported values are `es256`, `rs256` and `eddsa`.

`es256` denotes ECDSA over NIST P-256 with SHA-256. `rs256` denotes 2048-bit RSA with PKCS#1.5 padding and SHA-256. `eddsa` denotes EDDSA over Curve25519 with SHA-512.

Note that your authenticator may choose not to support some algorithms.

`--fido2-device=PATH`
Takes a path to a Linux `hidraw` device (e.g. `/dev/hidraw1`), referring to a FIDO2 security token implementing the `hmac-secret` extension that shall be able to unlock the user account. A random salt value is generated on the host and passed to the FIDO2 device, which calculates a HMAC hash of the salt using an internal secret key. The result is then used as the key to unlock the user account. The random salt is included in the user record, so that whenever authentication is needed it can be passed to the FIDO2 token again.

Instead of a valid path to a FIDO2 `hidraw` device the special strings `list` and `auto` may be specified. If `list` is passed, a brief table of suitable discovered FIDO2 devices is shown. If `auto` is passed, a suitable FIDO2 token is automatically selected, if exactly one is discovered. The latter is a useful shortcut for the most common case where a single FIDO2 hardware token is plugged in.

Note that FIDO2 devices suitable for this option must implement the `hmac-secret` extension. Most current devices (such as the YubiKey 5 series) do. If the extension is not implemented the device cannot be used for unlocking home directories.

The FIDO2 device may be subsequently removed by setting the device path to an empty string (e.g. `homectl update $USER --fido2-device=""`).

Note that many hardware security tokens implement both FIDO2 and PKCS#11/PIV (and thus may be used with either `--fido2-device=` or `--pkcs11-token-uri=`), for a discussion see above.

`--fido2-with-client-pin=BOOL`
When enrolling a FIDO2 security token, controls whether to require the user to enter a PIN when unlocking the account (the FIDO2 `clientPin` feature). Defaults to `yes`. (Note: this setting is without effect if the security token does not support the `clientPin` feature at all, or does not allow enabling or disabling it.)

`--fido2-with-user-presence=BOOL`
When enrolling a FIDO2 security token, controls whether to require the user to verify presence (tap the token, the FIDO2 `up` feature) when unlocking the account. Defaults to `yes`. (Note: this setting is without effect if the security token does not support the `up` feature at all, or does not allow enabling or disabling it.)

`--fido2-with-user-verification=BOOL`
When enrolling a FIDO2 security token, controls whether to require user verification when unlocking the account (the FIDO2 `uv` feature). Defaults to `no`. (Note: this setting is without effect if the security token does not support the `uv` feature at all, or does not allow enabling or disabling it.)

`--recovery-key=BOOL`
Accepts a boolean argument. If enabled a recovery key is configured for the account. A recovery key is a computer generated access key that may be used to regain access to an account if the password has been forgotten or the authentication token lost. The key is generated and shown on screen, and should be printed or otherwise transferred to a secure location. A recovery key may be entered instead of a regular password to unlock the account.

`--recovery-key-file=PATH`
Attempts to write the generated recovery key to the specified file. The file is created with access mode `0600` and must not exist yet. This option may only be used together with `--recovery-key=yes`. If the file cannot be written, a warning is shown and the account operation still succeeds. The recovery key is always shown on screen.

Blob Directory User Record Properties

`--blob=PATH`; `-b` <PATH>; `--blob=FILENAME=PATH`; `-b` <FILENAME>=<PATH>
Accepts either a directory path, or a file name followed by a file path. If just a directory path is specified, then the user's entire blob directory is replaced the specified path. Note that this replacement is performed before per-file manipulations are applied, which means these per-file manipulations will be applied on top of the specified directory. If a filename and file path are specified, then the single specified blob file will be overwritten with the specified path. If completely blank, the entire blob directory is emptied out (which also resets all previous blob-related flags up to this point). If a filename is specified but the corresponding path is blank, that single file will be deleted from the blob directory. All changes are performed in temporary copies of the specified files in directories, which means that the originals specified on the command line are not modified. See User Record Blob Directories for more information about blob directories.

`--avatar=PATH`; `--login-background=PATH`
Accept a file path. If set, the specified file is used to overwrite the corresponding file in the user's blob directory. If blank, the corresponding file is deleted from the blob directory. Essentially, these options are shortcuts to `--blob=FILENAME=PATH` for the known filenames defined in User Record Blob Directories.

Account Management User Record Properties

`--locked=BOOLEAN`
Takes a boolean argument. Specifies whether this user account shall be locked. If true logins into this account are prohibited, if false (the default) they are permitted (of course, only if authorization otherwise succeeds).

`--not-before=TIMESTAMP`; `--not-after=TIMESTAMP`
These options take a timestamp string, in the format documented in `systemd.time(7)` and configures points in time before and after logins into this account are not permitted.

`--rate-limit-interval=SECS`; `--rate-limit-burst=NUMBER`
Configures a rate limit on authentication attempts for this user. If the user attempts to authenticate more often than the specified number, on a specific system, within the specified time interval authentication is refused until the time interval passes. Defaults to 10 times per 1min.

`--password-hint=TEXT`
Takes a password hint to store alongside the user record. This string is stored accessible only to privileged users and the user itself and may not be queried by other users. Example: `--password-hint="My first pet's name"`.

`--enforce-password-policy=BOOL`; `-P`
Takes a boolean argument. Configures whether to enforce the system's password policy for this user, regarding quality and strength of selected passwords. Defaults to on. `-P` is short for `--enforce-password-policy=no`.

`--password-change-now=BOOL`
Takes a boolean argument. If true the user is asked to change their password on next login.

`--password-change-min=TIME`; `--password-change-max=TIME`; `--password-change-warn=TIME`; `--password-change-inactive=TIME`
Each of these options takes a time span specification as argument (in the syntax documented in `systemd.time(7)`) and configures various aspects of the user's password expiration policy. Specifically, `--password-change-min=` configures how much time has to pass after changing the password of the user until the password may be changed again. If the user tries to change their password before this time passes the attempt is refused. `--password-change-max=` configures how soon after it has been changed the password expires and needs to be changed again. After this time passes logging in may only proceed after the password is changed. `--password-change-warn=` specifies how much earlier than then the time configured with `--password-change-max=` the user is warned at login to change their password as it will expire soon. Finally, `--password-change-inactive=` configures the time which has to pass after the password as expired until the user is not permitted to log in or change the password anymore. Note that these options only apply to password authentication, and do not apply to other forms of authentication, for example PKCS#11-based security token authentication.

Resource Management User Record Properties

`--disk-size=BYTES`
Either takes a size in bytes as argument (possibly using the usual K, M, G, … suffixes for 1024 base values), a percentage value, or the special strings `min` or `max`, and configures the disk space to assign to the user. If a percentage value is specified (i.e. the argument suffixed with `%`) it is taken relative to the available disk space of the backing file system. If specified as `min` assigns the minimal disk space permitted by the constraints of the backing file system and other limits, when specified as `max` assigns the maximum disk space available. If the LUKS2 backend is used this configures the size of the loopback file and file system contained therein. For the other storage backends configures disk quota using the filesystem's native quota logic, if available. If not specified, defaults to 85% of the available disk space for the LUKS2 backend and to no quota for the others.

`--nice=NICE`
Takes the numeric scheduling priority ("nice level") to apply to the processes of the user at login

time. Takes a numeric value in the range -20 (highest priority) to 19 (lowest priority).

`--rlimit=LIMIT=VALUE:VALUE`
Allows configuration of resource limits for processes of this user, see `getrlimit(2)` for details. Takes a resource limit name (e.g. `LIMIT_NOFILE`) followed by an equal sign, followed by a numeric limit. Optionally, separated by colon a second numeric limit may be specified. If two are specified this refers to the soft and hard limits, respectively. If only one limit is specified the setting sets both limits in one.

`--tasks-max=TASKS`
Takes a non-zero unsigned integer as argument. Configures the maximum number of tasks (i.e. threads, where each process is at least one thread) the user may have at any given time. This limit applies to all tasks forked off the user's sessions, even if they change user identity via `su(1)` or a similar tool. Use `--rlimit=LIMIT_NPROC=` to place a limit on the tasks actually running under the UID of the user, thus excluding any child processes that might have changed user identity. This controls the `TasksMax=` setting of the per-user systemd slice unit `user-$UID.slice`. See `systemd.resource-control(5)` for further details.

`--memory-high=BYTES`; `--memory-max=BYTES`
Set a limit on the memory a user may take up on a system at any given time in bytes (the usual K, M, G, … suffixes are supported, to the base of 1024). This includes all memory used by the user itself and all processes they forked off that changed user credentials. This controls the `MemoryHigh=` and `MemoryMax=` settings of the per-user systemd slice unit `user-$UID.slice`. See `systemd.resource-control(5)` for further details.

`--cpu-weight=WEIGHT`; `--io-weight=WEIGHT`
Set CPU and IO scheduling weights of the processes of the user, including those of processes forked off by the user that changed user credentials. Takes a numeric value in the range 1…10000. This controls the `CPUWeight=` and `IOWeight=` settings of the per-user systemd slice unit `user-$UID.slice`. See `systemd.resource-control(5)` for further details.

Storage User Record Properties

`--tmp-limit=BYTES`; `--tmp-limit=PERCENT`; `--dev-shm-limit=BYTES`; `--dev-shm-limit=PERCENT`
Controls the per-user quota on `/tmp/` and `/dev/shm/` that is applied when the user logs in. Takes either an absolute value in bytes (with the usual K, M, G, T suffixes to the base of 1024), or a percentage. In the latter case the limit is applied relative to the size of the respective file system. This limit is only applied if the relevant file system is `tmpfs` and has no effect otherwise. Note that if these options are not used, a default quota might still be enforced (typically 80%.)

`--storage=STORAGE`
Selects the storage mechanism to use for this home directory. Takes one of `luks`, `fscrypt`, `directory`, `subvolume`, `cifs`. For details about these mechanisms, see above. If a new home directory is created and the storage type is not specifically specified, `homed.conf(5)` defines which default storage to use.

`--image-path=PATH`
Takes a file system path. Configures where to place the user's home directory. When LUKS2 storage is used refers to the path to the loopback file, otherwise to the path to the home directory (which may be in `/home/` or any other accessible filesystem). When unspecified defaults to `/home/$USER.home` when LUKS storage is used and `/home/$USER.homedir` for the other storage mechanisms. Not defined for the `cifs` storage mechanism. To use LUKS2 storage on a regular block device (for example a USB stick) pass the path to the block device here. Specifying the path to a directory here when using LUKS2 storage is not allowed. Similar, specifying the path to a regular file or device node is not allowed if any of the other storage backends are used.

`--drop-caches=BOOL`
Automatically flush OS file system caches on logout. This is useful in combination with the fscrypt storage backend to ensure the OS does not keep decrypted versions of the files and directories in memory (and accessible) after logout. This option is also supported on other backends, but should not bring any benefit there. Defaults to off, except if the selected storage backend is fscrypt, where it defaults to on. Note that flushing OS caches will negatively influence performance of the OS shortly after logout.

LUKS Storage User Record Properties

`--fs-type=TYPE`
When LUKS2 storage is used configures the file system type to use inside the home directory LUKS2 container. One of `btrfs`, `ext4`, `xfs`. If not specified `homed.conf(5)` defines which default file system type to use. Note that `xfs` is not recommended as its support for file system resizing is too limited.

`--luks-discard=BOOL`
When LUKS2 storage is used configures whether to enable the `discard` feature of the file system. If enabled the file system on top of the LUKS2 volume will report empty block information to LUKS2 and the loopback file below, ensuring that empty space in the home directory is returned to the backing file system below the LUKS2 volume, resulting in a "sparse" loopback file. This option mostly defaults to off, since this permits over-committing home directories which results in I/O errors if the underlying file system runs full while the upper file system wants to allocate a block. Such I/O errors are generally not handled well by file systems nor applications. When LUKS2

storage is used on top of regular block devices (instead of on top a loopback file) the discard logic defaults to on.

`--luks-offline-discard=BOOL`
Similar to `--luks-discard=`, controls the trimming of the file system. However, while `--luks-discard=` controls what happens when the home directory is active, `--luks-offline-discard=` controls what happens when it becomes inactive, i.e. whether to trim/allocate the storage when deactivating the home directory. This option defaults to on, to ensure disk space is minimized while a user is not logged in.

`--luks-extra-mount-options=OPTIONS`
Takes a string containing additional mount options to use when mounting the LUKS volume. If specified, this string will be appended to the default, built-in mount options. Defaults to "compress=zstd:1,noacl,user_subvol_rm_allowed".

`--luks-cipher=CIPHER`; `--luks-cipher-mode=MODE`; `--luks-volume-key-size=BYTES`; `--luks-pbkdf-type=TYPE`; `--luks-pbkdf-hash-algorithm=ALGORITHM`; `--luks-pbkdf-force-iterations=ITERATIONS`; `--luks-pbkdf-time-cost=SECONDS`; `--luks-pbkdf-memory-cost=BYTES`; `--luks-pbkdf-parallel-threads=THREADS`; `--luks-sector-size=BYTES`
Configures various cryptographic parameters for the LUKS2 storage mechanism. See `cryptsetup(8)` for details on the specific attributes.

Note that `homectl` uses bytes for key size, like `/proc/crypto`, but `cryptsetup(8)` uses bits.

`--auto-resize-mode=`
Configures whether to automatically grow and/or shrink the backing file system on login and logout. Takes one of the strings `off`, `grow`, `shrink-and-grow`. Only applies to the LUKS2 backend currently, and if the btrfs file system is used inside it (since only then online growing/shrinking of the file system is supported). Defaults to `shrink-and-grow`, if LUKS2/btrfs is used, otherwise is off. If set to `off` no automatic shrinking/growing during login or logout is done. If set to `grow` the home area is grown to the size configured via `--disk-size=` should it currently be smaller. If it already matches the configured size or is larger no operation is executed. If set to `shrink-and-grow` the home area is also resized during logout to the minimal size the used disk space and file system constraints permit. This mode thus ensures that while a home area is activated it is sized to the configured size, but while deactivated it is compacted taking up only the minimal space possible. Note that if the system is powered off abnormally or if the user otherwise not logged out cleanly the shrinking operation will not take place, and the user has to re-login/logout again before it is executed again.

`--rebalance-weight=`
Configures the weight parameter for the free disk space rebalancing logic. Only applies to the LUKS2 backend (since for the LUKS2 backend disk space is allocated from a per-user loopback file system instead of immediately from a common pool like the other backends do it). In regular intervals free disk space in the active home areas and their backing storage is redistributed among them, taking the weight value configured here into account. Expects an integer in the range 1…10000, or the special string `off`. If not specified, defaults to 100. The weight is used to scale

free space made available to the home areas: a home area with a weight of 200 will get twice the free space as one with a weight of 100; a home area with a weight of 50 will get half of that. The backing file system will be assigned space for a weight of 20. If set to `off` no automatic free space distribution is done for this home area. Note that resizing the home area explicitly (with `homectl resize` see below) will implicitly turn off the automatic rebalancing. To reenable the automatic rebalancing use `--rebalance-weight=` with an empty parameter.

Mounting User Record Properties

`--nosuid=BOOL`; `--nodev=BOOL`; `--noexec=BOOL`
Configures the `nosuid`, `nodev` and `noexec` mount options for the home directories. By default, `nodev` and `nosuid` are on, while `noexec` is off. For details about these mount options see `mount(8)`.

CIFS User Record Properties

`--cifs-domain=DOMAIN`; `--cifs-user-name=USER`; `--cifs-service=SERVICE`; `--cifs-extra-mount-options=OPTIONS`
Configures the Windows File Sharing (CIFS) domain and user to associate with the home directory/user account, as well as the file share ("service") to mount as directory. The latter is used when `cifs` storage is selected. The file share should be specified in format `//host/share/directory/…`. The directory part is optional — if not specified, the home directory will be placed in the top-level directory of the share. The `--cifs-extra-mount-options=` setting allows specifying additional mount options when mounting the share, see `mount.cifs(8)` for details.

Login Behaviour Record Properties

`--stop-delay=SECS`
Configures the time the per-user service manager shall continue to run after the all sessions of the user ended. The default is configured in `logind.conf(5)` (for home directories of LUKS2 storage located on removable media this defaults to 0 though). A longer time makes sure quick, repetitive logins are more efficient as the user's service manager does not have to be started every time.

`--kill-processes=BOOL`
Configures whether to kill all processes of the user on logout. The default is configured in `logind.conf(5)`.

`--auto-login=BOOL`
Takes a boolean argument. Configures whether the graphical UI of the system should automatically log this user in if possible. Defaults to off. If less or more than one user is marked this way automatic login is disabled.

`--session-launcher=LAUNCHER`
Takes a string argument. Configures the user's preferred session launcher .desktop entry file (i.e. `gnome`, `plasma`, or other names that appear in `/usr/share/xesssions/` or `/usr/share/wayland-sessions`). This is read by the display manager to pick the default session that is launched when the user logs in.

`--session-type=TYPE`
Takes a string argument. Configures the user's preferred session type (i.e. `x11`, `wayland`, and other values accepted by `$XDG_SESSION_TYPE`). This is read by the display manage to pick the default session type the user is logged into.

Commands

The following commands are understood:

`list`
List all home directories (along with brief details) currently managed by `systemd-homed.service`. This command is also executed if none is specified on the command line. (Note that the list of users shown by this command does not include users managed by other subsystems, such as system users or any traditional users listed in `/etc/passwd`.)

`activate` <USER> [<USER…>]
Activate one or more home directories. The home directories of each listed user will be activated and made available under their mount points (typically in `/home/$USER`). Note that any home activated this way stays active indefinitely, until it is explicitly deactivated again (with `deactivate`, see below), or the user logs in and out again and it thus is deactivated due to the automatic deactivation-on-logout logic.

Activation of a home directory involves various operations that depend on the selected storage mechanism. If the LUKS2 mechanism is used, this generally involves: inquiring the user for a password, setting up a loopback device, validating and activating the LUKS2 volume, checking the file system, mounting the file system, and potentially changing the ownership of all included files to the correct UID/GID.

`deactivate` <USER> [<USER…>]
Deactivate one or more home directories. This undoes the effect of `activate`.

`inspect` <USER> [<USER…>]
Show various details about the specified home directories. This shows various information about the home directory and its user account, including runtime data such as current state, disk use and similar. Combine with `--json=` to show the detailed JSON user record instead, possibly combined with `--export-format=` to suppress certain aspects of the output.

`authenticate` <USER> [<USER…>]
Validate authentication credentials of a home directory. This queries the caller for a password (or similar) and checks that it correctly unlocks the home directory. This leaves the home directory in the state it is in, i.e. it leaves the home directory in inactive state if it was inactive before, and in active state if it was active before.

`create` <USER>; `create` `--identity=PATH` [<USER>]
Create a new home directory/user account of the specified name. Use the various user record property options (as documented above) to control various aspects of the home directory and its user accounts.

The specified user name should follow the strict syntax described on User/Group Name Syntax.

`adopt` <PATH> [<PATH>…]
Adopts one or more existing home directories on the local system. Takes one or more paths to `*.home` LUKS home directories or `*.homedir/` standalone home directories or subvolumes previously created by `systemd-homed` and makes them available locally for login. The referenced files are not moved. This is an alternative for moving such home directories into `/home/` (where they would be picked up automatically).

`register` <FILE> [<FILE>…]
Registers one or more users, without creating their home directories. Takes one or more paths to JSON user record files. If the path is specified as `-` reads the JSON user record from standard input.

Registering a user makes it accessible on the local system without creating a new home directory. This is particularly useful for making a user accessible on a system it was not originally created on.

Here's an example how to make a local user account with its home directory accessible on a remote system, using SMB/CIFS file sharing. With Samba installed in its default configuration invoke as `root`:

```
# smbpasswd -a lennart
```

Continue as regular user `lennart`:

```
$ homectl update lennart --ssh-authorized-keys=… -N --storage=cifs --cifs-
service="//$HOSTNAME/lennart"
$ homectl get-signing-key | ssh targetsystem homectl add-signing-key --key-
name="$HOSTNAME".public
$ homectl inspect -E lennart | ssh targetsystem homectl register -
$ ssh lennart@targetsystem
```

This first ensures the user account `lennart` is known to and accessible by Samba. It then registers a local SSH access that shall be used for accessing this user, and configures CIFS as default storage for non-local systems on the account. It then adds the local system's account signing key to the target system. Then it registers the local user account with the target system. Finally it logs into the account on the target system. The target system will then connect back via SMB/CIFS to access the home directory.

`unregister` <USER>…
Unregisters one or more user accounts. This only removes the user record from the local system, it does not delete the home directory. The home directory can be readded via the `register` or `adopt` command later, on this or another system. Note that unregistering a user whose home directory is placed in `/home/` will not make the user disappear from the local user database, as all supported home directories placed there will show up in the user database. However, the user record will become "unfixated", i.e. lose its binding to the local system. When logged into it will automatically regain the binding, and acquire a local UID/GID pair.

`remove` <USER>
Remove a home directory/user account. This will remove both the home directory's user record and the home directory itself, and thus delete all files and directories owned by the user.

`update` <USER>; `update` `--identity=PATH` [<USER>]
Update a home directory/user account. Use the various user record property options (as documented above) to make changes to the account, or alternatively provide a full, updated JSON user record via the `--identity=` option.

Note that changes to user records not signed by a cryptographic private key available locally are not permitted, unless `--identity=` is used with a user record that is already correctly signed by a recognized private key.

`passwd` <USER>
Change the password of the specified home directory/user account.

`resize` <USER> <BYTES>
Change the disk space assigned to the specified home directory. If the LUKS2 storage mechanism is used this will automatically resize the loopback file and the file system contained within. Note that if `ext4` is used inside of the LUKS2 volume, it is necessary to deactivate the home directory before shrinking it (i.e the user has to log out). Growing can be done while the home directory is active. If `xfs` is used inside of the LUKS2 volume the home directory may not be shrunk

whatsoever. On all three of `ext4`, `xfs` and `btrfs` the home directory may be grown while the user is logged in, and on the latter also shrunk while the user is logged in. If the `subvolume`, `directory`, `fscrypt` storage mechanisms are used, resizing will change file system quota. The size parameter may make use of the usual suffixes B, K, M, G, T (to the base of 1024). The special strings `min` and `max` may be specified in place of a numeric size value, for minimizing or maximizing disk space assigned to the home area, taking constraints of the file system, disk usage inside the home area and on the backing storage into account.

`lock` <USER>
Temporarily suspend access to the user's home directory and remove any associated cryptographic keys from memory. Any attempts to access the user's home directory will stall until the home directory is unlocked again (i.e. re-authenticated). This functionality is primarily intended to be used during system suspend to make sure the user's data cannot be accessed until the user re-authenticates on resume. This operation is only defined for home directories that use the LUKS2 storage mechanism.

`unlock` <USER>
Resume access to the user's home directory again, undoing the effect of `lock` above. This requires authentication of the user, as the cryptographic keys required for access to the home directory need to be reacquired.

`lock-all`
Execute the `lock` command on all suitable home directories at once. This operation is generally executed on system suspend (i.e. by `systemctl suspend` and related commands), to ensure all active user's cryptographic keys for accessing their home directories are removed from memory.

`deactivate-all`
Execute the `deactivate` command on all active home directories at once. This operation is generally executed on system shut down (i.e. by `systemctl poweroff` and related commands), to ensure all active user's home directories are fully deactivated before `/home/` and related file systems are unmounted.

`with` <USER> <COMMAND…>
Activate the specified user's home directory, run the specified command (under the caller's identity, not the specified user's) and deactivate the home directory afterwards again (unless the user is logged in otherwise). This command is useful for running privileged backup scripts and such, but requires authentication with the user's credentials in order to be able to unlock the user's home directory.

`rebalance`
Rebalance free disk space between active home areas and the backing storage. See `--rebalance-weight=` above. This executes no operation unless there's at least one active LUKS2 home area that has disk space rebalancing enabled. This operation is synchronous: it will only complete once disk space is rebalanced according to the rebalancing weights. Note that rebalancing also takes place automatically in the background in regular intervals. Use this command to synchronously ensure disk space is properly redistributed before initiating an operation requiring large amounts of disk

space.

`firstboot`
This command is supposed to be invoked during the initial boot of the system. It checks whether any regular home area exists so far, and if not queries the user interactively on the console for user name and password and creates one (only if `--prompt-new-user` is specified). Alternatively, if one or more service credentials whose name starts with `home.create.` are passed to the command (containing a user record in JSON format) these users are automatically created at boot.

This command is invoked by the `systemd-homed-firstboot.service` service unit.

`list-signing-keys`
Show a list of public keys that home directories can be signed with to be allowed for local login. One such key (`local.public`) will be generated automatically for signing locally created home directories, but additional public keys may be registered to accept home directories from other origins too (see `add-signing-key` below).

`get-signing-key` [<NAME...>]
Write the public key identified by the specified name to standard output (in PEM format). If no name is specified defaults to `local.public`, i.e. the automatically generated key for locally created home directories.

`add-signing-key` [<FILE...>]
Add public key(s) from the specified PEM key file(s) to the list of keys that home areas have to be signed by to be permitted for local login. If a path of `-` is specified, or if no file is specified at all, the key will be read from standard input. The key file name(s) must carry the `.public` suffix, and the file name(s) will be used to name the key(s) once added, too. If a key is added from standard input the key name must be specified explicitly via `--key-name=`, see above.

This command is useful for permitting local home directories to be used on a remote system. Example:

```
homectl get-signing-key | ssh myotherhost homectl add-signing-key --key-
name="$HOSTNAME".public
```

`remove-signing-key` <NAME...>
Remove the public key identified by the specified name from the list of keys that control from which origins to permit home directories for login.

Credentials

When invoked with the `firstboot` command, `homectl` supports the service credentials logic as implemented by `ImportCredential=`/`LoadCredential=`/`SetCredential=` (see `systemd.exec(5)` for

details). The following credentials are used when passed in:

`home.create.*`
If one or more credentials whose names begin with `home.create.`, followed by a valid UNIX username are passed, a new home area is created, one for each specified user record.

Kernel Command Line

`systemd.firstboot=`
This boolean will disable the effect of `homectl firstboot` command. It's primarily interpreted by `systemd-firstboot(1)`.

Exit status

On success, 0 is returned, a non-zero failure code otherwise.

When a command is invoked with `with`, the exit status of the child is propagated. Effectively, `homectl` will exit without error if the command is successfully invoked *and* finishes successfully.

Examples

Create a user `waldo` in the administrator group `wheel`, and assign 500 MiB disk space to them.

```
homectl create waldo --real-name="Waldo McWaldo" -G wheel --disk-size=500M
```

Create a user `wally` on a USB stick, and assign a maximum of 500 concurrent tasks to them.

```
homectl create wally --real-name="Wally McWally" --image-path=/dev/disk/by-id/usb-
SanDisk_Ultra_Fit_476fff954b2b5c44-0:0 --tasks-max=500
```

Change nice level of user `odlaw` to +5 and make sure the environment variable `$SOME` is set to the string `THING` for them on login.

```
homectl update odlaw --nice=5 --setenv=SOME=THING
```

Set up authentication with a YubiKey security token using PKCS#11/PIV:

```
# Clear the Yubikey from any old keys (careful!)
ykman piv reset

# Generate a new private/public key pair on the device, store the public key in 'pubkey.pem'.
ykman piv generate-key -a RSA2048 9d pubkey.pem

# Create a self-signed certificate from this public key, and store it on the device.
ykman piv generate-certificate --subject "Knobelei" 9d pubkey.pem

# We do not need the public key on disk anymore
rm pubkey.pem

# Allow the security token to unlock the account of user 'lafcadio'.
homectl update lafcadio --pkcs11-token-uri=auto
```

Set up authentication with a FIDO2 security token:

```
# Allow a FIDO2 security token to unlock the account of user 'nihilbaxter'.
homectl update nihilbaxter --fido2-device=auto
```

Add a recovery key to an existing user account:

```
# Generate and add a recovery key for user 'emily'.
homectl update emily --recovery-key=yes
```

See Also

`systemd(1)`, `systemd-homed.service(8)`, `homed.conf(5)`, `userdbctl(1)`, `useradd(8)`, `cryptsetup(8)`

□□□ #1
□ □□□ □□□ 2026-10-31 14:46:30 UTC
□ □□□ □□□ 2026-10-31 14:46:31 UTC