userdbctl

- Description
- Options
- Commands
- Well-Known Services
- Integration with SSH
- Exit status
- See Also

systemd

userdbctl

1

userdbctl

Inspect users, groups and group memberships

userdbctl

OPTIONS

COMMAND

NAME

Description

`userdbctl` may be used to inspect user and groups (as well as group memberships) of the system. This client utility inquires user/group information provided by various system services, both operating on JSON user/group records (as defined by the JSON User Records and JSON Group Records definitions), and classic UNIX NSS/glibc user and group records. This tool is primarily a

client to the User/Group Record Lookup API via Varlink, and may also pick up drop-in JSON user and group records from `/etc/userdb/`, `/run/userdb/`, `/run/host/userdb/`, `/usr/lib/userdb/` (see `nss-systemd(8)` for details about drop-in user records).

Options

The following options are understood:

`--output=MODE`
Chooses the output mode. Takes one of `classic`, `friendly`, `table` or `json`. If `classic`, an output very close to the format of `/etc/passwd` or `/etc/group` is generated. If `friendly`, a more comprehensive and user friendly, human-readable output is generated. If `table`, a minimal, tabular output is generated. If `json`, a JSON formatted output is generated. Defaults to `friendly` if a user/group is specified on the command line, `table` otherwise.

Note that most output formats do not show all available information. In particular, `classic` and `table` show only the most important fields. Various modes also do not show password hashes. Use `json` to view all fields, including any authentication fields.

`--json=FORMAT`
Selects JSON output mode (like `--output=json`) and selects the precise display mode. Takes one of `pretty` or `short`. 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.

`--service=SERVICE:SERVICE…`; `-s` <SERVICE>:<SERVICE…>
Controls which services to query for users/groups. Takes a list of one or more service names, separated by `:`. See below for a list of well-known service names. If not specified, all available services are queried at once.

`--with-nss=BOOL`
Controls whether to include classic glibc/NSS user/group lookups in the output. If `--with-nss=no` is used, any attempts to resolve or enumerate users/groups provided only via glibc NSS is suppressed. If `--with-nss=yes` is specified, such users/groups are included in the output (which is the default).

`--with-varlink=BOOL`
Controls whether to include Varlink user/group lookups in the output, i.e. those done via the User/Group Record Lookup API via Varlink. If `--with-varlink=no` is used, any attempts to resolve or enumerate users/groups provided only via Varlink are suppressed. If `--with-varlink=yes` is specified, such users/groups are included in the output (which is the default).

`--with-dropin=BOOL`
Controls whether to include user/group lookups in the output that are defined using drop-in files in

`/etc/userdb/`, `/run/userdb/`, `/run/host/userdb/`, `/usr/lib/userdb/`. If `--with-dropin=no` is used, these records are suppressed. If `--with-dropin=yes` is specified, such users/groups are included in the output (which is the default).

`--synthesize=BOOL`
Controls whether to synthesize records for the root and nobody users/groups if they are not defined otherwise, as well as the user/groups for the "foreign" UID range. By default (or with `yes`), such records are implicitly synthesized if otherwise missing since they have special significance to the OS. When `no`, this synthesizing is turned off.

`-N`
This option is short for `--with-nss=no` `--synthesize=no`. Use this option to show only records that are natively defined as JSON user or group records, with all NSS/glibc compatibility and all implicit synthesis turned off.

`--multiplexer=BOOL`
Controls whether to do lookups via the multiplexer service (if specified as true, the default) or do lookups in the client (if specified as false). Using the multiplexer service is typically preferable, since it runs in a locked down sandbox.

`--chain`
When used with the `ssh-authorized-keys` command, this will allow passing an additional command line after the user name that is chain executed after the lookup completed. This allows chaining multiple tools that show SSH authorized keys.

`--fuzzy`; `-z`
When used with the `user` or `group` command, do a fuzzy string search. Any specified arguments will be matched against the user name, the real name of the user record, the email address, and other descriptive strings of the user or group record. Moreover, instead of precise matching, a substring match or a match allowing slight deviations in spelling is applied.

`--disposition=`
When used with the `user` or `group` command, filters by disposition of the record. Takes one of `intrinsic`, `system`, `regular`, `dynamic`, `container`. May be used multiple times, in which case only users matching any of the specified dispositions are shown.

`-I`; `-S`; `-R`
Shortcuts for `--disposition=intrinsic`, `--disposition=system`, `--disposition=regular`, respectively.

`--uid-min=`; `--uid-max=`
When used with the `user` or `group` command, filters the output by UID/GID ranges. Takes numeric minimum or maximum UID/GID values, respectively. Shows only records within the specified range. When applied to the `user` command, it matches against UIDs. When applied to the `group` command, matches against GIDs (despite the name of the switch). If unspecified, defaults to 0 (for the minimum) and 4294967294 (for the maximum), i.e. by default, no filtering is applied, as the whole UID/GID range is covered.

`--uuid=`
When used with the `user` or `group` command, filters the output to the record with the specified UUID. If unspecified, no UUID-based filtering is applied.

`--boundaries=`
When used with the `user` or `group` command, controls whether to show relevant UID/GID range boundary information in the tabular output. Takes a boolean. Defaults to true.

`-B`
Shortcut for `--boundaries=no`.

`--from-file=PATH`; `-F`
When used with the `user` or `group` command, read the user definition in JSON format from the specified file, instead of querying it from the system. If the path is specified as `-`, reads the JSON data from standard input. This is useful to validate and introspect JSON user or group records quickly, and check how they would be interpreted on the local system.

Commands

The following commands are understood:

`user` [<USER>…]
List all known users records or show details of one or more specified user records. Use `--output=` to tweak output mode.

If used in conjunction with `--from-file=` the user record data is read in JSON format from the specified file instead of querying it from the system. For details see above.

`group` [<GROUP>…]
List all known group records or show details of one or more specified group records. Use `--output=` to tweak the output mode.

If used in conjunction with `--from-file=` the group record data is read in JSON format from the specified file instead of querying it from the system. For details see above.

`users-in-group` [<GROUP>…]
List users that are members of the specified groups. If no groups are specified, list all user/group memberships defined. Use `--output=` to tweak the output mode.

`groups-of-user` [<USER>…]
Lists groups that the specified users are members of. If no users are specified, lists all user/group memberships defined (in this case, `groups-of-user` and `users-in-group` are equivalent). Use `--output=` to tweak the output mode.

`services`
List all services currently providing user/group definitions to the system. See below for a list of well-known services providing user information.

`ssh-authorized-keys`
Show SSH authorized keys for this account. This command is intended to be used to allow the SSH daemon to pick up authorized keys from user records, see below.

`load-credentials`
When specified, the following credentials are used when passed in:

`userdb.user.*`; `userdb.group.*`; `userdb.transient.user.*`; `userdb.transient.group.*`
These credentials should contain valid JSON User and JSON Group records. For `userdb.user.*` and `userdb.group.*` credentials, records are placed in `/etc/userdb/`, while for `userdb.transient.user.*` and `userdb.transient.group.*` credentials they are placed in `/run/userdb/`. These records implement the interface described in `nss-systemd(8)`. Any passed user records must contain valid UID and GID fields. Any passed group records must contain a GID field (i.e. automatic UID/GID allocation is not supported). For both user and group records, the credential suffix (for `userdb.user.foobar` the suffix is `foobar`) must match the user or group name encoded in the record.

Well-Known Services

The `userdbctl services` command will list all currently running services that provide user or group definitions to the system. The following well-known services are shown among this list:

`io.systemd.DynamicUser`
This service is provided by the system service manager itself (i.e. PID 1) and makes all users (and their groups) synthesized through the `DynamicUser=` setting in service unit files available to the system (see `systemd.exec(5)` for details about this setting).

`io.systemd.Home`
This service is provided by `systemd-homed.service(8)` and makes all users (and their groups) belonging to home directories managed by that service available to the system.

`io.systemd.Machine`
This service is provided by `systemd-machined.service(8)` and synthesizes records for all users/groups used by a container that employs user namespacing.

`io.systemd.Multiplexer`
This service is provided by `systemd-userdbd.service(8)` and multiplexes user/group look-ups to all other running lookup services. This is the primary entry point for user/group record clients, as it simplifies client side implementation substantially, since they can ask a single service for lookups

instead of asking all running services in parallel. `userdbctl` uses this service preferably, too, unless `--with-nss=` or `--service=` are used, in which case finer control over the services to talk to is required.

`io.systemd.NameServiceSwitch`
This service is (also) provided by `systemd-userdbd.service(8)` and converts classic NSS/glibc user and group records to JSON user/group records, providing full backwards compatibility. Use `--with-nss=no` to disable this compatibility, see above. Note that compatibility is actually provided in both directions: `nss-systemd(8)` will automatically synthesize classic NSS/glibc user/group records from all JSON user/group records provided to the system, thus using both APIs is mostly equivalent and provides access to the same data, however the NSS/glibc APIs necessarily expose a more reduced set of fields only.

`io.systemd.DropIn`
This service is (also) provided by `systemd-userdbd.service(8)` and picks up JSON user/group records from `/etc/userdb/`, `/run/userdb/`, `/run/host/userdb/`, `/usr/lib/userdb/`.

Note that `userdbctl` has internal support for NSS-based lookups too. This means that if neither `io.systemd.Multiplexer` nor `io.systemd.NameServiceSwitch` are running, look-ups into the basic user/group databases will still work.

Integration with SSH

The `userdbctl` tool may be used to make the list of SSH authorized keys possibly contained in a user record available to the SSH daemon for authentication. For that, configure the following in `sshd_config(5)`:

```
…
AuthorizedKeysCommand /usr/bin/userdbctl ssh-authorized-keys %u
AuthorizedKeysCommandUser root
…
```

Sometimes, it is useful to allow chain invocation of another program to list SSH authorized keys. By using the `--chain` option, such a tool may be chain executed by `userdbctl ssh-authorized-keys` once a lookup completes, regardless of whether an SSH key was found or not. Example:

```
…
AuthorizedKeysCommand /usr/bin/userdbctl ssh-authorized-keys %u --chain /usr/bin/othertool %u
AuthorizedKeysCommandUser root
…
```

The above will first query the userdb database for SSH keys, and then chain execute `/usr/bin/othertool` to also be queried.

Exit status

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

See Also

`systemd(1)`, `systemd-userdbd.service(8)`, `systemd-homed.service(8)`, `nss-systemd(8)`, `getent(1)`

版本号 #1

由 壳壳虫 创建于 2026-10-31 15:08:41 UTC

由 壳壳虫 更新于 2026-10-31 15:08:41 UTC