Rocky Enterprise Linux 9.2 Manual Pages on command 'udisks.8'

$ man udisks.8

UDISKS(8) System Administration UDISKS(8)

NAME

udisks - Disk Manager

DESCRIPTION

udisks provides interfaces to enumerate and perform operations on disks and storage devices. Any application (including unprivileged ones) can access the udisksd(8) daemon via the name org.freedesktop.UDisks2 on the system message bus[1]. In addition to the D-Bus API, a library, libudisks2 is also provided. This library can be used from C/C++ and any high-level language with GObjectIntrospection[2] support such as Javascript and Python. udisks is only indirectly involved in what devices and objects are shown in the user interface.

ACCESS CONTROL

By default, logged-in users in active log-in sessions are permitted to perform operations (for example, mounting, unlocking or modifying) on devices attached to the seat their session is on. Access-control is fine-grained and based on polkit(8), see the ?Authorization Checks? chapter in the udisks documentation for more information. Note that the

x-udisks-auth option can be used in the /etc/fstab and /etc/crypttab files to specify that additional authorization is required to mount resp. unlock the device (typically requiring the user to authenticate as an administrator).

DRIVE CONFIGURATION

At start-up and when a drive is connected, udisksd(8) will apply configuration stored in the file /etc/udisks2/IDENTIFIER.conf where IDENTIFIER is the value of the Drive:Id property for the drive. If the file changes on disk its new contents will also be applied to the drive. Typically, users or administrators will never need to edit drive configuration files as they are effectively managed through graphical applications such as gnome-disks(1). Manually editing configuration files is however supported ? the file format is a simple .ini-like format (see the Desktop Entry Specification[3] for the exact syntax). New groups and keys may be added in the future.

ATA group

The ATA group is for settings that apply to drives using the ATA command-set. The following keys are supported:

StandbyTimeout

The standby timeout. A value of zero means "timeouts are disabled": the device will not automatically enter standby mode. Values from 1 to 240 specify multiples of 5 seconds, yielding timeouts from 5 seconds to 20 minutes. Values from 241 to 251 specify from 1 to 11 units of 30 minutes, yielding timeouts from 30 minutes to 5.5 hours. A value of 252 signifies a timeout of 21 minutes. A value of 253 sets a vendor-defined timeout period between 8 and 12 hours, and the value 254 is reserved. 255 is interpreted as 21 minutes plus 15 seconds. Note that some older drives may have very different interpretations of these values. This is similar to the -S option in hdparm(8).

APMLevel

The Advanced Power Management level. A low value means aggressive power management and a high value means better performance.

Possible settings range from values 1 through 127 (which permit spin-down), and values 128 through 254 (which do not permit spin-down). The highest degree of power management is attained with a setting of 1, and the highest I/O performance with a setting of 254. A value of 255 can be used to disable Advanced Power Management altogether on the drive (not all drives support disabling it, but most do). This is similar to the -B option in hdparm(8).

AAMLevel

The Automatic Acoustic Management level. Most modern harddisk drives have the ability to speed down the head movements to reduce their noise output. The possible values are between 0 and 254. 128 is the most quiet (and therefore slowest) setting and 254 the fastest (and loudest). Some drives have only two levels (quiet / fast), while others may have different levels between 128 and 254. At the moment, most drives only support 3 options, off, quiet, and fast. These have been assigned the values 0, 128, and 254 at present, respectively, but integer space has been incorporated for future expansion, should this change. This is similar to the -M option in hdparm(8).

WriteCacheEnabled

A boolean specifying whether to enable or disable the Write Cache. Valid values for this key are ?true? and ?false?. This is similar to the -W option in hdparm(8). This key was added in 2.1.

ReadLookaheadEnabled

A boolean specifying whether to enable or disable the Read Look-ahead. Valid values for this key are ?true? and ?false?. This is similar to the -A option in hdparm(8). This key was added in 2.6.0.

DEVICE INFORMATION

udisks relies on recent versions of udev(7) and the Linux kernel. Influential device properties in the udev database include:

UDISKS_SYSTEM

If set, this overrides the value of the HintSystem property.

UDISKS_IGNORE

If set, this overrides the value of the HintIgnore property.

UDISKS_AUTO

If set, this overrides the value of the HintAuto property.

UDISKS_CAN_POWER_OFF

If set, this overrides the value of the CanPowerOff property.

UDISKS_NAME

The name to use for the device when presenting it in an user interface. This corresponds to the HintName property.

UDISKS_ICON_NAME

The icon to use for the device when presenting it in an user interface. If set, the name must adhere to the freedesktop.org icon theme specification[4]. This corresponds to the HintIconName property.

UDISKS_SYMBOLIC_ICON_NAME

The icon to use for the device when presenting it in an user interface using a symbolic icon. If set, the name must adhere to the freedesktop.org icon theme specification[4]. This corresponds to the HintSymbolicIconName property.

UDISKS_FILESYSTEM_SHARED

If set to 1, the filesystem on the device will be mounted in a shared directory (e.g. /media/VolumeName) instead of a private directory (e.g. /run/media/$USER/VolumeName) when the Filesystem.Mount() method is handled.

ID_SEAT

The physical seat the device is attached to. If unset or set to the empty string, ?seat0? (the first seat) is assumed.

API STABILITY

udisks guarantees a stable D-Bus API within the same major version and this guarantee also extends to the client-side library libudisks2. Additionally, several major versions of udisks can be installed and operate at the same time although interoperability may be limited - for

example, a device mounted using the udisks N.x API may require additional authorization if attempting to unmount it through the the (N-1).x API.

The udisks developers do not anticipate breaking API but does reserve the right to do so and if it happens, promises to bump the major version and ensure the new major version of udisks is parallel-installable with any older major version. However, note that programs, man pages and other artifacts may change name (for example, adopt a ?2? suffix) to make room for the next major version. Therefore, applications can not rely on tools like e.g. udisksctl(1) to be available. Additionally, there is no guarantee that the options, command-line switches etc. of command-line tools or similar will remain stable.

Instead, applications should only use the D-Bus API, the libudisks2 library or tools such as dbus-send(1) or gdbus(1) to interact with udisksd(8).

AUDIENCE

The intended audience of udisks include operating system developers working on the higher-level parts of the operating system, for example the desktop shell (such as GNOME[5]) and disk management applications (e.g. GNOME's Disks[6] application). Software on this level typically depend on a specific (major) version of udisks and may even have support for previous versions of udisks or alternative interfaces performing the same role as udisks.

While udisks indeed provides a stable API and a clear upgrade path, it may not be an appropriate dependency for third party applications. For example, if the operating system switches to udisks version N.x and an application is still using the udisks (N-1).x API, the application will not work unless udisks (N-1).x is installed. While this situation is still workable (since both udisks N.x and udisks (N-1).x can be installed) it may not be desirable to ask the user to install the old version - in fact, the operating system vendor may not even provide a packaged version of the old version. Hence, if an application does not

want to tie itself to a specific version of the operating system, it should not use udisks.

Viable alternatives to udisks are APIs that are guaranteed to be around for longer time-frames, including:

? Low-level APIs and commands such as e.g. sysfs[7], libudev[8], /proc/self/mountinfo[9] and util-linux[10].

? High-level APIs such as GVolumeMonitor[11].

In particular, for desktop applications it is a much better idea to use something like GVolumeMonitor since it will make the application show the same devices as the desktop shell (e.g. file manager, file chooser and so on) is showing.

AUTHOR

This man page was originally written for UDisks2 by David Zeuthen <zeuthen@gmail.com> with a lot of help from many others.

BUGS

Please send bug reports to either the distribution bug tracker or the upstream bug tracker at https://github.com/storaged-project/udisks/issues.

SEE ALSO

udev(7), polkit(8), udisksd(8), udisksctl(1), umount.udisks2(8), gnome-disks(1)

NOTES

1. system message bus
   http://www.freedesktop.org/wiki/Software/dbus
2. GObjectIntrospection
   https://live.gnome.org/GObjectIntrospection
3. Desktop Entry Specification
   http://freedesktop.org/wiki/Specifications/desktop-entry-spec
4. freedesktop.org icon theme specification
   http://www.freedesktop.org/wiki/Specifications/icon-theme-spec
5. GNOME
   http://www.gnome.org
6. Disks

https://live.gnome.org/Design/Apps/Disks

7. sysfs

http://en.wikipedia.org/wiki/Sysfs

8. libudev

https://www.freedesktop.org/software/systemd/man/libudev.html

9. /proc/self/mountinfo

http://www.kernel.org/doc/Documentation/filesystems/proc.txt

10. util-linux

http://en.wikipedia.org/wiki/Util-linux

11. GVolumeMonitor

http://developer.gnome.org/gio/stable/volume_mon.html