# Hallyd

***Release 0.9***

**unknown**

**Feb 16, 2025**

# CONTENTS

Hallyd is a library that contains all kinds of tools for various tasks around filesystem, and other system features. This is intended to be used by other projects available here. External developers should double-check if it actually suffices their needs!

CHAPTER

ONE

# LICENSE

Hallyd is distributed under the terms of the AGPL 3 license. This also affects all included files without a license header (non-source files like images), unless they are explicitly mentioned as third-party content. Read the 'Dependencies' section for included third-party stuff.

CHAPTER

TWO

# UP-TO-DATE?

Are you currently reading from another source than the homepage? If you are in doubt whether your package is up-to-date, you should visit the project homepage and check that. You are currently reading the documentation for version 0.9.

CHAPTER

THREE

# USER MANUAL

## 3.1 Getting Started

### 3.1.1 Foo

This section is about getting ready.

CHAPTER

FOUR

# API REFERENCE

## 4.1 hallyd package

### 4.1.1 Subpackages

#### hallyd._aux namespace

#### Submodules

#### hallyd._aux.services_helper__call_action module

hallyd._aux.services_helper__call_action.**main**(*token*)

**Parameters**
**token** (*str*) –

**Return type**
None

#### hallyd._aux.subprocess_helper__call_function module

hallyd._aux.subprocess_helper__call_function.**main**()

### 4.1.2 Submodules

### 4.1.3 hallyd.bindle module

**exception** hallyd.bindle.**SerializingError**

Bases: ValueError

hallyd.bindle.**_by_qualname**(*module*, *qualname*)

hallyd.bindle.**_deserialize_object_json_loads_object_hook**(*dict_*)

hallyd.bindle.**_enum**(*enum_type*, *name*)

hallyd.bindle.**_filter_unneeded_dict_entries**(*obj_type*, *obj_dict*)

**Parameters**
**obj_dict** (*dict*) –

**Return type**
dict

hallyd.bindle.**_new_bytes**(*b64str*)

hallyd.bindle.**_new_datetime_datetime**(*timestamp*)

hallyd.bindle.**_new_datetime_timedelta**(*seconds*)

hallyd.bindle.**_new_functools_partial**(*func*, *args*, *kwargs*)

hallyd.bindle.**_new_set**(*items*)

hallyd.bindle.**_serialize_object_json_dumps_default**(*obj*)

hallyd.bindle.**dict_from_object**(*obj*)

**Parameters**
**obj** (*Optional[Any]*) –

**Return type**
*Optional*[dict[str, *Optional*[*Any*]]]

hallyd.bindle.**dump**(*obj*, *fp*)
Serialize an object to a file object. See also load().

**Parameters**

- **obj** (*Optional[Any]*) –
- **fp** (*IO*) –

**Return type**
None

hallyd.bindle.**dumps**(*obj*)
Serialize an object to a string. See also loads().

**Parameters**
**obj** (*Optional[Any]*) –

**Return type**
str

hallyd.bindle.**load**(*fp*)
Deserialize an object from a file object. See also dump() and loads().

**Parameters**
**fp** (*IO*) –

**Return type**
*Optional*[*Any*]

hallyd.bindle.**loads**(*sj*)
Deserialize an object from a string. See also dumps() and load().

**Parameters**
**sj** (*AnyStr*) –

**Return type**
*Optional*[*Any*]

### 4.1.4 hallyd.cleanup module

Cleanup tasks can do arbitrary things after your process is terminated, like removing temporary files or other system resources that are not cleaned up automatically by the operating system.

**class** hallyd.cleanup.**_CleanupTask**(*task_id*)

Bases: object

**Parameters**
**task_id** (*Union[str, Path]*) –

**Id**

alias of str

**remove**()

**Return type**
None

**property task_id: str**

hallyd.cleanup.**_add_cleanup_task**(*func*, *args*, *kwargs*)

**Parameters**
- **func** (*Union[Callable, SupportsQualifiedName]*) –
- **args** (*tuple*) –
- **kwargs** (*dict*) –

**Return type**
_CleanupTask

hallyd.cleanup.**_current_cleanup_scope**()

**Return type**
str

hallyd.cleanup.**_do_cleanup**()

**Return type**
None

hallyd.cleanup.**_guarded_cleanup**(*process_permanent_id*)

**Parameters**
**process_permanent_id** (*str*) –

**Return type**
None

hallyd.cleanup.**_have_same_fs_root**(*p1*, *p2*)

**Parameters**
- **p1** (*str*) –
- **p2** (*str*) –

**Return type**
bool

hallyd.cleanup.**add_cleanup_task**(*func*, **args*, ***kwargs*)

Add a cleanup task to be executed when the current process is terminated, and return a task controller object that allows early execution and other things. This will run in a separate process, once the current process is fully terminated.

**Parameters**

- **func** (*Union[Callable, SupportsQualifiedName]*) – The function to call. Must not be a (non-static) method of some object.
- **args** – The function call args.
- **kwargs** – The function call kwargs.

**Return type**
_CleanupTask

hallyd.cleanup.**cleanup_after_exit**()

Make sure that cleanup will take place once the current process is terminated. You usually do not need to call it.

hallyd.cleanup.**cleanup_task_by_id**(*task_id*)

Return the task controller object by task id.

**Parameters**
**task_id** (*str*) – The task id.

**Return type**
_CleanupTask

hallyd.cleanup.**mark_current_process_as_cleanup_scope**()

Mark the current process as cleanup scope, so even for child processes, cleanup will not happen before this process is terminated.

**Return type**
None

### 4.1.5 hallyd.coding module

**class** hallyd.coding.**Editor**(*srcfile*)

Bases: object

**Parameters**
**srcfile** (Path) –

**class** **_ClassHandle**(*editor*, *class_name*)

Bases: *_Handle*, *_WithCodePositionsHandleMixin*, *_WithDecorationSupportHandleMixin*, *_RemovableHandleMixin*

**_abc_impl = <_abc._abc_data object>**

**property _ast: Optional[ClassDef]**

**add_method**(*body*)

**Parameters**
**body** (*str*) –

method_by_name(*method_name*)

property methods

property name: str

class _FunctionHandle(*editor*, *function_name*, *class_handle=None*)

Bases: *_Handle*, *_WithCodePositionsHandleMixin*, *_WithDecorationSupportHandleMixin*, *_RemovableHandleMixin*

**Parameters**

- **editor** (Editor) –
- **function_name** (*str*) –

_abc_impl = <_abc._abc_data object>

property _ast: Optional[FunctionDef]

property name: str

class _Handle(*editor*)

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract property _ast: Optional[AST]

property _editor

class _RemovableHandleMixin

Bases: object

remove()

class _WithCodePositionsHandleMixin

Bases: object

property ends_at_position

property starts_at_position

class _WithDecorationSupportHandleMixin

Bases: object

_WithDecorationSupportHandleMixin__decoration_ast_to_code_indexes(*decoration_ast*)

add_decoration(*code*)

**Parameters**

**code** (*str*) –

property decorations

remove_decoration(*idx*)

__indentation_for_code_fragment()

**Parameters**

**code** (*str*) –

**Return type**
str

property _ast: AST

_fix_body(*body*)

**Parameters**
**body** (*str*) –

**Return type**
str

static _function_name_from_body(*body*)

add_class(*class_name*, *, *derived_from=None*, *docstring='TODO add some documentation here'*)

add_function(*body*)

add_import(*module_name*)

**Parameters**
**module_name** (*str*) –

class_by_name(*class_name*)

**Parameters**
**class_name** (*str*) –

**Return type**
_ClassHandle

property code: str

property indentation: str

property path: *Path*

### 4.1.6 hallyd.disk module

class hallyd.disk.**Disk**(*dev_path*)

Bases: object

**Parameters**
**dev_path** (Path) –

__idpath()

**Return type**
*Optional*[str]

__lsblk()

**Return type**
dict[str, *Any*]

__udev_property(*name*)

**Parameters**
**name** (*str*) –

**Return type**
*Optional*[str]

**_has_path**(*path*)

**Parameters**
**path** (*str*) –

**Return type**
bool

**property is_disk: bool**

**property is_removable: bool**

**partition**(*part_no*)

**Parameters**
**part_no** (*int*) –

**Return type**
DiskPartition

**partition_path**(*part_no*)

**Parameters**
**part_no** (*int*) –

**Return type**
Path

**property partitions: list[*hallyd.disk.DiskPartition*]**

**property path: *Path***

**property size_bytes: int**

**property stable_path: Optional[str]**

**stable_udev_filter**()

**Return type**
str

**class** hallyd.disk.**DiskIntent**(*disk*, *setup*)

Bases: object

**Parameters**

- **disk** (Disk) –
- **setup** (DiskSetup) –

**__reread_partition_table**()

**__write_partition_table**(*repartitionspecs*)

**property disk**

**repartition**()

**Return type**
None

property setup

udev_rule_for_alias()

**Return type**
str

class hallyd.disk.**DiskPartition**(*disk*, *part_no*)

Bases: *Partition*

**Parameters**

- **disk** (Disk) –
- **part_no** (*int*) –

property disk

property fstype: Optional[*_PartitionType*]

property part_no

class hallyd.disk.**DiskSetup**(**partitions*, *identify_by*, *do_repartition=True*, *partition_table_type='gpt'*, *name=None*)

Bases: object

**Parameters**

- **path** – The path to the disk device file.
- **repartition** – If to write a new partition table (or reuse the existing one).
- **partitions** (PartitionSetup) –
- **identify_by** (*Iterable[str]*) –
- **do_repartition** (*bool*) –
- **partition_table_type** (*str*) –
- **name** (*Optional[str]*) –

hallyd.disk.**EfiPartitionSetup**(***kwargs*)

class hallyd.disk.**Mountpoint**(*partition*, *mountpoint*, *fstype*)

Bases: object

**Parameters**

- **partition** (Partition) –
- **mountpoint** (*str*) –
- **fstype** (_PartitionType) –

fstab_line()

**Return type**
str

mount(*, *prefix=None*, *create_before=True*)

**Parameters**

- **prefix** (*Optional[str]*) –

- **create_before** (*bool*) –

**umount**()

hallyd.disk.**NotEfiPartitionSetup**()

**class** hallyd.disk.**OrderedPartitionSetupsEntry**(*part_no*, *partition_setup*)

Bases: object

**Parameters**

- **part_no** (*int*) –
- **partition_setup** (PartitionSetup) –

**property part_no: int**

**property partition_setup: Optional[*PartitionSetup*]**

**class** hallyd.disk.**Partition**(*path*)

Bases: object

**Parameters**
**path** (Path) –

**static by_partuuid**(*uuid_*)

**Parameters**
**uuid_** (*str*) –

**Return type**
Partition

**static by_uuid**(*uuid_*)

**Parameters**
**uuid_** (*str*) –

**Return type**
Partition

**property partuuid: Optional[str]**

**property path: *Path***

**property stable_path: *Path***

**property uuid: Optional[str]**

**class** hallyd.disk.**PartitionSetup**(***, *index=None*, *fs_type=None*, *mountpoint=None*, *label=None*, *do_format=True*, *use_in_raid=None*, *size=None*, *start_at_mb=None*, *flag_bootable=False*)

Bases: *_PartitionSetup*

**Parameters**

- **index** (*Optional[int]*) – Partition index (counted from 1). Only needed in exotic cases.
- **use_in_raid** (*Optional[str]*) – Name of the raid to use this partition for. See also the raid_name parameter of RaidPartitionSetup.__init__.
- **size** (*Optional[Union[int, Callable[[PartitionSizingEvent], int]]]*) – The size in bytes.

- **start_at_mb** (*Optional[float]*) – The start offset in units of 1024^2 bytes.
- **flag_bootable** (*bool*) – If to flag this partition as bootable.
- **fs_type** (*Optional[_PartitionType]*) –
- **mountpoint** (*Optional[str]*) –
- **label** (*Optional[str]*) –
- **do_format** (*bool*) –

class hallyd.disk.**PartitionSizingEvent**(*disk_size*)

Bases: object

**Parameters**
**disk_size** (*int*) –

**property disk_size: int**

class hallyd.disk.**PartitionTypes**

Bases: object

**EFI = hallyd.disk._PartitionType()**

**EXT4 = hallyd.disk._PartitionType()**

**RAID = hallyd.disk._PartitionType()**

**SWAP = hallyd.disk._PartitionType()**

**UNUSED = hallyd.disk._PartitionType()**

class hallyd.disk.**RaidPartition**(*path*)

Bases: *Partition*

**Parameters**
**path** (Path) –

**stop**()

class hallyd.disk.**RaidPartitionSetup**(*raid_name*, ***, *fs_type=None*, *mountpoint=None*, *label=None*, *do_format=True*, *do_create_raid=True*)

Bases: *_PartitionSetup*

**Parameters**

- **raid_name** (*str*) – The raid to use for this partition. See also the use_in_raid parameter of Partition.__init__.
- **fs_type** (*Optional[_PartitionType]*) –
- **mountpoint** (*Optional[str]*) –
- **label** (*Optional[str]*) –
- **do_format** (*bool*) –
- **do_create_raid** (*bool*) –

class hallyd.disk.**RaidSetup**(*name*, *partitions*)

Bases: object

**Parameters**

- **name** (*str*) –
- **partitions** (*list[hallyd.fs.Path]*) –

**create**(*hostname*, ***, *do_create=True*)

**Parameters**

- **hostname** (*str*) –
- **do_create** (*bool*) –

**Return type**
RaidPartition

**class** hallyd.disk.**_PartitionSetup**(***, *fs_type=None*, *mountpoint=None*, *label=None*, *do_format=True*)

Bases: object

Base class for all kinds of partitions.

**Parameters**

- **fs_type** (*Optional[_PartitionType]*) – The filesystem type name.
- **mountpoint** (*Optional[str]*) – The mountpoint to use .
- **label** (*Optional[str]*) – The partition label to assign.
- **do_format** (*bool*) – If to format the partition (or leaving it with its existing data).

**make_filesystem**(*partition_dev_path*)

**Parameters**
**partition_dev_path** (*Union[str, Path]*) –

**Return type**
None

**mountpoint_spec**(*partition*)

**Parameters**
**partition** (Partition) –

**Return type**
*Optional*[Mountpoint]

**class** hallyd.disk.**_PartitionType**(*gpt_uuid*, *mbr_id=None*, *mkfs_command=None*, *fstab_type_name=None*)

Bases: object

**Parameters**

- **gpt_uuid** (*str*) –
- **mbr_id** (*Optional[str]*) –
- **mkfs_command** (*Optional[list[str]]*) –
- **fstab_type_name** (*Optional[str]*) –

**static by_gpt_uuid**(*parttype_uuid*)

**Parameters**
**parttype_uuid** (*str*) –

**Return type**
*Optional*[_PartitionType]

property fstab_type_name

property gpt_uuid

make_filesystem(*partition_path*)

**Parameters**
**partition_path** (*Union[str, Path]*) –

property mbr_id

hallyd.disk._disks_sort_key(*disk*)

**Parameters**
**disk** (Disk) –

hallyd.disk._find_disk_for_setup(*disks*, *disk_setup*)

**Parameters**

- **disks** (*list[hallyd.disk.Disk]*) –
- **disk_setup** (DiskSetup) –

**Return type**
Disk

hallyd.disk._lsblk(*params*)

**Return type**
dict[str, *Any*]

hallyd.disk.combine_disks_to_setups(*disks*, *disk_setups*)

**Parameters**

- **disks** (*list[hallyd.disk.Disk]*) –
- **disk_setups** (*list[hallyd.disk.DiskSetup]*) –

**Return type**
list[*hallyd.disk.DiskIntent*]

hallyd.disk.effective_partition_setup_order(*partition_setups*)

**Parameters**
**partition_setups** (*list[hallyd.disk.PartitionSetup]*) –

**Return type**
list[*hallyd.disk.OrderedPartitionSetupsEntry*]

hallyd.disk.find_disks_for_setups(*disks*, *disk_setups*)

**Parameters**

- **disks** (*list[hallyd.disk.Disk]*) –
- **disk_setups** (*list[hallyd.disk.DiskSetup]*) –

**Return type**
dict[*hallyd.disk.DiskSetup*, *hallyd.disk.Disk*]

hallyd.disk.**find_partition_for_setup**(*disk*, *disk_setup*, *partition*)

**Parameters**

- **disk** (Disk) –
- **disk_setup** (DiskSetup) –
- **partition** (PartitionSetup) –

**Return type**
DiskPartition

hallyd.disk.**host_disks**()

**Return type**
list[*hallyd.disk.Disk*]

hallyd.disk.**host_partition_for_fs_path**(*fs_path*)

**Parameters**
**fs_path** (*Union[str, Path]*) –

**Return type**
Partition

hallyd.disk.**host_raid_partitions**()

**Return type**
list[*hallyd.disk.RaidPartition*]

hallyd.disk.**mount**(*dev_path*, *target_path*)

**Parameters**

- **dev_path** (Path) –
- **target_path** (Path) –

hallyd.disk.**partition_path**(*diskpath*, *part_no*)

Returns a partition device path for a given disk device path and a partition number. Examples:

- "/dev/sda", 1 -> "/dev/sda1"
- "/dev/loop2", 1 -> "/dev/loop2p1"

**Parameters**

- **diskpath** (Path) –
- **part_no** (*int*) –

**Return type**
Path

hallyd.disk.**partition_tuple**(*partition_dev*)

Returns the disk device path partition number for a given partition device path. Examples:

- "/dev/sda1" -> "/dev/sda", 1
- "/dev/loop2p1" -> "/dev/loop2", 1

**Parameters**
**partition_dev** (*Union[str, Path]*) –

**Return type**
tuple[*hallyd.fs.Path*, int]

hallyd.disk.**raid_setups_from_disk_intents**(*disk_intents*)

**Parameters**
**disk_intents** (*list[hallyd.disk.DiskIntent]*) –

**Return type**
dict[str, *hallyd.disk.RaidSetup*]

hallyd.disk.**reload_devices**()

hallyd.disk.**umount**(*path*)

**Parameters**
**path** (Path) –

### 4.1.7 hallyd.fs module

**class** hallyd.fs.**OnRemoveError**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

**FAIL_INSTANTLY = 3**

**SKIP_AND_FAIL_LATER = 2**

**SKIP_AND_IGNORE = 1**

**class** hallyd.fs.**OnRemovePassingFileSystemBoundary**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

**CONTINUE_REMOVING_BEHIND_BOUNDARY = 2**

**ERROR = 1**

**TRY_UNMOUNTING = 3**

**TRY_UNMOUNTING_FORCEFULLY = 4**

**class** hallyd.fs.**Path**(**args*, ***kwargs*)

Bases: PosixPath

Construct a PurePath from one or several strings and or existing PurePath objects. The strings and path objects are combined so as to yield a canonicalized path, which is incorporated into the new PurePath object.

**__change_access__call_ch_func**(**args*, ***kwargs*)

**__copy**(*destination*, ***, *remove_source*, *sparse*, *transfer_perms*, *mode*, *owner*, *group*, *readable_by_all*, *executable*)

**Parameters**

- **source** (Path) –
- **destination** (Path) –

- **remove_source** (*bool*) –
- **sparse** (*bool*) –
- **transfer_perms** (*bool*) –
- **mode** (*Optional[Union[str, int, bool]]*) –
- **owner** (*Optional[Union[str, int, bool]]*) –
- **group** (*Optional[Union[str, int, bool]]*) –
- **readable_by_all** (*Optional[bool]*) –
- **executable** (*Optional[bool]*) –

**Return type**
None

**__fallback_safely_create**(*, *mode=416*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

**Parameters**

- **destination** (Path) –
- **mode** (*Optional[Union[str, int, bool]]*) –
- **owner** (*Optional[Union[str, int, bool]]*) –
- **group** (*Optional[Union[str, int, bool]]*) –
- **readable_by_all** (*Optional[bool]*) –
- **executable** (*Optional[bool]*) –

**__file_ends_with_newline**()

**Parameters**
**fd** (*int*) –

**Return type**
bool

**__open**(*, *exist_ok*, *not_exist_ok*, *create_directory*)

**Parameters**

- **exist_ok** (*bool*) –
- **not_exist_ok** (*bool*) –
- **create_directory** (*bool*) –

**Return type**
tuple[int, bool, bool]

**__parse_ownership**(*lookup_func*, *true_result*, *none_result*)

**Parameters**

- **owner** (*Optional[Union[str, int, bool]]*) –
- **lookup_func** (*Callable*) –
- **true_result** (*int*) –
- **none_result** (*int*) –

**Return type**
int

__parse_permission_mode(*was_created*, *fd_stat*, *readable_by_all*, *executable*)

**Parameters**

- **mode** (*Optional[Union[str, int, bool]]*) –
- **was_created** (*bool*) –
- **fd_stat** (*stat_result*) –
- **readable_by_all** (*Optional[bool]*) –
- **executable** (*Optional[bool]*) –

**Return type**
int

__parse_permission_mode__part_re = re.compile('([ugoa]*)([+=-]?)([ugorwxXst]*)')

__parse_permission_mode__rmask_for_letters = {'r': 292, 's': 3072, 't': 512, 'w': 146, 'x': 73}

__parse_permission_mode__str_helper(*current_mode*, *is_directory*)

**Parameters**

- **mode_str** (*str*) –
- **current_mode** (*int*) –
- **is_directory** (*bool*) –

**Return type**
int

__parse_permission_mode__str_helper__expand(*lvalue*)

**Parameters**

- **mode** (*int*) –
- **lvalue** (*str*) –

**Return type**
int

__parse_permission_mode__str_helper__lmask()

**Parameters**
**lvalue** (*str*) –

**Return type**
int

__parse_permission_mode__str_helper__rmask(*current_mode*, *is_directory*)

**Parameters**

- **rvalue** (*str*) –
- **current_mode** (*int*) –
- **is_directory** (*bool*) –

**Return type**
int

**__path_depth**()

**Parameters**
**path** (*Path*) –

**Return type**
int

**__path_trimmed_to_depth**(*max_depth*)

**Parameters**

- **path** (*Path*) –
- **max_depth** (*int*) –

**Return type**
Path

**__remove__on_error_func**()

**Parameters**
**on_error** (*Union[Callable, OnRemoveError]*) –

**Return type**
*Callable*

**__remove__on_passing_filesystem_boundary_func**()

**Parameters**
**on_passing_filesystem_boundary** (*Union[Callable, OnRemovePassingFileSystemBoundary]*) –

**Return type**
*Callable*

**__remove__open**(*not_exist_ok*, *on_error*, *had_errors*)

**__remove__subtree**(*dir_dev_id*, *dir_path*, *on_passing_filesystem_boundary*, *on_error*, *had_errors*)

**__set_data**(*data*, ***, *as_directory=False*, *exist_ok=True*, *not_exist_ok=True*, *append=False*, *ensure_head_newline=True*, *ensure_tail_newline=True*, *preserve_perms=False*, *mode=416*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

**Parameters**

- **data** (*Optional[Union[AnyStr, RawIOBase]]*) –
- **as_directory** (*bool*) –
- **exist_ok** (*bool*) –
- **not_exist_ok** (*bool*) –
- **append** (*bool*) –
- **ensure_head_newline** (*bool*) –
- **ensure_tail_newline** (*bool*) –
- **preserve_perms** (*bool*) –
- **mode** (*Optional[Union[str, int, bool]]*) –

- **owner** (*Optional[Union[str, int, bool]]*) –
- **group** (*Optional[Union[str, int, bool]]*) –
- **readable_by_all** (*Optional[bool]*) –
- **executable** (*Optional[bool]*) –

**Return type**
None

**append_data**(*data*, ***, *exist_ok=True*, *ensure_head_newline=True*, *ensure_tail_newline=True*, *preserve_perms=False*, *mode=416*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

Appends data to the end of the file at this path and apply some permission settings.

Per default, those permission settings will be applied even if the file already exists (see *preserve_perms*)!

**Parameters**

- **data** (*Union[AnyStr, RawIOBase]*) – The data to write.
- **exist_ok** (*bool*) – Whether it is okay if this file already exists.
- **ensure_head_newline** (*bool*) – For appending, whether to ensure a newline where *data* begins.
- **ensure_tail_newline** (*bool*) – For appending, whether to ensure a newline where *data* end.
- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the file already exists.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See *change_access()*.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See *change_access()*. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See *change_access()*. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See *change_access()*. Default has no effect.
- **executable** (*Optional[bool]*) – See *change_access()*. Default has no effect.

**Return type**
*Self*

**apply_substitutions**(**substitutions*, *source=None*, *preserve_perms=True*, *mode=416*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

Apply regexp substitution patterns to this file.

**Parameters**

- **substitutions** (*tuple[str, str]*) – Each substitution is a tuple of the match pattern and the replacement pattern.
- **source** (*Union[str, Path]*) – The source path to read the original content from. Default: This file.
- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the file already exists.

- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See *change_access()*.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See *change_access()*. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See *change_access()*. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See *change_access()*. Default has no effect.
- **executable** (*Optional[bool]*) – See *change_access()*. Default has no effect.

**Return type**
Path

**change_access**(*mode=None*, ***, *follow_symlinks=True*, *recursive=False*, *owner=None*, *group=None*, *readable_by_all=None*, *executable=None*)

Change file permissions and ownership.

**Parameters**

- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. If *None*, leave it unchanged (unless some other flags are set).
- **follow_symlinks** (*bool*) – If this is a symlink, whether to change the permission settings of the target item instead.
- **recursive** (*bool*) – Whether to apply the specified changes to the entire subtree (useful for directories).
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. If *True*, it's the (effective) current user. If *False*, leave it unchanged. Otherwise either a UID or user name.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. If *True*, it's the (effective) current user's primary group. If *False*, leave it unchanged. Otherwise either a GID or group name.
- **readable_by_all** (*Optional[bool]*) – If set, the mode will be extended to readable for all (and maybe executable).
- **executable** (*Optional[bool]*) – Whether to set normal files to executable for everyone with read privileges. If *True* or *False*, enable or disable this flag. If *None*, leave untouched.

**Return type**
*Self*

**copy_to**(*destination*, ***, *exist_ok=False*, *merge=False*, *sparse=False*, *transfer_perms=False*, *mode=True*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

Copy the item at this path to a destination.

If you copy a directory, all settings (incl. permission flags, ...) are applied to the entire tree.

**Parameters**

- **destination** (*Union[str, Path]*) – The destination path.
- **exist_ok** (*bool*) – Whether it is not an error if the destination path already exists (per default it will be removed then).
- **merge** (*bool*) – Whether to merge the source directory file-wise into the destination directory, instead of relying on an empty destination.
- **sparse** (*bool*) – If to copy in sparse mode. Usually not needed.

- **transfer_perms** (*bool*) – Whether to transfer the permission settings of the source file.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See *change_access()*. Default: u=rw,g=r and also +x if the source is executable.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See *change_access()*. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See *change_access()*. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See *change_access()*. Default has no effect.
- **executable** (*Optional[bool]*) – See *change_access()*. Default has no effect.

**Return type**
Path

**expand_archive_to**(*destination*, ***, *take_1st_level=False*, *exist_ok=False*, *mode=True*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

Expand this archive file (zip or tar-based) to a destination.

**Parameters**

- **destination** (*Union[str, Path]*) – The destination path.
- **take_1st_level** (*bool*) – Whether to take the top-level item from the archive (must be the only one on top level) and extract this one to the destination.
- **exist_ok** (*bool*) – Whether it is not an error if the destination path already exists (it will be removed then).
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See *change_access()*. Default: u=rw,g=r and also +x if the source is executable.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See *change_access()*. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See *change_access()*. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See *change_access()*. Default has no effect.
- **executable** (*Optional[bool]*) – See *change_access()*. Default has no effect.

**Return type**
Path

**classmethod home_dir**(*user=None*)

**Parameters**
**user** (*Optional[str]*) –

**Return type**
Path

**iterdir**()

Iterate over the files/subdirectories/etc in this directory.

Does not include the special items . and ...

**Return type**
*Iterable*[Path]

**make_dir**(*, *until=None*, *exist_ok=False*, *parent_exist_ok=True*, *preserve_perms=False*, *mode=488*, *owner=True*, *group=True*, *readable_by_all=None*)

Create a directory at this path.

With *readable_by_all=True* and *preserve_perms=True* this function behaves like `mkdir()`.

**Parameters**

- **until** (*Optional[Union[str, Path]]*) – Similar to *parents*, but only creates all superdirectories up to (excluding) that one.
- **exist_ok** (*bool*) – Whether it is okay if this directory already exists (it will set permission related attributes anyway).
- **parent_exist_ok** (*bool*) – Whether it is okay if a *parent* directory up to *until* already exists (it will NOT set permission related attributes for them).
- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the directory already exists.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See *change_access()*.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See *change_access()*. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See *change_access()*. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See *change_access()*. Default has no effect.

**Return type**
Path

**make_file**(*, *exist_ok=True*, *preserve_perms=False*, *mode=416*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

Create a file at this path.

With *readable_by_all=True* and *preserve_perms=True* this function behaves like `touch()`.

**Parameters**

- **exist_ok** (*bool*) – Whether it is okay if this file already exists (it will set permission related attributes anyway).
- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the file already exists.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See *change_access()*.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See *change_access()*. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See *change_access()*. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See *change_access()*. Default has no effect.
- **executable** (*Optional[bool]*) – See *change_access()*. Default has no effect.

**Return type**
Path

**move_to**(*destination*, ***, *exist_ok=False*)

Move the item at this path to a destination. Permission settings will be retained.

**Parameters**

- **destination** (*Union[str, Path]*) – The destination path.
- **exist_ok** (*bool*) – Whether it is not an error if the destination path already exists (it will be removed then).

**Return type**
Path

TODO merge ?!

**non_existent**()

**Return type**
bool

**relative_to**(*other*, *strict=True*)

Return the relative path to another path identified by the passed arguments. If the operation is not possible (because this is not a subpath of the other path), raise ValueError.

**Parameters**

- **other** (*Union[str, Path]*) –
- **strict** (*bool*) –

**Return type**
Path

**remove**(***, *not_exist_ok=False*, *on_error=OnRemoveError.SKIP_AND_FAIL_LATER*, *on_passing_filesystem_boundary=OnRemovePassingFileSystemBoundary.TRY_UNMOUNTING*)

Remove the item at this path (for directories, including the entire tree).

**Parameters**

- **not_exist_ok** (*bool*) – Whether it is okay if this item does not exist.
- **on_error** (*Union[Callable, OnRemoveError]*) – How to behave when errors occur.
- **on_passing_filesystem_boundary** (*Union[Callable, OnRemovePassingFileSystemBoundary]*) – How to behave when the removal of a directory tree would pass a filesystem boundary (i.e. if there is some other filesystem mounted somewhere in the tree).

**Return type**
*Self*

**set_data**(*data*, ***, *exist_ok=True*, *preserve_perms=False*, *mode=416*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

Write data into a file at this path and apply some permission settings.

Per default, those permission settings will be applied even if the file already exists (see *preserve_perms*)!

With *readable_by_all=True* and *preserve_perms=True* this function behaves like `write_text()` or `write_bytes()`.

**Parameters**

- **data** (*Union[AnyStr, RawIOBase]*) – The data to write.
- **exist_ok** (*bool*) – Whether it is okay if this file already exists.

- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the file already exists.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See *change_access()*.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See *change_access()*. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See *change_access()*. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See *change_access()*. Default has no effect.
- **executable** (*Optional[bool]*) – See *change_access()*. Default has no effect.

**Return type**
*Self*

**classmethod temp_dir**(*, *mode=488*, *owner=True*, *group=True*, *readable_by_all=None*, *temp_root_path='/tmp'*)

Create a fresh temporary directory and return its path. You must use it for a with-block; it will be removed after that block in usual cases. Otherwise at a somewhat later time (usually after the process terminated). This removal will fail if you do not have permissions to do so.

**Parameters**

- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See *change_access()*.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See *change_access()*. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See *change_access()*. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See *change_access()*. Default has no effect.
- **temp_root_path** (*Union[str, Path]*) – The parent directory where to create the temporary directory.

**Return type**
*ContextManager*[*Self*]

**class** hallyd.fs.**_Archive**(*archive_fileobj*)

Bases: ABC

**Parameters**
**archive_fileobj** (*RawIOBase*) –

**_abc_impl = <_abc._abc_data object>**

**abstract classmethod check_file_supported_by_begin**(*begin*)

**Parameters**
**begin** (*bytes*) –

**Return type**
bool

**abstract extract_all**(*destination*)

**Parameters**
**destination** (Path) –

**Return type**
None

**abstract property take_1st_level_default: bool**

class hallyd.fs.**_DiskSpaceInfo**(*total: int*, *used: int*, *free: int*)

Bases: object

**Parameters**

- **total** (*int*) –
- **used** (*int*) –
- **free** (*int*) –

**free: int**

**total: int**

**used: int**

class hallyd.fs.**_TarArchive**(*archive_fileobj*)

Bases: *_Archive*

**Parameters**
**archive_fileobj** (*RawIOBase*) –

**_abc_impl = <_abc._abc_data object>**

**classmethod check_file_supported_by_begin**(*begin*)

**extract_all**(*destination*)

**take_1st_level_default = True**

class hallyd.fs.**_ZipArchive**(*archive_fileobj*)

Bases: *_Archive*

**Parameters**
**archive_fileobj** (*RawIOBase*) –

**_abc_impl = <_abc._abc_data object>**

**classmethod check_file_supported_by_begin**(*begin*)

**extract_all**(*destination*)

**take_1st_level_default = False**

hallyd.fs.**_archive**(*archive_fileobj*)

hallyd.fs.**byte_size_to_human_readable**(*size*)

Return a human readable format for a size in bytes.

**Parameters**
**size** (*int*) – The number to format in bytes.

**Return type**
str

hallyd.fs.**disk_space**(*fs_root_dir*)

**Parameters**
**fs_root_dir** (*Union[str, Path]*) –

hallyd.fs.**disk_usage**(*path*)

Return the disk usage for a file or directory.

**Parameters**
**path** (*Union[str, Path]*) – The path.

**Return type**
int

hallyd.fs.**expand_archive**(*source*, *destination*, *, *take_1st_level=None*, *exist_ok=False*, *mode=True*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

Expand an archive file (zip or tar-based) to a destination.

**Parameters**

- **source** (*Union[str, Path, RawIOBase]*) – The source archive to expand.
- **destination** (*Union[str, Path]*) – The destination path.
- **take_1st_level** (*Optional[bool]*) – Whether to take the top-level item from the archive (must be the only one on top level) and extract this one to the destination. If None: Depends on the archive format.
- **exist_ok** (*bool*) – Whether it is not an error if the destination path already exists (it will be removed then).
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See change_access(). Default: u=rw,g=r and also +x if the source is executable.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See change_access(). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See change_access(). Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See change_access(). Default has no effect.
- **executable** (*Optional[bool]*) – See change_access(). Default has no effect.

**Return type**
Path

hallyd.fs.**temp_dir**(*, *mode=488*, *owner=True*, *group=True*, *readable_by_all=None*, *temp_root_path='/tmp'*)

Create a fresh temporary directory and return its path. You must use it for a with-block; it will be removed after that block in usual cases. Otherwise, at a somewhat later time (usually after the process terminated). This removal will fail if you do not have permissions to do so.

**Parameters**

- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See change_access().
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See change_access(). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See change_access(). Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See change_access(). Default has no effect.

- **temp_root_path** (*Union[str, Path]*) – The parent directory where to create the temporary directory.

**Return type**
*ContextManager*[Path]

### 4.1.8 hallyd.fs_monitor module

**class** hallyd.fs_monitor.**FilesystemMonitor**(**paths*, *trigger_initially=True*)

Bases: ABC

**Parameters**

- **paths** (*_fs.TInputPath*) –
- **trigger_initially** (*bool*) –

**class _Thread**(*paths*, *trigger_initially*, *monitor*)

Bases: Thread

This constructor should always be called with keyword arguments. Arguments are:

*group* should be None; reserved for future extension when a ThreadGroup class is implemented.

*target* is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

*args* is a list or tuple of arguments for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

**Parameters**

- **paths** (*Iterable[_fs.Path]*) –
- **trigger_initially** (*bool*) –
- **monitor** (FilesystemMonitor) –

**check_if_changed**(*wait_until_handled*)

**Parameters**
**wait_until_handled** (*bool*) –
**Return type**
None

**force_changed**(*wait_until_handled*)

**Parameters**
**wait_until_handled** (*bool*) –
**Return type**
None

**run**()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

stop()

_abc_impl = <_abc._abc_data object>

abstract _changed()

**Return type**
None

check_if_changed(*, *wait_until_handled=True*)

**Parameters**
**wait_until_handled** (*bool*) –

**Return type**
None

force_changed(*, *wait_until_handled=True*)

**Parameters**
**wait_until_handled** (*bool*) –

**Return type**
None

property paths: Iterable[*Path*]

class hallyd.fs_monitor.**Watcher**

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract wait_changed(*, *timeout=None*)

**Parameters**
**timeout** (*Optional[float]*) –

**Return type**
bool

class hallyd.fs_monitor.**_Watcher**(*paths*, *trigger_initially*)

Bases: *Watcher*

**Parameters**

- **paths** (*Iterable[_fs.TInputPath]*) –
- **trigger_initially** (*bool*) –

class PollingWatchThread(*watcher*)

Bases: *WatchThread*

This constructor should always be called with keyword arguments. Arguments are:

*group* should be None; reserved for future extension when a ThreadGroup class is implemented.

*target* is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

*args* is a list or tuple of arguments for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

**Parameters**
**watcher** (_Watcher) –

**_abc_impl = <_abc._abc_data object>**

**_wait_event()**

**class WatchThread**(*watcher*)

Bases: Thread, ABC

This constructor should always be called with keyword arguments. Arguments are:

*group* should be None; reserved for future extension when a ThreadGroup class is implemented.

*target* is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

*args* is a list or tuple of arguments for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

**Parameters**
**watcher** (_Watcher) –

**_abc_impl = <_abc._abc_data object>**

**_setup()**

**_teardown()**

**_wait_event()**

**Return type**
bool

**property _watcher: _Watcher**

**run()**

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

**stop()**

**classmethod try_create**(*watcher*)

**Parameters**
**watcher** (_Watcher) –
**Return type**
*Optional*[WatchThread]

class WatchdogWatchThread(*watcher*)

Bases: *WatchThread*

This constructor should always be called with keyword arguments. Arguments are:

*group* should be None; reserved for future extension when a ThreadGroup class is implemented.

*target* is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

*args* is a list or tuple of arguments for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

**Parameters**
**watcher** (*_Watcher*) –

_abc_impl = <_abc._abc_data object>

_setup()

_teardown()

_WatchThreadTypes = (<class 'hallyd.fs_monitor._Watcher.PollingWatchThread'>, <class 'hallyd.fs_monitor._Watcher.WatchdogWatchThread'>)

_abc_impl = <_abc._abc_data object>

_check_if_changed(*, *wait_until_handled=False*)

**Parameters**
**wait_until_handled** (*bool*) –

**Return type**
None

_force_changed(*, *wait_until_handled=False*)

**Parameters**
**wait_until_handled** (*bool*) –

**Return type**
None

property paths: Iterable[*Path*]

wait_changed(*, *timeout=None*)

hallyd.fs_monitor.**watch**(**paths*, *trigger_initially=True*)

**Parameters**

- **paths** (*Union[str, Path]*) –
- **trigger_initially** (*bool*) –

**Return type**
Watcher

### 4.1.9 hallyd.io module

class hallyd.io.**Lock**

Bases: ABC

**_abc_impl = <_abc._abc_data object>**

**abstract acquire**(*blocking=True*, *timeout=-1*)

**Parameters**

- **blocking** (*bool*) –
- **timeout** (*float*) –

**Return type**
bool

**abstract locked**()

**Return type**
bool

**abstract release**()

**Return type**
None

class hallyd.io.**_Lock**(*lock_path*, *is_reentrant*, *peek_interval*)

Bases: *Lock*

**Parameters**

- **lock_path** (*_fs.Path*) –
- **is_reentrant** (*bool*) –
- **peek_interval** (*float*) –

**_abc_impl = <_abc._abc_data object>**

**acquire**(*blocking=True*, *timeout=-1*)

**locked**()

**release**()

class hallyd.io.**_LoopDevice**(***kwargs*)

Bases: object

**property back_file:** *Path*

**detach**()

**property dev_path:** *Path*

hallyd.io.**_detach_loop_device**(*dev_path*, *back_file*)

hallyd.io.**all_loop_devices**()

hallyd.io.**connect_diskimage**(*disk_image_path*)

**Parameters**
**disk_image_path** (Path) –

**Return type**
*ContextManager*[Path]

hallyd.io.**connect_diskimage_buffered**(*dev_path*, *, *buffer_size_gb*)

**Parameters**

- **dev_path** (Path) –
- **buffer_size_gb** (*float*) –

**Return type**
*ContextManager*[Path]

hallyd.io.**create_diskimage**(*path*, *, *size_gb*)

**Parameters**

- **path** (*Union[str, Path]*) –
- **size_gb** (*float*) –

**Return type**
None

hallyd.io.**lock**(*lock_path*, *, *is_reentrant=True*, *peek_interval=0.25*)

**Parameters**

- **lock_path** (*Union[str, Path]*) –
- **is_reentrant** (*bool*) –
- **peek_interval** (*float*) –

**Return type**
Lock

hallyd.io.**loop_device_by_dev_path**(*dev_path*)

**Parameters**
**dev_path** (Path) –

**Return type**
*Optional*[_LoopDevice]

### 4.1.10 hallyd.ipc module

**exception** hallyd.ipc.**BadConnectionError**(*message*)

Bases: OSError

**Parameters**
**message** (*str*) –

**class** hallyd.ipc.**Enablable**

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract disable()

**Return type**
None

abstract enable()

**Return type**
None

abstract property is_enabled: bool

exception hallyd.ipc.IPCServerPathAlreadyExistsError(*ipc_path*)

Bases: *BadConnectionError*

**Parameters**
**ipc_path** (Path) –

property ipc_path

exception hallyd.ipc.IPCServerUnavailableError(*ipc_path*)

Bases: *BadConnectionError*

**Parameters**
**ipc_path** (Path) –

property ipc_path

exception hallyd.ipc.MethodCallErroneousError(*message*)

Bases: RuntimeError

**Parameters**
**message** (*str*) –

class hallyd.ipc._Client

Bases: *object*

class Proxy(*request_func*)

Bases: object

abstract _request(*method_name*, *args*, *kwargs*)

property object

class hallyd.ipc._LocalClient(*path*)

Bases: *_Client*

**Parameters**
**path** (Path) –

_request(*method_name*, *args*, *kwargs*)

class hallyd.ipc._NetworkClient(*connection*, *path*)

Bases: *_Client*

**Parameters**

- **connection** (*_net.Connection*) –
- **path** (Path) –

**_request**(*method_name*, *args*, *kwargs*)

**class** hallyd.ipc.**_ThreadedServer**(*object*, *, *path*)

Bases: *Enablable*

**Parameters**

- **object** (*Any*) –
- **path** (Path) –

**class _MainThread**(*server*)

Bases: Thread

This constructor should always be called with keyword arguments. Arguments are:

*group* should be None; reserved for future extension when a ThreadGroup class is implemented.

*target* is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

*args* is a list or tuple of arguments for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

**Parameters**

**server** (_ThreadedServer) –

**run**()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

**stop**()

**class _Request**(*connection*, *method_name*, *args*, *kwargs*)

Bases: object

**Parameters**

- **method_name** (*str*) –
- **args** (*list[Optional[Any]]*) –
- **kwargs** (*dict[str, list[Optional[Any]]]*) –

**answer**(*, *result=None*, *error=None*)

**_ThreadedServer__get_request**(*connection*)

**Return type**

*Optional*[_Request]

**_ThreadedServer__worker**(*request*, *connection*)

**_abc_impl = <_abc._abc_data object>**

_try_process_request()

**Return type**
None

disable()

enable()

property is_enabled

hallyd.ipc._socket_path(*ipc_path*)

**Parameters**
**ipc_path** (Path) –

**Return type**
Path

hallyd.ipc.client(*path*, *, *connection=None*)

**Parameters**

- **path** (Path) –
- **connection** (*Optional[Connection]*) –

**Return type**
*Any*

hallyd.ipc.threaded_server(*object_*, *, *path*)

**Parameters**

- **object_** (*Any*) –
- **path** (Path) –

**Return type**
Enablable

### 4.1.11 hallyd.ipc_hub module

class hallyd.ipc_hub.Hub(*path*)

Bases: Generic[_RequestT, _ResponseT]

Hubs provide one way of distributed computing via hallyd.ipc.

Clients of a hub can put requests on a hub, and eventually get answer data.

A hub contains an IPC server for an internal hub controller, which provides a pluggable interface. Connect *HubWorker* instances to the hub for request processing.

You must use it for a with-block.

**Parameters**
**path** (Path) –

__verify_entered()

**get_answer**(*request_id*)

Return the answer for a request made in the past.

This can only be called for each request.

**Parameters**
**request_id** (*int*) – The request id.

**Return type**
*_ResponseT*

**property path**

The IPC path where hub workers can be plugged into by means of *HubWorker.plug_into_hub()*.

**put_request**(*request*)

Make a request to the hub and return the request id for further actions.

**Parameters**
**request** (*_RequestT*) –

**Return type**
int

**class** hallyd.ipc_hub.**HubWorker**(*ipc_dialog_hub_object*, *, *poll_interval=0.5*)

Bases: Generic[_RequestT, _ResponseT]

Hub workers can be plugged into *Hub* instances in order to implement a routine that processes requests.

Plug a worker into a hub using *plug_into_hub()*.

**Parameters**
- **ipc_dialog_hub_object** (*_HubIpcObject*) –
- **poll_interval** (*float*) –

**__run_worker_thread**()

**answer_request**(*request_id*, *response*)

**Parameters**
- **request_id** (*int*) –
- **response** (*_ResponseT*) –

**Return type**
bool

**classmethod plug_into_hub**(*path*, *connection=None*, ***kwargs*)

Creates a new instance of this hub worker type and connects it to a hub while active.

You must use it for a with-block.

**Parameters**
- **path** (*_fs.Path*) –
- **connection** (*Optional[_net.Connection]*) –
- **kwargs** –

**Return type**
t.Self

abstract request_arrived(*request*)

**Parameters**
**request** (*_RequestT*) –

**Return type**
None

request_disappeared(*request*)

**Parameters**
**request** (*_RequestT*) –

**Return type**
None

try_lock_request(*request*)

**Parameters**
**request** (*_RequestT*) –

**Return type**
bool

class hallyd.ipc_hub._HubIpcObject

Bases: Generic[_RequestT, _ResponseT]

The internal controller used by each hubs in order to provide pluggability.

add_answer(*request_id*, *response*)

Store the answer for a request.

**Parameters**

- **request_id** (*int*) –
- **response** (*_ResponseT*) –

**Return type**
bool

add_request(*request*)

Add a new request to the hub.

**Parameters**
**request** (*_RequestT*) –

**Return type**
int

pending_request_ids()

Return request ids for all requests that a currently open.

pop_answer(*request_id*)

Return and delete the answer for a request.

**Parameters**
**request_id** (*int*) –

**Return type**
*_ResponseT*

**request_by_id**(*request_id*)

Return the request by request id.

**Parameters**
**request_id** (*int*) –

**Return type**
*Optional*[_HubRequest[*_RequestT*]]

class hallyd.ipc_hub.**_HubRequest**(*request_id*, *payload*)

Bases: Generic[_RequestT]

One request that was made to the hub.

**property id**

**property payload**

**property request_id**

**property request_payload**

### 4.1.12 hallyd.lang module

class hallyd.lang.**AllAbstractMethodsProvidedByTrick**

Bases: Generic[_T]

class hallyd.lang.**Counter**

Bases: object

**next**()

class hallyd.lang.**_AllAbstractMethodsProvidedByTrickMeta**(*name*, *bases*, *namespace*)

Bases: ABCMeta, Generic[_T]

class hallyd.lang.**_ExecuteParallelThread**(*fct*)

Bases: Thread

This constructor should always be called with keyword arguments. Arguments are:

*group* should be None; reserved for future extension when a ThreadGroup class is implemented.

*target* is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

*name* is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.

*args* is a list or tuple of arguments for the target invocation. Defaults to ().

*kwargs* is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

**Parameters**
**fct** (*Callable[[], None]*) –

**run**()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

hallyd.lang.**call_now_with_retry**(*, *tries=8*, *interval=30*, *interval_fact=1*, *retry_on=None*)

**Parameters**

- **tries** (*int*) –
- **interval** (*float*) –
- **interval_fact** (*float*) –
- **retry_on** (*Optional[Iterable[type[Exception]]]*) –

**Return type**
*Callable*

hallyd.lang.**execute_in_parallel**(*funcs*)

**Parameters**
**funcs** (*list[Callable[[], NoneType]]*) –

**Return type**
None

hallyd.lang.**unique_id**(*, *numeric_only=False*)

**Parameters**
**numeric_only** (*bool*) –

**Return type**
str

hallyd.lang.**with_friendly_repr_implementation**(*, *skip=()*)

**Parameters**
**skip** (*Iterable[str]*) –

hallyd.lang.**with_retry**(*, *tries=8*, *interval=30*, *interval_fact=1*, *retry_on=None*)

**Parameters**

- **tries** (*int*) –
- **interval** (*float*) –
- **interval_fact** (*float*) –
- **retry_on** (*Optional[Iterable[type[Exception]]]*) –

**Return type**
*Callable*

### 4.1.13 hallyd.net module

**exception** hallyd.net.**AccessDeniedError**

Bases: OSError

**class** hallyd.net.**Connection**

Bases: object

**class ExecutionResult**(*returncode*, *output*, *error_output*)

Bases: object

**Parameters**

- **returncode** (*int*) –
- **output** (*str*) –
- **error_output** (*str*) –

**property error_output: str**

**property output: str**

**property returncode**

**abstract exec**(*command*)

**Parameters**
**command** (*list[str]*) –

**Return type**
ExecutionResult

**is_alive**()

**Return type**
bool

**abstract mount**(*remote_path*, *local_path*)

**Parameters**

- **remote_path** (Path) –
- **local_path** (Path) –

**Return type**
None

**abstract umount**(*local_path*)

**Parameters**
**local_path** (Path) –

**Return type**
None

**exception** hallyd.net.**CouldNotConnectError**

Bases: OSError

class hallyd.net.**SshConnection**(*host*, ***, *port*, *user*, *password*, *connect_timeout=60*)

Bases: *Connection*

**Parameters**

- **host** (*str*) –
- **port** (*int*) –
- **user** (*str*) –
- **password** (*str*) –
- **connect_timeout** (*float*) –

**exec**(*command*)

**mount**(*remote_path*, *local_path*)

**umount**(*local_path*)

### 4.1.14 hallyd.py_import module

hallyd.py_import.**_file_name_to_module_name**(*file_name*)

**Parameters**
**file_name** (*str*) –

**Return type**
str

hallyd.py_import.**import_module**(*module_file*, ***, *module_name_left='+hallyd..'*, *module_name_right=None*, *using_sys_modules=False*)

**Parameters**

- **module_file** (*Union[str, Path]*) –
- **module_name_left** (*str*) –
- **module_name_right** (*Optional[str]*) –
- **using_sys_modules** (*bool*) –

**Return type**
module

hallyd.py_import.**import_types_from_module_dir**(*base_type*, *module_dir*, ***, *file_name_glob_pattern='*.[Pp][Yy]'*, *module_name_left='+hallyd..'*, *using_sys_modules=False*)

**Parameters**

- **base_type** (*type[~_T]*) –
- **module_dir** (*Union[str, Path]*) –
- **file_name_glob_pattern** (*str*) –
- **module_name_left** (*str*) –
- **using_sys_modules** (*bool*) –

**Return type**
*Iterable*[type[~_T]]

hallyd.py_import.**types_from_module**(*base_type*, *module*)

**Parameters**

- **base_type** (*type[~_T]*) –
- **module** (*module*) –

**Return type**
*Iterable*[type[~_T]]

### 4.1.15 hallyd.services module

**class** hallyd.services.**CalendarTask**

Bases: *_Task*, ABC

**_abc_impl = <_abc._abc_data object>**

**class** hallyd.services.**CalendarTaskSetup**(*_name: Optional[str] = None*, *_runnable: Optional[ForwardRef('TRunnable')] = None*, *_description: Optional[str] = None*, *_user: Union[str, int, NoneType] = None*, *_group: Union[str, int, NoneType] = None*, *_discard_output: bool = False*, *_working_dir:* hallyd.fs.Path *= Path('/')*, *_calendars: Iterable[ForwardRef('_Calendar')] = ()*)

Bases: *_TimedTaskSetup*

**Parameters**

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[TRunnable]*) –
- **_description** (*Optional[str]*) –
- **_user** (*Optional[Union[str, int]]*) –
- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** (Path) –
- **_calendars** (*Iterable[_Calendar]*) –

**class _Calendar**

Bases: object

**class _DailyCalendar**(*time: Optional[str]*)

Bases: *_Calendar*

**Parameters**
**time** (*Optional[str]*) –

**time: Optional[str]**

class _MonthlyCalendar(*time: Optional[str]*, *day: int*)

Bases: _Calendar

**Parameters**

- **time** (*Optional[str]*) –
- **day** (*int*) –

day: int

time: Optional[str]

class _WeeklyCalendar(*time: Optional[str]*, *day_of_week: int*)

Bases: _Calendar

**Parameters**

- **time** (*Optional[str]*) –
- **day_of_week** (*int*) –

day_of_week: int

time: Optional[str]

class _YearlyCalendar(*time: Optional[str]*, *day: int*, *month: int*)

Bases: _Calendar

**Parameters**

- **time** (*Optional[str]*) –
- **day** (*int*) –
- **month** (*int*) –

day: int

month: int

time: Optional[str]

_calendars: Iterable[_Calendar] = ()

schedule_daily(*, *time=None*)

**Parameters**
**time** (*Optional[str]*) –

**Return type**
None

schedule_monthly(*, *time=None*, *day=1*)

**Parameters**

- **time** (*Optional[str]*) –
- **day** (*int*) –

**Return type**
None

**schedule_weekly**(*, *time=None*, *day_of_week=7*)

**Parameters**

- **time** (*Optional[str]*) –
- **day_of_week** (*int*) –

**Return type**
None

**schedule_yearly**(*, *time=None*, *day=1*, *month=1*)

**Parameters**

- **time** (*Optional[str]*) –
- **day** (*int*) –
- **month** (*int*) –

**Return type**
None

class hallyd.services.**IntervalTask**

Bases: *_Task*, ABC

**_abc_impl = <_abc._abc_data object>**

class hallyd.services.**IntervalTaskSetup**(*_name: Optional[str] = None*, *_runnable: Optional[ForwardRef('TRunnable')] = None*, *_description: Optional[str] = None*, *_user: Union[str, int, NoneType] = None*, *_group: Union[str, int, NoneType] = None*, *_discard_output: bool = False*, *_working_dir:* hallyd.fs.Path *= Path('/')*, *_intervals: Iterable[float] = ()*)

Bases: *_TimedTaskSetup*

**Parameters**

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path,* Runnable*, Callable[[], None], Callable[[*Runnable*], None]]]*) –
- **_description** (*Optional[str]*) –
- **_user** (*Optional[Union[str, int]]*) –
- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** (Path) –
- **_intervals** (*Iterable[float]*) –

**_intervals: Iterable[float] = ()**

**schedule_by_interval**(*seconds=0*, *, *minutes=0*, *hours=0*)

**Parameters**

- **seconds** (*float*) –
- **minutes** (*float*) –

- **hours** (*float*) –

**Return type**
None

class hallyd.services.**NextBootTask**

Bases: *_Task*, ABC

**_abc_impl = <_abc._abc_data object>**

class hallyd.services.**NextBootTaskSetup**(*_name: Optional[str] = None*, *_runnable: Optional[ForwardRef('TRunnable')] = None*, *_description: Optional[str] = None*, *_user: Union[str, int, NoneType] = None*, *_group: Union[str, int, NoneType] = None*, *_discard_output: bool = False*, *_working_dir:* hallyd.fs.Path *= Path('/')*, *_dependencies: Iterable[*hallyd.services._TaskSetup._WithDependencies.Dependency*] = ()*, *_interactive: bool = False*)

Bases: *_WithDependencies*, *_TaskSetup*

**Parameters**

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –
- **_description** (*Optional[str]*) –
- **_user** (*Optional[Union[str, int]]*) –
- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** (Path) –
- **_dependencies** (*Iterable[Dependency]*) –
- **_interactive** (*bool*) –

**_after_create**()

**_interactive: bool = False**

**run_interactively**(*run_interactively=True*)

**Parameters**
**run_interactively** (*bool*) –

**Return type**
None

class hallyd.services.**Runnable**

Bases: ABC

**exception _FinishAndReboot**

Bases: BaseException

**_abc_impl = <_abc._abc_data object>**

**reboot**()

**Return type**
None

**abstract run**()

**Return type**
None

class hallyd.services.**Service**

Bases: *_Task*, ABC

**_abc_impl = <_abc._abc_data object>**

**abstract is_active**()

**Return type**
None

**abstract reload**()

**Return type**
None

**abstract restart**()

**Return type**
None

**abstract start**()

**Return type**
None

**abstract stop**()

**Return type**
None

class hallyd.services.**ServiceSetup**(*_name: Optional[str] = None*, *_runnable: Optional[ForwardRef('TRunnable')] = None*, *_description: Optional[str] = None*, *_user: Union[str, int, NoneType] = None*, *_group: Union[str, int, NoneType] = None*, *_discard_output: bool = False*, *_working_dir:* hallyd.fs.Path *= Path('/')*, *_dependencies: Iterable[*hallyd.services._TaskSetup._WithDependencies.Dependency*] = ()*, *_start_instantly: bool = True*, *_enabled: bool = True*, *_restart_delay: Optional[int] = 2*, *_as_oneshot: bool = False*, *_startup_context: Optional[Iterable[str]] = None*, *_post_stop: Optional[ForwardRef('TCmdLineRunnable')] = None*, *_options: Iterable[Callable[[str], NoneType]] = ()*)

Bases: *_WithDependencies*, *_TaskSetup*

**Parameters**

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –
- **_description** (*Optional[str]*) –

- **_user** (*Optional[Union[str, int]]*) –
- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** (Path) –
- **_dependencies** (*Iterable[Dependency]*) –
- **_start_instantly** (*bool*) –
- **_enabled** (*bool*) –
- **_restart_delay** (*Optional[int]*) –
- **_as_oneshot** (*bool*) –
- **_startup_context** (*Optional[Iterable[str]]*) –
- **_post_stop** (*Optional[Union[Iterable[Union[str, Path]], str, Path]]*) –
- **_options** (*Iterable[Callable[[str], None]]*) –

**__DEFAULT_RESTART_DELAY = 2**

**_after_create**()

**_as_oneshot: bool = False**

**_enabled: bool = True**

**_options: Iterable[Callable[[str], None]] = ()**

**_post_stop: Optional[TCmdLineRunnable] = None**

**_restart_delay: Optional[int] = 2**

**_start_instantly: bool = True**

**_startup_context: Optional[Iterable[str]] = None**

**as_oneshot**(*as_oneshot=True*)

**Parameters**
**as_oneshot** (*bool*) –

**Return type**
None

**do_not_restart**(*do_not_restart=True*)

**Parameters**
**do_not_restart** (*bool*) –

**Return type**
None

**do_not_start_instantly**(*do_not_start_instantly=True*, ***, *not_even_enable=None*)

**Parameters**

- **do_not_start_instantly** (*bool*) –
- **not_even_enable** (*Optional[bool]*) –

**Return type**
None

**with_option**(*option*)

**Parameters**
**option** (*Callable[[str], None]*) –

**Return type**
None

**with_post_stop_command**(*post_stop_command*)

**Parameters**
**post_stop_command** (*str*) –

**Return type**
None

**with_restart_delay**(*restart_delay*)

**Parameters**
**restart_delay** (*Optional[int]*) –

**Return type**
None

**with_startup_context**(*startup_context*)

**Parameters**
**startup_context** (*str*) –

**Return type**
None

**class** hallyd.services.**_CalendarTaskBackend**

Bases: ABC

**_abc_impl = <_abc._abc_data object>**

**abstract calendar_task**(*name*)

**Parameters**
**name** (*str*) –

**Return type**
CalendarTask

**abstract create_calendar_task**(*setup*)

**Parameters**
**setup** (CalendarTaskSetup) –

**Return type**
None

**abstract remove_calendar_task**(*name*)

**Parameters**
**name** (*str*) –

**Return type**
None

class hallyd.services._FunctionRunnable(*func*)

Bases: *Runnable*

**Parameters**
**func** (*TCallableRunnable*) –

_abc_impl = <_abc._abc_data object>

run()

class hallyd.services._IntervalTaskBackend

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract create_interval_task(*setup*)

**Parameters**
**setup** (IntervalTaskSetup) –

**Return type**
None

abstract interval_task(*name*)

**Parameters**
**name** (*str*) –

**Return type**
IntervalTask

abstract remove_interval_task(*name*)

**Parameters**
**name** (*str*) –

**Return type**
None

class hallyd.services._NextBootTaskBackend

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract create_next_boot_task(*setup*)

**Parameters**
**setup** (NextBootTaskSetup) –

**Return type**
None

abstract next_boot_task(*name*)

**Parameters**
**name** (*str*) –

**Return type**
NextBootTask

abstract remove_next_boot_action(*name*)

**Parameters**
**name** (*str*) –

**Return type**
None

class hallyd.services._ServiceBackend

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract create_service(*setup*)

**Parameters**
**setup** (ServiceSetup) –

**Return type**
None

abstract remove_service(*name*)

**Parameters**
**name** (*str*) –

**Return type**
None

abstract service(*name*)

**Parameters**
**name** (*str*) –

**Return type**
Service

class hallyd.services._SystemdBackend

Bases: *_ServiceBackend*, *_CalendarTaskBackend*, *_IntervalTaskBackend*, *_NextBootTaskBackend*

static _SystemdBackend__create(*setup*)

**Return type**
None

_SystemdBackend__lock = <hallyd.io._Lock object>

static _SystemdBackend__remove(*service*)

**Parameters**
**service** (*_Unit*) –

**Return type**
None

static _SystemdBackend__runnable_to_spec_info(*runnable*, *unit_spec*, *service_spec_path*, *, *interactive=False*, *once=False*, *discard_output=False*)

**Parameters**

- **runnable** (*TRunnable*) –

- **unit_spec** (_UnitSpec) –
- **service_spec_path** (*Optional[Path]*) –
- **interactive** (*bool*) –
- **once** (*bool*) –
- **discard_output** (*bool*) –

**Return type**
str

static **_SystemdBackend__setup_to_object_spec**(*setup*, *service_spec_path=None*)

**Parameters**

- **setup** (_TaskSetup) –
- **service_spec_path** (*Optional[Path]*) –

**Return type**
*_UnitSpec*

**_SystemdBackend__unit**(*name*, ***, *type*, *also_allow_types=()*)

**Parameters**

- **type** (*str*) –
- **also_allow_types** (*Iterable[str]*) –

**Return type**
*_Unit*

class **_Unit**(*name*, *unit_type*)

Bases: *Service*, *IntervalTask*, *CalendarTask*

**Parameters**

- **name** (*str*) –
- **unit_type** (*str*) –

**_Unit__systemctl**(**args*)

**Return type**
None

**_abc_impl = <_abc._abc_data object>**

**disable**()

**enable**()

property **full_name**

**is_active**()

property **is_enabled**

property **name**

override(*wants=()*, *requires=()*, *after=()*, *before=()*, *wanted_by=()*, *required_by=()*, *reset_wants=False*, *reset_requires=False*, *reset_after=False*, *reset_before=False*, *reset_wanted_by=False*, *reset_required_by=False*)

**Parameters**
- **wants** (*list[str]*) –
- **requires** (*list[str]*) –
- **after** (*list[str]*) –
- **before** (*list[str]*) –
- **wanted_by** (*list[str]*) –
- **required_by** (*list[str]*) –
- **reset_wants** (*bool*) –
- **reset_requires** (*bool*) –
- **reset_after** (*bool*) –
- **reset_before** (*bool*) –
- **reset_wanted_by** (*bool*) –
- **reset_required_by** (*bool*) –

**Return type**
None

reload()

restart()

start()

stop()

property unit_type

class _UnitSpec

Bases: object

class _MultiDict

Bases: object

add_value(*key*, *value*)

**Parameters**
- **key** (*str*) –
- **value** (*str*) –

clear_values(*key*)

**Parameters**
**key** (*str*) –

set_value(*key*, *value*)

**Parameters**
- **key** (*str*) –
- **value** (*str*) –

property install: *_MultiDict*

property service: *_MultiDict*

property timer: *_MultiDict*

property unit: *_MultiDict*

_abc_impl = <_abc._abc_data object>

static _short_name(*name*)

**Parameters**
**name** (*str*) –

**Return type**
tuple[str, *Optional*[str]]

calendar_task(*name*)

create_calendar_task(*setup*)

create_interval_task(*setup*)

create_next_boot_task(*setup*)

create_service(*setup*)

interval_task(*name*)

next_boot_task(*name*)

remove_calendar_task(*name*)

remove_interval_task(*name*)

remove_next_boot_action(*name*)

remove_service(*name*)

service(*name*)

class hallyd.services._Task

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract disable()

**Return type**
None

abstract enable()

**Return type**
None

abstract property is_enabled: bool

abstract property name: str

class hallyd.services._TaskSetup(*_name: Optional[str] = None*, *_runnable: Optional[ForwardRef('TRunnable')] = None*, *_description: Optional[str] = None*, *_user: Union[str, int, NoneType] = None*, *_group: Union[str, int, NoneType] = None*, *_discard_output: bool = False*, *_working_dir:* hallyd.fs.Path *= Path('/')*)

Bases: object

**Parameters**

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –
- **_description** (*Optional[str]*) –
- **_user** (*Optional[Union[str, int]]*) –
- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** (Path) –

class **_WithDependencies**(*_dependencies: Iterable[hallyd.services._TaskSetup._WithDependencies.Dependency] = ()*)

Bases: object

**Parameters**
**_dependencies** (*Iterable[Dependency]*) –

class **Dependency**(*name: str*, *afterwards: bool*, *success_required: bool*, *optional: bool*)

Bases: object

**Parameters**
- **name** (*str*) –
- **afterwards** (*bool*) –
- **success_required** (*bool*) –
- **optional** (*bool*) –

**afterwards: bool**

**name: str**

**optional: bool**

**success_required: bool**

**_dependencies: Iterable[*Dependency*] = ()**

**add_dependency**(*unit_name*, ***, *afterwards=False*, *success_required=False*, *optional=False*)

**Parameters**
- **unit_name** (*str*) –
- **afterwards** (*bool*) –
- **success_required** (*bool*) –
- **optional** (*bool*) –

**_after_create**()

**Return type**
None

**_before_create**()

**Return type**
None

**_description: Optional[str] = None**

**_discard_output: bool = False**

**_group: Optional[Union[str, int]] = None**

**_name:** Optional[str] = None

**_runnable:** Optional[Union[Iterable[Union[str, Path]], str, Path, *Runnable*, Callable[[], None], Callable[[*Runnable*], None]]] = None

**_user:** Optional[Union[str, int]] = None

**_working_dir:** *Path* = Path('/')

**Parameters**
**paths** (*Union[str, Path]*) –

**Return type**
*Path*

**description**(*description*)

**Parameters**
**description** (*str*) –

**Return type**
None

**discard_output**(*discard_output=True*)

**Parameters**
**discard_output** (*bool*) –

**Return type**
None

**run_as_user**(*user*, *group=None*)

**Parameters**

- **user** (*Union[str, int]*) –
- **group** (*Optional[Union[str, int]]*) –

**Return type**
None

**run_in_working_dir**(*working_dir*)

**Parameters**
**working_dir** (*Union[str, Path]*) –

**Return type**
None

**class** hallyd.services.**_TimedTaskSetup**(*_name=None*, *_runnable=None*, *_description=None*, *_user=None*, *_group=None*, *_discard_output=False*, *_working_dir=Path('/')*)

Bases: *_TaskSetup*

**Parameters**

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –
- **_description** (*Optional[str]*) –
- **_user** (*Optional[Union[str, int]]*) –

- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** (Path) –

**_start_instantly: bool = False**

**start_instantly**(*start_instantly=True*)

**Parameters**
**start_instantly** (*bool*) –

**Return type**
None

hallyd.services.**_create_setup_context**(*name*, *runnable*, *setup_type*, *create_func*)

**Parameters**

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –
- **setup_type** (*type[~_TTaskSetup]*) –

**Return type**
*ContextManager*[*_TTaskSetup*]

hallyd.services.**calendar_task**(*runnable*)

**Parameters**
**runnable** (*Union[CalendarTask, str]*) –

**Return type**
CalendarTask

hallyd.services.**create_calendar_task**(*name*, *runnable*)

**Parameters**

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –

**Return type**
*ContextManager*[CalendarTaskSetup]

hallyd.services.**create_interval_task**(*name*, *runnable*)

**Parameters**

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –

**Return type**
*ContextManager*[IntervalTaskSetup]

hallyd.services.**create_next_boot_task**(*name*, *runnable*)

**Parameters**

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –

**Return type**
*ContextManager*[NextBootTaskSetup]

hallyd.services.**create_service**(*name*, *runnable*)

**Parameters**

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –

**Return type**
*ContextManager*[ServiceSetup]

hallyd.services.**interval_task**(*runnable*)

**Parameters**
**runnable** (*Union[IntervalTask, str]*) –

**Return type**
IntervalTask

hallyd.services.**next_boot_task**(*runnable*)

**Parameters**
**runnable** (*Union[NextBootTask, str]*) –

**Return type**
NextBootTask

hallyd.services.**remove_calendar_task**(*runnable*)

**Parameters**
**runnable** (*Union[CalendarTask, str]*) –

**Return type**
None

hallyd.services.**remove_interval_task**(*runnable*)

**Parameters**
**runnable** (*Union[IntervalTask, str]*) –

**Return type**
None

hallyd.services.**remove_next_boot_action**(*runnable*)

**Parameters**
**runnable** (*Union[NextBootTask, str]*) –

**Return type**
None

hallyd.services.**remove_service**(*runnable*)

**Parameters**
**runnable** (*Union[Service, str]*) –

**Return type**
None

hallyd.services.**service**(*runnable*)

**Parameters**
**runnable** (*Union[Service, str]*) –

**Return type**
Service

### 4.1.16 hallyd.subprocess module

**exception** hallyd.subprocess.**_ProcessInfoUnavailableError**

Bases: Exception

**exception** hallyd.subprocess.**_ProcessNotRunningError**

Bases: Exception

hallyd.subprocess.**check_call_with_stdin_string**(*cmd*, *, *stdin*, **kwargs*)

**Parameters**

- **cmd** (*list*) –
- **stdin** (*AnyStr*) –

**Return type**
None

hallyd.subprocess.**check_output_with_stdin_string**(*cmd*, *, *stdin*, **kwargs*)

**Parameters**

- **cmd** (*list*) –
- **stdin** (*AnyStr*) –

**Return type**
bytes

hallyd.subprocess.**is_process_running**(*process_permanent_id*)

**Parameters**
**process_permanent_id** (*str*) –

**Return type**
*Optional*[bool]

hallyd.subprocess.**pid_for_process_permanent_id**(*process_permanent_id*)

**Parameters**
**process_permanent_id** (*str*) –

**Return type**
int

hallyd.subprocess.**process_permanent_id_for_pid**(*pid*)

**Parameters**
**pid** (*int*) –

**Return type**
str

hallyd.subprocess.**start_function_in_new_process**(*func*, *args=None*, *kwargs=None*, ***, *interactive=False*, *capture_output=False*, *shell=()*)

**Parameters**
**func** (*Union[Callable, SupportsQualifiedName]*) –

**Return type**
*Popen*

hallyd.subprocess.**verify_tool_available**(*tool*)

Checks if a tool is installed (by calling it). Raises an exception if not.

**Parameters**
**tool** (*str*) –

**Return type**
None

### 4.1.17 hallyd.terminal module

**class** hallyd.terminal.**AnsiEscapedTextFormatter**(***, *with_rgb256*)

Bases: *TextFormatter*

**Parameters**
**with_rgb256** (*bool*) –

**__control_for_color**(*color*, *bare_block1_begin*, *bare_block2_begin*, *rgb256_code*)

**Parameters**

- **bare_block1_begin** (*int*) –
- **bare_block2_begin** (*int*) –
- **rgb256_code** (*int*) –

**_abc_impl = <_abc._abc_data object>**

**control_for_bg_color**(*color*)

**control_for_fg_color**(*color*)

**control_for_reset**()

**escape**(*s*)

**class** hallyd.terminal.**Color**

Bases: object

**as_bare_ansi_terminal_code**()

**Return type**
int

as_html()

**Return type**
str

as_rgb256()

**Return type**
tuple[float, float, float]

as_rgb_normed()

**Return type**
tuple[float, float, float]

as_xterm256_terminal_code()

**Return type**
int

by_bare_ansi_terminal_code(*code*)

by_html(*html_color*)

**Parameters**
**html_color** (*str*) –

**Return type**
Color

by_rgb256(*r*, *g*, *b*)

**Parameters**
- **r** (*float*) –
- **g** (*float*) –
- **b** (*float*) –

**Return type**
Color

by_rgb_normed(*r*, *g*, *b*)

**Parameters**
- **r** (*float*) –
- **g** (*float*) –
- **b** (*float*) –

**Return type**
Color

by_xterm256_terminal_code(*code*)

class hallyd.terminal.**PlainTextFormatter**

Bases: *TextFormatter*

_abc_impl = <_abc._abc_data object>

control_for_bg_color(*color*)

control_for_fg_color(*color*)

control_for_reset()

escape(*s*)

class hallyd.terminal.**Style**

Bases: object

has_key(*key*)

**Parameters**
**key** (*str*) –

**Return type**
bool

value(*key*)

**Return type**
*Any*

class hallyd.terminal.**Text**(*krz_shl_doc_xml*)

Bases: object

**Parameters**
**krz_shl_doc_xml** (*str*) –

__smart_strip()

**Parameters**
**s** (*str*) –

**Return type**
str

__traverse(*node_start_func*, *node_stop_func*)

**Return type**
Text

property as_xml: str

static by_plain_text(*text*, *, *smart_strip=True*)

**Parameters**

- **text** (*str*) –
- **smart_strip** (*bool*) –

**Return type**
Text

static by_xml(*krz_shl_doc_xml*)

**Parameters**
**krz_shl_doc_xml** (*str*) –

**Return type**
Text

format(*formatter*, *, *style*=<*hallyd.terminal._DefaultStyle object*>)

**Parameters**

- **formatter** (TextFormatter) –
- **style** (Style) –

**Return type**
str

indent(*width*, *, *char*=' ')

**Parameters**

- **width** (*int*) –
- **char** (*str*) –

**Return type**
Text

line_wrap_at_maximum_width(*width*)

**Parameters**
**width** (*Optional[int]*) –

**Return type**
Text

class hallyd.terminal.**TextFormatter**

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract control_for_bg_color(*color*)

**Parameters**
**color** (Color) –

**Return type**
str

abstract control_for_fg_color(*color*)

**Parameters**
**color** (Color) –

**Return type**
str

abstract control_for_reset()

**Return type**
str

abstract escape(*s*)

**Parameters**
**s** (*str*) –

**Return type**
str

class hallyd.terminal._BareAnsiTerminalColorSpace

Bases: *_TerminalColorSpace*

_COLORS = ((0, 0, 0), (128, 0, 0), (0, 128, 0), (128, 128, 0), (0, 0, 128), (128, 0, 128), (0, 128, 128), (192, 192, 192), (128, 128, 128), (255, 0, 0), (0, 255, 0), (255, 255, 0), (0, 0, 255), (255, 0, 255), (0, 255, 255), (255, 255, 255))

_abc_impl = <_abc._abc_data object>

color_to_index(*r*, *g*, *b*)

index_to_color(*idx*)

class hallyd.terminal._DefaultStyle

Bases: *Style*

class hallyd.terminal._TerminalColorSpace

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract color_to_index(*r*, *g*, *b*)

**Parameters**

- **r** (*float*) –
- **g** (*float*) –
- **b** (*float*) –

**Return type**
int

abstract index_to_color(*idx*)

**Parameters**
**idx** (*int*) –

**Return type**
tuple[float, float, float]

class hallyd.terminal._XTerm256TerminalColorSpace

Bases: *_TerminalColorSpace*

_COLOR_INDEX_BEGIN = 16

_COLOR_SYMBOLS_PER_CHANNEL = 6

_COLOR_VALUE_BEGIN = 95

_COLOR_VALUE_END = 255

_COLOR_VALUE_STEP = 40

_END_INDEX_BEGIN = 256

_GRAYSCALE_INDEX_BEGIN = 232

_GRAYSCALE_SYMBOLS_PER_CHANNEL = 24

**_GRAYSCALE_VALUE_BEGIN = 8**

**_GRAYSCALE_VALUE_END = 238**

**_GRAYSCALE_VALUE_STEP = 10**

**_XTerm256TerminalColorSpace__color_to_index__colors**(*r*, *g*, *b*)

**_XTerm256TerminalColorSpace__color_to_index__grayscale**(*r*, *g*, *b*)

**_XTerm256TerminalColorSpace__color_to_index__helper__channel**(*val*, *symbol_count*, *value_begin*, *value_step*)

**_XTerm256TerminalColorSpace__cost**(*c1*, *c2*)

**_abc_impl = <_abc._abc_data object>**

**color_to_index**(*r*, *g*, *b*)

**index_to_color**(*idx*)

hallyd.terminal.**is_interactive_shell**()

**Return type**
bool

hallyd.terminal.**is_superuser**()

hallyd.terminal.**terminal_width**(**, fallback_to=80*)

**Parameters**
**fallback_to** (*int*) –

**Return type**
int

### 4.1.18 hallyd.text module

hallyd.text.**match_format_string**(*pattern*, *string*)

**Parameters**

- **pattern** (*str*) –
- **string** (*str*) –

**Return type**
dict[str, str]

hallyd.text.**pretty_print_xml**(*input_xml*)

**Parameters**
**input_xml** (*str*) –

**Return type**
str

### 4.1.19 hallyd.typing module

class hallyd.typing.**SupportsQualifiedName**(**args*, ***kwargs*)

Bases: Protocol

**_abc_impl = <_abc._abc_data object>**

**_is_protocol = True**

**_is_runtime_protocol = True**

# PYTHON MODULE INDEX

## h

# INDEX

## Symbols

## A



## B

## C

## D



## E

## F



## G



## H



## I

## L



## M



## N

## O



## P



## R

## S

## T



## U



## V



## W