## Package ‘outsider.base’

June 14, 2020

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Base Package for ‘Outsider’</td>
</tr>
<tr>
<td>Version</td>
<td>0.1.3</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Dom Bennett <a href="mailto:dominic.john.bennett@gmail.com">dominic.john.bennett@gmail.com</a></td>
</tr>
<tr>
<td>Description</td>
<td>Base package for ‘outsider’ <a href="https://github.com/ropensci/outsider">https://github.com/ropensci/outsider</a>. The ‘outsider’ package and its sister packages enable the installation and running of external, command-line software within R. This base package is a key dependency of the user-facing ‘outsider’ package as it provides the utilities for interfacing between ‘Docker’ <a href="https://www.docker.com">https://www.docker.com</a> and R. It is intended that end-users of ‘outsider’ do not directly work with this base package.</td>
</tr>
<tr>
<td>License</td>
<td>MIT + file LICENSE</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>LazyData</td>
<td>true</td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>7.1.0</td>
</tr>
<tr>
<td>SystemRequirements</td>
<td>docker (&gt;=18.0.0)</td>
</tr>
<tr>
<td>BugReports</td>
<td><a href="https://github.com/ropensci/outsider.base/issues">https://github.com/ropensci/outsider.base/issues</a></td>
</tr>
<tr>
<td>Language</td>
<td>en-GB</td>
</tr>
<tr>
<td>Depends</td>
<td>R (&gt;= 3.3.0)</td>
</tr>
<tr>
<td>Imports</td>
<td>utils (&gt;= 3.1), crayon, devtools (&gt;= 1.1), jsonlite (&gt;= 1.1), sys (&gt;= 2.1), yaml (&gt;= 2.0), callr (&gt;= 3.0.0), withr (&gt;= 2.0), tibble, cli, praise</td>
</tr>
<tr>
<td>Suggests</td>
<td>ssh, testthat (&gt;= 2.0)</td>
</tr>
<tr>
<td>NeedsCompilation</td>
<td>no</td>
</tr>
<tr>
<td>Author</td>
<td>Dom Bennett [aut, cre] (<a href="https://orcid.org/0000-0003-2722-1359">https://orcid.org/0000-0003-2722-1359</a>), Hannes Hettling [ctb] (<a href="https://orcid.org/0000-0003-4144-2238">https://orcid.org/0000-0003-4144-2238</a>), Daniele Silvestro [ctb] (<a href="https://orcid.org/0000-0003-0100-0961">https://orcid.org/0000-0003-0100-0961</a>), Rutger Vos [ctb] (<a href="https://orcid.org/0000-0001-9254-7318">https://orcid.org/0000-0001-9254-7318</a>)</td>
</tr>
</tbody>
</table>
R topics documented:

arglist_get .................................................. 3
arglist_parse .................................................. 3
console-methods ............................................. 4
container-class .............................................. 5
default_log_set ............................................. 6
dirpath_get ................................................... 7
docker_build .................................................. 7
docker_cmd ..................................................... 8
docker_cp ....................................................... 9
docker_ids_get ............................................... 9
docker_img_ls ............................................... 10
docker_img_rm ............................................... 10
docker_ps_count ............................................ 11
docker_pull .................................................... 11
eexec_internal ............................................... 12
eexec_wait ..................................................... 13
filestosend_get .............................................. 14
image_install ............................................... 15
is_docker_available ....................................... 15
is_docker_installed ....................................... 16
is_docker_running ......................................... 16
is_filepath ................................................... 17
is_installed .................................................. 17
is_server_connected ....................................... 18
log_get ........................................................ 18
log_set ........................................................ 19
meta_get ....................................................... 20
modules_list .................................................. 20
outsider-class ............................................... 21
outsider.base ................................................. 23
pkg_install ................................................... 23
server_connect .............................................. 23
server_disconnect ......................................... 24
server_download ............................................ 24
server_fetch .................................................. 25
server_upload ............................................... 25
to_basename .................................................. 27
uninstall ....................................................... 27
wd_get ........................................................ 28

Alexandre Antonelli [ctb] (<https://orcid.org/0000-0003-1842-9297>),
Anna Kryštallí [rev]

Repository CRAN

Date/Publication 2020-06-14 14:40:13 UTC
**arglist_get**

*Generate vector of arguments*

**Description**

Convert all the arguments passed to this function, including those contained in '...', into character vector.

**Usage**

```r
arglist_get(...)
```

**Arguments**

`...`  
Any number of arguments

**Value**

Character vector

**Examples**

```r
calllib(outsider.base)
# return a character vector of all arguments provided
arglist_get('any', 'number', 'of', 'arguments...', 'number', 'or', 'anything', 1L, TRUE)
```

**arglist_parse**

*Normalise arguments for docker container*

**Description**

Utility function for parsing the arguments provided by a user. Drop any specified key:value pairs with `keyvals_to_drop` or drop any specific values `vals_to_drop`. With `normalise_paths` as `TRUE`, all filepaths in the arglist will be converted to basenames.

**Usage**

```r
arglist_parse(
  arglist,
  keyvals_to_drop = NULL,
  vals_to_drop = NULL,
  normalise_paths = TRUE
)
```
Arguments

arglist Arguments as character vector
keyvals_to_drop Argument keys to drop, e.g. -wd.
vals_to_drop Specific values to drop, e.g. --verbose.
normalise_paths Reduce paths to basenames? Default, TRUE.

Details

It is important the file paths are normalised, because they will not be available to the Docker container. The only files available will be those that have been transferred to the container as determined through the outsider\_init. These files will be located in the same directory as where the function is called and require no absolute file path.

Value

Character vector

Examples

library(outsider.base)
wd <- file.path(tempdir(), 'results')
dir.create(wd)
arglist <- c('-a', 10, '-b', 'model2', '-wd', wd, '--unwanted')
# drop unwanted key:value pairs
(arglist_parse(arglist = arglist, keyvals_to_drop = '-wd',
               normalise_paths = FALSE))
# drop unwanted argument values
(arglist_parse(arglist = arglist, vals_to_drop = '--unwanted',
               normalise_paths = FALSE))
# make paths relative, necessary for Docker:
 # paths must be relative to the working directory in the container
(arglist_parse(arglist = arglist, normalise_paths = TRUE))

# clean-up
unlink(wd, recursive = TRUE)

Description

Print to console using colours.
Usage

char(x)
stat(...)
func(x)
cat_line(...)

Arguments

x Character
... Objects to print

container-class  Docker container class and methods

Description

Return a list class that describes a Docker container. The resulting class object comes with a series of convenience methods for starting, stopping and interacting with a container.

Usage

container_init(pkgnm)

## S3 method for class 'container'
start(x)

## S3 method for class 'container'
halt(x)

## S3 method for class 'container'
exec(x, ...)

## S3 method for class 'container'
status(x)

## S3 method for class 'container'
copy(x, send = NULL, rtrn = NULL)

## S3 method for class 'container'
run(x, cmd, args)
**Arguments**

- **pkgnm**: Package name
- **x**: container
- **...**: Arguments
- **send**: Filepaths to send from host computer to container.
- **rtrn**: Directory on host computer where returning files should be sent.
- **cmd**: Command name, character
- **args**: List or vector of arguments, character

**Details**

All outsider modules have a `working_dir/` in which generated files are created and initiation files must be for the program to use. Files must be sent to this working directory and then returned before and after the program has run.

If no `send` or `rtrn` specified, returns TRUE.

**Value**

A list of class `container` with the following items:

- **pkgnm**: Package name of the outsider module
- **prgrm**: Command to be called in the container
- **cntnr**: Unique Docker container name
- **img**: Image ID

**See Also**

Other private-docker: `docker_build()`, `docker_cmd()`, `docker_cp()`, `docker_img_rm()`, `docker_ps_count()`, `docker_pull()`

---

**default_log_set**  
*Set default log streams*

**Description**

By default all streams are printed to console with the exception of `docker_out`.

**Usage**

`default_log_set()`
**dirpath_get**  
*Convert file path to directory path*

**Description**

Takes a file path and converts it to its directory path by dropping the file name and extension. If `flpth` is already a directory path, the argument will be returned unchanged. If nothing is provided, nothing is returned (i.e. character(0)).

**Usage**

```r
dirpath_get(flpth)
```

**Arguments**

- `flpth`  
  File path for which directory path will be returned.

**Value**

Character

**Examples**

```r
library(outsider.base)
# get the parent directory from a filepath
drpth <- tempdir()
flpth <- file.path(drpth, 'testfile')
file.create(flpth)
(dirpath_get(flpth = flpth) == drpth)
(dirpath_get(flpth = drpth) == drpth)
file.remove(flpth)
```

---

**docker_build**  
*Build a docker image*

**Description**

Runs `run build` command.

**Usage**

```r
docker_build(img, url_or_path, tag = "latest")
```

**Arguments**

- `img`  
  Image name

- `url_or_path`  
  Dockerfile URL

- `tag`  
  Docker tag, default 'latest'
**docker_cmd**

**Value**

Logical

**See Also**

Other private-docker: `container-class.docker_cmd()`, `docker_cp()`, `docker_img_rm()`, `docker_ps_count()`, `docker_pull()`

---

**docker_cmd**  |  *Run docker command*
---

**Description**

Runs a docker command with provided arguments

**Usage**

```
docker_cmd(args, std_out = TRUE, std_err = TRUE)
```

**Arguments**

- **args**  Vector of arguments
- **std_out** if and where to direct child process STDOUT. See `sys::exec`.
- **std_err** if and where to direct child process STDERR. See `sys::exec`.

**Value**

Logical

**See Also**

Other private-docker: `container-class.docker_build()`, `docker_cp()`, `docker_img_rm()`, `docker_ps_count()`, `docker_pull()`
**docker_cp**

**Copy files to and from container**

**Description**
Copy files to and from running Docker container

**Usage**
docker_cp(origin, dest)

**Arguments**
- origin Origin filepath
- dest Destination filepath

**Details**
Container folders are indicated with [container_id]:[filepath]. Files are uploaded/downloaded to/from the server based on the presence of ":" in origin/dest file paths.

**Value**
Logical

**See Also**
Other private-docker: container-class, docker_build(), docker_cmd(), docker_img_rm(), docker_ps_count(), docker_pull()

---

**docker_ids_get**

**Get docker names for a module**

**Description**
Return the image and container names for a module. Will attempt to build/pull image if missing.

**Usage**
docker_ids_get(pkgnm)

**Arguments**
- pkgnm Package name of module
**docker_img_ls**

**List the number of installed images**

**Description**

Return a table of all the available Docker images.

**Usage**

```r
docker_img_ls()
```

**Value**

tibble

**See Also**

Other ids: `meta_get()`, `modules_list()`

---

**docker_img_rm**

**Remove docker image**

**Description**

Deletes docker image from system.

**Usage**

```r
docker_img_rm(img)
```

**Arguments**

`img` Image name

**Value**

Logical

**See Also**

Other private-docker: `container-class`, `docker_build()`, `docker_cmd()`, `docker_cp()`, `docker_ps_count()`, `docker_pull()`
docker_ps_count  Count docker processes

Description
Count the number of running docker containers.

Usage
docker_ps_count()

Details
Use this to avoid creating multiple containers with the same ID.

Value
Integer

See Also
Other private-docker: container-class, docker_build(), docker_cmd(), docker_cp(), docker_img_rm(), docker_pull()

docker_pull  Pull an image from DockerHub.

Description
Speeds up outsider module installation by downloading compiled images.

Usage
docker_pull(img, tag = "latest")

Arguments
img Image name
tag Tag version, default latest.

Value
Logical

See Also
Other private-docker: container-class, docker_build(), docker_cmd(), docker_cp(), docker_img_rm(), docker_ps_count()
exec_internal

Execute system commands and wait for response

Description

Passes arguments to `sys::exec_internal`, if a server is connected arguments are passed to `ssh::ssh_exec_internal` instead.

Usage

```r
exec_internal(
  cmd,
  args = NULL,
  std_in = NULL,
  error = TRUE,
  timeout = 0,
  with_ssh = TRUE
)
```

Arguments

- **cmd**: Command
- **args**: Arguments
- **std_in**: Standard in
- **error**: Call an error? T/F
- **timeout**: Timeout
- **with_ssh**: Try and run with ssh, default TRUE

Value

- logical

See Also

Other private-sys: `exec_wait()`
exec_wait

**Description**

Passes arguments to `sys::exec_wait`, if a server is connected arguments are passed to `ssh::ssh_exec_wait` instead.

**Usage**

```r
exec_wait(
  cmd,
  args = NULL,
  std_out = stdout(),
  std_err = stderr(),
  std_in = NULL,
  timeout = 0,
  with_ssh = TRUE
)
```

**Arguments**

- **cmd**  Command
- **args**  Arguments
- **std_out**  Standard out
- **std_err**  Standard error
- **std_in**  Standard in
- **timeout**  Timeout
- **with_ssh**  Try and run with ssh, default TRUE

**Value**

logical

**See Also**

Other private-sys: [exec_internal](#)
filestosend_get  

_Determine which arguments are filepaths_

**Description**

Return filepaths from arguments. These filepaths can then be used to identify files/folders for sending to the Docker container.

**Usage**

```r
filestosend_get(arglist, wd = NULL)
```

**Arguments**

- `arglist`: Character vector of arguments
- `wd`: Working directory in which to look for files

**Value**

Character vector

**Examples**

```r
library(outsider.base)
# set-up: create wd and files to send
wd <- file.path(tempdir(), 'results')
dir.create(wd)
file1 <- file.path(wd, 'file1')
file.create(file1)
file2 <- file.path(wd, 'file2')
file.create(file2)

# identify files to be sent to container
arglist <- c('-in', file1, '-out', file2)
(filestosend_get(arglist = arglist))
# works with -wd
arglist <- c('-in', 'file1', '-out', 'file2', '-wd', wd)
(filestosend_get(arglist = arglist, wd = wd))

# clean-up
unlink(wd, recursive = TRUE)
```
Install module’s image

Description
Install the Docker image for an outsider module after the module package has been installed.

Usage
image_install(pkgnm, tag = "latest", pull = TRUE)

Arguments
pkgnm         Name of module’s R package
tag           Docker tag, default 'latest'
pull          Pull from Docker Hub or build locally? Default, FALSE.

Value
Integer

Check if Docker is installed and running

Description
Raises an error if docker is not available.

Usage
is_docker_available(call_error = TRUE)

Arguments
call_error    Call an error if no Docker detected? Default TRUE.
is_docker_installed  
*Check if Docker is installed*

**Description**

Docker is required to run outsider. This function tests whether Docker is installed.

**Usage**

`is_docker_installed()`

**Value**

Logical

**See Also**

Other private-check: `is_docker_running()`

---

is_docker_running  
*Check if Docker is running*

**Description**

Docker is required to run outsider. This function tests whether Docker is running.

**Usage**

`is_docker_running()`

**Value**

Logical

**See Also**

Other private-check: `is_docker_installed()`
is_filepath

Description
Return TRUE or FALSE for whether character(s) is a valid filepath.

Usage
is_filepath(x)

Arguments
x
Character vector

Value
Logical

See Also
Other private: log_get(), to_basename()

is_installed

Description
Return TRUE if module is installed.

Usage
is_installed(pkgnm)

Arguments
pkgnm
Package name

Value
logical(1)
**is_server_connected**  
*Is server connected?*

**Description**

Return TRUE if an ssh session exists with which outsider can interact.

**Usage**

`is_server_connected()`

**Details**

This requires installation of ssh package.

**Value**

logical

**See Also**

Other private-server: `server_download()`, `server_fetch()`, `server_upload()`

---

**log_get**  
*Return log stream option*

**Description**

Return the log stream setting for a given stream. If the stream is not set, the function will return TRUE (i.e. prints to console).

**Usage**

`log_get(log = c("program_out", "program_err", "docker_out", "docker_err"))`

**Arguments**

- `log`  
  Log stream

**See Also**

Other private: `is_filepath()`, `to_basename()`
log_set

Set log streams for console output

Description
Set if and where to send the console streams of the outsider modules.

Usage
log_set(log, val)

Arguments
log
Output stream one of program_out, program_err, docker_out or docker_err
val
Either logical, file or connection.

Details
See 'sys::exec'.

Examples

library(outsider.base)

# Manually install example module
# outsider.base contains the hello.world module in its package files
pkgnm <- 'om..hello.world'
mdl_flpth <- system.file('extdata', 'om..hello.world',
    package = "outsider.base")
# install and import (outsider::module_install performs these tasks)
pkg_install(flpth = mdl_flpth)
image_install(pkgnm = pkgnm)
# (outsider::module_import performs this task)
hello_world <- utils::getFromNamespace(x = 'hello_world', ns = pkgnm)

# control the log stream
# send output to file
tmpfl <- tempfile()
log_set(log = 'program_out', val = tmpfl)
hello_world()
(readLines(con = tmpfl))
file.remove(tmpfl)
# send docker and program output to console
log_set(log = 'program_out', val = TRUE)
log_set(log = 'docker_out', val = TRUE)
hello_world()

# clean-up
modules_list

uninstall(pkgnm)

meta_get  Get outsider module details

Description
Return a named list of all metadata associated with a module

Usage
meta_get(pkgnm)

Arguments
pkgnm Package name of module

Value
Named list

See Also
Other ids: docker_ids_get(), modules_list()

modules_list  List all installed outsider modules

Description
Return the R package names of all installed outsider modules

Usage
modules_list()

Value
Logical

See Also
Other ids: docker_ids_get(), meta_get()
outsider-class

Construct outsider object

Description

Returns an outsider object. The outsider object describes a outsider module’s program and arguments. The object is generated every time an outsider module program is called. It details the arguments of a call, the command as well as the files to send to the docker container.

Usage

```r
outsider_init(
  pkgnm,
  cmd = NA,
  arglist = NULL,
  wd = NULL,
  files_to_send = NULL,
  ignore_errors = FALSE
)
```

```r
run(x, ...)
```

## S3 method for class 'outsider'

```r
run(x, ...)
```

Arguments

- **pkgnm**: Name of the installed R package for the outsider module
- **cmd**: Command to be called in the container
- **arglist**: Arguments for command, character vector
- **wd**: Directory to which program generated files will be returned
- **files_to_send**: Files to be sent to container
- **ignore_errors**: Ignore raised errors? Default FALSE.
- **x**: outsider object
- **...**: Additional arguments

Details

The outsider module runs a docker container that acts like a separate machine on the host computer. All the files necessary for the program to be run must be sent to the remote machine before the program is called. The arguments, `wd` and `files_to_send` can all be defined after the outsider has been initiated using `$` notation. Once a outsider has been defined, the command can be run using `.run()`. The `arglist`, `wd` or `files_to_send` do not need to be defined for the outsider to be run.
Value

A list of class outsider with the following items:

- **pkgnm**: Package name of the outsider module
- **cmd**: Command to be called in the container
- **arglist**: Arguments for command, character vector
- **wd**: Directory to which program generated files will be returned
- **files_to_send**: Files to be sent to container
- **container**: Docker container object
- **ignore_errors**: Prevent errors being raised

Examples

```r
# Set-up: install "hello.world", ships with ubuntu
# we can make simple commands in bash via R using the module
library(outsider.base)

# Manually install example module
# outsider.base contains the hello.world module in its package files
pkgnm <- 'om..hello.world'
mdl_flpth <- system.file('extdata', 'om..hello.world',
    package = "outsider.base")
# install and import (outsider::module_install performs these tasks)
pkg_install(flpth = mdl_flpth)
image_install(pkgnm = pkgnm)

# Run echo
# create a outsider object that contains argument and Docker container details
otsdr <- outsider_init(pkgnm = pkgnm, cmd = 'echo', arglist = c('hello world!'))
# check details
print(otsdr)
# run the command
run(otsdr)

# Send a file
# an existing outsider object can be modified
tmppth <- tempdir()
flpth <- file.path(tmppth, 'testfile')
write(x = 'hello from within a file!', file = flpth)
otsdr$files_to_send <- flpth
otsdr$cmd <- 'cat'
otsdr$arglist <- 'testfile'
# check details
print(otsdr)
# run the command
run(otsdr)

# Return a file
# an existing outsider object can be modified
```
outsider.base

outsider.base: Base Package for outsider.

Description

For more information visit the outsider website (https://docs.ropensci.org/outsider/).

pkg_install

Install outsider module package

Description

Install outsider module’s package.

Usage

pkg_install(flpth, verbose = TRUE)

Arguments

flpth File path to module directory.
verbose Be verbose? Default TRUE.

Value

Logical(1)
server_connect

server_connect  
Connect to a server

Description
Connect to a server, make accessible to outsider and set-up for outsider interaction.

Usage
server_connect(session)

Arguments
session  ssh session, see 'ssh::ssh_connect'.

Details
This requires installation of ssh package.

Value
logical

See Also
Other public-server: server_disconnect()

Examples
library(outsider.base)

# NOT RUN
## Not run:
if (requireNamespace("ssh", quietly = TRUE)) {
  session <- ssh::ssh_connect(host = '[INSERT HOST IP]')
  server_connect(session = session)
  # run outsider.base commands, when finished
  server_disconnect()
}

## End(Not run)
server_disconnect  Disconnect from a server

Description
Disconnect from a server and remove from outsider

Usage
server_disconnect()

Details
This requires installation of ssh package.

Value
logical

See Also
Other public-server: server_connect()

Examples
library(outsider.base)

# NOT RUN
## Not run:
if (requireNamespace("ssh", quietly = TRUE)) {
  session <- ssh::ssh_connect(host = '[INSERT HOST IP]')
  server_connect(session = session)
  # run outsider.base commands, when finished
  server_disconnect()
}
## End(Not run)

server_download  Download from server

Description
Download file/folder from connected server. File is copied to a temporary folder before transferred to desired destination.
server_fetch

Usage
server_download(origin, dest)

Arguments
origin Origin filepath
dest Destination filepath

Value
Logical

See Also
Other private-server: is_server_connected(), server_fetch(), server_upload()

server_fetch Fetch server "session"

Description
Return connected session to server.

Usage
server_fetch(verbose)

Arguments
verbose Be verbose? Logical.

Details
See ‘ssh::ssh_connect’ for more details.

Value
ssh session

See Also
Other private-server: is_server_connected(), server_download(), server_upload()
server_upload

Description
Upload file/folder to connected server. File is placed in working dir on server.

Usage
server_upload(fl)

Arguments
fl File/folder to be transferred.

Details
This requires installation of ssh package.

Value
Logical

See Also
Other private-server: is_server_connected(), server_download(), server_fetch()

---

to_basename

Description
Return return a vector where all valid filepaths are converted to file basenames. E.g. "dir1/dir2/text.file" is converted to "text.file"

Usage
to_basename(x)

Arguments
x Character vector

Value
Character vector
See Also

Other private: is_filepath(), log_get()

uninstall  
Uninstall and remove a module

Description

Remove outsider module: uninstall package, delete Docker image.

Usage

uninstall(pkgnm)

Arguments

pkgnm  Package name

Details

If program is successfully removed TRUE is returned, else FALSE.

Value

Logical(1)

wd_get  
Return working directory

Description

Utility function for determining the working directory from arglist. The working directory can be determined from the arglist either by a key:value or an index. For example, the working directory may be determined by the key -wd in which case this function will identify whether this key exists in the arglist and will return its corresponding value. Alternatively, the working directory may be determined by the first argument (e.g. an input file), in which case setting i=1 will return the first argument in the arglist. If an input file is returned, a user can use dirpath_get to convert the file path to a directory path. If both key and i are provided, key takes precedence. If no key or i is provided and/or no working directory is found in the arguments, the function will return the R session's working directory. If no arguments are provided, returns empty character vector.

Usage

wd_get(arglist, key = NULL, i = NULL)
Arguments

arglist  Arguments as character vector
key     Argument key identifying the working directory, e.g. -wd
i       Index in the arglist that determines the working directory, e.g. 1.

Value

Character

Examples

library(outsider.base)
# wd is determined by key argument
arglist <- c('-a', 10, '-wd', 'path/to/wd', '-b', 'model2')
(wd_get(arglist = arglist, key = '-wd'))
# wd is determined by an index
arglist <- c('path/to/wd', '-a', 10, '-b', 'model2')
(wd_get(arglist = arglist, i = 1))