Package: adfExplorer2 (via r-universe)

September 11, 2024

Title Access and Manipulate Amiga Disk Files

Version 0.0.2

Author Pepijn de Vries [aut, cre] (0000-0002-7961-6646)

Maintainer Pepijn de Vries <pepijn.devries@outlook.com>

Description Amiga Disk Files (ADF) are virtual representations of 3.5
inch floppy disks for the Commodore Amiga. Most disk drives
from other systems (including modern drives) are not able to
read these disks. The adfExplorer2 package enables you to
establish R connections to files on such virtual DOS-formatted
disks, which can be use to read from and write to those files.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

LinkingTo cpp11

Collate 'adfExplorer2-package.R' 'blocks.R' 'compress.R' 'cpp11.R'
'connect_adf.R' 'device_info.R' 'device_create.R' 'helpers.R'
'directory.R' 'entry_info.R' 'imports.R' 'init.R' 'move.R'
'remove.R' 's3_methods.R' 'virtual_path.R'

Depends R (>= 3.5.0)

Imports methods, vctrs

Suggests adfExplorer, knitr, rmarkdown

VignetteBuilder knitr

URL https://pepijn-devries.github.io/adfExplorer2/

BugReports https://github.com/pepijn-devries/adfExplorer2/issues

Repository https://pepijn-devries.r-universe.dev

RemoteUrl https://github.com/pepijn-devries/adfExplorer2

RemoteRef master

RemoteSha 834c01e8b84053829c06b591973478430c53516b

Contents

Index   **[21](#)**

---

adf_directory	*Changing and creating directories on a virtual device*

Description

adf_directory() shows the current directory of a virtual device, when a file system is present. When connecting to or creating a new device, the current directory is the disk's root by default. To change the current directory, use adf_directory() in combination with the assign operator (<-).

Usage

```
adf_directory(dev, ...)

## S3 method for class 'adf_device'
adf_directory(dev, ...)

adf_directory(dev, ...) <- value

## S3 replacement method for class 'adf_device'
adf_directory(dev, ...) <- value

## S3 replacement method for class 'adf_device.character'
adf_directory(dev, ...) <- value

## S3 replacement method for class 'adf_device.virtual_path'
adf_directory(dev, ...) <- value

make_adf_dir(x, virtual_path, ...)
```

```
## S3 method for class 'adf_device'
make_adf_dir(x, virtual_path, ...)

## S3 method for class 'virtual_path'
make_adf_dir(x, virtual_path, ...)

## S3 method for class 'character'
make_adf_dir.adf_device(x, virtual_path, ...)

## S3 method for class 'virtual_path'
make_adf_dir.adf_device(x, virtual_path, ...)
```

Arguments

dev	The virtual adf device for which information needs to be obtained. It should be of class adf_device which can be created with [create_adf_device()](#) or [connect_adf()](#).
...	Ignored
value	A character string or a virtual_path (see [virtual_path()](#)) representing directory you wish to set as current.
x	An adf_device or virtual_path class object. The first specifies the device on which a directory needs to be created. The latter specifies both the directory and the device on which it needs to be created.
virtual_path	A character string or a virtual_path (see [virtual_path()](#)) specifying the name of the new directory to be created. Should be missing when x is of class virtual_path

Details

To create a new directory on a device use make_adf_dir() and use a full or relative path name to specify the new directory name.

See vignette("virtual_paths") for a note on file and directory names on the Amiga.

Value

make_adf_dir() returns the device connection. adf_directory() returns the current directory as a virtual_path class object.

Author(s)

Pepijn de Vries

Examples

```
## First setup a connection to a virtual device
adz_file <- system.file("example.adz", package = "adfExplorer2")
my_device <- connect_adf(adz_file)
```

```
## Show the current directory
adf_directory(my_device)

## Create a new directory
make_adf_dir(my_device, "DF0:s/newdir")

## Change the current dir to the new directory:
adf_directory(my_device) <- "DF0:s/newdir"

## Close the virtual device
close(my_device)
```

---

adf_entry_info   *Retrieve information from entry headers on virtual ADF devices*

Description

Retrieve information from entry (file and directory) headers on virtual ADF devices. Get information like entry name, modification date, file size etc.

Usage

```
adf_entry_info(x, virtual_path, ...)

## S3 method for class 'adf_device'
adf_entry_info(x, virtual_path, ...)

## S3 method for class 'virtual_path'
adf_entry_info.adf_device(x, virtual_path, ...)

## S3 method for class 'character'
adf_entry_info.adf_device(x, virtual_path, ...)

## S3 method for class 'virtual_path'
adf_entry_info(x, virtual_path, ...)

## S3 method for class 'adf_file_con'
adf_entry_info(x, virtual_path, ...)
```

Arguments

x	Either a virtual device or virtual path.
virtual_path	A [virtual_path()](#) to be checked for existence. Should be omitted when x is already a virtual path.
...	Ignored

Value

Returns a named list of entry properties. Elements included in the list depend on the type of entry (root, directory or file).

Author(s)

Pepijn de Vries

Examples

```
## First setup a connection to a virtual device
adz_file <- system.file("example.adz", package = "adfExplorer2")
my_device <- connect_adf(adz_file)

adf_entry_info(my_device, "DF0:")
adf_entry_info(my_device, "s")
adf_entry_info(my_device, "s/startup-sequence")

close(my_device)
```

---

adf_file_con   *Open a connection to a file on a virtual ADF device*

Description

Open a connection to a file on a virtual ADF device. The created connection (if valid) should be accepted by any R function that reads from or writes to a connection, such as readLines(), writeLines(), readBin(), writeBin(), etc.

Usage

```
adf_file_con(x, ..., writable = FALSE)

## S3 method for class 'adf_device'
adf_file_con(x, virtual_path, ..., writable = FALSE)

## S3 method for class 'character'
adf_file_con.adf_device(x, virtual_path, ..., writable = FALSE)

## S3 method for class 'virtual_path'
adf_file_con(x, ..., writable = FALSE)
```

Arguments

x   Either a connection to a virtual ADF device created with connect_adf(), or a
             virtual_path created with virtual_path().
...   Ignored.

writable	A `logical` value. When `TRUE` the connection can be used to write to the file on the virtual device. When `FALSE` it can only be used to read. Note that a writeable connection can only be setup on a virtual device that is not write protected.
virtual_path	Only required when `x` is a virtual device of class `adf_device`. In that case `virtual_path` should be a character string representing the path to the file on the virtual device. See also `vignette("virtual_paths")`.

Value

Returns an R connection that can be handled by any function that accepts a connection for reading or writing. Remember to call [close()](#) after use.

Author(s)

Pepijn de Vries

Examples

```
## First setup a connection to a virtual device
adz_file <- system.file("example.adz", package = "adfExplorer2")
my_device <- connect_adf(adz_file)

## Open a connection to a file on the virtual device
fcon <- adf_file_con(my_device, "DF0:s/startup-sequence")

## Read from the file
my_startup <- readLines(fcon, warn = FALSE)

## Close the file
close(fcon)

## Close the virtual device
close(my_device)
```

---

adf_file_exists   *Test if an entry exists on a virtual device*

Description

Test if an entry (file or directory) exists on a virtual ADF device. `adf_file_exists()` is the equivalent of [file.exists()](#) on a virtual ADF device. `adf_dir_exists()` is the equivalent of [dir.exists()](#) on a virtual ADF device.

Usage

```
adf_file_exists(x, virtual_path, ...)

## S3 method for class 'adf_device'
```

```
adf_file_exists(x, virtual_path, ...)

## S3 method for class 'virtual_path'
adf_file_exists(x, virtual_path, ...)

adf_dir_exists(x, virtual_path, ...)

## S3 method for class 'adf_device'
adf_dir_exists(x, virtual_path, ...)

## S3 method for class 'virtual_path'
adf_dir_exists(x, virtual_path, ...)
```

Arguments

x	Either a virtual device or virtual path.
virtual_path	A [virtual_path()](#) to be checked for existence. Should be omitted when x is already a virtual path.
...	Ignored

Value

adf_file_exists() returns TRUE if the path exists on the virtual device, FALSE otherwise. adf_dir_exists() returns TRUE when the path exists and is a directory, FALSE otherwise.

Author(s)

Pepijn de Vries

Examples

```
## First setup a connection to a virtual device
adz_file <- system.file("example.adz", package = "adfExplorer2")
my_device <- connect_adf(adz_file)

adf_file_exists(my_device, "s/startup-sequence")
adf_dir_exists(my_device, "s/startup-sequence")

close(my_device)
```

---

compress_adf   *Compress ADF to ADZ files and vice versa*

Description

The ADZ format is essentially a compressed (gzip) version of the Amiga Disk File (ADF) format. The adfExplorer2 allows you to connect to both formats. However, you can only open a 'read-only' connection to ADZ files. Use the compression and decompression functions documented here to move back and forth from and to ADF and ADZ formats.

Usage

```
compress_adf(source, destination)

decompress_adz(source, destination)
```

Arguments

source	Path to the source file to read.
destination	Path to the destination file to write.

Value

Returns NULL invisibly.

Author(s)

Pepijn de Vries

Examples

```
adz_file  <- system.file("example.adz", package = "adfExplorer2")
adf_file  <- tempfile(fileext = ".adf")
adz_file2 <- tempfile(fileext = ".adz")

decompress_adz(adz_file, adf_file)
compress_adf(adf_file, adz_file2)
```

---

connect_adf   *Create a connection to a virtual disk*

Description

Establish a connection to a virtual disk stored as Amiga Disk Files (ADF). You cannot write or read directly from this connection. Instead, use the methods provided in this package to retrieve information about the virtual disk or create connections to the files on the disk, to which you *can* write and read from (see [adf_file_con()](#)). Like any other connection, please use [close()](#) to close the connection after use.

Usage

```
connect_adf(filename, write_protected = TRUE)
```

Arguments

filename	Filename of the ADF or ADZ file containing the virtual disk
write_protected	
	A logical value indicating whether the virtual disk needs to be write protected. If TRUE, you can only open 'read only' connections and cannot write to the disk.

Value

Returns an R connection of class `adf_device`.

Author(s)

Pepijn de Vries

Examples

```
adz_file <- system.file("example.adz", package = "adfExplorer2")
my_device <- connect_adf(adz_file)

device_capacity(my_device)
close(my_device)
```

| copy_adf_entry | *Copy or move files between physical and virtual devices* |

Description

With these functions you can copy or move entries (files and directories) between a physical and virtual ADF device. With `copy_adf_entry()` the files are duplicated, with `move_adf_entry()` the files are moved (and deleted from its source).

Usage

```
copy_adf_entry(source, destination, ...)

## S3 method for class 'character'
copy_adf_entry(source, destination, ...)

## S3 method for class 'virtual_path'
copy_adf_entry(source, destination, ...)

## S3 method for class 'virtual_path'
copy_adf_entry.character(source, destination, ...)

## S3 method for class 'virtual_path'
copy_adf_entry.virtual_path(source, destination, ...)

## S3 method for class 'character'
copy_adf_entry.virtual_path(source, destination, ...)

move_adf_entry(source, destination, ...)

## S3 method for class 'character'
move_adf_entry(source, destination, ...)
```

```
## S3 method for class 'virtual_path'
move_adf_entry(source, destination, ...)

## S3 method for class 'virtual_path'
move_adf_entry.character(source, destination, ...)

## S3 method for class 'virtual_path'
move_adf_entry.virtual_path(source, destination, ...)

## S3 method for class 'character'
move_adf_entry.virtual_path(source, destination, ...)
```

Arguments

`source, destination`

> The `source` is a path to a file or directory that needs to be moved or copied.
> `destination` is a path to a directory to which the `source` needs to be copied or
> moved. When `source` or `destination` is a `character` string, it is assumed to be
> a path to a file or directory on a physical device. You can use a [virtual_path()](#)
> for either the `source` or `destination` or both. `source` and `destination` cannot
> both be a `character` string. For copying and moving files on a physical device
> you should you base function [file.copy()](#).

`...`   Ignored

Author(s)

Pepijn de Vries

Examples

```
## Create an Amiga Disk File
## and prepare a file system on the virtual device
my_device <-
  create_adf_device(
    tempfile(fileext = ".adf"),
    write_protected = FALSE) |>
  prepare_adf_device()

## Copy the packaged R scripts of this package to the virtual device
copy_adf_entry(
  system.file("R", package = "adfExplorer2"),
  virtual_path(my_device, "DF0:")
)

## List all entries on the virtual device
list_adf_entries(my_device, recursive = TRUE)

## Move the entire virtual device content to
## the tempdir on your physical device
move_adf_entry(
```

```
  virtual_path(my_device, "DF0:"),
  tempdir()
)

close(my_device)
```

---

create_adf_device   *Create and format a virtual ADF device*

Description

These functions help you to create an empty virtual device that can be used in Commodore Amiga
emulation. create_adf_device() simply creates a file of the proper size (the file size repre-
sents the device capacity) and fills it with raw zeros. In order to use the device in the Amiga
operating system, a file system needs to be installed on the device. This can be achieved with
prepare_adf_device(). Note that the file system itself will also consume disk space on the vir-
tual device.

Usage

```
create_adf_device(destination, type = "DD", ..., connect = TRUE)

prepare_adf_device(
  dev,
  name = "EMPTY",
  ffs = TRUE,
  international = TRUE,
  dircache = FALSE,
  bootable = TRUE,
  ...
)

## S3 method for class 'adf_device'
prepare_adf_device(
  dev,
  name = "EMPTY",
  ffs = TRUE,
  international = TRUE,
  dircache = FALSE,
  bootable = TRUE,
  ...
)
```

Arguments

destination   File path where the virtual device needs to be stored.

type   Specify the type of virtual device you wish to create. Should be one of "DD"
                 (double density floppy disk) or "HD" (high density floppy disk).

...	Ignored for `prepare_adf_device()`.
connect	A `logical` value. If set to `TRUE` a connection is opened to the newly created virtual device and is returned as a `adf_device` class object. If it is set to `FALSE`, the file is just created and no connection is opened. In the latter case `NULL` is returned invisibly.
dev	The virtual adf device for which information needs to be obtained. It should be of class `adf_device` which can be created with create_adf_device() or connect_adf().
name	A `character` string specifying the disk name for the volume on the virtual device. It will be truncated automatically when too long.
ffs	A `logical` value indicating which file system to be used. If `TRUE` the 'Fast File System' (FFS) is used, when `FALSE`, the 'Old File System' is used. See also `vignette("file_system_modes")`.
international	A `logical` value indicating whether the international mode should be used for file naming. See also `vignette("file_system_modes")`.
dircache	A `logical` value indicating whether directory caching should be used. See also `vignette("file_system_modes")`.
bootable	A `logical` value indicating whether you want to include executable code on the boot block. If set to `TRUE` minimal code will be added to the boot block. In an Amiga emulator, this code will load the Amiga Disk Operating System library and start the Amiga Command line interface (CLI). It will then run the startup sequence file from the disk (if available).
	If set to `FALSE` no such code is added. In that case the file system will still be accessible by the Amiga operating system (if the file system mode is compatible). You just can't use the disk to start up a (virtual) Amiga machine.

Value

Either an `adf_device` connection or `NULL` depending on the value of `connect`.

Author(s)

Pepijn de Vries

Examples

```
## Filepath to store the virtual device:
dest <- tempfile(fileext = ".adf")

## Create a blank unformated virtual device (a double density floppy disk):
my_device <- create_adf_device(dest, "DD", connect = TRUE, write_protected = FALSE)

print(my_device)

## Format the floppy and create a file system on the device:
prepare_adf_device(my_device, name = "foobar")

print(my_device)
```

```
## don't forget to close the device connection after use:
close(my_device)
```

---

device_type   *Obtain information about an* adf_device *connection*

Description

A collection of functions to retrieve information about the virtual device, or any volume (file system) available on the device. See examples for usage and results.

Usage

```
device_type(dev, ...)

## S3 method for class 'adf_device'
device_type(dev, ...)

device_capacity(dev, ...)

## S3 method for class 'adf_device'
device_capacity(dev, ...)

volume_capacity(dev, ...)

## S3 method for class 'adf_device'
volume_capacity(dev, vol = 0L, ...)

volume_name(dev, ...)

## S3 method for class 'adf_device'
volume_name(dev, vol = 0L, ...)

n_volumes(dev, ...)

## S3 method for class 'adf_device'
n_volumes(dev, ...)

bytes_free(dev, ...)

## S3 method for class 'adf_device'
bytes_free(dev, vol = 0L, ...)

is_bootable(dev, ...)

## S3 method for class 'adf_device'
```

```
is_bootable(dev, vol = 0L, ...)

is_fast_file_system(dev, ...)

## S3 method for class 'adf_device'
is_fast_file_system(dev, vol = 0L, ...)

is_international(dev, ...)

## S3 method for class 'adf_device'
is_international(dev, vol = 0L, ...)

is_dircache(dev, ...)

## S3 method for class 'adf_device'
is_dircache(dev, vol = 0L, ...)

is_write_protected(dev, ...)

## S3 method for class 'adf_device'
is_write_protected(dev, ...)
```

Arguments

dev	The virtual adf device for which information needs to be obtained. It should be of class adf_device which can be created with [create_adf_device()](#) or [connect_adf()](#).
...	Ignored
vol	Volume index number on the device starting at 0. Default is 0. Note that floppy disks can only have 1 volume installed.

Author(s)

Pepijn de Vries

Examples

```
adz_file <- system.file("example.adz", package = "adfExplorer2")
my_device <- connect_adf(adz_file)

device_type(my_device)

device_capacity(my_device) # in bytes

volume_capacity(my_device) # in bytes

n_volumes(my_device) # number of volumes available on device

volume_name(my_device) # name of the volume
```

```
bytes_free(my_device) # bytes available for writing

is_bootable(my_device) # tests if device is potentially bootable

is_fast_file_system(my_device) # tests if volume uses FFS

is_international(my_device) # tests if file system uses intl mode

is_dircache(my_device) # tests if file system uses dir caching

is_write_protected(my_device) # tests if device is protected against writing

close(my_device)
```

---

format.adf_device   *Basic methods for S3 class objects*

Description

Format and print methods for all S3 class objects created with adfExplorer2

Usage

```
## S3 method for class 'adf_device'
format(x, ...)

## S3 method for class 'adf_file_con'
format(x, ...)

## S3 method for class 'adf_block'
format(x, ...)

## S3 method for class 'virtual_path'
format(x, width = 20L, ...)

## S3 method for class 'adf_device'
print(x, ...)

## S3 method for class 'adf_file_con'
print(x, ...)

## S3 method for class 'adf_block'
print(x, ...)

## S3 method for class 'virtual_path'
print(x, ...)

## S3 method for class 'virtual_path'
as.character(x, ...)
```

Arguments

x	Object to be formatted or printed
...	Ignored or passed on to next methods
width	Set the text width for formatting virtual paths

---

list_adf_entries   *List entries in a directory of a virtual ADF device*

Description

Get an overview of all entries (files and directories) in a specific directory.

Usage

```
list_adf_entries(x, virtual_path, recursive = FALSE, ...)

## S3 method for class 'adf_device'
list_adf_entries(x, virtual_path, recursive = FALSE, ...)

## S3 method for class 'virtual_path'
list_adf_entries(x, virtual_path, recursive = FALSE, ...)

## S3 method for class 'character'
list_adf_entries.adf_device(x, virtual_path, recursive = FALSE, ...)

## S3 method for class 'virtual_path'
list_adf_entries.adf_device(x, virtual_path, recursive = FALSE, ...)
```

Arguments

x	Either an adf_device class object, in which case the virtual_path argument needs to be specified; or, a virtual_path class object.
virtual_path	The virtual path for which you wish to obtain a list of entries (see also vignette("virtual_paths")). When missing, entries for the current directory (adf_directory()) are returned, wen x is an adf_device class object. If x is a virtual_path class object, content of the path defined in that object is listed
recursive	A logical value. When set to TRUE, the function is called recursively for all subdirectories in virtual_path.
...	Ignored

Value

A vector of virtual_path class objects

Author(s)

Pepijn de Vries

Examples

```
## First setup a connection to a virtual device
adz_file <- system.file("example.adz", package = "adfExplorer2")
my_device <- connect_adf(adz_file)

## List all entries in the disk's root:
list_adf_entries(my_device)
## List all entries on the disk:
list_adf_entries(my_device, recursive = TRUE)

close(my_device)
```

---

read_adf_block *Read or write raw data blocks to a virtual device*

Description

The Amiga file system is structured around 512 byte blocks. A double density floppy disk consists of 1760 blocks of 512 bytes. read_adf_block and write_adf_block can be used to transform raw data from and to virtual devices (created with create_adf_device() or connect_adf()). Note that writing raw data to a disk could corrupt the file system on the device. So it is generally not advised unless you know what you are doing.

Usage

```
read_adf_block(dev, sector, ...)

## S3 method for class 'adf_device'
read_adf_block(dev, sector, ...)

write_adf_block(dev, sector, data, ...)

## S3 method for class 'adf_device'
write_adf_block(dev, sector, data, ...)

## S3 method for class 'raw'
write_adf_block.adf_device(dev, sector, data, ...)

## S3 method for class 'adf_block'
write_adf_block.adf_device(dev, sector, data, ...)

## Default S3 method:
write_adf_block.adf_device(dev, sector, data, ...)
```

```
as_adf_block(data, ...)

new_adf_block()
```

Arguments

dev	The virtual adf device for which information needs to be obtained. It should be of class adf_device which can be created with create_adf_device() or connect_adf().
sector	Sector ID of the block you wish to read/write. It is an integer value. For double density disks, the ID ranges from 0 to 1759.
...	Ignored
data	Block data (raw vector of length 512) you wish to write to a virtual device

Value

In case of write_adf_block NULL is returned invisibly. In case of read_adf_block the raw data is returned as a adf_block class object.

Author(s)

Pepijn de Vries

---

remove_adf_entry   *Remove entry (file / directory) from a virtual ADF device*

Description

This function removes an entry (file or directory) from a virtual ADF device. At the moment this function only removes a single entry per call, and in case the entry is a directory, the directory needs to be empty before it can be removed.

Usage

```
remove_adf_entry(x, virtual_path, flush = FALSE, ...)

## S3 method for class 'adf_device'
remove_adf_entry(x, virtual_path, flush = FALSE, ...)

## S3 method for class 'virtual_path'
remove_adf_entry(x, virtual_path, flush = FALSE, ...)

## S3 method for class 'character'
remove_adf_entry.adf_device(x, virtual_path, flush = FALSE, ...)

## S3 method for class 'virtual_path'
remove_adf_entry.adf_device(x, virtual_path, flush = FALSE, ...)
```

Arguments

x
: The virtual ADF device from which an entry needs to be deleted or a virtual path pointing at the entry to be deleted. In case of a virtual device, it should be of class adf_device which can be created with create_adf_device() or connect_adf(). In case of a virtual path use virtual_path().

virtual_path
: A character string or a virtual_path (see virtual_path()) representing a file or directory you wish to delete. Should be omitted when x is alreade a virtual path.

flush
: A logical value. When set to FALSE (default), only the entry's registry in its parent directory is removed and its flags in the bitmap block are set to 'available'. The entry's header data and if the entry is a file, the file data will still linger on the virtual disk. If you don't want that, set this argument to TRUE, in that case all file or directory data will be purged. Note that in the latter case, it won't be possible to recover your deleted file or directory.

...
: Ignored

Value

Returns the device connection

Author(s)

Pepijn de Vries

Examples

```
## We first need a writable connection to an ADF device.
## For this purpose we decompress the ADZ file that comes
## with this package and open a connection

adz_file <- system.file("example.adz", package = "adfExplorer2")
adf_file <- tempfile(fileext = ".adf")
decompress_adz(adz_file, adf_file)
my_device <- connect_adf(adf_file, write_protected = FALSE)

## List files in directory 'Devs':
list_adf_entries(my_device, "Devs")

## remove the file 'system-configuration' from the virtual device
remove_adf_entry(my_device, "devs/system-configuration")

## List files in directory 'Devs' again:
list_adf_entries(my_device, "Devs")

## close the connection to the virtual device
close(my_device)
```

---

virtual_path   *A path pointing to a file or directory on a virtual ADF device*

Description

This function creates a path pointing to a file or directory on a virtual ADF device (created with [connect_adf()](#) or [create_adf_device()](#)). The virtual path created with this function can be used to establish a readable or writable connection to a file, or obtain information about a file or directory. See also vignette("virtual_paths")

Usage

```
virtual_path(dev, virtual_path)
```

Arguments

dev   A virtual ADF device (created with [connect_adf()](#) or [create_adf_device()](#)).
       Make sure a file system is present on the virtual device or install first when miss-
       ing using [prepare_adf_device()](#).

virtual_path A character string representing the path to a file or directory on the virtual
       device.

Value

Returns a virtual_path class object.

Author(s)

Pepijn de Vries

Examples

```
adz_file <- system.file("example.adz", package = "adfExplorer2")

# Open a connection to a virtual device:
my_device <- connect_adf(adz_file)

# specify a virtual path:
my_path <- virtual_path(my_device, "DF0:s/startup-sequence")

# close the virtual device:
close(my_device)
```

Index