

Package: olereader (via r-universe)

October 28, 2026

Title Read OLE Compound Files

Version 0.1.0

Description Reads Microsoft compound files, also known as OLE2 or structured storage files, in pure R. Lists the streams in a file, returns their contents as raw vectors and extracts them to disk. Written from the MS-CFB specification
<https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-cfb/>.

License MIT + file LICENSE

URL <https://github.com/ethanbass/olereader>,
<https://ethanbass.github.io/olereader/>

BugReports <https://github.com/ethanbass/olereader/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Suggests testthat (>= 3.0.0), withr

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

Repository <https://ethanbass.r-universe.dev>

Date/Publication 2026-10-28 21:34:55 UTC

RemoteUrl <https://github.com/ethanbass/olereader>

RemoteRef HEAD

RemoteSha 7f355dc8893bc5a1d79d1d2bbbe2be891eda1c13

Contents

ole_extract	2
ole_open	3
ole_read	4
ole_size	5
ole_streams	5

Index	7
--------------	----------

ole_extract	<i>Extract OLE streams to files</i>
-------------	-------------------------------------

Description

Writes each stream to a file under `dir`, with one folder for each storage that contains it, so `c("storage", "stream")` becomes `dir/storage/stream`.

Usage

```
ole_extract(ole, dir, streams = NULL, overwrite = FALSE)
```

Arguments

<code>ole</code>	An <code>ole_file</code> object from <code>ole_open()</code> , or the path to a compound file, which is opened and parsed again on every call.
<code>dir</code>	Folder to write to. It is created if it does not exist.
<code>streams</code>	The streams to extract, or <code>NULL</code> (the default) for all of them. Either a character vector with one <code>/</code> -separated path per stream, such as <code>c("data", "notes/readme")</code> , or a list of paths like the one returned by <code>ole_streams()</code> . In a list, each path is a vector of names, such as <code>list("data", c("notes", "readme"))</code> , and strings are not split at <code>/</code> , so <code>list("notes/readme")</code> looks for a stream named <code>notes/readme</code> outside any storage.
<code>overwrite</code>	Whether to replace files that already exist. If <code>FALSE</code> (the default), extraction stops at the first file that already exists, including one written by an earlier stream in the same call.

Details

Names are changed where they would not work as file names on every platform. Control characters and `<>:"/\|?*` are replaced with `_`, trailing dots and spaces are removed, names left empty (including `.` and `..`) become `_`, and 'Windows' device names such as `CON` or `NUL.txt` get a `_` prefix. The names in a file cannot place a file outside `dir`.

`ole_extract()` stops before writing anything if a requested stream does not exist, if two streams would be written to the same file (ignoring case), or if the streams add up to more bytes than the file holds, which means it is truncated or has streams sharing sectors. Any other error stops extraction partway, and the files already written are left in place.

Value

The paths of the files written, invisibly, named by stream.

Examples

```
ole <- ole_open(system.file("extdata", "example.cfb", package = "olereader"))
dir <- file.path(tempdir(), "example")
ole_extract(ole, dir)
list.files(dir, recursive = TRUE)

ole_extract(ole, dir, streams = "notes/readme", overwrite = TRUE)
unlink(dir, recursive = TRUE)
```

ole_open

*Open an OLE compound file***Description**

Reads the header, the sector allocation tables and the directory of a Microsoft compound file (MS-CFB), also called OLE2 or structured storage. Stream contents are not read until you ask for them with [ole_read\(\)](#). Only reading is supported.

Usage

```
ole_open(path)
```

Arguments

path Path to the file.

Details

The file is not kept open: each call to [ole_read\(\)](#) reopens it. There is nothing to close.

The other functions also take a path in place of an `ole_file` object and call `ole_open()` on it themselves. A path is simpler for a single read, but the file is parsed again on every call, so open it once to read several streams.

Damaged files give an error rather than wrong data: a header or allocation table that does not fit the size of the file, a chain of sectors that loops, a directory entry that points past the end of the directory, or storages nested more than 64 levels deep.

Value

An `ole_file` object: a list holding

- `path`, version (3 or 4) and `file_size`, in bytes;
- `sector_size` and `mini_sector_size`, in bytes, and `n_sectors`, the number of sectors after the header;
- `fat` and `minifat`, which give for each sector or mini sector the 0-based number of the next sector in its chain, or a negative number marking the end of a chain or a free or reserved sector;
- `mini_chain`, the 0-based sectors holding the mini stream, or the error met while following them;

- dir, a data frame with one row per directory entry. Its columns are name; type (1 for a storage, 2 for a stream, 5 for the root); left, right and child, the 0-based IDs of the entry's siblings and first child, or -1 for none; start, its first sector; size, in bytes; parent, the row of the storage holding it, or 0 for the root and for entries not reached from it; and key, its name with the letters A to Z in lower case.

See Also

```
ole_streams(), ole_read(), ole_size(), ole_extract()
```

Examples

```
ole <- ole_open(system.file("extdata", "example.cfb", package = "olereader"))
ole
```

ole_read	<i>Read an OLE stream</i>
----------	---------------------------

Description

Read an OLE stream

Usage

```
ole_read(ole, stream)
```

Arguments

ole	An ole_file object from <code>ole_open()</code> , or the path to a compound file, which is opened and parsed again on every call.
stream	Path to the stream: a character vector of names, such as <code>c("storage", "stream")</code> , a single string separating them with <code>/</code> , or a list holding one such path, such as <code>ole_streams(ole)[1]</code> . A string in a list is not split at <code>/</code> , so a list also reaches streams whose names contain <code>/</code> . Names are matched ignoring the case of the letters A to Z. <code>ole_extract()</code> reads a character vector differently: each element is a separate stream.

Value

A raw vector, or NULL if the stream does not exist.

Examples

```
ole <- ole_open(system.file("extdata", "example.cfb", package = "olereader"))
rawToChar(ole_read(ole, c("notes", "readme")))
```

ole_size	<i>Size of an OLE stream</i>
----------	------------------------------

Description

Size of an OLE stream

Usage

```
ole_size(ole, stream)
```

Arguments

ole	An <code>ole_file</code> object from <code>ole_open()</code> , or the path to a compound file, which is opened and parsed again on every call.
stream	Path to the stream: a character vector of names, such as <code>c("storage", "stream")</code> , a single string separating them with <code>/</code> , or a list holding one such path, such as <code>ole_streams(ole)[1]</code> . A string in a list is not split at <code>/</code> , so a list also reaches streams whose names contain <code>/</code> . Names are matched ignoring the case of the letters A to Z. <code>ole_extract()</code> reads a character vector differently: each element is a separate stream.

Value

The size in bytes, or 0 if the stream does not exist. Storages also have size 0.

Examples

```
ole <- ole_open(system.file("extdata", "example.cfb", package = "olereader"))
ole_size(ole, "data")
```

ole_streams	<i>List the streams in an OLE file</i>
-------------	--

Description

Streams are listed depth-first, with the entries of each storage sorted by name, the same order as `listdir()` in the Python package 'olefile'. Storages themselves are not listed.

Usage

```
ole_streams(ole)
```

Arguments

ole	An <code>ole_file</code> object from <code>ole_open()</code> , or the path to a compound file, which is opened and parsed again on every call.
-----	--

Value

A named list of character vectors, each the path to a stream: the names of the storages that contain it, then its own name. The names of the list are the same paths joined with /.

Examples

```
ole <- ole_open(system.file("extdata", "example.cfb", package = "olereader"))
ole_streams(ole)
names(ole_streams(ole))
```

Index

`ole_extract`, [2](#)
`ole_extract()`, [4](#), [5](#)
`ole_open`, [3](#)
`ole_open()`, [2](#), [4](#), [5](#)
`ole_read`, [4](#)
`ole_read()`, [3](#), [4](#)
`ole_size`, [5](#)
`ole_size()`, [4](#)
`ole_streams`, [5](#)
`ole_streams()`, [2](#), [4](#)