

# Package: alfr (via r-universe)

September 5, 2024

**Type** Package

**Title** Connectivity to 'Alfresco' Content Management Repositories

**Version** 1.3.0.9000

**Author** Roy Wetherall <rwetherall@gmail.com>

**Maintainer** Roy Wetherall <rwetherall@gmail.com>

**Description** Allows you to connect to an 'Alfresco' content management repository and interact with its contents using simple and intuitive functions. You will be able to establish a connection session to the 'Alfresco' repository, read and upload content and manage folder hierarchies. For more details on the 'Alfresco' content management repository see <<https://www.alfresco.com/ecm-software/document-management>>.

**Depends** R (>= 3.5.0)

**License** GPL-3 | file LICENSE

**URL** <https://github.com/rwetherall/alfr>,  
<https://rwetherall.github.io/alfr/>

**BugReports** <https://github.com/rwetherall/alfr/issues>

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**Imports** httr, jsonlite, magrittr, stringr

**Suggests** devtools, httptest, roxygen2, testthat, knitr, rmarkdown, covr, remotes, spelling, fs

**VignetteBuilder** knitr

**SystemRequirements** Alfresco Content Repository (Community or Enterprise)

**Language** en-US

**Repository** <https://rwetherall.r-universe.dev>

**RemoteUrl** <https://github.com/rwetherall/alfr>

**RemoteRef** HEAD

**RemoteSha** b13aee707e9c160c1bf6297279a83ccea917441

## Contents

alfr . . . . .	2
alf_node . . . . .	3
alf_node.delete . . . . .	4
alf_node.new . . . . .	5
alf_session . . . . .	7
alf_session.invalidate . . . . .	8
alf_session.is_valid . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

alfr	<i>alfr: A package for connecting with Alfresco</i>
------	---

---

## Description

The alfr package provides a way to connect to Alfresco and interact with the contents of the repository.

### Session

- [alf\\_session](#) - connection session to an Alfresco repository
- [alf\\_session.is\\_valid](#) - determine whether the session connection to an Alfresco repository is still valid
- [alf\\_session.invalidate](#) - invalidates a session so it can no longer use used to connect to an Alfresco repository

### Nodes

- [alf\\_node](#) - get the details of a folder or content node
- [alf\\_node.new](#) - creates a new folder or content node
- [alf\\_node.delete](#) - deletes a folder or content node

## Author(s)

Roy Wetherall <[rwetherall@gmail.com](mailto:rwetherall@gmail.com)>

---

alf_node	<i>Get Alfresco node</i>
----------	--------------------------

---

**Description**

Gets the details of an Alfresco repository node matching `node_id` or, if provided, the node at `relative_path` relative to `node_id`.

**Usage**

```
alf_node(session, node_id = "-root-", relative_path = NULL)
```

**Arguments**

<code>session</code>	valid Alfresco repository session
<code>node_id</code>	node id, defaults to <code>-root-</code>
<code>relative_path</code>	relative path from <code>node_id</code> to required node, defaults to <code>NULL</code>

**Value**

Node details

**Examples**

```
# try to establish a connection to the alfresco content repository
my_session <- tryCatch(
  alf_session("http://localhost:8080", "admin", "admin"),
  error = function(e) NULL)

if (!is.null(my_session)) {

  # create document
  my_new_document <- alf_node.new(my_session, node_id="-root-",
    list(
      name = "example.txt",
      nodeType = "cm:content",
      relativePath = "example"
    ))

  # upload content
  my_new_document$content$update(
    system.file("extdata", "sample.txt", package="alfr"))

  # get details of document node
  my_document <- alf_node(my_session, relative_path = "example/example.txt")

  # output the name of the document
  print(my_document$name)
```

```

# output the details of documents content
print(my_document$content$mime_type)
print(my_document$content$mime_type_name)
print(my_document$content$size)
print(my_document$content$encoding)

# read document content
my_content_file <- file(my_document$content$as.file(), "r")
my_content <- readLines(my_content_file)
close(my_content_file)
print(my_content)

# upload new content
my_updated_document <- my_document$content$update(
  system.file("extdata", "modified_sample.txt", package="alfr"))

# print updated content size
print(my_updated_document$content$size)

# delete document
alf_node.delete(my_session, my_document$id)
}

```

---

alf_node.delete	<i>Deletes an Alfresco node</i>
-----------------	---------------------------------

---

## Description

Deletes an Alfresco node identified by node\_id. If the node is a folder then all the delete recurses through the primary children.

## Usage

```
alf_node.delete(session, node_id, permanent = FALSE)
```

## Arguments

session	valid Alfresco repository session
node_id	node id to delete
permanent	indicates whether the node is permanently deleted or places in the trashcan where where it can be recovered from. FALSE by default.

## Examples

```

# try to establish a connection to the alfresco content repository
my_session <- tryCatch(
  alf_session("http://localhost:8080", "admin", "admin"),
  error = function(e) NULL)

```

```
if (!is.null(my_session)) {  
  
  # create document  
  my_new_document <- alf_node.new(my_session, node_id="-root-",  
    list(  
      name = "example.txt",  
      nodeType = "cm:content",  
      relativePath = "example"  
    ))  
  
  # upload content  
  my_new_document$content$update(  
    system.file("extdata", "sample.txt", package="alfr"))  
  
  # get details of document node  
  my_document <- alf_node(my_session, relative_path = "example/example.txt")  
  
  # output the name of the document  
  print(my_document$name)  
  
  # output the details of documents content  
  print(my_document$content$mime_type)  
  print(my_document$content$mime_type_name)  
  print(my_document$content$size)  
  print(my_document$content$encoding)  
  
  # read document content  
  my_content_file <- file(my_document$content$as.file(), "r")  
  my_content <- readLines(my_content_file)  
  close(my_content_file)  
  print(my_content)  
  
  # upload new content  
  my_updated_document <- my_document$content$update(  
    system.file("extdata", "modified_sample.txt", package="alfr"))  
  
  # print updated content size  
  print(my_updated_document$content$size)  
  
  # delete document  
  alf_node.delete(my_session, my_document$id)  
}
```

---

alf\_node.new

*Create a new Alfresco node*

---

### **Description**

Creates a new Alfresco repository node as a child of node\_id.

**Usage**

```
alf_node.new(session, node_id, node_details)
```

**Arguments**

session	valid Alfresco repository session
node_id	node id
node_details	details of new node

**Value**

node details

**Examples**

```
# try to establish a connection to the alfresco content repository
my_session <- tryCatch(
  alf_session("http://localhost:8080", "admin", "admin"),
  error = function(e) NULL)

if (!is.null(my_session)) {

  # create document
  my_new_document <- alf_node.new(my_session, node_id="-root-",
    list(
      name = "example.txt",
      nodeType = "cm:content",
      relativePath = "example"
    ))

  # upload content
  my_new_document$content$update(
    system.file("extdata", "sample.txt", package="alfr"))

  # get details of document node
  my_document <- alf_node(my_session, relative_path = "example/example.txt")

  # output the name of the document
  print(my_document$name)

  # output the details of documents content
  print(my_document$content$mime_type)
  print(my_document$content$mime_type_name)
  print(my_document$content$size)
  print(my_document$content$encoding)

  # read document content
  my_content_file <- file(my_document$content$as.file(), "r")
  my_content <- readLines(my_content_file)
  close(my_content_file)
  print(my_content)
```

```

# upload new content
my_updated_document <- my_document$content$update(
  system.file("extdata", "modified_sample.txt", package="alfr"))

# print updated content size
print(my_updated_document$content$size)

# delete document
alf_node.delete(my_session, my_document$id)
}

```

---

alf\_session

*Get connection session to Alfresco content repository*


---

### Description

Validates authentication details with Alfresco content repository, returning ticket, server details and endpoints if successful.

### Usage

```
alf_session(server, username, password)
```

### Arguments

server	Alfresco server URL
username	user name
password	password

### Value

Connection session to Alfresco repository

### Examples

```

# try to establish a connection to the alfresco content repository
my_session <-
  tryCatch(
    alf_session("http://localhost:8080", "admin", "admin"),
    error = function(e) NULL)

if (!is.null(my_session)) {

  # output session information
  print(paste("Session: [ticket = ", my_session$ticket,
             ", server = ", my_session$server, "]", sep=""))

  # verify that the session is valid

```

```
if (alf_session.is_valid(my_session)) print("Session verified as valid.")

# invalidate the session so that it can no longer be used
alf_session.invalidate(my_session)
}
```

---

alf\_session.invalidate

*Invalidates a session.*

---

### Description

Invalidates a valid session so it can no longer be used to connect to an Alfresco repository.

### Usage

```
alf_session.invalidate(session)
```

### Arguments

session            session

### Value

TRUE if session has been successfully invalidated, FALSE if session was already invalid.

### Examples

```
# try to establish a connection to the alfresco content repository
my_session <-
  tryCatch(
    alf_session("http://localhost:8080", "admin", "admin"),
    error = function(e) NULL)

if (!is.null(my_session)) {

  # output session information
  print(paste("Session: [ticket = ", my_session$ticket,
             ", server = ", my_session$server, "]" , sep=""))

  # verify that the session is valid
  if (alf_session.is_valid(my_session)) print("Session verified as valid.")

  # invalidate the session so that it can no longer be used
  alf_session.invalidate(my_session)
}
```

---

alf\_session.is\_valid *Determine whether a session is valid.*

---

**Description**

Determines whether a given session is still valid or not.

**Usage**

```
alf_session.is_valid(session)
```

**Arguments**

session            session

**Value**

TRUE if the session is valid, FALSE otherwise

**Examples**

```
# try to establish a connection to the alfresco content repository
my_session <-
  tryCatch(
    alf_session("http://localhost:8080", "admin", "admin"),
    error = function(e) NULL)

if (!is.null(my_session)) {

  # output session information
  print(paste("Session: [ticket = ", my_session$ticket,
             ", server = ", my_session$server, "]", sep=""))

  # verify that the session is valid
  if (alf_session.is_valid(my_session)) print("Session verified as valid.")

  # invalidate the session so that it can no longer be used
  alf_session.invalidate(my_session)
}
```

# Index

alf\_node, [2, 3](#)  
alf\_node.delete, [2, 4](#)  
alf\_node.new, [2, 5](#)  
alf\_session, [2, 7](#)  
alf\_session.invalidate, [2, 8](#)  
alf\_session.is\_valid, [2, 9](#)  
alfr, [2](#)  
alfr-package (alfr), [2](#)