

Package ‘rcom’

June 17, 2009

Version 2.2-1

Date 2009-04-07

Title R COM Client Interface and internal COM Server

Author Thomas Baier

Maintainer Thomas Baier <thomas.baier@univie.ac.at>

Description R functions to interface with COM objects, R exposed to COM Clients

License GPL-2

Depends R (>= 2.8.0), rscproxy, utils

LinkingTo rscproxy

SystemRequirements Windows, statconnDCOM (>= 3.1-0B0)

OS_type windows

Repository CRAN

Date/Publication 2009-06-17 08:01:13

R topics documented:

comCreateObject	2
comGetObject	3
comGetObjectInfo	4
comGetPicture	4
comGetProperty	5
comGetVersion	6
comInvoke	7
comIsValidHandle	7
comRegisterRegistry	8
comRegisterServer	9
comSetProperty	9

comThis	10
comTrace	11
comTraceObject	11
comUnregisterRegistry	12
comUnregisterServer	12
installstatconnDCOM	13
[[.COMObject	13
[[<-.COMObject	14
\$.COMObject	15

Index	16
--------------	-----------

comCreateObject	<i>Create COM Objects</i>
-----------------	---------------------------

Description

Creates a new COM object specified by its program identifier.

The object will be released automatically when the last reference to the object is removed. To speed up this process, assign some NULL value as soon as you don't need the object any more.

Usage

```
comCreateObject (progid)
```

Arguments

progid	ProgId of COM object to create, e.g. "Excel.Application"
--------	----------------------------------------------------------

Value

comCreateObject returns an object of class "COMObject".

Author(s)

Thomas Baier

Examples

```
# start up excel
## Not run: x<-comCreateObject("Excel.Application")

# and make it visible
## Not run: comSetProperty(x, "Visible", TRUE)

# do something now...
```

comGetObject	<i>Get Access to Existing COM Objects</i>
--------------	-------------------------------------------

Description

Tries to find a running instance of the specified `progid` and returns this object.

Use this function to use services provided by an already running application, e.g. the currently visible instance of Microsoft Excel.

The object will be released automatically when the last reference to the object is removed. To speed up this process, assign some NULL value as soon as you don't need the object any more.

Usage

```
comGetObject (progid)
```

Arguments

<code>progid</code>	ProgId of COM object to get access to, e.g. "Excel.Application"
---------------------	-----------------------------------------------------------------

Value

`comGetObject` returns an object of class "COMObject".

Author(s)

Thomas Baier

Examples

```
# get access to excel
## Not run: x<-comGetObject("Excel.Application")

# and make it visible
## Not run: comSetProperty(x, "Visible", TRUE);

# do something now...
```

comGetObjectInfo *Get Information about a COM Objects*

Description

This function will return information about the COM object passed as parameter.

Use this function to find out what `rCOM` thinks about the object.

Use this function for debugging purposes to find out why `rCOM` fails.

Not yet implemented!

Usage

```
comGetObjectInfo(handle)
```

Arguments

`handle` COM object (class "COMObject") as returned by e.g. `comCreateObject`

Value

`comGetObject` will return a list of all functions and their descriptions

Author(s)

Thomas Baier

comGetPicture *Get the clipboard contents as IPictureDisp Object*

Description

This function checks the contents of the (local) clipboard for an enhanced metafile object. If found, a COM object implementing the standard COM interface `IPictureDisp` is created from the clipboard contents.

Enhanced metafiles in the clipboard are created using R's graphics functions if the device `win.metafile` with an empty filename is used.

You have to close the device before retrieving the picture object.

Does not work at the moment due to problems in Microsoft's `IPictureDisp`.

Usage

```
comGetPicture()
```

Value

comGetObject returns an object of class "COMObject".

Author(s)

Thomas Baier

Examples

```
## Not run: win.metafile()  
## Not run: plot(sin(1:100))  
## Not run: dev.off()  
## Not run: pic<-comGetPicture()
```

comGetProperty *Read One of a COM Object's Properties*

Description

Reads one of the COM object's properties.

Usage

```
comGetProperty(handle, property, ...)
```

Arguments

- handle COM object (class "COMObject") as returned by e.g. comCreateObject
- property name of property to get as a character string
- ... optional additional arguments for property specification (e.g. index for an array)

Value

The return value depends on the (data) type of the property to read.

Author(s)

Thomas Baier

See Also

[comInvoke](#), [comSetProperty](#)

Examples

```
# start up excel
## Not run: x<-comCreateObject("Excel.Application")

# retrieve the "Visible" property
## Not run: v <- comGetProperty(x,"Visible")

# add a new workbook to Excel and gain access to the first worksheet
## Not run: newwb <- comInvoke(comGetProperty(x,"Workbooks"),"Add")
## Not run: ws <- comGetProperty(newwb,"Worksheets",1)

# get a specific range
## Not run: r <- comGetProperty(ws,"Range","A1","B4")

# do something now...
```

comGetVersion

Get rcom version information

Description

This function returns the version of the currently installed `rcom` package as a floating point number.

Use this function to find out which version you are using. The version must be specified when reporting problems.

Usage

```
comGetVersion()
```

Value

`comGetVersion` returns the version number in format `major.minor`.

Author(s)

Thomas Baier

comInvoke	<i>Invoke a function (method) on a COM Object</i>
-----------	---------------------------------------------------

Description

Invokes a function (method) on a COM object.

Usage

```
comInvoke(handle, method, ...)
```

Arguments

handle	COM object (class "COMObject") as returned by e.g. <code>comCreateObject</code>
method	name of method to call as a character string
...	parameters passed when calling the method

Value

The return value of `comInvoke` depends on the function (method) called.

Author(s)

Thomas Baier

See Also

[comSetProperty](#), [comGetProperty](#)

comIsValidHandle	<i>Check if the COM object is valid</i>
------------------	-----------------------------------------

Description

Check if the passed object is a valid COM object

Usage

```
comIsValidHandle(handle)
```

Arguments

handle	COM object (class "COMObject") as returned by e.g. <code>comCreateObject</code>
--------	---------------------------------------------------------------------------------

Value

`com.isValid` returns TRUE if the object is valid, FALSE otherwise.

Author(s)

Thomas Baier

See Also

[comCreateObject](#), [comGetObject](#)

`comRegisterRegistry`

Register the COM server in the registry

Description

This function will register the COM server and its associated type library with the system (registry).

Technically, this will register the type library and add local-server entries for the prog and class ids.

This function has to be called when installing the package. Otherwise, neither the type library will be available (automation clients cannot use the server) nor a mapping from version independent and version-dependant prog-id will be available.

Usage

```
comRegisterRegistry()
```

Author(s)

Thomas Baier

See Also

[comUnregisterRegistry](#)

comRegisterServer *Initialize the COM server*

Description

This function registers the COM server at runtime. It is called automatically when the package `rcom` is loaded.

Technically, this will register the class factory in the system, so calls to `CoCreateInstance()` from client applications will succeed.

Remark: The type library will be loaded on demand later on

`comUnregisterServer()` is used to unregister the class object (class factory) again.

Usage

```
comRegisterServer()
```

Author(s)

Thomas Baier

See Also

[comUnregisterServer](#), [comRegisterRegistry](#)

comSetProperty *Set One of the COM Object's Properties*

Description

Set one of the COM object's properties.

In case the property is an indexed property, pass the index value before the property value.

Usage

```
comSetProperty(handle, property, ...)
```

Arguments

<code>handle</code>	COM object (class "COMObject") as returned by e.g. <code>comCreateObject</code>
<code>property</code>	name of property to get as a character string
<code>...</code>	the value the property should be set to

Author(s)

Thomas Baier

See Also

[comInvoke](#), [comGetProperty](#)

Examples

```
# start up excel
## Not run: x<-comCreateObject("Excel.Application")

# and make it visible
## Not run: comSetProperty(x, "Visible", TRUE);

# set the value of index 3 of the indexed property ArrayProperty
## Not run: comSetProperty(x, "ArrayProperty", 3, TRUE);

# do something now...
```

comThis

Return the COM Object for this R Instance

Description

Returns the COM object for this R Instance.

Use this function to e.g. pass a handle to the running R COM server to another COM object.

The object will be released automatically when the last reference to the object is removed. To speed up this process, assign some NULL value as soon as you don't need the object any more.

Usage

```
comThis()
```

Value

comThis returns an object of class "COMObject" representing the IStatConnector/IDispatch of the current R.

Author(s)

Thomas Baier

Examples

```
# get access to R
## Not run: x<-comThis();

# do something with R
## Not run: comInvoke(x, "SetSymbol", "abc", 1);
```

`comTrace`*Trace information about SEXP*

Description

This function will trace all passed parameters (variables, constants etc) to the R console.

Use this function to find out what `rcom` thinks about the parameters (e.g. which data types are recognized).

Use this function for debugging purposes to find out why `rcom` fails.

Usage

```
comTrace(x)
```

Arguments

`x` expression (variable, constant) to trace

Author(s)

Thomas Baier

`comTraceObject`*Trace information about the passed object*

Description

This function will trace information about the COM object passed as parameter.

Use this function to find out what `rcom` thinks about the object.

Use this function for debugging purposes to find out why `rcom` fails.

Usage

```
comTraceObject(handle)
```

Arguments

`handle` COM object (class "COMObject") as returned by e.g. `comCreateObject`

Author(s)

Thomas Baier

comUnregisterRegistry

Deregister the COM server from the registry

Description

This function will deregister the COM server and its associated type library from the registry/system. Call this function before uninstalling the package to clean-up the registry.

Usage

```
comUnregisterRegistry()
```

Author(s)

Thomas Baier

See Also

[comRegisterRegistry](#)

comUnregisterServer

Terminate the COM server

Description

This function will deregister the COM server at runtime.

Technically, this will unregister the class factory in the system, so future calls to `CoCreateInstance()` from client applications will fail.

Usage

```
comUnregisterServer()
```

Author(s)

Thomas Baier

See Also

[comRegisterServer](#), [comUnregisterRegistry](#)

```
installstatconnDCOM
      install statconnDCOM
```

Description

Installs RExcel, an Excel add-in connecting R and Excel on Windows, by running a Windows installer program creating its own uninstaller.

Usage

```
installstatconnDCOM()
```

Details

rcom requires an installation of statconnDCOM to work. `installstatconnDCOM()` downloads a binary installer `statconnDCOM.latest.exe` from its web site, <http://rcom.univie.ac.at/>.

It is possible to install (or update) statconnDCOM on a machine without Internet access. To accomplish this, download the current version from <http://rcom.univie.ac.at/>, transfer it to the machine without Internet access and run in on this machine. Administrator privileges are needed to run this installer.

```
[ [.COMObject          Read One of a COM Object's Properties
```

Description

Reads one of the COM object's properties.

Usage

```
## S3 method for class 'COMObject':
handle[[property,...]]
```

Arguments

<code>handle</code>	COM object (class "COMObject") as returned by e.g. <code>comCreateObject</code>
<code>property</code>	name of property to get as a character string
<code>...</code>	optional additional arguments for property specification (e.g. index for an array)

Value

When reading a property, the return value depends on the (data) type of the property to read.

Author(s)

Gabor Grothendieck

See Also[comGetProperty](#), [\\$.COMObject](#)**Examples**

```
# start up excel
## Not run: x<-comCreateObject("Excel.Application")

# retrieve the "Visible" property
## Not run: v <- x[["Visible"]]
```

[[<-COMObject *Write One of a COM Object's Properties*

Description

Writes one of the COM object's properties.

Usage

```
## S3 replacement method for class 'COMObject':
handle[[property, ...]]<-value
```

Arguments

handle	COM object (class "COMObject") as returned by e.g. <code>comCreateObject</code>
property	name of property to get as a character string
...	optional additional arguments for property specification (e.g. index for an array)
value	the value the property should be set to

Value

The return value is the object handle.

Author(s)

Gabor Grothendieck

See Also[comSetProperty](#), [\\$.COMObject](#)

Examples

```
# start up excel
## Not run: x<-comCreateObject("Excel.Application")

# and make it visible
## Not run: x[["Visible"]] <- TRUE;
```

\$.COMObject	<i>Invoke a function (method) on a COM Object</i>
--------------	---------------------------------------------------

Description

Invokes a function (method) on a COM object.

Usage

```
## S3 method for class 'COMObject':
handle$method, ...
```

Arguments

handle	COM object (class "COMObject") as returned by e.g. <code>comCreateObject</code>
method	name of method to call as a character string
...	parameters passed when calling the method

Value

The return value of `$.COMObject` depends on the function (method) called.

Author(s)

Gabor Grothendieck

See Also

[comInvoke](#), [\[\\$.COMObject](#), [\[\[<- \\$.COMObject](#)

Index

*Topic **interface**

- [[|.COMObject](#), [12](#)
- [[\[<-|.COMObject](#), [13](#)
- [\\$.COMObject](#), [14](#)
- [comCreateObject](#), [2](#)
- [comGetObject](#), [2](#)
- [comGetObjectInfo](#), [3](#)
- [comGetPicture](#), [4](#)
- [comGetProperty](#), [5](#)
- [comGetVersion](#), [6](#)
- [comInvoke](#), [6](#)
- [comIsValidHandle](#), [7](#)
- [comRegisterRegistry](#), [7](#)
- [comRegisterServer](#), [8](#)
- [comSetProperty](#), [8](#)
- [comThis](#), [9](#)
- [comTrace](#), [10](#)
- [comTraceObject](#), [10](#)
- [comUnregisterRegistry](#), [11](#)
- [comUnregisterServer](#), [11](#)

*Topic **programming**

- [[|.COMObject](#), [12](#)
- [[\[<-|.COMObject](#), [13](#)
- [\\$.COMObject](#), [14](#)
- [comCreateObject](#), [2](#)
- [comGetObject](#), [2](#)
- [comGetObjectInfo](#), [3](#)
- [comGetPicture](#), [4](#)
- [comGetProperty](#), [5](#)
- [comGetVersion](#), [6](#)
- [comInvoke](#), [6](#)
- [comIsValidHandle](#), [7](#)
- [comRegisterRegistry](#), [7](#)
- [comRegisterServer](#), [8](#)
- [comSetProperty](#), [8](#)
- [comThis](#), [9](#)
- [comTrace](#), [10](#)
- [comTraceObject](#), [10](#)
- [comUnregisterRegistry](#), [11](#)

- [comUnregisterServer](#), [11](#)

*Topic **utilities**

- [installstatconnDCOM](#), [12](#)
- [[|.COMObject](#), [12](#), [14](#)
- [[\[<-|.COMObject](#), [13](#), [14](#)
- [\\$.COMObject](#), [13](#), [14](#)

- [comCreateObject](#), [2](#), [7](#)
- [comGetObject](#), [2](#), [7](#)
- [comGetObjectInfo](#), [3](#)
- [comGetPicture](#), [4](#)
- [comGetProperty](#), [5](#), [7](#), [9](#), [13](#)
- [comGetVersion](#), [6](#)
- [comInvoke](#), [5](#), [6](#), [9](#), [14](#)
- [comIsValidHandle](#), [7](#)
- [comRegisterRegistry](#), [7](#), [8](#), [11](#)
- [comRegisterServer](#), [8](#), [11](#)
- [comSetProperty](#), [5](#), [7](#), [8](#), [13](#)
- [comThis](#), [9](#)
- [comTrace](#), [10](#)
- [comTraceObject](#), [10](#)
- [comUnregisterRegistry](#), [8](#), [11](#), [11](#)
- [comUnregisterServer](#), [8](#), [11](#)

- [installstatconnDCOM](#), [12](#)