Package: RDCOMClient (via r-universe)

August 27, 2024

Version 0.95-1

Title R-DCOM client

Author Duncan Temple Lang <duncan@r-project.org>

Maintainer Duncan Temple Lang <duncan@r-project.org>

Description Provides dynamic client-side access to (D)COM applications from within R.

License GPL2

LazyLoad yes

Collate classes.R COMLists.S COMError.R com.R debug.S zzz.R runTime.S

URL http://www.omegahat.net/RDCOMClient, http://www.omegahat.net http://www.omegahat.net/bugs

Depends R (>= 3.2.0)

OS_type windows

Repository https://rickhelmus.r-universe.dev

RemoteUrl https://github.com/rickhelmus/RDCOMClient

RemoteRef HEAD

RemoteSha 708b5089b1cb4357d8d6abf05541de8f5f194e63

Contents

`.COM` *Full access to client COM invocation*

Description

This is the S function that provides full access to the C routines that perform the invocation of methods in COM servers. This allows one to control the specification of the dispatch method, whether the result is returned.

Usage

```
.COM(obj, name, ..., .dispatch = as.integer(3), .return = TRUE,
      .ids = numeric(0), .suppliedArgs)
```

Arguments

`obj`	the COM object reference, usually obtained via `COMCreate`.
`name`	the name of the method or property to be accessed.
`...`	arguments to be passed to the method. If names are provided for these arguments, these are used as the names in the COM call. (Not yet!)
`.dispatch`	one or more of the `DispatchMethods` values indicating the target of the invocation: a method or a property, and whether to get or set the property. In some cases, one wants to specify both a method and a property which is done by OR'ing the values in `DispatchMethods` in the bit-wise sense of OR'ing.
`.return`	a logical value indicating whether to bother returning the result of the call. This can be used to discard the result of the invocation when only the side-effect is of interest.
`.ids`	an optional numeric vector which, if given, provides the `MEMBERID` values which identify the names of the method and parameters used in the call. Supplying these avoids the need for an extra communication step with the COM object to map the names to identifiers. One must compute these values using the type library (see `getNameIDs`) or via other type information gathered from the object or another type library tool, e.g. oleviewer, Visual Basic Editor's type brower.

`.suppliedArgs` a logical vector indicating which of the arguments provided by ... are to be used. In general, this argument is not used in interactive use. However, when the code calling the `.COM` function is generated, this provides a way to specify which are actual arguments and which are default values for arguments.

Value

An arbitrary value obtained from converting the value returned from the COM invocation.

Note

We have madee PROPERTYGET|METHOD the default for method invocation. In this case, this function would become less commonly used.

Also, we will add code to handle the DispatchMethods enumeration symbollically in the same we have for Gtk enumerations in RGtk.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

See Also

COMCreate COMAccessors getNameIDs

Examples

```
e <- COMCreate("Excel.Application")
books <- e[["Workbooks"]]
books$Add()

 # Now for the example!
books$Item(1)

sheets <- e[["Sheets"]]
sheets$Item(1)
## Not run:
  # We can index the list of sheets by sheet name.
  # This is not run here as the name is different for
  # different languages.
sheets$Item("Sheet1")

## End(Not run)

# Now tidy up.
e$Quit()
rm(list = c("books", "e", "sheets"))
```

```
 gc()

## Not run:
o = COMCreate("Excel.Application")

.COM(o, "Count", .dispatch = 2, .ids = id)

## End(Not run)
```

.COMInit *Activate and de-activate COM facilities in R*

Description

This function allows one to turn on and off the COM facilities in an R session. Generally, one calls this at the start of the session and does not turn it off.

Usage

```
.COMInit(status = TRUE)
```

Arguments

status	a logical value with TRUE meaning to activate the COM mechanism and FALSE meaning to turn it off.

Details

This merely calls CoInitialize or CoUninitialize.

Value

NULL.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

See Also

COMCreate .COM

Examples

```
.COMInit()
.COMInit(TRUE)

.COMInit(FALSE)
```

---

asCOMArray *Create COM SAFEARRAY from R matrix*

Description

This creates a COM array from the contents of a two-dimensional R matrix or data frame. This can be used within R to explicitly coerce an R object before it is passed to the COM mechanism as an argument or return value of a function. Otherwise, the automatic converter mechanism creates a dynamic COM object that points to this S object which loses no information (e.g. column or row names, etc.) This currently only handles basic types in S, i.e. integer, numeric, logical and character elements. In the future, we will provide functions for creating an arbitrary SAFEARRAY and populating it in steps directly from R.

Usage

```
asCOMArray(obj)
```

Arguments

obj	an S object that is converted to a matrix and whose resulting contents are copied into the COM array.

Value

An object which is an external pointer to the newly created SAFEARRAY.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs

See Also

.COM createCOMObject

Examples

```
## Not run:
  r = sheet$Range("A1:C10")
  r[["Value"]] <- asCOMArray(matrix(rnorm(30, 10, 3)))

## End(Not run)
```

COMAccessors *COM Object Accesors*

Description

These operators provide a more S-like syntax for accessing methods and properties in a dynamic COM object. One calls a COM object method using the `$` operator. The values of COM object properties are retrieved and set using `[[` and `[[<-`, respectively.

Value

Setting a property returns `NULL`.

Invoking a method and getting a property value returns an S object representing the COM value. Primitive COM values are converted to the corresponding S objects. COM objects are returned as COMIDispatch objects.

Methods

x = "COMIDispatch" the COMIDispatch object whose method or property is to be invoked.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

See Also

.COM COMCreate

COMList-class COMTypedList-class COMTypedNamedList-class

Examples

```
e <- COMCreate("Excel.Application")

 # Boolean/Logical
e[["Visible"]]
 # Setting a value.
e[["Visible"]] <- TRUE

 # String
e[["Path"]]
e[["Version"]]

 # Double
e[["Width"]]

 # Long
e[["SheetsInNewWorkbook"]]

 # Object
books <- e[["Workbooks"]]

books$Add()

 # Use this as a container, so can have integer indices, 1-based.
books[[1]]
e[["Workbooks"]][[1]]

## Not run:
books$Open("C:\mySheet.xls")

## End(Not run)

e$CheckSpelling("This is a spell check") # okay
e$CheckSpelling("This is a spell chck") # error

## Not run:
e$SaveWorkspace()

## End(Not run)
e$Quit()

rm(list= c("e", "books"))
gc()
```

COMCreate *Create a COM object*

Description

These functions are used from R to either create a COM object of a specific type or to obtain a reference to an existing COM application/object.

Usage

```
COMCreate(name, ..., existing = TRUE)
getCOMInstance(guid, force = TRUE, silent = FALSE)
```

Arguments

name	the name identifying the type of COM object. This is usually the name of the class, such as Excel.Application. In the future, this may support finding class IDs separately and passing them as this argument
...	currently ignored. In the future, we will have arguments controlling how the server is created, e.g. the location of the machine, etc.
guid	a string (character vector of length 1) that identifies the GUID - Global Unique Identifier - for the COM interface of interest. This should have the curly braces ('...').
existing	a logical value. If this is TRUE, then first we try to connect to an existing instance of the specified DCOM server, If no such instance exists, then we create a new one. If this is FALSE, then we just create a new instance and don't check whether there is one already running. This guarantees that we get
force	a logical value. If this is TRUE, this will (attempt to) create a new instance of the specified DCOM application if there is no instance already in existence. In other words, getCOMInstance will look for an existing instance and if that fails, it will create a new one. If this is FALSE, getCOMInstance will return after attempting to find an existing instance. As a result, the return value might be NULL.
silent	a logical value that is used when force is TRUE and a new object needs to be created. If this is TRUE, the warning message about creating the new instance is supressed. Otherwise, the warning is issued.

Details

This resolves the class ID given the name and then creates the instance using CoCreateInstance. The class ID is found using CLSIDFromString and if that fails using CLSIDFromProgID.

Value

An object of class COMIDispatch.

Note

getCOMInstance works with Word and Excel but for some reason on at least some machines fails to find an existing Internet Explorer application. As a result, it will create a new instance of the application.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

See Also

The $ and [[operators for the COMIDispatch class - COMAccessors.

Also, see how one can generate "compiled" or processed interfaces to any or all of the types in an application using the SWinTypeLibs.

Examples

```
## Not run:
 COMCreate("Excel.Application")
 getCOMInstance("{000208D5-0000-0000-C000-000000000046}")

## End(Not run)
```

COMDate-class *Classes for representing COM VARIANT values as numbers*

Description

Several types of VARIANTs in COM represent their value as a number which is interpreted appropriately based on the type of the VARIANT. For example, a date is the number of days since January 1, 1900, i.e. Midnight Jan 1, 1900 is 2. Similarly, a date-time value contains the hour, minute and second information in the decimal part of the number. In order to be able to associate the VARIANT type with the value when converting the VARIANT to R, we use these classes which provide the numeric value but also class information corresponding to the VARIANT.

Objects from the Class

Objects can be created by calls of the form new("COMDate", ...), etc. They are currently created implicitly in the C code that performs the conversion from VARIANT objects to R. In the future, we will provide facilities for the reverse direction.

Slots

.Data: Object of class "numeric" this is the numeric vector in which the value is stored.

Extends

Class "numeric", from data part. Class "vector", by class "numeric".

Methods

No methods defined with class "COMDate" in the signature.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

See Also

.COM

COMIDispatch-class *Representation of generic COM object in R*

Description

These classes are used to represent in S an arbitrary COM object. IUnknown is the most basic and provides us with very little information about the underlying COM object. COMIDispatch is the work-horse for this package and provides a mechanism by which one can access the methods and properties of the COM object using the IDispatch interface, i.e. dynamic bindings rather than compiled ones.

COMIDispatch is a trivial extension of IUnknown that provides type information which we use to dispatch methods. The IUnknown class is merely a reference to the C/C++-level COM object.

Reference counting is done automatically in the C code so that the COM object should persist as long as there is an S object that refers to it and will be released when no S value refers to it. Of course, other clients can clobber the COM object and the S references will be meaningless.

Objects from the Class

Objects can be created by calls of the form COMCreate or implicitly when a COM object is returned from a COM method call.

Slots

ref: Object of class "externalptr" this is the C++ value identifying the COM object.

Methods

\$ `signature(x = "COMIDispatch")`: accessor for a method in the COM object. This returns a function that can be used to invoke the named method See `$` in COMAccessors

\$<- `signature(x = "COMIDispatch", "ANY")`: generates an error as one cannot assign to a function/method. This is implemeted this way for symmetry so that assigning to a property (`x[["foo"]] <- 1`) has the same basic syntax as accessing it `x[["foo"]]`. If we made this operator a property assignment, we would have `x$foo <- 1` work but `x$foo` failing. See COMAccessors

[[`signature(x = "COMIDispatch", "numeric")`: access an element of a list via the DCOM object's `Item` method, assuming we it has one. See COMAccessors

[[<- `signature(x = "COMIDispatch", "character")`: set a property value using the specified name. See COMAccessors

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

See Also

COMCreate

COMList *Create an instance of COMList class*

Description

This is a function that creates an instance of a COMList-class class or some derived class of this. The COMList-class class allows certain types of COM objects to be treated as an ordered container with a length, much like the R list concept.

Usage

```
COMList(obj, class = "COMList")
```

Arguments

`obj`	the COMIDispatch-class object which is to be treated as a class.
`class`	the name of the class which is to be instantiated. This argument allows one to use the name of a derived class such as ExcelCOMList or COMTypedList.

Value

An object of the specified class.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

See Also

.COM

COMList-class *COMList collection types*

Description

The COMList type is used in R to represent a COM type which has both a Count() and Item() method. Such a type arises frequentyly in Office applications such as Excel and Word for representing collections or orderd lists of COM objects. For example, Workbooks is a list of Workbook objects, Worksheets is a collection of Worksheet objects, and Addins is a collection of Addin objects. This class provides a way to treat such a container as an R list with methods to compute the length, access elements by index, and iterate over the elements using the s/lapply functions. For each of these, we never convert the list to an R list, but perform the computations via the COM methods.

The COMTypedList-class class is an extension of COMList and should be considered a virtual class. (It is not for implementation reasons only.) This class has the property that when one extracts individual elements from the container in R, the class of that resulting R object is determined from the class of the COMTypedList. This is done in a very simple fashion by translating the name of the COMTypedList to its singular form (in English). So for a COMTypedList of class `Workbooks`, say, the expression `x[[1]]` would result in an object of class `Workbook`. It does this computation dynamically. An extension of the class could compute this value at the time of definition and use that explicitly. This is an example of the utility for class slots.

The COMTypedList should be treated as virtual. One should define an extension of this and the associated class for the elements in tandem, e.g. `setClass("Workbooks", contains = "COMTypedList")` and `setClass("Workbook", contains = "COMIDispatch")`.

Objects from the Class

The constructor function `COMList` should be used to create objects of either of these types. Alternatively, one can use the canonical form `new("COMList", comObject)`.

Slots

`ref`: Object of class `"externalptr"`. This is inherited from the base class `COMIDispatch-class`.

Extends

Class `"COMIDispatch"`, directly. Class `"IUnknown"`, by class `"COMIDispatch"`.

Methods

[[`signature(x = "COMList", i = "numeric")`: this is the method to access an individual element in the COM container. This amounts to a call to the method `Item` in the COM object.

[[<- `signature(x = "COMList", i = "numeric")`: this sets the value of an individual element in the COM container. In general, this does nothing except return the `COMList` object so that we can make inline'd assigments of the form `docs[[1]][["Range"]][["Text"]] = "Some text"` and have the `docs[[1]]` perform correctly.

lapply `signature(X = "COMList")`: this iterates over the elements of the list and invokes the specified function with that element as the first argument. See `lapply`

sapply `signature(X = "COMList")`: a version of `sapply` that we currently need to copy to here.

length `signature(x = "COMList")`: computes the number of elements currently in the container.

Author(s)

Duncan Temple Lang <duncan@r-project.org>

References

Excel types. http://www.omegahat.net/RDCOMClient

See Also

`COMList` `.COM` `COMTypedList-class` `COMTypedNamedList-class`

Examples

```
## Not run:
  e = COMCreate("Excel.Application")
  e$Workbooks()$Add()
  e$Workbooks()$Add()

  l = COMList(e$Workbooks())
  length(l)  # should be 2
  l[[1]]  # First Workbook

  setClass("Workbooks", contains = "COMTypedList")
  setClass("Workbook", contains = "COMIDispatch")

  l = COMList(e$Workbooks, "Workbooks")
  l[[1]]  # class is "Workbook"

## End(Not run)
```

CompiledCOMIDispatch-class *DCOM Interface object with knowledge of its methods and their types.*

Description

This class is used as a simple extension of COMIDispatch-class to illustrate that it has information about its methods and properties and can make use of this information in coercing arguments and return types apporpriately. This class is used by "compiled" or generated S code that is derived from reading type libraries via the SWinTypeLibs package.

The key things we use this class for are to provide special methods for the $ and [[operators which can take advantage of the pre-computed information. The class is "never" used directly but rather is extended to created derived classes for particular DCOM interfaces.

Objects from the Class

Objects of this class are not created directly, but typically are implicitly created by calling methods in the compiled/generated code created from the type library processing.

Slots

ref: Object of class "externalptr", inherited from IUnknown-class

Extends

Class "COMIDispatch", directly. Class "IUnknown", by class "COMIDispatch".

Methods

\$ signature(x = "CompiledCOMIDispatch"): access a property or call a method.

\$<- signature(x = "CompiledCOMIDispatch", name = "character"): ...

[[signature(x = "CompiledCOMIDispatch", i = "character"): ...

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

See Also

COMIDispatch-class generateInterface writeCode

COMStop *General Error function for COM errors*

Description

This is a general function for raising an exception in a DCOM computation from within R. It takes a message and a status code (an integer from the DCOM operation) and raises an error. The class of the error can be specified allowing for different types of errors to be distinguished. It is the ability to specify the class of the error that makes this general.

Usage

```
COMStop(msg, status, class)
```

Arguments

`msg`	a message string to display in the error, intended to be human readable.
`status`	an integer that identifies the type of the DCOM error. These are interpreted relative to a table of errors. We should make this available from the R language by accessing the C structures.
`class`	a character vector giving the name of the class(es) for the error being created. One can specify different class names to allow error handlers to easily differentiate between types of errors using `tryCatch`.

Details

This uses the exception mechanism in R to provide extensible error objects that can be caught in a flexible manner.

Value

An error object of the class specified by `class`.

Author(s)

Duncan Temple Lang <duncan@r-project.org>

References

http://www.omegahat.net/RDCOMServer http://www.omegahat.net/RDCOMClient

See Also

`.COM`

Examples

```
## Not run:
   COMStop("A fake error message", 1, "FakeCOMError")

## End(Not run)
```

COMTypedList-class *Classes for representing DCOM objects that have list-like facilities*

Description

These classes extend the basic COMList. They are used to represent DCOM types that have methods that support a list-like interface. In Office applications, this means that they have methods named Item and Count which correspond to index accessors ([[) and length in R. We can therefore make these objects feel more like R lists using methods defined for these classes.

Like COMList-class, these classes are not intended to be used directly, but are to be used in creating derived or sub-classes.

These particular classes provide additional functionality for the COMList-class by specifying the type of the elements. COMTypedList knows that its elements have a type/class name given by the singular of its own class. For example, if we had a class named Workbooks that extended COMTypedList, then elements returned via its Item method would be coerced to class Workbook the singular of Workbooks.

The class COMTypedNamedList does essentially the same thing, but does not rely on the class name of its elements being the singular form of its own class name. Instead, the class name is stored with the COMTypedNamedList object and is used to coerce an element returned from the Item method to the appropriate R class. (If we had class slots in the S4 system, we would use that rather than putting the name into each instance of a class. However, the overhead is small.) This is used, for example, in the case of our Workbooks illustration above. In Excel, the Workbooks type behaves like a list but returns elements which are of class _Workbook. In this case, the name field in the derived class Workbooks would be _Workbook and elements would be coerced to that R type.

These classes form part of the run-time infrastructure for "compiled" or pre-processed classes to DCOM interfaces that are generated from the type library describing a collection of COM interfaces. See the SWinTypeLibs package and the generateInterface function.

Objects from the Class

Objects can be created from these classes manually using the familiar new function. However, when used in the "compiled" DCOM interfaces, they are automatically created as return values or arguments in methods where appropriate.

Slots

ref: Object of class "externalptr", inherited

name: character string (i.e. vector of length 1). This is the field in the COMTypedNamedList that specifies the name of the class for an element of the list.

Extends

Class "COMList", directly. Class "COMIDispatch", by class "COMList". Class "IUnknown", by class "COMList".

Methods

[[signature(x = "COMTypedList", i = "ANY"): get the i-th element of the DCOM list and coerce the result to the appropriate type specified by the type of elements associated with the class.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

See Also

COMList-class .COM

---

createCOMReference *Creates S object for COM reference*

Description

This is the function that is called anytime a COM object is being created or returned from a C language call to S. This function can examine the object and determine what is the best representation in S for it. It can use the generic IUnknown or COMIDispatch classes to simply represent the pointer value. Alternatively, it might dynamically generate a new S class and accessor methods for accessing properties and functions for that object using the SWinTypeLibs package. Or it might lookup a previously compiled collection of type information and match the GUID of the object's type to find the associated S class name.

This function is not intended to be called from S, but primarily from C code that has access to the COM references.

Usage

```
createCOMReference(ref, className)
```

Arguments

ref	the S object that is an external pointer containing the reference to the COM object.
className	The name of the class that is "suggested" by the caller.

Value

An S object that should almost definitely contain the `ref` value. The class of the object should probably extend `IUnknown`.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs

See Also

COMCreate .COM

`DispatchMethods` *S-level constants*

Description

Constants used to communicate between S and C code to indicate how a COM call should be invoked. This can be used as the value of the `.dispatch` argument in the `.COM` function to control whether to treat the call as a property accessor, method call or both.

Usage

```
data(DispatchMethods)
```

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

EnumValue *Create instance of enumeration value class in R*

Description

This generic function is used to create an instance of a value from an enumeration constant definition type. See EnumValue-class for more information about these named symbolic constants.

Usage

```
EnumValue(id, value, obj = new("EnumValue"))
```

Arguments

id	a string giving the symbolic name of the value.
value	the numeric value for this EnumValue instance.
obj	an instance of the class to be created. This allows the caller to reuse this function with for different derived classes of EnumValue-class. One creates the new object via its default prototype and passes this to EnumValue.

Value

The value of the obj argument with is .Data slot filled in with a named integer vector of length 1.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

See Also

EnumValue-class setAs

EnumValue-class *A class to represent a value of an enumerated constant*

Description

This class is intended to be used to represent a value that originates from an enumerated constant. The idea is that this value carries around the value and the symbolic name. Additionally, if an object of a enumerated type is expected, an integer or character string could be supplied and its value compared to a definition of the enumeration in R.

It is expected that users would create new classes derived from this one for each new enumeration type that is encountered. The definition of the enumerated constant values would be identified from the class name of the new class. See EnumValue.

Objects from the Class

Objects can be created by calls of the form `new("EnumValue", ...)` or via the function `EnumValue` with an instance of the appropriate target class specified.

Slots

`.Data`**:** Object of class `"integer"`, giving the value. This must have the associated name which makes it the symbolic constant.

Extends

Class `"integer"`, from data part. Class `"vector"`, by class `"integer"`. Class `"numeric"`, by class `"integer"`.

Methods

show `signature(object = "EnumValue")`: this takes care of displaying the value as a regular named integer giving the human-readable form and its value.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

See Also

`EnumValue`

Examples

```
  # Define a new Enumeration class.
setClass("Color", contains = "EnumValue")
      # Define the enumeration definition table.
ColorEnum = c(Red = 1, Green = 2, Blue = 3)
storage.mode(ColorEnum) = "integer"

 # Set the coercion methods to take a number or a string
 # to this class type.
setAs("character", "Color",
        function(from)
           EnumValue(from, new("Color")))

setAs("numeric", "Color",
        function(from)
           EnumValue(from, new("Color")))


  # Now we can use this class.
 as(1, "Color")
 as("Red", "Color")

 as("Blue", "Color")
```

```
  # These should give errors, so we enclose them in a call to try.
try(as("Orange", "Color"))
try(as(20, "Color"))
```

---

getCLSID *Get the UUID/GUID from the human-readable name of an application.*

Description

This function provides a way to find the unique identifier for an application or interface given its human-readable form, e.g. "Excel.Application". This is convenient if you want to attempt to load the type library without having an instance of the application or if you want to use this information for looking in the Windows registry (- see the SWinRegistry package).

Usage

```
getCLSID(appName)
```

Arguments

appName	the human-readable string (i.e. character vector of length 1) whose UUID we want to find.

Details

This is a interface to the C routines CLSIDFromString and CLSIDFromProgID

Value

A string which gives the UUID. If the UUID could not be found, this (currently) returns the UUID of all zeros: "{00000000-0000-0000-0000-000000000000}"

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs http://www.omegahat.net/SWinRegistry

Examples

```
getCLSID("Excel.Application")
```

SCOMErrorInfo-class *S version of COM Error structure*

Description

This is a class that we use to represent an error that was generated in a COM call. If an object of this class is returned, then we know that a call to .COM failed. An object of this class contains information about the error status, a description of the error and a string giving the source of the error (e.g. the name of the COM server).

The C++ code to use this is in place but is currently not activated.

Objects from the Class

Created in the C++ code when a COM call fails.

Slots

status: Object of class "numeric" the error status for the error. This can be used with system codes to find the symbolic name. See .COMErrors in the package.

source: Object of class "character" a string giving the source of the error, e.g. the application name.

description: Object of class "character" the human-readable description of the error.

Methods

No methods defined with class "SCOMErrorInfo" in the signature.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs

See Also

.COM

VARIANT-class *Reference to a C-level VARIANT object*

Description

The VARIANT class provides a way to represent a C-level VARIANT object in S and assign it to S variable names. The other classes are trivial extensions that provide type information about a variant so that we can dispatch methods on such an object.

Objects from the Class

Currently created in C code when returning values from COM calls. In the future, one will be able to create variant objects directly from R and use them as arguments in .COM calls. Also we will provide methods for accessing the values of the variants, and converting them to different S values.

Slots

ref: Object of class "externalptr" reference to the C-level address of the variant data structure.

kind: Object of class "integer" the kind of the variant, e.g. I8, I4, R8, BSTR, DISPATCH, etc. represented as a named integer which has the value from the C-level enumeration.

Methods

No methods defined with class "VARIANT" in the signature.

Author(s)

Duncan Temple Lang (duncan@r-project.org)

References

http://www.omegahat.net/RDCOMClient http://www.omegahat.net/RDCOMServer http://www.omegahat.net/SWinTypeLibs

See Also

.COM

`writeErrors` *Query or set whether DCOM errors are written to a file*

Description

This function allows one to query or set the flag that controls whether DCOM errors are appended to a file when they are encountered.

Usage

```
writeErrors(val = logical(), file = character())
```

Arguments

`val`	a logical value. If this is not specified, the function queries the current logical value controlling whether DCOM errors are written to a file. If this is a logical value (not an empty vector), that value of the first element is used to set whether the DCOM errors are written to a file or not.
`file`	a character value. If this is not specified, the function queries the current location of the log file. If this is a character value (not an empty vector) the value of the first element is used as the location of the log file. The file will be created if it does not exist already. If the log file is not specified, but errors are being written, this will create a temporary log file.

Value

A list with an entry val, with the current status of whether errors are appended to the file and an entry file with the location of the log file, if any.

Author(s)

Duncan Temple Lang

Examples

```
writeErrors(TRUE, file = "C:/temp/RDCOM_err.log")
writeErrors(FALSE)
writeErrors()
writeErrors(TRUE)
```

Index