Package: rSharp (via r-universe)

July 11, 2026

Type Package

Title Accessing .NET from R

Version 1.2.2

Description Enable low-level access to .NET runtime from R. Provides a set of functions to create and manipulate .NET objects, call methods, and access properties and fields.

URL https://github.com/Open-Systems-Pharmacology/rsharp/, http://www.open-systems-pharmacology.org/rSharp/

Encoding UTF-8

Depends R (>= 4.1)

Imports R6, methods, cli, lifecycle

Suggests devtools, covr, testthat (>= 3.0.0), knitr, rmarkdown

License file LICENSE

SystemRequirements dotnet8

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://open-systems-pharmacology.r-universe.dev

Date/Publication 2026-07-10 13:12:47 UTC

RemoteUrl https://github.com/Open-Systems-Pharmacology/rSharp

RemoteRef v1.2.2

RemoteSha fdea119f2e44f0394c9689fae3ddfa79fa9993d6

Contents

`.clrTypeNameExtPtr` *Gets the type name of an object*

Description

Gets the type name of an object, given the SEXP external pointer to this .NET object.

Usage

```
.clrTypeNameExtPtr(extPtr)
```

Arguments

`extPtr`	external pointer to a .NET object (not a 'cobjRef' S4 or a 'NetObject' object)

Value

a character string, the type name

Examples

```
## Not run:
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
.clrTypeNameExtPtr(testObj$pointer)

## End(Not run)
```

`.getCurrentConvertedObject` *System function to get a direct access to an object*

Description

This function is highly unlikely to be of any use to an end user, even an advanced one. This is indirectly needed to unlock the benefits of using R.NET convert data structures between R and .NET.

Usage

```
.getCurrentConvertedObject()
```

Value

a ‘cobjRef‘ S4 object

`.mkClrObjRef` *Create if possible an S4 ‘cobjRef‘ object.*

Description

Create if possible and adequate the S4 object that wraps the external pointer to a ‘cobjRef‘ object.

Usage

```
.mkClrObjRef(obj, clrtype = NULL)
```

Arguments

`obj`	the presumed external pointer.
`clrtype`	character; the name of the type for the object. If NULL, rSharp retrieves the type name.

Value

a cobjRef S4 object if the argument is indeed an external pointer, otherwise returned unchanged.

callStatic *Call a static method on a .NET type*

Description

Call a static method on a .NET type

Usage

```
callStatic(typename, methodName, ...)
```

Arguments

typename	type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAsse
methodName	the name of a static method of the type
...	additional method arguments passed to .External (e.g., arguments to the method)

Value

an object resulting from the call. May be a 'NetObject' object, or a native R object for common types. Can be NULL.

Examples

```
cTypename <- getRSharpSetting("testCasesTypeName")
callStatic(cTypename, "IsTrue", TRUE)
```

castToRObject *Create if possible an object of the R6 class 'NetObject'*

Description

Create if possible an object of the R6 class 'NetObject'

Usage

```
castToRObject(obj, recursive = TRUE)
```

Arguments

obj	the presumed external pointer.
recursive	logical; if TRUE, the function is applied recursively to the list elements.

Details

Create if possible and adequate the R6 object of the class ‘NetObject‘ that wraps the external pointer to a .NET object. If ‘obj‘ is not a pointer, returns ‘obj‘ unchanged.

Value

A ‘NetObject‘ R6 object if the argument is indeed an external pointer, otherwise returned unchanged. If ‘recursive‘ is TRUE and ‘obj‘ is a list, the function is applied recursively to the list elements.

Examples

```
castToRObject(1)
castToRObject("a")
castToRObject(TRUE)
castToRObject(FALSE)
castToRObject(1L)
castToRObject(1.1)
castToRObject(1.1 + 1i)
castToRObject(list(1, 2, 3))
castToRObject(data.frame(a = 1:3, b = c("a", "b", "c")))
```

---

getConstructors *List the public constructors of a CLR Type*

Description

List the public constructors of a CLR Type

Usage

```
getConstructors(type)
```

Arguments

type	.NET Type, or a (character) type name that can be successfully parsed

Value

a list of constructor signatures

Examples

```
testClassName <- "ClrFacade.TestObject"
getConstructors(testClassName)
```

getEnumNames *Gets the names of a .NET Enum value type*

Description

Gets the names of a .NET Enum value type

Usage

```
getEnumNames(enumType)
```

Arguments

enumType	a .NET object, System.Type or type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAssemblyName'.

Value

a character vector of the names for the enum

Examples

```
enumName <- "ClrFacade.TestEnum"
getEnumNames(enumName)
# Get enum names from object
enumObj <- newObjectFromName(enumName)
getEnumNames(enumObj)
```

getLoadedAssemblies *List the names of loaded assemblies*

Description

List the names of loaded assemblies

Usage

```
getLoadedAssemblies(fullname = FALSE, filenames = FALSE)
```

Arguments

fullname	should the full name of the assemblies be returned. 'FALSE' by default.
filenames	if TRUE, return a data frame where the second column is the URI (usually file path) of the loaded assembly. 'FALSE' by default.

Value

the names of loaded assemblies

Examples

```
getLoadedAssemblies()
```

---

getRSharpSetting *getRSharpSetting*

Description

Get the value of a global rSharp setting.

Usage

```
getRSharpSetting(settingName)
```

Arguments

settingName	String name of the setting

Value

Value of the setting stored in rSharpEnv. If the setting does not exist, an error is thrown.

Examples

```
getRSharpSetting("nativePkgName")
```

---

getSexpType *Get the type code for a SEXP*

Description

Get the type code for a SEXP, as returned by the TYPEOF macro

Usage

```
getSexpType(sexp)
```

Arguments

sexp	an R object

Value

the type code, an integer, as defined in Rinternals.h

Examples

```
getSexpType(1)
getSexpType("a")
getSexpType(1:10)
```

`getStatic` *Gets the value of a static field or property of a class*

Description

Gets the value of a static field or property of a class

Usage

```
getStatic(type, name)
```

Arguments

`type`	Type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAss
`name`	the name of a field/property of the object

Value

An object resulting from the call. May be a 'NetObject' object, or a native R object for common types. Can be NULL.

Examples

```
testClassName <- getRSharpSetting("testObjectTypeName")
getStatic(testClassName, "StaticPropertyIntegerOne")
```

getStaticFields *Gets the static fields for a type*

Description

Gets the static fields for a type

Usage

```
getStaticFields(objOrType, contains = "")
```

Arguments

objOrType	a 'NetObject' object, or type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAssemblyName'.
contains	a string that the property names returned must contain

getStaticMembers *Gets the static members for a type*

Description

Gets the static members for a type

Usage

```
getStaticMembers(objOrType)
```

Arguments

objOrType	a .NET object, or type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAssemblyName'.

Examples

```
cTypename <- getRSharpSetting("testCasesTypeName")
getStaticMembers(cTypename)
```

getStaticMemberSignature *Gets the signature of a static member of a type*

Description

Gets the signature of a static member of a type

Usage

```
getStaticMemberSignature(typename, memberName)
```

Arguments

typename	type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAsse
memberName	The exact name of the member (i.e. field, property, method) to search for

getStaticMethods *Gets the static methods for a type*

Description

Gets the static methods for a type

Usage

```
getStaticMethods(objOrType, contains = "")
```

Arguments

objOrType	a 'NetObject' object, or type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAssemblyName'.
contains	a string that the property names returned must contain

getStaticProperties *Gets the static properties for a type*

Description

Gets the static properties for a type

Usage

```
getStaticProperties(objOrType, contains = "")
```

Arguments

objOrType	a 'NetObject' object, or type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAssemblyName'.
contains	a string that the property names returned must contain

getType *Gets the pointer to the 'System.RuntimeType' of a 'NetObject' object or a .NET type name.*

Description

Returns a 'NetObject' object with external pointer to the object of type 'System.RuntimeType' that represents the type of the .NET object. To get a string representation of the type, call 'toStringNET' on the returned object.

Usage

```
getType(objOrTypename)
```

Arguments

objOrTypename	An object of class 'NEtObject' or a character vector of length one. It can be the full file name of the assembly to load, or a fully qualified assembly name, or as a last resort a partial name.

Value

A 'NetObject' to the pointer of 'System.RuntimeType' of 'objOrTypename'.

Examples

```
testClassName <- "ClrFacade.TestObject"
type <- getType(testClassName)
toStringNET(type)

testObj <- newObjectFromName(testClassName)
type <- getType(testObj)
toStringNET(type)
```

`getTypesInAssembly` *Get a list of .NET type names exported by an assembly*

Description

Get a list of .NET type names exported by an assembly

Usage

```
getTypesInAssembly(assemblyName)
```

Arguments

`assemblyName`	the name of the assembly

Value

The names of the types exported by the assembly

Examples

```
getTypesInAssembly("ClrFacade")
```

`inspectArgs` *Peek into the structure of R objects 'as seen from C code'*

Description

Inspect one or more R object to get information on its representation in the engine. This function is mostly useful for R/rSharp developers. It is derived from the 'showArgs' example in the R extension manual

Usage

```
inspectArgs(...)
```

Arguments

...	one or more R objects

Value

NULL. Information is printed, not returned.

Examples

```
inspectArgs(1, "a", 1:10)
```

---

isAssemblyLoaded *Is the assembly loaded?*

Description

Is the assembly loaded?

Usage

```
isAssemblyLoaded(assemblyName)
```

Arguments

assemblyName	The name of the assembly, e.g. 'ClrFacade'

Value

TRUE if the assembly is loaded, FALSE otherwise

Examples

```
isAssemblyLoaded("ClrFacade")
```

`loadAssembly` *Loads a .NET assembly.*

Description

Loads a .NET assembly.

Usage

```
loadAssembly(name)
```

Arguments

`name`	a character vector of length one. It can be the full file name of the assembly to load, or a fully qualified assembly name, or as a last resort a partial name.

Details

Note that this is loaded in the single application domain that is created by rSharp, not a separate application domain.

Value

Name of the loaded assembly, if successfull.

See Also

`.C` which this function wraps

Examples

```
## Not run:
f <- file.path("SomeDirectory", "YourDotNetBinaryFile.dll")
f <- path.expand(f)
stopifnot(file.exists(f))
loadAssembly(f)

## End(Not run)
```

NetObject *NetObject*

Description

Base wrapper class for the pointers to .NET objects. Offers basic methods to interact with the .NET objects.

Active bindings

`type` String representation of the type of the .NET object. Read-only

`pointer` The external pointer to the .NET object. Read-only

Methods

Public methods:

- `NetObject$new()`
- `NetObject$getFields()`
- `NetObject$getStaticFields()`
- `NetObject$getProperties()`
- `NetObject$getStaticProperties()`
- `NetObject$getMethods()`
- `NetObject$getStaticMethods()`
- `NetObject$getMemberSignature()`
- `NetObject$call()`
- `NetObject$get()`
- `NetObject$set()`
- `NetObject$.printLine()`
- `NetObject$.printClass()`
- `NetObject$print()`

Method `new()`**:** Initializes the object.

Usage:

```
NetObject$new(pointer)
```

Arguments:

`pointer` The external pointer to the .NET object

Returns: The initialized object

Examples:

```
testClassName <- "ClrFacade.Tests.RefClasses.LevelOneClass"
o <- .External("r_create_clr_object", testClassName, PACKAGE = getRSharpSetting("nativePkgName"))
x <- newObjectFromName(testClassName)
print(x)
```

Method `getFields()`**:** List the fields of the object

Usage:

```
NetObject$getFields(contains = "")
```

Arguments:

`contains` a string that the field names returned must contain

Returns: a list of names of the fields of the object

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getFields()
testObj$getFields("ieldInt")
```

Method `getStaticFields()`**:** List the static fields of the object

Usage:

```
NetObject$getStaticFields(contains = "")
```

Arguments:

`contains` a string that the field names returned must contain

Returns: a list of names of the static fields of the object

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getStaticFields()
testObj$getStaticFields("ieldInt")
```

Method `getProperties()`**:** List the properties of the object

Usage:

```
NetObject$getProperties(contains = "")
```

Arguments:

`contains` a string that the property names returned must contain

Returns: a list of names of the properties of the object

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getProperties()
testObj$getProperties("One")
```

Method `getStaticProperties()`**:** List the static properties of the object

Usage:

```
NetObject$getStaticProperties(contains = "")
```

Arguments:

`contains` a string that the property names returned must contain

Returns: a list of names of the static properties of the object

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getStaticProperties()
testObj$getStaticProperties("One")
```

Method `getMethods()`**:** List the methods the object

Usage:

```
NetObject$getMethods(contains = "")
```

Arguments:

`contains` a string that the methods names returned must contain

Returns: a list of names of the methods of the object

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getMethods()
testObj$getMethods("Get")
```

Method `getStaticMethods()`**:** List the static methods the object

Usage:

```
NetObject$getStaticMethods(contains = "")
```

Arguments:

`contains` a string that the methods names returned must contain

Returns: a list of names of the static methods of the object

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getStaticMethods()
testObj$getStaticMethods("Get")
```

Method `getMemberSignature()`**:** Gets a string representation of the signature of a member (i.e. field, property, method). Mostly used to interactively search for what arguments to pass to a method.

Usage:

```
NetObject$getMemberSignature(memberName)
```

Arguments:

`memberName` The exact name of the member (i.e. field, property, method) to search for

Returns: a character vector with summary information on the method/member signatures

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getMemberSignature("set_PropertyIntegerOne")
testObj$getMemberSignature("FieldIntegerOne")
testObj$getMemberSignature("PropertyIntegerTwo")
```

Method `call()`**:** Call a method of the object

Usage:

```
NetObject$call(methodName, ...)
```

Arguments:

`methodName` the name of a method of the object

`...` additional method arguments

Returns: An object resulting from the call. May be a 'NetObject' object, or a native R object for common types. Can be NULL.

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$call("GetFieldIntegerOne")
```

Method `get()`**:** Gets the value of a field or property of the object

Usage:

```
NetObject$get(name)
```

Arguments:

`name` the name of a field/property of the object

Returns: An object resulting from the call. May be a 'NetObject' object, or a native R object for common types. Can be NULL.

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$get("FieldIntegerOne")
```

Method `set()`**:** Sets the value of a field or property of the object.

Usage:

```
NetObject$set(name, value, asInteger = FALSE)
```

Arguments:

`name` the name of a field/property of the object

`value` the value to set the field with

`asInteger` Boolean whether to convert the value to an integer. Used for cases where .NET signature requires an integer. Ignored if 'value' is not numeric.

Examples:

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$set("FieldIntegerOne", as.integer(42))
```

Method `.printLine()`**:** DEPRECATED: Internal method for printing a line

Usage:

```
NetObject$.printLine(entry, value = NULL, addTab = TRUE)
```

Arguments:

`entry` The entry text

`value` The value to print

`addTab` Whether to add a tab before the entry

Method `.printClass()`**:** DEPRECATED: Internal method for printing class name

Usage:

```
NetObject$.printClass()
```

Method `print()`**:** Prints a summary of the object.

Usage:

```
NetObject$print()
```

Examples

```
## ------------------------------------------------
## Method `NetObject$new`
## ------------------------------------------------

testClassName <- "ClrFacade.Tests.RefClasses.LevelOneClass"
o <- .External("r_create_clr_object", testClassName, PACKAGE = getRSharpSetting("nativePkgName"))
x <- newObjectFromName(testClassName)
print(x)

## ------------------------------------------------
## Method `NetObject$getFields`
## ------------------------------------------------

testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getFields()
testObj$getFields("ieldInt")

## ------------------------------------------------
## Method `NetObject$getStaticFields`
## ------------------------------------------------

testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getStaticFields()
testObj$getStaticFields("ieldInt")

## ------------------------------------------------
## Method `NetObject$getProperties`
## ------------------------------------------------

testClassName <- getRSharpSetting("testObjectTypeName")
```

```
testObj <- newObjectFromName(testClassName)
testObj$getProperties()
testObj$getProperties("One")

## ------------------------------------------------
## Method `NetObject$getStaticProperties`
## ------------------------------------------------

testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getStaticProperties()
testObj$getStaticProperties("One")

## ------------------------------------------------
## Method `NetObject$getMethods`
## ------------------------------------------------

testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getMethods()
testObj$getMethods("Get")

## ------------------------------------------------
## Method `NetObject$getStaticMethods`
## ------------------------------------------------

testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getStaticMethods()
testObj$getStaticMethods("Get")

## ------------------------------------------------
## Method `NetObject$getMemberSignature`
## ------------------------------------------------

testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$getMemberSignature("set_PropertyIntegerOne")
testObj$getMemberSignature("FieldIntegerOne")
testObj$getMemberSignature("PropertyIntegerTwo")

## ------------------------------------------------
## Method `NetObject$call`
## ------------------------------------------------

testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$call("GetFieldIntegerOne")

## ------------------------------------------------
## Method `NetObject$get`
## ------------------------------------------------
```

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$get("FieldIntegerOne")

## ------------------------------------------------
## Method `NetObject$set`
## ------------------------------------------------

testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
testObj$set("FieldIntegerOne", as.integer(42))
```

`newObjectFromName` *Create a new NetObject R6 object given the type name.*

Description

Create a new NetObject R6 object given the type name.

Usage

```
newObjectFromName(typename, ..., R6objectClass = NetObject)
```

Arguments

`typename`	type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAsse
`...`	additional method arguments passed to the object constructor via the call to .External
`R6objectClass`	the R6 class of the object to be created, defaults to 'NetObject'. Can be used to create custom R6 classes that inherit from 'NetObject'.

Value

a 'NetObject' R6 object

Examples

```
testClassName <- getRSharpSetting("testObjectTypeName")
testObj <- newObjectFromName(testClassName)
# object with a constructor that has parameters
testObj <- newObjectFromName(testClassName, as.integer(123))
```

newPointerFromName *Create a new external pointer to a .NET object given the type name.*

Description

Create a new external pointer to a .NET object given the type name.

Usage

```
newPointerFromName(typename, ...)
```

Arguments

typename	type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAsse
...	additional method arguments passed to the object constructor via the call to .External

Value

an external pointer to a .NET object

Examples

```
testClassName <- getRSharpSetting("testObjectTypeName")
testPtr <- newPointerFromName(testClassName)
# Now we can create a NetObject from the pointer
testObj <- NetObject$new(testPtr)
```

printTraceback *Prints the last .NET exception*

Description

This is roughly the equivalent of the traceback function of R.

Usage

```
printTraceback()
```

Examples

```
## Not run:
callStatic(
  getRSharpSetting("testCasesTypeName"), "
           ThrowException",
  10L
) # will be truncated by the Rf_error API
printTraceback() # prints the full stack trace

## End(Not run)
```

`rSharpSettingNames` *Names of the settings stored in rSharpEnv Can be used with 'getRSharpSetting()'*

Description

Names of the settings stored in rSharpEnv Can be used with 'getRSharpSetting()'

Usage

```
rSharpSettingNames
```

Format

An object of class `character` of length 9.

`rToDotNetType` *Gets the type of a .NET object resulting from converting an R object*

Description

Gets the type of a .NET object resulting from converting an R object. This function is mostly for documentation purposes, but may be of use to end users.

Usage

```
rToDotNetType(x)
```

Arguments

`x`	An R object

Value

A list, with columns including mode, type, class, length and the string of the corresponding .NET type.

Examples

```
rToDotNetType(1)
```

---

setConvertAdvancedTypes *Turn on/off the conversion of advanced data types with R.NET*

Description

Turn on/off the conversion of advanced data types with R.NET. This will turn off the conversion of classes such as dictionaries into R lists, as these are not bidirectional and you may want to see and manipulate external pointers to dictionaries in some circumstances.

Usage

```
setConvertAdvancedTypes(enable = TRUE)
```

Arguments

enable	if true enable, otherwise disable

Examples

```
library(rSharp)
cTypename <- getRSharpSetting("testCasesTypeName")
callStatic(cTypename, "CreateStringDictionary")
setConvertAdvancedTypes(FALSE)
callStatic(cTypename, "CreateStringDictionary")
```

---

setStatic *Sets the value of a field or property of an object or class*

Description

Sets the value of a field or property of an object or class

Usage

```
setStatic(type, name, value, asInteger = FALSE)
```

Arguments

type	Type name, possibly namespace and assembly qualified type name, e.g. 'My.Namespace.MyClass,MyAss
name	the name of a field/property of the object
value	The value to set the field with
asInteger	Boolean whether to convert the value to an integer. Used for cases where .NET signature requires an integer. Ignored if ‘value‘ is not numeric.

Examples

```
testClassName <- getRSharpSetting("testObjectTypeName")
setStatic(testClassName, "StaticPropertyIntegerOne", as.integer(42))
```

`toStringNET` *Calls the ToString method of an object*

Description

Calls the ToString method of an object as represented in the .NET. This function is here to help quickly test object equivalence from the R interpreter, for instance on the tricky topic of date-time conversions

Usage

```
toStringNET(x)
```

Arguments

`x`	any R object, which is converted to a .NET object on which to call ToString

Value

The string representation of the object in .NET

Examples

```
library(rSharp)
dt <- as.POSIXct("2001-01-01 02:03:04", tz = "UTC")
toStringNET(dt)
```

Index