

R documentation

of all in 'pkg/CORBA/man'

December 15, 2000

R topics documented:

| | |
|--------------------------------------|----|
| CORBAinit | 1 |
| .Corba | 2 |
| CorbaOpMode | 4 |
| .CorbaServer | 4 |
| addCorbaConverter | 5 |
| baseMatrixHandlerGenerator | 6 |
| checkCorbaObjectExists | 7 |
| corbaCommandLineArgs | 8 |
| corbaServices | 9 |
| corbaTypeCode | 9 |
| corbaTypeCodeNames | 10 |
| createNamingContext | 10 |
| deferredRequestList | 11 |
| getCorbaConverters | 12 |
| getIRContainerContents | 12 |
| getNextCompletedTask | 14 |
| getRequestServer | 15 |
| getRequestStatus | 16 |
| getRequestValue | 18 |
| idleElementType | 19 |
| idlInterface | 20 |
| idlType | 21 |
| interfaceRepository | 22 |
| ior | 23 |
| iorDump | 24 |
| namedCorbaObject | 25 |
| namingService | 26 |
| namingServiceContents | 27 |
| primitiveIORDump | 28 |
| removeCorbaConverter | 29 |
| removeNamedCorbaObject | 30 |
| removeRequest | 31 |
| sendRequests | 32 |
| super | 34 |

| | |
|-----------|---|
| CORBAinit | <i>Initialize the connection to the CORBA ORB and BOA and internal library variables.</i> |
|-----------|---|

Description

Establishes a connection to the CORBA Object Request Broker (ORB) and optionally the Basic Object Adapter. Connections to the naming service and interface repository are done as needed in other calls. This function is usually invoked by `.First.lib()`. Calling it more than once usually causes the CORBA implementation to display a warning as most implementations do not support re-initializing an ORB or BOA.

Usage

```
CORBAinit(args=corbaCommandLineArgs(), which=c(T, T))
```

Arguments

| | |
|--------------------|--|
| <code>args</code> | character vector passed to the initialization of the ORB and BOA to parameterize their creation allowing run-time configuration to override default settings. The useful values are typically ORB dependent. Most Orbacus (the default CORBA implementation we use) options can be provided in the file identified by the environment variable <code>ORBACUS_CONFIG</code> . |
| <code>which</code> | logical vector controlling whether the ORB and/or BOA should be initialized. This allows one to defer initializing the BOA. Rarely needed. |

Value

logical vector of length 2 (or the same length as the argument `which` indicating whether the instantiation of the ORB and BOA were successful.

Note

Author(s)

Duncan Temple Lang

References

Henning and Vinoski, Advanced CORBA Programming in C++ Alan Pope, The CORBA Reference Guide, Orbacus Reference Manual <http://www.ooc.com/ob/download>

See Also

[interfaceRepository](#), [namingService](#)

Examples

```
CORBAinit()
# Initialize the BOA to listen on the port 2001
# along with the other command line arguments.
CORBAinit(c("-OApport", "2001", commandArgs()))
```

.Corba

Invoke operation on CORBA server from R.

Description

Invokes an operation or field accessor in a CORBA server. If the method identifies a CORBA attribute, and no arguments are supplied, its value is retrieved. Alternatively, if a single value is provided, an attempt to set the attribute of the CORBA server is made. If the `.methodName` identifies The available methods can be dynamically discovered using the function `idlInterface`.

Usage

```
.Corba(.server, .methodName, ..., .wait=T, .deferred=character(1), .send=T)
```

Arguments

| | |
|--------------------------|--|
| <code>.server</code> | CORBA object identifier (IOR or CORBA name) identifying the server whose method will be invoked. |
| <code>.methodName</code> | Name of the server's method to be invoked. |
| <code>...</code> | R arguments to the CORBA call, converted using the default converters. |
| <code>.wait</code> | with <code>.deferred</code> , controls whether the call should be blocking, one-way or deferred. |
| <code>.deferred</code> | If specified, provides the name of a grouping in which one or more background requests will be stored. These can then be queried later. |
| <code>.send</code> | if <code>FALSE</code> , the template request is stored for dispatch in the future. Useful for constructing a collection of calls when arguments are available and deferring their calls until a future time. |

Details

The arguments can be primitive or more complex R objects. An attempt is made to convert these to the IDL types identified by the parameters of the CORBA operation definition. Conversion to CORBA primitives are defined by the C++ code. Conversion to IDL interfaces can be controlled by registering functions or C/C++ routines to convert R objects. If no converters are registered for the target IDL type, the dynamic conversion process is used, which takes the R object and stores it in a proxy CORBA server. All CORBA calls to it are transferred to it as function calls via the function `CorbaMethodEval`. In this way, R arguments can become CORBA servers. These function calls will take place in the calling frame of the `.Corba`.

If any of the arguments are IDL 'out' arguments, they can be omitted provided all of the other arguments are named with the same names as used in the IDL. A simple argument matching is performed.

If the call is performed as a background invocation and not one-way, then the `.deferred` object identifies a table or list in which the request will be stored. The functions `getRequestStatus()` and `getRequestValue` can be used to poll the status of the outstanding calls and obtain the result when it has finished.

Value

In a standard blocking call, the value of the method invocation is returned. If this is a CORBA primitive or sequence of primitives, it is converted to an appropriate R type. Otherwise, we search for a converter from the IDL type of the result to an R object. If one exists it is invoked and the result returned. Otherwise, a dynamic reference to the CORBA object is returned, and this can be used in other `.Corba` calls.

If an exception occurs, this is

Author(s)

Duncan Temple Lang

References

Henning and Vinoski, Allan Pope, OMG CORBA Spec, Papers accompanying this library/
<http://www.omegahat.org/CORBA/>

See Also

`.CorbaServer`, `idlInterface`, `addCorbaConverter`

Examples

```
# Assume there is a CORBA server with the name
# R/Server and that it has a plot method
# that takes a CORBA matrix.
server <- namedCorbaObject("R","Server")
# Assume that the default and registered converters
# will convert an R matrix to a CORBA matrix
.Corba(server,"plot", matrix(rnorm(3*100),100,3))
```

CorbaOpMode

Symbolic constants defining Operation Invocation Modes

Description

Named/symbolic identifiers for flags to be passed to C++ code that control the mode of method invocation for blocking, non-blocking and "oneway" CORBA calls.

`.CorbaServer`

Create a CORBA server with a particular IDL type and local R data.

Description

This creates a CORBA server which is implemented via interpreted R code using the Dynamic CORBA Server facilities (DSI).

Usage

```
.CorbaServer(idlTypes, ..., name="", evalFunc="", block=T, activate=T, checkTypes=T, handlerType)
```

Arguments

| | |
|--------------------------|---|
| <code>idlTypes</code> | a string identifier for the IDL type being implemented by this server. This is usually <code>IDL:module/sub-module/.../Interface:1.0</code> or <code>module::sub-module::...::Interface</code> |
| <code>...</code> | a list of the R objects which are being used to implement the CORBA servers. These are |
| <code>name</code> | if the name(s) of the servers are non-simple names (i.e. nested names within a naming context), the can be specified here. |
| <code>evalFunc</code> | the name of the R function which is used to handle the generic CORBA operation call and map it to the server. This is used if the object is not a closure. In that case, the call is made directly into to the function identified by the operation name and accessible in the closure. |
| <code>block</code> | logical indicating whether we should enter the request event loop immediately after creating this server or not so that requests will not be processed at this point. |
| <code>activate</code> | |
| <code>checkTypes</code> | logical indicating whether we should check that the IDL type is defined in the Interface Repository. |
| <code>handlerType</code> | flag indicating what type of interpreted object is being used to implement the server - a simple object, a closure, a function, etc. |

Details

Value

If the invocation is blocking, the return value is the resulting state of the server when it is deactivated and the request loop exited. Otherwise, it is the IOR of the CORBA server created.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[.Corba](#)

Examples

```
.CorbaServer("IDL:Omegahat/Matrix:1.0", Rmatrix=baseMatrixHandlerGenerator())
```

`addCorbaConverter` *Register an R function or C++ routine to convert between R and CORBA/IDL type.*

Description

This allows a user to register a specialized converter (rather than the standard dynamic CORBA mechanism) when converting between R and a CORBA object. This stores the `name` in an internal C++ table which is referenced when we convert between R and CORBA. If the name of a function is supplied, this should generate a suitable R object that can be used as a CORBA object - namely a closure. If a C/C++ routine is supplied, it must have an appropriate signature. See the more detailed documentation.

Usage

```
addCorbaConverter(name, idlName, userFunction, toCorba=T)
```

Arguments

| | |
|---------------------------|--|
| <code>name</code> | a character vector of length 1 identifying the R function or C/C++ routine that will perform the conversion. |
| <code>idlName</code> | the identifier for the IDL type associated with this converter. |
| <code>userFunction</code> | logical indicating whether the <code>name</code> argument refers to an R function or a C/C++ routine. |
| <code>toCorba</code> | logical indicating whether the direction of the conversion is from R to CORBA (TRUE) or vice-versa (FALSE) |

Details

Value

A list containing a details of the converter previously registered for that IDL type and conversion direction, or the default converter entry.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA/References>

See Also

[removeCorbaConverter](#), [.CorbaServer](#)

Examples

```
addCorbaConverter(baseMatrixHandlerGenerator, "IDL:Omegahat/Matrix:1.0",T)
addCorbaConverter(baseMatrixHandlerGenerator, "Omegahat::Matrix",T)
```

`baseMatrixHandlerGenerator`*Converter for R matrix to CORBA/IDL matrix*

Description

Create a simple closure that contains methods that implement the basic operations of a matrix without requiring the matrix to be explicitly required. This is necessary in object oriented interfaces such as CORBA and Java in which we invoke a method on an object rather than a function with the object as the argument.

Usage

```
baseMatrixHandlerGenerator(this)
```

Arguments

`this` the matrix to be enclosed in the resulting closure which acts as the implicit source and target of the operations. This objects is the equivalent of the 'this' in an object oriented language like C++ or Java.

Details

This is a closure mechanism that is used to implement a simple class mechanism, with simple lookup of "inherited methods".

Value

A closure containing a list of functions defined to implement the basic methods of a matrix.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[.CorbaServer](#)

Examples

```
server <- baseMatrixHandlerGenerator(matrix(1:9,3,3))
server$nrow()
server$setElement(1,1,10.3)
server$getElement(1,1)

.CorbaServer("IDL:Omegahat/Matrix:1.0", Rmatrix=server)
```

checkCorbaObjectExists

Determines whether a give name exists is registered in the CORBA naming service.

Description

Determines whether the specified CORBA object exists in the naming service

Usage

```
checkCorbaObjectExists(..., server="")
```

Arguments

| | |
|--------|--|
| ... | a CORBA object identifier - a collection of character vectors defining a nested name relative to the default naming service. |
| server | the default naming service relative to which a nested name should be resolved. |

Value

logical, with TRUE indicating the object does exist (but is not necessarily active) or FALSE indicating that no object is bound to that name.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[namingServiceContents](#)

Examples

```
checkCorbaObjectExists("a")
checkCorbaObjectExists("a", "b")
```

`corbaCommandLineArgs` *Creates command line arguments suitable for initializing ORB and BOA.*

Description

Merges the regular command line arguments used to start R with any others that should be passed to the initialization of the ORB and BOA for the particular CORBA implementation being used.

Usage

```
corbaCommandLineArgs()
```

Value

character vector of values that can be used as arguments to another application, etc. suitable for use with CORBA.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>, <http://www.ooc.com/ob>

See Also

[CORBAinit](#), [commandArgs](#)

Examples

```
corbaCommandLineArgs()
```

`corbaServices` *Default CORBA services offered by the CORBA implementation.*

Description

List the available default CORBA services such as the interface repository, naming service, event, trading service, etc.

Usage

```
corbaServices()
```

Value

a character vector identifying the different services as returned by the CORBA implementation's `list_initial_services()`.

Author(s)

Duncan Temple Lang

References

Usual CORBA references.

See Also

Events

Examples

```
corbaServices()
```

| | |
|----------------------------|---|
| <code>corbaTypeCode</code> | <i>Named list of CORBA codes identifying object types to translate between C++ and R.</i> |
|----------------------------|---|

Description

An integer vector containing the identifiers for basic CORBA types with corresponding human-interpretable names. These are used to map between R/S specification of a type and internal C++ usage of that type.

| | |
|---------------------------------|--|
| <code>corbaTypeCodeNames</code> | <i>Names of basic CORBA types indexed by 'corbaTypeCode' to facilitate representation in R from C++.</i> |
|---------------------------------|--|

Description

The names of the `corbaTypeCode` vector that provide more interpretable identifiers for symbolic constant integers representing CORBA types.

See Also

[corbaTypeCode](#)

`createNamingContext` *Create a new CORBA naming context within the naming service.*

Description

Creates a new naming context into which CORBA servers can be registered.

Usage

```
createNamingContext(..., recursive=F, server=NULL)
```

Arguments

| | |
|------------------------|---|
| <code>...</code> | one or more character elements which are treated either as the names of individual naming contexts to be created, or which are collected to define a path identifying the new naming context relative to the top-level container of the default CORBA naming service. |
| <code>recursive</code> | controls whether multiple names in <code>...</code> are treated as multiple new naming context names or a single new naming context at depth <code>length(...)-1</code> . |
| <code>server</code> | CORBA identifier (IOR, name relative to the current top-level naming service in which the new naming context will be created. |

Value

A CORBA object (IOR and name) with IDL type `NamingContext` which can be used in future calls.

Author(s)

Duncan Temple Lang

References

Usual CORBA references

See Also

[namedCorbaObject](#), [namingService](#), [namingServiceContents](#), [removeNamedCorbaObject](#)

Examples

```
# Can run if naming server is running.
createNamingContext("R")
namingServiceContents()
# create A within R.
createNamingContext("A", server=namedCorbaObject("R"))
namingServiceContents()
# create B within R.
createNamingContext("R", "B", recursive=TRUE)
namingServiceContents()
createNamingContext("X", "Y", "Z", recursive=FALSE)
namingServiceContents()
```

`deferredRequestList` *Obtain information about each CORBA task group or list.*

Description

This iterates over all the CORBA task lists managed internally and returns the number of elements within each such list.

Usage

```
deferredRequestList()
```

Value

A list containing an element for each named task list of CORBA requests. The names of the returned list are the names of the internal task lists. Each element contains the number of tasks within that task list.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[getRequestStatus](#), [getRequestValue](#), [removeRequest](#) [getNextCompletedTask](#)

Examples

```
for(i in 1:10)
  .Corba("A", "doit", .deferred="x", .send=F)

deferredRequestList()[["x"]]
```

`getCorbaConverters` *Retrieve a list of the functions/routines registered to convert between R and CORBA/IDL types.*

Description

Returns a list of all the currently registered “methods” for converting between R and CORBA in a particular direction.

Usage

```
getCorbaConverters(to=T)
```

Arguments

`to` indicates whether we are interested in the converters from R to CORBA (TRUE) or vice-versa (FALSE)

Value

A list in which each element contains information about the IDL type being converter to or from, the name of the C routine or R function and whether it is a routine or function.

Author(s)

Duncan Temple Lang

References

<http://www.omg.org>, <http://www.omegahat.org>

See Also

[addCorbaConverter](#), [removeCorbaConverter](#)

Examples

```
getIRContainerContents
```

Retrieve the contents of an element of the Interface Repository.

Description

Retrieve a description of the elements or sub-elements of a container in the CORBA Interface Repository (IR). These recursively/hierarchically describe the modules, interfaces, operations, arguments and return types for the different CORBA servers. These define what the dynamic CORBA facilities know about the overall CORBA system. By default, this retrieves the elements from the top-level container. One can specify an identifier to restrict the elements to come from a particular sub-element.

Usage

```
getIRContainerContents(name=NULL, repository=NULL)
```

Arguments

`name` The name of the IDL entry within the repository. If omitted, all top-level elements (modules, interfaces, etc.) are returned. Otherwise, this is usually a string of the form `IDL:module/interface:1.0`

`repository` The identifier for the CORBA object that is the Interface Repository server. If omitted, this defaults to the current value obtained by calling [interfaceRepository](#)

Details

One can use this function along with [idlInterface](#) to get a complete description of a CORBA interface definition.

Value

A list whose elements correspond to the elements in the interface repository's container being queried..

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>, <http://www.omg.org>

See Also

[idlInterface](#), [idlType](#), [namedCorbaObject](#), [ior](#), [.Corba](#)

Examples

```
getIRContainerContents()

# Equivalent calls to get the Omegahat module
# definition as a named list.
getIRContainerContents("Omegahat")
getIRContainerContents("IDL:Omegahat:1.0")

getIRContainerContents("Omegahat::ArgList")

getIRContainerContents("Omegahat::ArgList")

sapply(getIRContainerContents("Omegahat::XYPlot"), )
```

`getNextCompletedTask` *Identifies the next completed CORBA task among a collection of background tasks.*

Description

This iterates over an internal list of CORBA requests waiting for one to complete. It then returns the details of this task.

Usage

```
getNextCompletedTask(listName, which=NULL)
```

Arguments

| | |
|-----------------|--|
| listName | The name of the internal task list created and augmented via the <code>.Corba</code> function and its <code>.deferred</code> argument. |
| which | the indices of the elements within the specified task/request list that are to be examined for completion status. By default, this is all of the values. |

Details

This repeatedly searches over the elements in the specified task list and finds the first task that has finished, either successfully or in error, and returns the information about that task.

Value

A list containing

| | |
|---------------|--|
| value | The object returned by the CORBA server as a result of the operation invocation, converted to an R object using the basic conversion mechanism. |
| index | The index within the task list of the task that was completed. |
| server | The CORBA server associated with the task that was completed. This can be used as the next idle server to which a new task can be sent (assuming that the currently completed task did not end in a fatal server error). |
| status | The status of the task invocation, either ok or error. |

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

`.Corba`, `getRequestValue`, `getRequestServer`, `getRequestStatus`, `deferredRequestList`

Examples

```
numServers <- length(servers)
numDispatched <- 1
numCompleted <- 0
values <- vector("list", 100)
for(i in servers) {
  .Corba(i, "doit", .deferred = "parallel")
  numDispatched <- numDispatched + 1
}
while(numCompleted <= 100) {
  # get the first completed task
  task <- getNextCompletedTask("parallel")
  idleServer <- task$server
  numCompleted <- numCompleted + 1
}
```

```

values[[numCompleted] <- task$value

if(numDispatched <= 100) {
  .Corba(idleServer, "doit", .deferred = "parallel")
  numDispatched <- numDispatched + 1
}
}

```

| | |
|------------------|--|
| getRequestServer | <i>Reference to the CORBA server currently performing the given task</i> |
|------------------|--|

Description

This returns a reference to the CORBA object to which this task was previously dispatched. This is mainly used when iterating over a collection of tasks.

Usage

```
getRequestServer(name, index=1)
```

Arguments

| | |
|--------------------|--|
| <code>name</code> | The name of the group or collection of tasks that is managed in internal (C++) code. |
| <code>index</code> | The index of the task of interest within this task group identified by <code>name</code> |

Details

This assumes that one has previously created a task group via the `.Corba` using the `.deferred` argument. This method is frequently replaced by the complete action `getNextCompletedTask` and these are used typically when iterating over a collection of tasks. In these situations, we need to determine which server was given a particular task so as to be able to a) reuse that server when the task has been processed, or b) re-dispatch the tasks assigned to a server if that fails.

Value

An object of class CORBA object, containing the IOR of the server.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

`getRequestValue`, `getRequestStatus`, `deferredRequestList`, `removeRequest`, `.Corba` `getNextCompletedTask`

Examples

```
.Corba(findServer(),"operation", .deferred="background")
... # do other things.
getRequestServer("background")

numServers <- length(servers)
numDispatched <- 1
numCompleted <- 0
values <- vector("list", 100)
for(i in servers) {
  .Corba(i, "doit", .deferred = "parallel")
  numDispatched <- numDispatched + 1
}
while(numCompleted <= 100) {
  # get the first completed task
  task <- getNextCompletedTask("parallel")
  idleServer <- task$server
  numCompleted <- numCompleted + 1
  values[[numCompleted]] <- task$value

  if(numDispatched <= 100) {
    .Corba(idleServer, "doit", .deferred = "parallel")
    numDispatched <- numDispatched + 1
  }
}
```

| | |
|------------------|---|
| getRequestStatus | <i>Query whether a background CORBA operation has completed or not.</i> |
|------------------|---|

Description

This allows us to determine if a remote background call has terminated. We can optionally wait for it to complete or return immediately and continue on with local computations, periodically returning.

Usage

```
getRequestStatus(requestName, requestNumber=0, block=F)
```

Arguments

| | |
|----------------------|--|
| requestName | the name of the group in which the background/deferred request has been stored. This is the same as the value for the <code>.deferred</code> argument used in the <code>.Corba</code> call generating the request. |
| requestNumber | The index of the request in the list of requests stored under the name <code>requestName</code> . This uses 1-based counting. |
| block | logical indicating whether we should wait until the call has returned (or completed with an error) or just query the current status and immediately return |

Details

The requests are stored in an internal C++ table and will not persist across sessions.

Value

a vector of the same length as the argument `requestNumber` in which each element indicates whether that request has completed (TRUE) or not (FALSE).

Note

In the future, it would be good to add an error status.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[getRequestValue](#), [getRequestServer](#), [deferredRequestList](#), [removeRequest](#), [.Corba](#)

Examples

```
.Corba("server", "test", .deferred="name")
while(getRequestStatus("name",1) == 0)
  cat("Still waiting\n")
```

getRequestValue

Retrieve the value of a background CORBA operation invocation.

Description

This provides access to the return “value” of a background request invoked via the `.Corba` function. The actual return value and the *out* and *inout* arguments are all returned in a list.

Usage

```
getRequestValue(requestName, requestIndex=1)
```

Details

If the identified request has not completed when this function is invoked, the call blocks waiting for it to complete. Use [getRequestStatus](#) to avoid this.

Value

A list containing the returned value from the operation and also each of the *out* and *inout* arguments in the operation definition.

Note

Calling this more than once for the same request causes the operation to block. In the future, we might cache the result to avoid this.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[getRequestStatus](#), [.Corba](#), [deferredRequestList](#), [sendRequests](#), [removeRequest](#)

Examples

```
#
.Corba("sleeper1", "sleep", 20, .deferred="sleeping")
# do something else
# ...

# now get the result of the operation.
getRequestValue("sleeping", 1)
```

| | |
|----------------|---|
| idlElementType | <i>Get the IDL type for a particular element within a CORBA object's IDL interface.</i> |
|----------------|---|

Description

This indicates whether the IDL interface of the CORBA object or identified specifically by name is an operation, an attribute or something else including non-existent. This can be used

Usage

```
idlElementType(server, element)
```

Arguments

| | |
|---------|--|
| server | A CORBA object - either (nested) name or IOR - or the (fully qualified) name of an IDL interface |
| element | the name of the element of interest within the IDL interface - an attribute or operation. |

Details

If a CORBA object is specified, either by name or explicitly by an IOR, the interface it implements is queried. If no such object is found, we assume the identifier is the name of the IDL interface. At this point, we query the Interface Repository for the IDL interface corresponding to this name and lookup the element within it. Its type is available to the C++ code from the object returned to us by the repository.

Value

If the element in the interface is an operation, the value 2 is returned. If it is an attribute, the value 1 is returned. Otherwise, -1 is returned.

Note

This is used in S when we try to handle the syntactically convenient `server$name` and we do not know whether this will be followed by a `()` or not. Hence, if `name` refers to an attribute, we want to evaluate it and retrieve the attribute's value. Otherwise, we create a reference to the function so that when it is invoked, it turns into an appropriate call to `.Corba`.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org>

See Also

`idlInterface`, `idlType`, `.Corba`

Examples

```
# Ask about the element "sleep" in the
# interface directly.
idlElementType("RSExamples::Sleep","sleep")

# Now try it on a server object that implements this.
#
idlElementType("Sleeper","sleep")
```

`idlInterface`

Retrieve a description of an IDL interface contained in the Interface Repository.

Description

This communicates with the Interface Repository (IR) to get a complete listing of the interface defined by the CORBA object specified by the ... arguments.

Usage

```
idlInterface(..., asObject=T)
```

Arguments

... a CORBA object identifier – a sequence of character strings which are interpreted as a path relative to the top-level naming service, or a single string supplying a CORBA IOR.

asObject treats the reference identifier as a CORBA object (e.g. IOR or name) rather than a IDL interface string/identifier (e.g. "IDL:Omegahat/Matrix:1.0")

Details

The appropriate interface repository must be running and be the default interface repository recognized by this session as specifiable via [interfaceRepository](#). There are two types of identifiers - the IDL:module/sub-module/.../interface or module::sub-module::...::interface. The relevant interface must have been "installed" in the repository. Using Orbacus, one can use the `irfeed`, `irdel` applications.

Value

A list containing the attributes, operations, super classes from which this interface is derived, and the IDL identifier for the interface. The attributes and operations are themselves lists containing type information and

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[idlType](#), [namedCorbaObject](#), [getIRContainerContents](#)

Examples

```
# get the interface for the first element
# returned from the Interface Repository.
idlInterface(names(getIRContainerContents())[1])
```

idlType

Get the IDL identifier for a CORBA object.

Description

This is a mechanism to find the IDL interface implemented by a particular server. This is similar to finding its class.

Usage

```
idlType(...)
```

Arguments

... a CORBA object identifier - an IOR or a sequence of character strings which are combined to define a path in the default namign service identifying a CORBA object.

Details

This makes a query

Value

A character vector of length containing the IDL identifier for the interface. This will be of the form `IDL:module/sub-module/.../interface:1.0` where the `1.0` is the version of the interface and may be different.

Note

Exceptions can be thrown if the identifier does not resolve to a CORBA object or if the resulting server is not active.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[idlInterface](#), [interfaceRepository](#), [getIRContainerContents](#)

Examples

```
# Let's create a known object (without needing
# IDL in the repository, etc.)
createNamingContext("R")
# Now find the IDL type for this object
# Should produce
# "IDL:omg.org/CosNaming/NamingContext:1.0"
idlType("R")
idlType(namedCorbaObject("R"))
# similar (but more specific) information
# within the iorDump output.
iorDump(namedCorbaObject("R"))
```

`interfaceRepository` *Retrieve or set the current (default) CORBA Interface Repository.*

Description

Allows one to query or set the identity of the Interface Repository (IR) containing the definitions of the CORBA interfaces that we query during dynamic CORBA calls and server implementations. Many functions consult this internally stored value, and this allows us to set this value dynamically, rather than once at initialization.

Usage

```
interfaceRepository(id=character(0))
```

Arguments

`id` a CORBA identifier - a single character string specifying an IOR, or alternatively one or more character strings that are used as an iterative path identifying a (nested) name in the default CORBA naming service.

Details

Functions that access the interface repository usually consult the default repository stored in a C++ structure. This function allows us to query and set that value.

Value

A CORBA IOR string identifying an Interface Repository. If the function is called with no arguments, this is the IOR for the current default repository. If an argument is provided, this value identifies the previous default that has now been replaced. This allows one to temporarily set the default repository and arrange to have the original value restored at the end of a computation. See the examples below.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[idlInterface](#), [getIRContainerContents](#), [namingService](#)

Examples

```
# get the current repository
interfaceRepository()
iorDump(interfaceRepository())

# Start a new Interface Repository
```

| | |
|---------|--|
| iorDump | <i>Display a more human readable version of a CORBA IOR.</i> |
|---------|--|

Description

Provides information that is encoded in the IOR (stringified reference) of a CORBA object such as the IDL type, the machine and port on which the server is listening the endianness of the machine.

Usage

```
iorDump(...)
```

Arguments

... usual identifier of a single CORBA object either as an IOR or a collection of strings identifying the server in the naming service.

Value

a character vector containing lines from the output of IOR dump. These can be parsed again to produce a list. This is implementation dependent.

Note

This is implementation dependent and may not exist for all ORBs. This is based on the Orbacus <http://www.ooc.com/ob> implementation.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[ior](#), [namedCorbaObject](#), [.CorbaServer](#)

Examples

```
iorDump("IOR:010000002f00000049444c3a6f6f632e636f6d2f436f734e616d696e672f4578744e616d696e67436f6e7465")
```

`namedCorbaObject` *Retrieve a reference to a CORBA object in the naming service.*

Description

Returns a CORBA identifier (an IOR in R and an object of class `CORBAObject` in S) representing the element registered with the specified name in the specified naming context.

Usage

```
namedCorbaObject(..., .server=character(0))
```

Arguments

`...` a sequence of one or more character strings that identify a path in the naming context defined by `.server`. Note that an IOR is not acceptable here, so this differs from the functions that take an arbitrary CORBA identifier.

`.server` the naming context relative to which the names specified by the `...` argument should be resolved. By default, this is the current top-level naming service.

Details

Value

In R, this is currently a simple string that is an IOR - InterOperable Object Reference - created by the CORBA (ORB actually) routine `object_to_string`.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[namingService](#), [namingServiceContents](#), [createNamingContext](#), [removeNamedCorbaObject](#)

Examples

```
# create an object that is a simple
createNamingContext("R")
# Now get its IOR.
namedCorbaObject("R")

# Now use the IOR in another function call
# that expects a CORBA identifier
```

```
idlType(namedCorbaObject("R"))
# Same as above.
idlType("R")
```

| | |
|---------------|--|
| namingService | <i>Retrieve or set the current (default) CORBA naming service.</i> |
|---------------|--|

Description

Allows one to query or set the value of the default naming service that is used in many functions to identify a named CORBA object. This can be used to simply re-root the default context searched for values or specify a totally different naming service process. For example, we may have a nested naming context called "Project1" and several objects within it. Rather than identifying them as "Project1", "A" and "Project1", "B" we can temporarily make "Project1" the default naming service and refer to these objects simply as "A" and "B".

Alternatively, we can switch to another naming service which may be managed by a different group, or contain objects for a different project, etc. This is a managerial/configuration decision.

Usage

```
namingService(id=character(0))
```

Arguments

| | |
|-----------|---|
| id | either no value if one is querying the current value or a CORBA identifier - IOR or character vector identifying a nested path in the naming context - specifying a CORBA Naming Context which is to be used as the new default naming service value. |
|-----------|---|

Value

If no argument is supplied, the return value is the IOR of the current value for the default naming context. Otherwise, the value is the IOR of the previous naming context that has been replaced with the new value identified by the argument. This allows one to temporarily set the default to a value, storing the return value in order to restore it to its original value at the end of a computation. See the examples below.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[createNamingContext](#), [namedCorbaObject](#), [namingServiceContents](#)

Examples

```
namingService()
  # change the default to point to
  # a naming context within the current default on.
namingService("R")

# start a new naming service and trap its IOR
system("nameserv -OApport 5678 --ior > /tmp/Name.ref &")
old <- namingService(ior("/tmp/Name.ref"))
on.exit(namingService(old))
```

namingServiceContents

Retrieve a list of the names contained in a particular CORBA naming context, identifying which are nested naming contexts and regular objects.

Description

This resolves the naming context specified in the ... argument and queries its contents using the CORBA method

Usage

```
namingServiceContents(..., maxNum=0)
```

Arguments

| | |
|--------|--|
| ... | a CORBA identifier - a single string specifying an IOR or a sequence of one or more strings specifying a (nested) path in the naming service relative to the default top-level naming context. |
| maxNum | a very rarely used argument that controls how many elements we want returned from the naming context being queried. This is an option the CORBA specification allows us. |

Details

In theory, it is possible to implement this using the `.Corba` function. This would assume that the interface was registered with the default Interface Repository and there were suitable converters registered for the elements of the return value.

Value

A logical vector with names matching the elements within the naming context. An element whose value is `TRUE` identifies a nested sub-naming context. Elements with value `FALSE` are regular objects

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[namingService](#), [namedCorbaObject](#) [createNamingContext](#)

Examples

```
# list the contents of the top-level naming service
namingServiceContents()
# Add a new sub-naming context.
createNamingContext("R")
# now see the list again, containing the newly added R.
namingServiceContents()

# List the contents of the (empty) R naming context.
# showing how one can specify the naming context of interest.
namingServiceContents("R")
# create a new naming context within R.
createNamingContext("R","S", recursive=T)
# now list contents of R.
namingServiceContents("R")
```

| | |
|------------------|---|
| primitiveIORDump | <i>Internal mechanism to display information about a CORBA IOR.</i> |
|------------------|---|

Description

This is function that performs the low-level work to fetch the contents of the dump of the IOR. It is usually called from `iorDump`.

Usage

```
primitiveIORDump(obj)
```

Arguments

`obj` a CORBA identifier - either an IOR or a character vector with one or more elements identifying an object in the default naming service - which is to be “dumped” into human readable format.

Details

This is an Orbacus-specific implementation and may not even be available in all CORBA implementations.

Value

A single string with elements separated by new-lines identifying different characteristics of the CORBA object identified by this IOR. These characteristics include the IDL type, machine and port, byte ordering convention, etc.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[iorDump](#), [ior](#), [namedCorbaObject](#) .[Corba](#)

Examples

```
primitiveIORDump(namingService())
creatNamingContext("R")
creatNamingContext("R","S", recursive=T)

primitiveIORDump("R")
primitiveIORDump(c("R","S"))
#
iorDump(namingService())
```

removeCorbaConverter *Un-register a converter between R and CORBA/IDL object for a given IDL type.*

Description

This function is the reciprocal of `addCorbaConverter` and arranges to remove the registered routine or function name from the internal list of object converters which are used when converting objects between R and CORBA. This conversion happens when objects are passed as arguments to CORBA calls and return values from CORBA servers implemented in R.

Usage

```
removeCorbaConverter(idlName, toCorba=T)
```

Arguments

| | |
|----------------------|--|
| <code>idlName</code> | string identifying the IDL type with which the converter was previously registered. |
| <code>toCorba</code> | logical indicating whether the converter is from R to CORBA (<code>TRUE</code>) or the opposite direction (<code>FALSE</code>) |

Details

The lists of converters are stored internally, but can be retrieved and restored using the `getCorbaConverters` and `addCorbaConverter`.

Value

Returns the value of the element in the converter list for the specified IDL type. This allows one to restore this at a later stage. The value is a list containing two elements - **symbols** which is a named character vector contain the name of the C/C++ routine or R function, and **types**, which is a named integer vector indicating the type the corresponding **symbols** element: an interpreted function or native routine. The names of each of these vectors are the IDL type.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[addCorbaConverter](#), [getCorbaConverters](#)

Examples

```
removeNamedCorbaObject
```

Unbind name(s) in CORBA naming service

Description

Removes the specified names from the CORBA naming context identified by the arguments. If an object being removed is itself a naming context, all its contents are also removed, at least in Orbacus.

Usage

```
removeNamedCorbaObject(..., recursive=T)
```

Arguments

| | |
|------------------------|--|
| <code>...</code> | one or more character strings identifying either different objects in the top-level naming context, or a path in the naming service identifying a single object. |
| <code>recursive</code> | controls whether multiple names are treated as a path identifying a single nested name (TRUE) or multiple names at the top-level |

Value

logical indicating whether the removal was successful.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

[createNamingContext](#), [.CorbaServer](#)

Examples

```
createNamingContext("A")
createNamingContext("B")
createNamingContext("C")

# now remove one and then two at a time
removeNamedCorbaObject("A")
removeNamedCorbaObject("C", "B", recursive=FALSE)

createNamingContext("A")
createNamingContext("A", "B")
removeNamedCorbaObject("A", "B")
```

| | |
|----------------------------|--|
| <code>removeRequest</code> | <i>Remove a background CORBA operation call from a group of outstanding calls.</i> |
|----------------------------|--|

Description

Removes one or more elements from the list of requests grouped by some naming scheme.

Usage

```
removeRequest(requestName, requestIndex=-1)
```

Arguments

`requestName` identifies the group of requests by the name used to create them via [.Corba](#) or [.Corba](#)

`requestIndex` specifies which elements of the request list, or by default all of them.

Details

The request lists are stored internally in C++ tables. The request functions in this package provide facilities to manage these tables. They can be used to implement distributed task management. Ideally, we will implement internal and user-level threads to facilitate this so that we can provide parallel servers as arguments in these calls rather than serializing the operation calls from the different slaves. Much more to be said on this. See <http://www.omegahat.org>.

Value

logical vector of the same length as `requestIndex` with each element indicating whether the operation was successful for the corresponding element of the list. If the value of `requestIndex` is -1, then the value is either TRUE or FALSE indicating whether the entire action was successful or not.

Author(s)

Duncan Temple Lang

References<http://www.omegahat.org/CORBA>**See Also**[deferredRequestList](#), [getRequestStatus](#), [getRequestValue](#)**Examples**

```
.Corba("x", "foo", .deferred="foo")
.Corba("y", "foo", .deferred="foo")

sendRequests("foo")
# now clear out all the requests
removeRequests("foo")
```

| | |
|--------------|---|
| sendRequests | <i>Send one or more background requests to CORBA objects.</i> |
|--------------|---|

Description

Perform the invocation of the specified CORBA task elements in the internal request list to the servers stored in each request element. These are tasks that were created via a call to the `.Corba` function with the argument `.send` being `FALSE` indicating that the task should be merely stored as a potential method invocation rather than dispatched to the CORBA server immediately. Also, the `.deferred` argument must have been specified so that the template task was stored internally for later dispatch. The value of the `.deferred` argument to the `.Corba` call should be the same as the name of the task list identified in this function.

Usage

```
sendRequests(name, which=-1, how=CorbaOpMode[["deferred"]])
```

Arguments

| | |
|--------------|---|
| name | the character identifier (i.e. string) used to identify which list of requests to dispatch. This is the value passed as the parameter <code>.deferred</code> in <code>.Corba</code> to create and add the task to the request list. |
| which | an integer vector identifying which elements of the request list are to be dispatched. If this is -1 (the default), all requests are dispatched. This allows one to avoid querying the length of the internal C++ list. |
| how | An integer indicating how the tasks should be dispatched. The possible invocation modes are <code>invoke</code> , <code>deferred</code> and <code>oneway</code> corresponding to a) each is a regular, blocking call; b) invoke each task as an asynchronous/background request whose result can be retrieved at a later time; c) invoke each task as a background request whose result (or return value) is discarded and unavailable to the caller. (These typically are specified as values from the <code>CorbaOpMode</code> vector.) |

Details

This passes control to the internal routines that manage and operate on the task lists created via the `.deferred` and `.send` arguments of the `.Corba` function.

Value

An integer vector indicating how many tasks were invoked. This is the length of the task list if the value of the `which` argument is `-1`. Otherwise, it is the length of the `which` argument.

Author(s)

Duncan Temple Lang

References

<http://www.omegahat.org/CORBA>

See Also

`.Corba`, `getRequestValue`, `getRequestServer`, `getRequestStatus`, `deferredRequestList`

Examples

```
.Corba("A", "doit", .deferred="wait",.send=F)
...
sendRequests("wait")
```

super

Inheritance-style lookup of function from within a closure.

Description

Finds a function in ancestor environments of the current one, so that we can avoid local definitions. This is used in the CORBA facilities within a closure when we provide a local function with an implicit `this`. Then we use `super` to find the real function and supply it the explicit `this`.

Usage

```
super(fun)
```

Arguments

`fun` the name of the function we are looking for

Details**Value**

a function definition for the specified name.

Author(s)

Duncan Temple Lang, Ross Ihaka

See Also

closure

Examples

```
# define a closure.
cl <- function(this) {
  # define a local nrow which calls the global one.
  nrow <- function() {
    super(nrow)(this)
  }
  return(list(nrow=nrow))
}
# create a closure from the generator
mobj <- cl(matrix(1:9,3,3))

# call the closure's nrow() function
mobj$nrow()
```

Index

- *Topic **CORBA, Background method invocation**
 - deferredRequestList, 11
- *Topic **CORBA, Class, Inheritance, Closure**
 - super, 34
- *Topic **CORBA, Distributed Computing**
 - .CorbaServer, 4
- *Topic **CORBA, class, closure**
 - baseMatrixHandlerGenerator, 6
- *Topic **CORBA, initialization, shell script**
 - corbaCommandLineArgs, 8
- *Topic **CORBA**
 - .Corba, 2
 - addCorbaConverter, 5
 - checkCorbaObjectExists, 7
 - CORBAinit, 1
 - corbaServices, 9
 - createNamingContext, 10
 - getCorbaConverters, 12
 - getIRContainerContents, 12
 - getNextCompletedTask, 14
 - getRequestServer, 15
 - getRequestStatus, 16
 - getRequestValue, 18
 - idleElementType, 19
 - idlInterface, 20
 - idlType, 21
 - interfaceRepository, 22
 - ior, 23
 - iorDump, 24
 - namedCorbaObject, 25
 - namingService, 26
 - namingServiceContents, 27
 - primitiveIORDump, 28
 - removeCorbaConverter, 29
 - removeNamedCorbaObject, 30
 - removeRequest, 31
 - sendRequests, 32
- *Topic **Remote Method Invocation**
 - .Corba, 2
- *Topic **interface**
 - .Corba, 2
 - checkCorbaObjectExists, 7
 - getNextCompletedTask, 14
 - getRequestServer, 15
 - getRequestStatus, 16
 - namingService, 26
 - namingServiceContents, 27
 - primitiveIORDump, 28
 - removeCorbaConverter, 29
 - removeNamedCorbaObject, 30
 - removeRequest, 31
- *Topic **keyword**
 - sendRequests, 32
- .Corba, 2, 5, 13–19, 29, 31–33
- .CorbaServer, 3, 4, 6, 7, 25, 31
- addCorbaConverter, 3, 5, 12, 30
- baseMatrixHandlerGenerator, 6
- checkCorbaObjectExists, 7
- commandArgs, 8
- corbaCommandLineArgs, 8
- CORBAinit, 1, 8
- CorbaOpMode, 4, 33
- corbaServices, 9
- corbaTypeCode, 9, 10
- corbaTypeCodeNames, 10
- createNamingContext, 10, 25, 27, 28, 31
- deferredRequestList, 11, 14, 16–18, 32, 33
- getCorbaConverters, 12, 30
- getIRContainerContents, 12, 21, 23
- getNextCompletedTask, 11, 14, 15, 16
- getRequestServer, 14, 15, 17, 33
- getRequestStatus, 11, 14, 16, 16, 18, 32, 33
- getRequestValue, 11, 14, 16, 17, 18, 32, 33
- idleElementType, 19

idlInterface, 3, 13, 19, 20, 21, 23
idlType, 13, 19, 21, 21
interfaceRepository, 2, 13, 20, 21, 22
ior, 13, 23, 25, 29
iorDump, 24, 29

namedCorbaObject, 11, 13, 21, 24, 25, 25,
27-29
namingService, 2, 11, 23, 25, 26, 28
namingServiceContents, 8, 11, 25, 27, 27

primitiveIORDump, 28

removeCorbaConverter, 6, 12, 29
removeNamedCorbaObject, 11, 25, 30
removeRequest, 11, 16-18, 31

sendRequests, 18, 32
super, 34