Package 'RCy3'

April 12, 2018

Type Package

Title Display and manipulate graphs in Cytoscape >= 3.3.0

Version 1.8.0

Date 2017-01-03

Author Tanja Muetze, Georgi Kolishovski, Paul Shannon

Maintainer Tanja Muetze <tanja.muetze14@alumni.imperial.ac.uk>, Georgi Kolishovski <g_kolishv@yahoo.com>, Paul Shannon <pshannon@systemsbiology.org>

Depends R (>= 3.2), graph (>= 1.48.0)

Imports httr, methods, RCurl, RJSONIO

Suggests BiocGenerics, RUnit, knitr, igraph, RColorBrewer, paxtoolsr, rmarkdown

SystemRequirements Cytoscape (>= 3.3.0), CyREST (>= 3.3.7), Java (>=8)

Description Vizualize, analyze and explore graphs, connecting R to Cytoscape (>= 3.3.0).

License Artistic-2.0

URL https://github.com/tmuetze/Bioconductor_RCy3_the_new_RCytoscape

BugReports https://github.com/tmuetze/Bioconductor_RCy3_the_new_RCytoscape/issues

LazyLoad yes

biocViews Visualization, GraphAndNetwork, ThirdPartyClient, Network

NeedsCompilation no

VignetteBuilder knitr

R topics documented:

`addCyEdge` *addCyEdge*

Description

Given a CytoscapeWindow containing a (possibly empty) graph, this method adds an edge. Edge attributes are added separately, via successive calls to sendEdgeAttributesDirect. The two nodes must already exist in the Cytoscape network.

Usage

```
addCyEdge(obj, sourceNode, targetNode, edgeType, directed)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`sourceNode`	a `character string` object.
`targetNode`	a `character string` object.
`edgeType`	a `character string` object.
`directed`	a `boolean` object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

sendEdgeAttributesDirect addCyNode

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())
window.name <- 'demo addCyEdge'
cw <- CytoscapeWindow (window.name, graph=makeSimpleGraph ())
displayGraph (cw)
layoutNetwork(cw)
directed = TRUE
addCyEdge (cw, 'A', 'B', 'synthetic rescue', directed)

## End(Not run)
```

---

addCyNode *addCyNode*

Description

Given a CytoscapeWindow containing a (possibly empty) graph, this method adds a node. Node attributes are added separately, via successive calls to sendNodeAttributesDirect. The new node must be unique – not already a member of the graph as known to Cytoscape.

Usage

```
addCyNode(obj, nodeName)
```

Arguments

obj	a CytoscapeWindowClass object.
nodeName	a character string object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

sendNodeAttributesDirect addCyEdge

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())
window.name <- 'demo addCyNode'
cw <- CytoscapeWindow (window.name, graph=makeSimpleGraph ())
displayGraph (cw)
layoutNetwork (cw)
```

```
  addCyNode (cw, 'A NEW NODE')
  layoutNetwork (cw)
  # redraw (cw) --> function no longer required

## End(Not run)
```

---

addGraphToGraph *addGraphToGraph*

Description

Given a CytoscapeWindow containing a graph, this method adds new nodes, edges, and their attributes. Thus, it is the way to extend a graph – to merge a new graph with an existing one. A typical use would be to add a second KEGG pathway to a CytoscapeWindow upon discovering that two KEGG pathways overlap, which share some enzymes and some reactions. No existing attributes are overwritten.

Usage

```
addGraphToGraph(obj, other.graph)
```

Arguments

obj	a CytoscapeWindowClass object.
other.graph	a graph object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())
window.name <- 'demo addGraphToGraph'
cw3 <- CytoscapeWindow (window.name, graph=makeSimpleGraph ())
displayGraph (cw3)
redraw (cw3)
layoutNetwork(cw3)

  # create a new graph, which adds two nodes, and edges between them
  # and an existing node, A

g2 <- new("graphNEL", edgemode = "directed")
g2 <- graph::addNode ('A', g2)
g2 <- graph::addNode ('D', g2)
g2 <- graph::addNode ('E', g2)
```

```
g2 <- initNodeAttribute (g2, "label", "char", "default node label")
g2 <- initEdgeAttribute (g2, "edgeType", "char", "unspecified")
g2 <- initEdgeAttribute (g2, "probability", "numeric", 0.0)

nodeData (g2, 'D', 'label') <- 'Gene D'
nodeData (g2, 'E', 'label') <- 'Gene E'

g2 <- graph::addEdge ('D', 'E', g2)
g2 <- graph::addEdge ('A', 'E', g2)

edgeData (g2, 'D', 'E', 'probability') <- 0.95
edgeData (g2, 'D', 'E', 'edgeType') <- 'literature'
edgeData (g2, 'A', 'E', 'edgeType') <- 'inferred'

addGraphToGraph (cw3, g2)
redraw (cw3)
layoutNetwork(cw3)


## End(Not run)
```

---

clearSelection *clearSelection*

Description

If any nodes are selected in the current Cytocape window, they will be unselected.

Usage

```
clearSelection(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

Nothing

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())
cw <- CytoscapeWindow ('clearSelection.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw)
```

```
  selectNodes (cw, 'A')
  print (getSelectedNodeCount (cw))   # should be 1
  clearSelection (cw)
  print (getSelectedNodeCount (cw))   # should be 0

## End(Not run)
```

---

copyVisualStyle *copyVisualStyle*

Description

Once you have designed a visual style, you may wish to duplicate it, perhaps in preparation for adding further mapping rules. Another scenario arises when style rules have been added to the 'default' style, and you wish to create a Cytoscape session file with your current network and this default style. However, the default style is not saved into a session, only explicitly named styles are. Use this method to achieve this.

Usage

```
copyVisualStyle(obj, from.style, to.style)
```

Arguments

obj	a CytoscapeConnectionClass object or CytoscapeWindow object.
from.style	a character string specifying the name of an existing style you wish to copy.
to.style	a character string, the name of an as yet non-existent style.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getVisualStyleNames setVisualStyle

Examples

```
## Not run:

  # create the usual demo graph and Cytoscape window, then
 # specify that all the edges should be 5 pixels wide.  This affects the 'default' style only
  # in order to save this style for later use, copy it to a
  # new style named 'fatEdgeStyle'
  # the related method 'setVisualStyle' must be called in order for
  # the fatEdgestyle to be associated with this window (and saved
  # into the CytoscapeSession file from the Cytoscape application's
  # File menu)
```

```
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.name = 'demo.copyVisualStyle'
  cw = CytoscapeWindow (window.name, graph=makeSimpleGraph ())
  setDefaultEdgeLineWidth (cw, 5);
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw)

     # create a unique style name, using millisecond precision, so should be unique
  time.msec = proc.time()[['elapsed']]
  new.unique.style.name = paste ('fatEdgeStyle', time.msec, sep='.')

  copyVisualStyle (cw, 'default', new.unique.style.name)
  new.names = getVisualStyleNames (cw)
  setVisualStyle (cw, new.unique.style.name)

   # save the session from the Cytoscape application menu. The new
   # style name will be saved along with the network and its attributes

## End(Not run)
```

---

createWindow *createWindow*

Description

Request that Cytoscape create a new window for the supplied CytoscapeWindowClass object. It will hold a new network, using the title supplied when the object's constructor was called.

This method will probably not often be useful: it is called behind the scenes by the CytoscapeWindow constructor unless you specify (in calling the constructor) 'create.window=FALSE'. In that case, or if you interactively delete the window in Cytoscape, or if you call the 'deleteWindow' or 'deleteAllWindows' methods, you can create a new window by calling this method.

Usage

```
createWindow(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
 # first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

 # create a CytoscapeWindowClass object by calling the constructor
c2 <- CytoscapeWindow ('cwc demo', makeSimpleGraph ())

 # delete the window and then recreate it
deleteAllWindows(CytoscapeConnection())
createWindow(c2)

## End(Not run)
```

---

createWindowFromSelection
createWindowFromSelection

Description

All selected nodes, their connecting edges, and associated attributes are copied into a new CytoscapeWindow, with the supplied title.

Usage

```
createWindowFromSelection(obj, new.windowTitle, return.graph)
```

Arguments

obj	a CytoscapeWindowClass object.
new.windowTitle	a String.
return.graph	an logical object.

Value

A new CytoscapeWindow object, with the graph slot populated with the new selected subgraph, if requested. If not requested, the graph slot holds an empty graph.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

selectNodes

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
title <- 'createWindowFromSelection demo'

cw <- CytoscapeWindow (title, makeSimpleGraph ())
displayGraph (cw)
layoutNetwork (cw)
selectNodes (cw, c ('A', 'C'))

new.window.title <- 'NEW WINDOW'
c2 <- createWindowFromSelection (cw, new.window.title, TRUE)
layoutNetwork (c2, 'force-directed')
print (getEdgeCount(c2))  # should be 1

## End(Not run)
```

`cy2.edge.names` *cy2.edge.names*

Description

Bioconductor graph edges are named, i.e., A~B. The same edge in the Cytoscape domain would be 'A (<edgeType>) B', where '<edgeType>' might be 'phosphorylates' or 'represses'.

Usage

```
cy2.edge.names(graph, R.edge.names=NA)
```

Arguments

`graph`	An R graph
`R.edge.names`	one or more R graph-style edge names. default NA, in which case all edges in the graph are translated to cy2-style.

Value

A named list, in with Cytoscape edges names are the content, and bioc graph edge names are their names.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

g <- makeSimpleGraph ()
cy2.edge.names (g)
  #                     A~B                       B~C                       C~A
  #    "A (phosphorylates) B" "B (synthetic lethal) C"       "C   (undefined) A"
cy2.edge.names (g, R.edge.names="B~C")
  #                     B~C
  # "B (synthetic lethal) C"


## End(Not run)
```

---

cyPlot *cyPlot*

Description

Given a node attribute data frame (node.df) with the node names in column 1, and an edge attribute data.frame (edge.df) with node names in the first two columns, cyPlot creates a graphNEL object with nodes, edges, and their attributes that can be loaded into Cytoscape with CytoscapeWindow.

Usage

```
cyPlot(node.df, edge.df)
```

Arguments

node.df	a data.frame with node names in the first column.
edge.df	a data.frame with node names in the first two columns.

Value

a graphNEL object with nodes, edges, and their attributes

Author(s)

Mark Grimes

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

# Create node and edge data frames
  node.tbl <- data.frame(Gene.Name=c("FOX", "CENT", "RTK"), Numeric.Data=as.numeric(c(10, 12, 7)), nodeType=c
  edge.tbl <- data.frame(Gene.1=c("FOX", "FOX", "CENT", "CENT"), Gene.2=c("CENT", "RTK", "FOX", "RTK"), Weigh
  window.name <- 'demo cyPlot'
```

```
# Create graph and send to Cytoscape
  cg <- cyPlot(node.tbl, edge.tbl)
  cw <- CytoscapeWindow (window.name, graph=cg)
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw)

## End(Not run)
```

CytoscapeConnection *CytoscapeConnection*

Description

The constructor for the CytoscapeConnectionClass. This class is both the base class for CytoscapeWindow objects, and quite usefully, and instantiable object in its own right. It is very useful for calling the many RCytoscape methods which do not address a single window in particular: getWindowList, getWindowCount, deleteWindow, getNodeShapes, etc.

Usage

```
CytoscapeConnection (host = "localhost", port = 1234)
```

Arguments

host	Defaults to 'localhost', this is the domain name of a machine which is running Cytoscape with the appropriate CyREST plugin.
port	Defaults to 1234, this may be any port to which CyREST is listening.

Value

An object of the CytoscapeConnection Class.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

ping version getWindowCount getWindowID getWindowCount getWindowList deleteWindow deleteAllWindows getNodeShapes getAttributeClassNames getLineStyles getArrowShapes getLayoutNames haveNodeAttribute haveEdgeAttribute getGraphFromCyWindow hidePanel dockPanel floatPanel

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
deleteAllWindows (cy)
getNodeShapes (cy)
```

```
  # [1] "DIAMOND"         "OCTAGON"         "TRIANGLE"        "VEE"
  # [5] "HEXAGON"         "RECTANGLE"       "PARALLELOGRAM"   "ROUND_RECTANGLE"
  # [9] "ELLIPSE"
  hidePanel (cy, 'control')

## End(Not run)
```

`CytoscapeConnectionClass-class` *Class "CytoscapeConnectionClass"*

Description

A class providing access to operations of the Cytoscape application which are not specific to a particular window.

Slots

`uri:` An `attrData`, the address of the Cytoscape server.

Methods

ping

version

getWindowcount

getWindowID

getWindowCount

getWindowList

destroyWindow

destroyAllWindows

getNodeShapes

getAttributeClassNames

getLineStyles

getArrowShapes

haveNodeAttribute

haveEdgeAttribute

copyNodeAttributesFromCyGraph

copyEdgeAttributesFromCyGraph

getGraphFromCyWindow

hidePanel

dockPanel

floatPanel

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())
# create a CytoscapeConnectionClass object by calling the constructor
cy <- CytoscapeConnection (host='localhost', port=1234)


## End(Not run)
```

CytoscapeWindow *CytoscapeWindow*

Description

The constructor for the CytoscapeWindowClass

Usage

```
CytoscapeWindow(title, graph = new("graphNEL", edgemode='directed'),
                host = "localhost", port = 1234,
            create.window = TRUE, overwriteWindow=FALSE, collectTimings=FALSE)
```

Arguments

title	A character string, this is the name you will see on the Cytoscape network window. Multiple windows with the same name are not permitted.
graph	A Bioconductor graph.
host	Defaults to 'localhost', this is the domain name of a machine which is running Cytoscape with the appropriate CyREST plugin.
port	Defaults to 1234, this may be any port to which CyREST is listening.
create.window	Defaults to TRUE, but if you want a CytoscapeWindow just to call what in Java we would call 'class methods' – getWindowList (), for instance, a CytoscapeWindow without an actual window can be useful.
overwriteWindow	Every Cytoscape window must have a unique title. If the title that you supply is already in use, this method will fail unless you specify TRUE for this parameter, in which case the pre-existing window with the same title will be deleted before this new one is created.
collectTimings	Defaults to FALSE. Will record and report the time required to send a graph to Cytoscape.

Value

An object of the CytoscapeWindow Class.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

CytoscapeWindow existing.CytoscapeWindow, predictTimeToDisplayGraph

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('new.demo', new ('graphNEL'))

## End(Not run)
```

CytoscapeWindowClass-class *Class "CytoscapeWindowClass"*

Description

A class providing access to the Cytoscape application.

Slots

`title`: An `attrData`, the name of the window.

`window.id`: An `attrData` Cytoscape's identifier.

`graph`: An `attrData`, a graph instance.

`collectTimings`: A `logical` object.

`suid.name.dict`: A `list` object.

`edge.suid.name.dict`: A `list` object.

`view.id`: A `numeric` object.

`uri`: An `attrData` the address of the Cytoscape CyREST API.

Methods

createWindow

destroyWindow

destroyAllWindows

displayGraph

firstNeighbors

getArrowShapes

getLayoutNames

getLineStyles

getNodeShapes

getWindowCount

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  # create a CytoscapeWindowClass object by calling the constructor
  c2 <- CytoscapeWindow ('cwc demo', makeSimpleGraph ())


## End(Not run)
```

---

deleteAllWindows *deleteAllWindows*

Description

Delete all the network windows currently held by Cytoscape, removing them from the screen, and deleting Cytoscape's copy of all of the graphs. The R graphs are unchanged.

Usage

```
deleteAllWindows(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  cy <- CytoscapeConnection ()
  cw1 = CytoscapeWindow ('cw1')
  cw2 = CytoscapeWindow ('cw2')
  deleteAllWindows (cy)

## End(Not run)
```

deleteEdgeAttribute *deleteEdgeAttribute*

Description

Node and edge attributes are usually added to a Cytoscape network by defining them on the graph used to construct a CytoscapeWindow. Once Cytoscape has been passed an attribute, however, it persists until you exit the application or delete it by using the Cytoscape graphical user interface or by calling this method.

Usage

```
deleteEdgeAttribute(obj, attribute.name)
```

Arguments

obj	a CytoscapeConnectionClass object or CytoscapeWindow object.
attribute.name	a character string, the name of the attribute you wish to delete.

Value

nothing

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getEdgeAttributeNames addEdgeAttribute deleteNodeAttribute

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.name = 'demo.deleteEdgeAttribute'
  cw = CytoscapeWindow (window.name, graph=makeSimpleGraph ())
  setDefaultEdgeLineWidth (cw, 5);
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw)

    # before:
  print (getEdgeAttributeNames (cw))
    # [1] "name"        "interaction" "edgeType"    "score"       "misc"
  deleteEdgeAttribute (cw, 'score')
    # after
  print (getEdgeAttributeNames (cw))
    # [1] "name"        "interaction" "edgeType"    "misc"

## End(Not run)
```

deleteNodeAttribute *deleteNodeAttribute*

Description

Node and node attributes are usually added to a Cytoscape network by defining them on the graph used to construct a CytoscapeWindow. Once Cytoscape has been passed an attribute, however, it persists until you exit the application or delete it by using the Cytoscape graphical user interface or by calling this method.

Usage

```
deleteNodeAttribute(obj, attribute.name)
```

Arguments

obj	a CytoscapeConnectionClass object or CytoscapeWindow object.
attribute.name	a character string, the name of the attribute you wish to delete.

Value

nothing

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getNodeAttributeNames addNodeAttribute

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.name = 'demo.deleteNodeAttribute'
  cw = CytoscapeWindow (window.name, graph=makeSimpleGraph ())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw)

    # before:
  print (getNodeAttributeNames (cw))
    # [1] "name"  "type"  "lfc"   "label" "count"
  deleteNodeAttribute (cw, 'count')
    # after:
  print (getNodeAttributeNames (cw))
    # [1] "name"  "type"  "lfc"   "label"


## End(Not run)
```

deleteSelectedEdges *deleteSelectedEdges*

Description

This function removes all selected edges in Cytoscape. These edges will still exist in the corresponding R graph until you delete them there as well.

Usage

```
deleteSelectedEdges(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

selectEdges cy2.edge.names deleteSelectedNodes

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('deleteSelectedEdges.test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layoutNetwork(cw, 'force-directed')
print (cy2.edge.names (cw@graph))   # find out Cytoscape's names for  these edges
selectEdges (cw, "B (synthetic lethal) C")
deleteSelectedEdges (cw)
redraw (cw)

## End(Not run)
```

deleteSelectedNodes *deleteSelectedNodes*

Description

This function deletes all the selected nodes in Cytoscape. Edges originating or terminating in these nodes will also be deleted. The nodes and edges will still exist in the corresponding R graph until you explicitly delete them there as well.

Usage

```
deleteSelectedNodes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

selectNodes deleteSelectedEdges

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('deleteSelectedNodes.test', graph=makeSimpleGraph())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw, 'force-directed')
  print (nodes (cw@graph))
    # [1] "A" "B" "C"
  selectNodes (cw, "B")
  deleteSelectedNodes (cw)

## End(Not run)
```

deleteWindow *deleteWindow*

Description

Delete the window associated with the supplied CytoscapeConnection object. In addition, Cytoscape's copy of the network is deleted from Cytoscape's memory store, but the R graph object is unaffected. There are two different ways to use this method. First, if you call it on a CytoscapeWindow object, using the default window.title value of NA, the Cytoscape window itself will be deleted. Alternatively, if you supply a window.title as the second argument – independent of whether or not the first argument is a CytoscapeConnection object, or its subclass, a CytoscapeWindow object, the named window is deleted.

Usage

```
deleteWindow(obj, window.title=NA)
```

Arguments

obj	a CytoscapeConnectionClass object, or subclass
window.title	a string object, optional title

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.title <- 'demo deleteWindow'
  cw <- CytoscapeWindow (window.title, graph=makeSimpleGraph())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw)
  deleteWindow (cw)
  cw2 <- CytoscapeWindow ('demo 2')
  cy = CytoscapeConnection ()
  deleteWindow (cy, 'demo 2')

## End(Not run)
```

demoSimpleGraph *demoSimpleGraph*

Description

Create, display and render the 3-node, 3-edge graph, with some biological trappings.

Usage

```
demoSimpleGraph()
```

Value

Returns a CytoscapeWindow object for subsequent manipulation

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cwd <- demoSimpleGraph ()

## End(Not run)
```

displayGraph *displayGraph*

Description

This method transmits the CytoscapeWindowClass's graph data, from R to Cytoscape: nodes, edges, node and edge attributes, and displays it in a window titled as specified by the objects 'title' slot. With large graphs, this transmission may take a while. The resulting view of the network in Cytoscape will need layout and vizmap rendering; layout so that all the nodes and edges can be seen; rendering so that data attributes can control the appearance of the the nodes and edges.

Usage

```
displayGraph(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('displayGraph.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
redraw (cw)

## End(Not run)
```

---

dockPanel *dockPanel*

Description

The specified panel is returned to its 'home' position in the Cytoscape Desktop if it had been previously floating or hidden. The `panelName` parameter is very flexible: a match is defined as a case-independent match of the supplied panelName to any starting characters in the actual panelName. Thus, 'd' and 'DA' both identify 'Data Panel'. Possible options also include: 'WEST', 'EAST', 'SOUTH', 'SOUTH_WEST'. The 'SOUTH' panel is the Data Panel and the 'WEST' panel is the control panel.

Usage

```
dockPanel(obj, panelName)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.
`panelName`	a character string, providing a partial or complete case-independent match to the start of the name of an actual panel.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

floatPanel hidePanel

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  dockPanel (cy, 'Control Panel')
   # or
  dockPanel (cy, 'c')

  dockPanel (cy, 'EAST')

## End(Not run)
```

---

eda *eda*

Description

Obtain the value of the specified edge attribute for every edge in the graph.

Usage

```
eda(graph, edge.attribute.name)
```

Arguments

graph	typically, a bioc graphNEL object
edge.attribute.name	a character string

Details

The edge.attribute.name may be obtained from the function, eda.names.

Value

A list, the contents of which are the attribute values, the names of which are the names of the edges.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

eda.names

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

g  <- makeSimpleGraph()
eda (g, 'edgeType')
  #             A|B                B|C                C|A
  # "phosphorylates" "synthetic lethal"        "undefined"


## End(Not run)
```

`eda.names` *eda.names*

Description

Retrieve the names of the edge attributes in the specified graph. These are typically strings like 'score', 'weight', 'link', and (strongly recommended when you create a graph) 'edgeType'. Once you are reminded of the names of the edge attributes, you can use the method 'eda' to get all the values of this attribute for the edges in the graph.

Usage

```
eda.names (graph)
```

Arguments

`graph`	typically, a bioc graphNEL)

Value

A list, the contents of which are the attribute values, the names of which are the names of the edges.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

noa, eda, noa.names

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

g <- makeSimpleGraph()
eda.names (g)
 # "edgeType" "score"    "misc"

## End(Not run)
```

existing.CytoscapeWindow *existing.CytoscapeWindow*

Description

The constructor for the CytoscapeWindowClass, used when Cytoscape already contains and displays a network.

Usage

```
existing.CytoscapeWindow (title, host='localhost', port=1234, copy.graph.from.cytoscape.to.R=FALS
```

Arguments

title	A character string, this is the name of an existing Cytoscape network window. This name enables RCy3 to identify and connect to the proper Cytoscape window and network that it contains.
host	Defaults to 'localhost', this is the domain name of a machine which is running Cytoscape with the appropriate CyREST plugin.
port	Defaults to 1234, this may be any port to which CyREST is listening.
copy.graph.from.cytoscape.to.R	Defaults to FALSE, but you may want a copy in R, for further exploration.

Value

An object of the existing.CytoscapeWindow class.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  cw <- CytoscapeWindow ('demo.existing', graph=makeSimpleGraph ())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  cw2 <- existing.CytoscapeWindow ('demo.existing',  copy.graph.from.cytoscape.to.R=TRUE)

## End(Not run)
```

fitContent *fitContent*

Description

Display the current graph using all of the available window (the Cytoscape drawing canvas).

Usage

```
fitContent(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setZoom fitSelectedContent

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('fitContent.test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layoutNetwork(cw, 'force-directed')
setZoom (cw, 0.1)
fitContent (cw)
setZoom (cw, 10.0)
fitContent (cw)

## End(Not run)
```

`fitSelectedContent` *fitSelectedContent*

Description

Using all of the available window (the Cytoscape drawing canvas) display the current graph.

Usage

```
fitSelectedContent(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setZoom fitContent

Examples

```
  ## Not run:
  #WARNING: function currently not supported

  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('fitSelectedContent.test', graph=makeSimpleGraph())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw, 'force-directed')
  setZoom (cw, 0.1)
  selectNodes (cw, 'A')
  fitSelectedContent (cw)
  setZoom (cw, 10.0)
  fitSelectedContent (cw)

## End(Not run)
```

floatPanel *floatPanel*

Description

The specified panel will 'float' detached from its 'home' position in the Cytoscape Desktop. The panelName parameter is very flexible: a match is defined as a case-independent match of the supplied panelName to any starting characters in the actual panelName. Thus, 'd' and 'DA' both identify 'Data Panel'. Possible options also include: 'WEST', 'EAST', 'SOUTH', 'SOUTH_WEST'. The 'SOUTH' panel is the Data Panel and the 'WEST' panel is the control panel.

Usage

```
floatPanel(obj, panelName)
```

Arguments

obj	a CytoscapeConnectionClass object.
panelName	a character string, providing a partial or complete case-independent match to the start of the name of an actual panel.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

hidePanel dockPanel

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy  <- CytoscapeConnection ()
  floatPanel (cy, 'Control Panel')
    # or with less typing
  floatPanel (cy, 'c')

  floatPanel (cy, 'SOUTH_WEST')

## End(Not run)
```

getAdjacentEdgeNames *getAdjacentEdgeNames*

Description

Given one or more node names, this method returns the 'cy2-style' names of the immediately adjacent edges – suitable for being passed, for instance, to selectEdges, and thereby extending the selection.

Usage

```
getAdjacentEdgeNames(graph, node.names)
```

Arguments

graph	An R graph
node.names	character strings

Value

Zero or more cy2-style edge names.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

cy2.edge.names, selectEdges, getSelectedNodes, selectFirstNeighborsOfSelectedNodes

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  g <- makeSimpleGraph ()
  print (nodes (g))
    # [1] "A" "B" "C"
  print (getAdjacentEdgeNames (g, 'A'))
    # [1] "A (phosphorylates) B" "C (undefined) A"

## End(Not run)
```

getAllEdgeAttributes *getAllEdgeAttributes*

Description

Create a data frame with all the edge attributes for the graph contained by the supplied CytoscapeWindow object. Only the local copy of the graph is queried. If you want all the (possibly different) edge attributes from the Cytoscape network which corresponds to this graph, one option is to create a new CytoscapeWindow; see the existing.CytoscapeWindow function.

Usage

```
getAllEdgeAttributes(obj, onlySelectedEdges=FALSE)
```

Arguments

obj	a CytoscapeWindowClass object object.
onlySelectedEdges	a logical variable, used to restrict the query.

Value

A data frame, with a column for each attribute, a row for each edge.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getEdgeAttribute deleteEdgeAttribute getAllNodeAttributes

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

window.name = 'demo.getAllEdgeAttributes'
cw = CytoscapeWindow (window.name, graph=makeSimpleGraph ())
displayGraph (cw)
redraw (cw)
layoutNetwork(cw)
  # get all attributes for all edges
tbl.eda = getAllEdgeAttributes (cw, onlySelectedEdges=FALSE)
tbl.eda
  #             edgeType source target score         misc
  # A|B   phosphorylates      A      B    35 default misc
  # B|C synthetic lethal      B      C   -12 default misc
  # C|A        undefined      C      A     0 default misc


## End(Not run)
```

getAllEdges *getAllEdges*

Description

Retrieve all edges in the current graph, expressed in the standard Cytoscape notation.

Usage

```
getAllEdges(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('getAllEdges.test', graph=makeSimpleGraph())
  displayGraph (cw)
  print (getAllEdges (cw))
    # [1] "A (phosphorylates) B"   "B (synthetic lethal) C" "C (undefined) A"

## End(Not run)
```

getAllNodeAttributes *getAllNodeAttributes*

Description

Create a data frame with all the node attributes for the graph contained by the supplied CytoscapeWindow object. Only the local copy of the graph is queried. If you want all the (possibly different) node attributes from the Cytoscape network which corresponds to this graph, one option is to create a new CytoscapeWindow; see the existing.CytoscapeWindow function.

Usage

```
getAllNodeAttributes(obj, onlySelectedNodes=FALSE)
```

Arguments

obj	a CytoscapeWindowClass object object.
onlySelectedNodes	a logical variable, used to restrict the query.

Value

A data frame, with a column for each attribute, a row for each node.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getNodeAttribute deleteNodeAttribute

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.name <- 'demo.getAllNodeAttributes'
  cw <- CytoscapeWindow (window.name, graph=makeSimpleGraph ())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork (cw, 'force-directed')
    # get all attributes for all nodes
  tbl.noa <- getAllNodeAttributes (cw, onlySelectedNodes=FALSE)
  tbl.noa
  #                   type lfc  label count
  # A               kinase  -3 Gene A     2
  # B transcription factor   0 Gene B    30
  # C         glycoprotein   3 Gene C   100

  selectNodes (cw, 'A')
  getAllNodeAttributes (cw, TRUE)
  #     type lfc  label count
  # A kinase  -3 Gene A     2

## End(Not run)
```

---

getAllNodes *getAllNodes*

Description

Retrieve the identifiers of all the nodes in the current graph - a list of strings.

Usage

```
getAllNodes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings. Note that node names are returned – their original and primary identifiers – and that these may be different from the node labels that you see when you look at the graph in Cytoscape.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('getAllNodes.test', graph=makeSimpleGraph())
  displayGraph (cw)
  print (getAllNodes (cw))
    # [1] "A" "B" "C"

## End(Not run)
```

---

getArrowShapes *getArrowShapes*

Description

Retrieve the names of the currently supported 'arrows' – the decorations can (optionally) appear at the ends of edges, adjacent to the nodes they connect, and conveying information about the nature of the nodes' relationship.

Usage

```
getArrowShapes(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

A list of character strings, e.g., 'DIAMOND', 'T', 'ARROW'

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
getArrowShapes (cy)
  # [1] "DIAMOND_SHORT_1" "HALF_TOP"        "DELTA_SHORT_2"   "DELTA"
  # [5] "HALF_BOTTOM"     "CIRCLE"          "DIAMOND"         "DIAMOND_SHORT_2"
  # [9] "NONE"            "T"               "ARROW"           "ARROW_SHORT"
  # [13] "DELTA_SHORT_1"

## End(Not run)
```

`getAttributeClassNames` *getAttributeClassNames*

Description

Retrieve the names of the recognized and supported names for the class of any node or edge attribute. Two or three options are provided for each of the basic types, with the intention that you can use names that seem natural to you, and RCytoscape will recognize them.

Usage

```
getAttributeClassNames(obj)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.

Value

A list of character strings group, e.g., "floating|numeric|double", "integer|int", "string|char|character"

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cyw <- CytoscapeConnection ()
getAttributeClassNames (cyw)
 # [1] "floating|numeric|double" "integer|int"            "string|char|character"

## End(Not run)
```

getCenter *getCenter*

Description

This method returns the coordinates of the current center of the visible Cytoscape canvas, or drawing surface. The intial values are a little unpredictable, but seem to be on the order of 100 for both x and y.

Usage

```
getCenter(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A names list, x and y.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getCenter getZoom setZoom

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

window.title = 'getCenter demo'
cw <- CytoscapeWindow (window.title, graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layoutNetwork(cw, 'force-directed')
print (getCenter (cw))

## End(Not run)
```

getDefaultBackgroundColor *getDefaultBackgroundColor*

Description

Retrieve the default background color for the next CytoscapeWindow.

Usage

```
getDefaultBackgroundColor(obj, vizmap.style.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
vizmap.style.name	a character object, 'default' by default

Value

A character string, a hexadecimal, e.g. #000000

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  print (getDefaultBackgroundColor (cy)) # a hex number
    # [1] "#FFFFFF"

## End(Not run)
```

getDefaultEdgeReverseSelectionColor *getDefaultEdgeReverseSelectionColor*

Description

Retrieve the default color used to display not selected edges.

Usage

```
getDefaultEdgeReverseSelectionColor(obj, vizmap.style.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
vizmap.style.name	a character object, 'default' by default

Value

A character string, a hexadecimal, e.g. #000000

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  print (getDefaultEdgeReverseSelectionColor (cy))
    # [1] "#323232"

## End(Not run)
```

---

getDefaultEdgeSelectionColor *getDefaultEdgeSelectionColor*

Description

Retrieve the default color used to display selected edges.

Usage

```
getDefaultEdgeSelectionColor(obj, vizmap.style.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
vizmap.style.name	a character object, 'default' by default

Value

A character string, a hexadecimal, e.g. #000000

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  print (getDefaultEdgeSelectionColor (cy))
    # [1] "#FF0000"

## End(Not run)
```

`getDefaultNodeReverseSelectionColor` *getDefaultNodeReverseSelectionColor*

Description

Retrieve the default color used to display not selected nodes.

Usage

```
getDefaultNodeReverseSelectionColor(obj, vizmap.style.name)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.
`vizmap.style.name`	a `character` object, 'default' by default

Value

A character string, a hexadecimal, e.g. #000000

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  print (getDefaultNodeReverseSelectionColor (cy))
    # [1] "#1E90FF"

## End(Not run)
```

getDefaultNodeSelectionColor
getDefaultNodeSelectionColor

Description

Retrieve the default color used to display selected nodes.

Usage

```
getDefaultNodeSelectionColor(obj, vizmap.style.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
vizmap.style.name	a character object, 'default' by default

Value

A character string, a hexadecimal, e.g. #000000

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  print (getDefaultNodeSelectionColor (cy))
    # [1] "#FFFF00"

## End(Not run)
```

getDirectlyModifiableVisualProperties
getDirectlyModifiableVisualProperties

Description

Retrieve the names of those visual attributes which can be set directly, bypassing vizmap rules.

Usage

```
getDirectlyModifiableVisualProperties(obj, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
vizmap.style.name	a visual style name.

Value

A list of about 106 character strings, e.g., "EDGE_LABEL_FONT_SIZE" and "NODE_SHAPE"

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeConnection ()
getDirectlyModifiableVisualProperties (cw, vizmap.style.name="default")

## End(Not run)
```

---

getEdgeAttribute *getEdgeAttribute*

Description

Node and edge attributes are usually added to a Cytoscape network by defining them on the graph used to construct a CytoscapeWindow. The small family of methods described here, however, provide another avenue for adding an edge attribute, for learning which are currently defined, and for deleting an edge attribute.

Note that edge (and node) attibutes are defined, not just for a specific, single CytoscapeWindow, but for an entire Cytoscape application session.Thus if you have two nodes (or edges) with the same ID in two different windows, adding a node attribute results in both nodes having that attribute.

Usage

```
getEdgeAttribute(obj, edge.name, attribute.name)
```

Arguments

obj	a CytoscapeConnectionClass object or CytoscapeWindow object.
edge.name	a character string specifying the Cytoscape-style name of an edge.
attribute.name	a character string, the name of the attribute you wish to retrieve.

Value

The attribute in question, which may be of any scalar type.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getEdgeAttributeNames deleteEdgeAttribute

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.name = 'demo.getEdgeAttribute'
  cw = CytoscapeWindow (window.name, graph=makeSimpleGraph ())
  setDefaultEdgeLineWidth (cw, 5);
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw)

  score.bc = getEdgeAttribute (cw, "B (synthetic lethal) C", 'score')
  print (paste ("should be -12: ", score.bc))

## End(Not run)
```

---

getEdgeAttributeNames *getEdgeAttributeNames*

Description

Node and edge attributes belong to the Cytoscape session as a whole, not to a particular window. Use this method to find out the name of the currently defined edge attributes.

Usage

```
getEdgeAttributeNames(obj)
```

Arguments

obj	a CytoscapeConnectionClass object or CytoscapeWindow object.

Value

A list of names.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getEdgeAttribute deleteEdgeAttribute getNodeAttributeNames

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('demo.getEdgeAttributeNames', graph=makeSimpleGraph())
displayGraph (cw)
print (getEdgeAttributeNames (cw))
  # [1] "name"        "interaction" "edgeType"    "score"       "misc"

## End(Not run)
```

`getEdgeCount` *getEdgeCount*

Description

Reports the number of the edges in the current graph.

Usage

```
getEdgeCount(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

A list of character strings.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('getEdgeCount.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
redraw (cw)
getEdgeCount (cw)
# [1] 3

## End(Not run)
```

getFirstNeighbors *getFirstNeighbors*

Description

Returns a non-redundant ('uniquified') list of all of the first neighbors of the supplied list of nodes.

Usage

```
getFirstNeighbors(obj, node.names, as.nested.list = FALSE)
```

Arguments

obj	a CytoscapeWindowClass object.
as.nested.list	a Boolean object of if a nested list or a concatenated list should be returned
node.names	a String list object.

Value

A list of node names.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

selectNodes selectFirstNeighborsOfSelectedNodes

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('getFirstNeighbors.test', graph=makeSimpleGraph())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw, 'grid')
  print (getFirstNeighbors (cw, 'A'))
  selectNodes (cw, getFirstNeighbors (cw, 'A'))  # note that A is not selected

## End(Not run)
```

getGraph *getGraph*

Description

Returns the Bioconductor graph object which belongs to the specified CytoscapeWindow object

Usage

```
getGraph(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A graph object.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('getGraph.test', graph=makeSimpleGraph())
  displayGraph (cw)
  print (getGraph (cw))

## End(Not run)
```

getGraphFromCyWindow *getGraphFromCyWindow*

Description

Returns the Cytoscape network as a Bioconductor graph

Usage

```
getGraphFromCyWindow(obj, window.title)
```

Arguments

obj	a CytoscapeConnectionClass object.
window.title	a string object.

Value

A Bioconductor graph object.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('getGraphFromCyWindow.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw)
  g.cy <- getGraphFromCyWindow (cw, 'getGraphFromCyWindow.test')
  print (g.cy)  # 3 nodes, 3 edges

## End(Not run)
```

---

getLayoutNameMapping *getLayoutNameMapping*

Description

The Cytoscape 'Layout' menu lists many layout algorithms, but the names presented there are different from the names by which these algorithms are known to layout method. This method returns a named list in which the names are from the GUI, and the values identify the names you must use to choose an algorithms in the programmatic interface.

Usage

```
getLayoutNameMapping(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

A named list of strings.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getLayoutNames getLayoutPropertyNames getLayoutPropertyType getLayoutPropertyValue setLayoutProperties

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
layout.name.map <- getLayoutNameMapping (cy)
print (head (names (layout.name.map), n=3))
  # [1] "Attribute Circle Layout"     "Stacked Node Layout"
  # [3] "Degree Sorted Circle Layout"
print (head (as.character (layout.name.map), n=3))
  # [1] "attribute-circle"    "stacked-node-layout" "degree-circle"

## End(Not run)
```

`getLayoutNames` *getLayoutNames*

Description

Retrieve the names of the currently supported layout algorithms. These may be used in subsequent calls to the 'layoutNetwork' function. Note that some of the more attractive layout options, from yFiles, cannot be run except from the user interface; their names do not appear here.

Usage

```
getLayoutNames(obj)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.

Value

A list of character strings, e.g., "jgraph-circle" "attribute-circle" "jgraph-annealing"

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getLayoutNameMapping getLayoutNames getLayoutPropertyNames getLayoutPropertyType getLayoutPropertyValue setLayoutProperties

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
getLayoutNames (cy)
 #  [1] "jgraph-circle" "attribute-circle" "jgraph-annealing"  ...

## End(Not run)
```

---

getLayoutPropertyNames

getLayoutPropertyNames

Description

Returns a list of the tunable properties for the specified layout.

Usage

```
getLayoutPropertyNames(obj, layout.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
layout.name	a string object.

Value

A named list of strings.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

layout getLayoutNames getLayoutNameMapping getLayoutPropertyType getLayoutPropertyValue setLayoutProperties

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
prop.names <- getLayoutPropertyNames (cy, 'isom')
print (prop.names)
  # [1] "maxEpoch"           "radiusConstantTime" "radius"             "minRadius"
  # [5] "initialAdaptation"  "minAdaptation"      "sizeFactor"         "coolingFactor"
```

```
    # [9] "singlePartition"

## End(Not run)
```

---

getLayoutPropertyType *getLayoutPropertyType*

Description

Returns the type of one of the tunable properties (property.name) for the specified layout.

Usage

```
getLayoutPropertyType(obj, layout.name, property.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
layout.name	a string object.
property.name	a string object.

Value

A character string specifying the type. These types do not always necessarily need to be R types.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

layout getLayoutNames getLayoutNameMapping getLayoutPropertyNames getLayoutPropertyValue setLayoutProperties

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
prop.names <- getLayoutPropertyNames (cy, 'isom')
print (prop.names)
  # "maxEpoch"           "radiusConstantTime" "radius"             "minRadius"
  # "initialAdaptation"  "minAdaptation"      "sizeFactor"         "coolingFactor"
  # "singlePartition"

sapply (prop.names, function (pn) getLayoutPropertyType (cy, 'isom', pn))
  # maxEpoch radiusConstantTime             radius          minRadius
  #          "int"              "int"              "int"              "int"
  # initialAdaptation      minAdaptation         sizeFactor      coolingFactor
  #          "double"           "double"           "double"           "double"
```

```
    # singlePartition
    #       "boolean"

## End(Not run)
```

---

getLayoutPropertyValue

getLayoutPropertyValue

Description

Returns the appropriately typed value of the specified tunable property for the specified layout.

Usage

```
getLayoutPropertyValue(obj, layout.name, property.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
layout.name	a string object.
property.name	a string object.

Value

Typically an integer, numeric or string value, the current setting of this property for this layout.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

layout getLayoutNames getLayoutNameMapping getLayoutPropertyNames getLayoutPropertyType setLayoutProperties

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
prop.names <- getLayoutPropertyNames (cy, 'isom')
print (prop.names)
  # "maxEpoch"           "radiusConstantTime" "radius"             "minRadius"
  # "initialAdaptation"  "minAdaptation"      "sizeFactor"         "coolingFactor"
  # "singlePartition"
sapply (prop.names, function (pn) getLayoutPropertyValue (cy, 'isom', 'coolingFactor'))
  # maxEpoch radiusConstantTime             radius          minRadius
  #       2                 2                  2                  2
  # initialAdaptation      minAdaptation         sizeFactor      coolingFactor
```

```
    #              2                 2                 2                 2
    # singlePartition
    #              2

## End(Not run)
```

---

getLineStyles *getLineStyles*

Description

Retrieve the names of the currently supported line types – values which can be used to render edges, and thus can be used in calls to 'setEdgeLineStyleRule'.

Usage

```
getLineStyles(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

A list of character strings, e.g., 'SOLID', 'DOT'

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  getLineStyles (cy)
  #  [1] "SOLID" "LONG_DASH" "EQUAL_DASH" ...

## End(Not run)
```

getNodeAttribute *getNodeAttribute*

Description

Node and node attributes are usually added to a Cytoscape network by defining them on the graph used to construct a CytoscapeWindow. The small family of methods described here, however, provide another avenue for adding an node attribute, for learning which are currently defined, and for deleting and node attribute.

Note that node (and node) attibutes are defined not just for a specific, single CytoscapeWindow, but for an entire Cytoscape application session. Thus if you have two nodes (or nodes) with the same ID (the same name) in two different windows, adding a node attribute results in both nodes having that attribute.

Usage

```
getNodeAttribute(obj, node.name, attribute.name)
```

Arguments

obj	a CytoscapeConnectionClass object or CytoscapeWindow object.
node.name	a character string specifying the Cytoscape-style name of an node.
attribute.name	a character string, the name of the attribute you wish to retrieve.

Value

The attribute in question, which may be of any scalar type.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getNodeAttributeNames deleteNodeAttribute

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

window.name = 'demo.getNodeAttribute'
cw = CytoscapeWindow (window.name, graph=makeSimpleGraph ())
displayGraph (cw)
redraw (cw)
layoutNetwork(cw)

count.B = getNodeAttribute (cw, "B", 'count')
count.B
  # [1] 30
```

```
## End(Not run)
```

---

getNodeAttributeNames *getNodeAttributeNames*

Description

Node and node attributes belong to the Cytoscape session as a whole, not to a particular window. Use this method to find out the names of the currently defined node attributes.

Usage

```
getNodeAttributeNames(obj)
```

Arguments

obj	a CytoscapeConnectionClass object or CytoscapeWindow object.

Value

A list of names.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getNodeAttribute deleteNodeAttribute getEdgeAttributeNames

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('demo.getNodeAttributeNames', graph=makeSimpleGraph())
  displayGraph (cw)
  # displayGraph is required. If you do not display the graph,
  # your R graph data has not been sent to Cytoscape yet.
  print (getNodeAttributeNames (cw))
    # [1] "name"  "type"  "lfc"   "label" "count"

## End(Not run)
```

getNodeCount *getNodeCount*

Description

Reports the number of nodes in the current graph.

Usage

```
getNodeCount(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('getNodeCount.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  redraw (cw)
  getNodeCount (cw)
  # [1] 3

## End(Not run)
```

getNodePosition *getNodePosition*

Description

Get the position of the specified nodes on the CytoscapeWindow canvas. Useful in retrieving the current position of nodes in the window.

Usage

```
getNodePosition(obj, node.names)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	a list of strings, the names of nodes to select.

Value

A names list of x,y pairs; names are the identifiers of nodes supplied when the graph was created.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('getNodePosition.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw)
getNodePosition (cw, c ('A', 'B', 'C'))

## End(Not run)
```

---

getNodeShapes *getNodeShapes*

Description

Retrieve the names of the currently supported node shapes, which can then be used in calls to setNodeShapeRule and setDefaultVizMapValue.

Usage

```
getNodeShapes(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

A list of character strings, e.g. 'ELLIPSE', 'RECTANGLE'.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
getNodeShapes(cy)
 # [1] "VEE"              "RECTANGLE"       "PARALLELOGRAM"   "DIAMOND"
 # [5] "ROUND_RECTANGLE" "ELLIPSE"         "TRIANGLE"        "HEXAGON"
 # [9] "OCTAGON"

## End(Not run)
```

---

getNodeSize *getNodeSize*

Description

Get the size of the specified nodes on the CytoscapeWindow canvas.

Usage

```
getNodeSize(obj, node.names)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	a list of strings, the names of nodes to select.

Value

A named list containing two equal-lengthed vectors, width and height. Unless node dimensions are 'unlocked' these two vectors will be identical.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeSizeRule, setNodeSizeDirect, lockNodeDimensions

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('getNodeSize.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw)
sizes <- getNodeSize (cw, c ('A', 'B', 'C'))
```

```
print (sizes$width)
  # [1] 75 75 75
print (sizes$height)
  # [1] 35 35 35
setNodeSizeDirect (cw, 'A',  18)
redraw (cw)
print (getNodeSize (cw, 'A'))
# lockNodeDimensions (cw, FALSE) # not required anymore
setNodeWidthDirect (cw, 'A', 30)
setNodeHeightDirect (cw, 'A', 10)
redraw (cw) # not required anymore
sizes <- getNodeSize (cw, 'A')
print (sizes$width)
print (sizes$height)
# lockNodeDimensions (cw, TRUE) # not required anymore
setNodeSizeDirect (cw, 'A', 80)
redraw (cw) # not required anymore
print (getNodeSize (cw, 'A'))

## End(Not run)
```

`getSelectedEdgeCount` *getSelectedEdgeCount*

Description

Returns the number of edges currently selected.

Usage

```
getSelectedEdgeCount(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

An integer.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('getSelectedEdgeCount.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
```

```
redraw (cw)
clearSelection (cw)
getSelectedEdgeCount (cw)
  # [1] 0
  # in Cytoscape, interactively select an edge, or programmatically
selectEdges (cw, "A (phosphorylates) B")
getSelectedEdgeCount (cw)
  # [1] 1

## End(Not run)
```

---

getSelectedEdges *getSelectedEdges*

Description

Retrieve the identifiers of all the edges selected in the current graph.

Usage

```
getSelectedEdges(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('getSelectedEdges.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
redraw (cw)
  # in Cytoscape, interactively select an edge or call the selectEdges function
selectEdges (cw, "A (phosphorylates) B")
getSelectedEdges (cw)
  # [1] "A (phosphorylates) B"

## End(Not run)
```

getSelectedNodeCount *getSelectedNodeCount*

Description

Returns the number of node currently selected.

Usage

```
getSelectedNodeCount(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

An integer.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('getSelectedNodeCount.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  redraw (cw)
    # in Cytoscape, interactively select two nodes, or call selectNodes
  selectNodes (cw, c ('A','B'))
  getSelectedNodeCount (cw)
  # [1] 2

## End(Not run)
```

getSelectedNodes *getSelectedNodes*

Description

Retrieve the names of all the nodes selected in the current graph.

Usage

```
getSelectedNodes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('getSelectedNodes.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
redraw (cw)
  # in Cytoscape, interactively select two nodes, or call selectNodes
selectNodes (cw, c ('A','B'))
getSelectedNodes (cw)
# [1] "A" "B"

## End(Not run)
```

---

getViewCoordinates *getViewCoordinates*

Description

This method returns the four numbers (top.x, top.y, bottom.x, bottom.y) which implicitly specify the bounds of the current window.

Usage

```
getViewCoordinates(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A named list of four numbers, with these names: top.x, top.y, bottom.x, bottom.y

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getViewCoordinates getZoom setZoom

Examples

```
  ## Not run:
  # WARNING: Method RCy3::getViewCoordinates() is not implemented in RCy3!

  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.title = 'getViewCoordinates demo'
  cw <- CytoscapeWindow (window.title, graph=makeSimpleGraph())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork (cw, 'force-directed')
  #print (getViewCoordinates (cw))

## End(Not run)
```

`getVisualStyleNames` *getVisualStyleNames*

Description

Cytoscape provides a number of canned visual styles, to which you may add your own. Use this method to find out the names of those which are currently defined.

Usage

```
getVisualStyleNames(obj)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object or CytoscapeWindow object.

Value

a `list` of character strings.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

copyVisualStyle setVisualStyle

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  print (getVisualStyleNames (cy))

## End(Not run)
```

---

getWindowCount *getWindowCount*

Description

Returns the number of windows which currently exist in the Cytoscape Desktop.

Usage

```
getWindowCount(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

An integer.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  getWindowCount (cy)
    # [1] 0
  cw2 <- CytoscapeWindow ('getWindowCount.test1', graph=makeSimpleGraph())
  cw3 <- CytoscapeWindow ('getWindowCount.test2', graph=makeSimpleGraph())
  getWindowCount (cy)
    # [1] 2

## End(Not run)
```

getWindowID *getWindowID*

Description

Windows in Cytoscape have both a title and an identifier. The title is useful for human readers; the identifier is used by Cytoscape internals, and is sometimes useful to obtain.This method returns the identifier associated with the window title.

Usage

```
getWindowID(obj, window.title)
```

Arguments

obj	a CytoscapeConnectionClass object.
window.title	a string.

Value

The identifier (id) of a window, which is always a string – even if the identifer appears to be an integer.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getWindowList

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  cw <- CytoscapeWindow ('getWindowID.test', graph=makeSimpleGraph())
  displayGraph (cw)
  getWindowID (cy, 'getWindowID.test')

## End(Not run)
```

getWindowList *getWindowList*

Description

Returns a named list of windows in the current Cytoscape Desktop.

Usage

```
getWindowList(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

A named list, in which the values are the titles of the windows; the names of the list are integers.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  getWindowList (cy)
    # NULL
  cw2 <- CytoscapeWindow ('getWindowList.test1', graph=makeSimpleGraph())
  cw3 <- CytoscapeWindow ('getWindowList.test2', graph=makeSimpleGraph())
  getWindowList (cy)
    # [1] "getWindowList.test2" "getWindowList.test1"

## End(Not run)
```

getZoom *getZoom*

Description

This method returns the zoom level of the CytoscapeWindow. A value of 1.0 typically renders the graph with an ample margin. A call to fitContent produces a zoom level of about 1.5.

Usage

```
getZoom(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A names list, x and y.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setZoom getCenter setCenter getViewCoordinates fitContent

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.title = 'getZoom demo'
  cw <- CytoscapeWindow (window.title, graph=makeSimpleGraph())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw, 'force-directed')
  print (getZoom (cw))

## End(Not run)
```

---

hideAllPanels *hideAllPanels*

Description

All panels will be hidden, and no longer visible in the Cytoscape Desktop, even if floating, elsewhere on the computer screen.

Usage

```
hideAllPanels(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

floatPanel dockPanel hidePanel

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  hideAllPanels (cy)

## End(Not run)
```

`hideNodes` *hideNodes*

Description

Hide (but do not delete) nodes. Highly recommended: save the current layout before hiding, since 'unhideAll' will, in addition to restoring hidden nodes to view, will place then in unpredictable locations on the screen.

Usage

```
hideNodes(obj, node.names)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.names`	a `character` list object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

hideSelectedNodes unhideAll saveLayout restoreLayout

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('hideNodes.test', graph=makeSimpleGraph())
displayGraph(cw)
redraw (cw)
layoutNetwork(cw, 'force-directed')
saveLayout (cw, 'layout.tmp.RData')
hideNodes (cw, c ('A', 'B'))
unhideAll (cw)
restoreLayout (cw, 'layout.tmp.RData')

## End(Not run)
```

`hidePanel` *hidePanel*

Description

The specified panel will be hidden, and no longer visible in the Cytoscape Desktop of, if floating, elsewhere on the computer screen. The `panelName` parameter is very flexible: a match is defined as a case-independent match of the supplied panelName to any starting characters in the actual panelName. Thus, 'd' and 'DA' both identify 'Data Panel'. Possible options also include: 'WEST', 'EAST', 'SOUTH', 'SOUTH_WEST'. The 'SOUTH' panel is the Data Panel and the 'WEST' panel is the control panel.

Usage

```
hidePanel(obj, panelName)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.
`panelName`	a character string, providing a partial or complete case-independent match to the start of the name of an actual panel.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

floatPanel dockPanel hideAllPanels

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
hidePanel (cy, 'Control Panel')
 # or
hidePanel (cy, 'c')
  # or
hidePanel (cy, 'WEST')

## End(Not run)
```

`hideSelectedEdges` *hideSelectedEdges*

Description

Hide (but do not delete) the currently selected edges.

Usage

```
hideSelectedEdges(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

unhideAll

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('hideSelectedEdges.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'grid')

selectEdges (cw, 'B (synthetic lethal) C')
```

```
  hideSelectedEdges (cw)
  # unhideAll (cw) # makes the edges appear again
   # previously, Cytoscape required that you render these edges,
   # and redo the layout, so that they are visible again.
   # This is no longer required.

## End(Not run)
```

---

hideSelectedNodes *hideSelectedNodes*

Description

Hide (but do not delete) the currently selected nodes. We strongly recommened that you save the current layout before hiding any nodes: 'unhideAll' previously often placed restored nodes in unpredictable positions.

Usage

```
hideSelectedNodes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

unhideAll

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('hideSelectedNodes.test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layoutNetwork(cw, 'force-directed')
saveLayout (cw, 'layout.tmp.RData')
selectNodes (cw, c ('A', 'B'))
hideSelectedNodes (cw)
unhideAll (cw)
restoreLayout (cw, 'layout.tmp.RData')
```

```
## End(Not run)
```

`initEdgeAttribute` *initEdgeAttribute*

Description

Create the edge attribute slot that the Bioconductor graph class requires, including a default value, and then specifying what the base type (or 'class') is – 'char', 'integer', or 'numeric' – which is needed by RCy3. This method converts these standard R data type names, to the forms needed by Cytoscape.

Usage

```
initEdgeAttribute(graph, attribute.name, attribute.type, default.value)
```

Arguments

`graph`	a Bioconductor `graph` object.
`attribute.name`	a `string`, the name of the new edge attribute.
`attribute.type`	a `string`, either 'char', 'integer', or 'numeric'
`default.value`	something sensible, of the right type

Value

Returns the modified graph.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

initNodeAttribute makeSimpleGraph

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  g = new ('graphNEL', edgemode='directed')
  g = initEdgeAttribute (g, 'edgeType', 'char', 'associates with')

## End(Not run)
```

initNodeAttribute *initNodeAttribute*

Description

Create the node attribute slot that the Bioconductor graph class requires, including a default value, and then specifying what the base type (or 'class') is – 'char', 'integer', or 'numeric' – which is needed by RCy3. This method converts these standard R data type names, to the forms needed by Cytoscape.

Usage

```
initNodeAttribute(graph, attribute.name, attribute.type, default.value)
```

Arguments

graph	a Bioconductor graph object.
attribute.name	a string, the name of the new node attribute.
attribute.type	a string, either 'char', 'integer', or 'numeric'
default.value	something sensible, of the right type

Value

Returns the modified graph.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

initEdgeAttribute makeSimpleGraph

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  g = new ('graphNEL', edgemode='directed')
  g = initNodeAttribute (g, 'lfc', 'numeric', 1.0)

## End(Not run)
```

invertEdgeSelection *invertEdgeSelection*

Description

Invert edge selection, i.e. select all edges that were not selected and deselect all edges that were selected.

Usage

```
invertEdgeSelection(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

clearSelection invertNodeSelection

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('invertEdgeSelection demo', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  invertEdgeSelection (cw)
    # all edges should be selected, since none were before

## End(Not run)
```

invertNodeSelection *invertNodeSelection*

Description

Invert the node selection, i.e. select all nodes that were not selected and deselect all nodes that were selected.

Usage

```
invertNodeSelection(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

clearSelection invertEdgeSelection

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('invertNodeSelection demo', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
invertNodeSelection (cw)
   # all nodes should be selected, since none were before
invertNodeSelection (cw)
   # no node should be selected, since all were before
selectNodes (cw, 'A')
invertNodeSelection (cw)
  # only B and C should be selected

## End(Not run)
```

layoutNetwork *layoutNetwork*

Description

Layout the current graph according to the specified algorithm.

Usage

```
layoutNetwork(obj, layout.name='grid')
```

Arguments

obj	a CytoscapeWindowClass object.
layout.name	a string, one of the values returned by getLayoutNames, 'grid' by default.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getLayoutNameMapping getLayoutNames restoreLayout saveLayout

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('layout.test', graph=makeSimpleGraph())
  displayGraph (cw)
  possible.layout.names <- getLayoutNames(cw)
  # choose one of the layouts e.g.:
  layoutNetwork (cw, possible.layout.names[1])
  # or:
  layoutNetwork (cw, 'force-directed')

## End(Not run)
```

lockNodeDimensions *lockNodeDimensions*

Description

Lock the node dimensions. Required for setting the same height and width if the node size is set.

Usage

```
lockNodeDimensions(obj, new.state, visual.style.name='default')
```

Arguments

obj	a CytoscapeConnectionClass object.
new.state	a boolean object, TRUE or FALSE
visual.style.name	a string object, naming the visual style whose 'locked' you wish to change. Defaults to 'default'.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeSizeDirect setNodeWidthDirect setNodeHeightDirect setNodeSizeRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('lockNodeDimensions demo', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw)
  lockNodeDimensions (cw, FALSE) # not required anymore as this function
   # is called by the other functions.
  setNodeWidthDirect (cw, 'A', 100)
  setNodeHeightDirect (cw, 'A', 50)

## End(Not run)
```

makeRandomGraph *makeRandomGraph*

Description

Create a random undirected graphNEL, useful for testing.Two default edge attributes are added, for demonstration purposes.

Usage

```
makeRandomGraph(node.count=12, seed=123)
```

Arguments

node.count	the number of nodes you wish to see in the graph
seed	an integer which, when supplied, allows reproducibility

Value

Returns (by default) a 12-node, rather dense undirected graph, with some attributes on the nodes and edges.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  g = makeRandomGraph (node.count=12, seed=123)

  ## The function is currently defined as
  function (node.count = 12, seed = 123)
  {
      set.seed(seed)
      node.names = as.character(1:node.count)
      g = randomGraph(node.names, M <- 1:2, p = 0.6)
      attr(edgeDataDefaults(g, attr = "weight"), "class") = "DOUBLE"
      edgeDataDefaults(g, "pmid") = "9988778899"
      attr(edgeDataDefaults(g, attr = "pmid"), "class") = "STRING"
      return(g)
  }

## End(Not run)
```

makeSimpleGraph *makeSimpleGraph*

Description

A 3-node, 3-edge graph, with some biological trappings, useful for demonstrations.

Usage

```
makeSimpleGraph()
```

Value

Returns a 3-node, 3-edge graph, with some attributes on the nodes and edges.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

g = makeSimpleGraph ()

## The function is currently defined as
function ()
{
    g = new("graphNEL", edgemode = "directed")
    nodeDataDefaults(g, attr = "type") = "undefined"
    attr(nodeDataDefaults(g, attr = "type"), "class") = "STRING"
    nodeDataDefaults(g, attr = "lfc") = 1
    attr(nodeDataDefaults(g, attr = "lfc"), "class") = "DOUBLE"
    nodeDataDefaults(g, attr = "label") = "default node label"
    attr(nodeDataDefaults(g, attr = "label"), "class") = "STRING"
    nodeDataDefaults(g, attr = "count") = "0"
    attr(nodeDataDefaults(g, attr = "count"), "class") = "INTEGER"
    edgeDataDefaults(g, attr = "edgeType") = "undefined"
    attr(edgeDataDefaults(g, attr = "edgeType"), "class") = "STRING"
    edgeDataDefaults(g, attr = "score") = 0
    attr(edgeDataDefaults(g, attr = "score"), "class") = "DOUBLE"
    edgeDataDefaults(g, attr = "misc") = ""
    attr(edgeDataDefaults(g, attr = "misc"), "class") = "STRING"
    g = graph::addNode("A", g)
    g = graph::addNode("B", g)
    g = graph::addNode("C", g)
    nodeData(g, "A", "type") = "kinase"
    nodeData(g, "B", "type") = "transcription factor"
    nodeData(g, "C", "type") = "glycoprotein"
    nodeData(g, "A", "lfc") = "-3.0"
    nodeData(g, "B", "lfc") = "0.0"
    nodeData(g, "C", "lfc") = "3.0"
```

```
    nodeData(g, "A", "count") = "2"
    nodeData(g, "B", "count") = "30"
    nodeData(g, "C", "count") = "100"
    nodeData(g, "A", "label") = "Gene A"
    nodeData(g, "B", "label") = "Gene B"
    nodeData(g, "C", "label") = "Gene C"
    g = graph::addEdge("A", "B", g)
    g = graph::addEdge("B", "C", g)
    g = graph::addEdge("C", "A", g)
    edgeData(g, "A", "B", "edgeType") = "phosphorylates"
    edgeData(g, "B", "C", "edgeType") = "synthetic lethal"
    edgeData(g, "A", "B", "score") = 35
    edgeData(g, "B", "C", "score") = -12
    return(g)
  }

## End(Not run)
```

---

noa *noa*

Description

Retrieve the value of the specified node attribute for every node in the graph.

Usage

```
noa(graph, node.attribute.name)
```

Arguments

graph	typically, a bioc graphNEL)
node.attribute.name	a character string

Value

A list, the contents of which are the attribute values, the names of which are the names of the nodes.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

noa.names

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

g <- makeSimpleGraph ()
noa (g, 'type')
  #             A                     B                     C
  #      "kinase" "transcription factor"         "glycoprotein"

## End(Not run)
```

---

noa.names *noa.names*

Description

Retrieve the names of the node attributes in the specified graph. These are typically strings like 'type', 'label', 'count', and (strongly recommended when you create a graph) 'nodeType'. Once you are reminded of the names of the edge attributes, you can use the method 'noa' to get all the values of this attribute for the nodes in the graph.

Usage

```
noa.names(graph)
```

Arguments

graph	typically, a bioc graphNEL)

Value

A list, the contents of which are the attribute values, the names of which are the names of the edges.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

noa eda eda.names

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

g <- makeSimpleGraph()
noa.names (g)
  #  [1] "type"  "lfc"   "label" "count"

## End(Not run)
```

ping *ping*

Description

Test the connection to Cytoscape.

Usage

```
ping(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

"It works!"

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  ping (cy)
    # "It works!"

## End(Not run)
```

pluginVersion *pluginVersion*

Description

Get the currently used plugin version number of CyREST.

Usage

```
pluginVersion(obj)
```

Arguments

obj	a CytoscapeConnectionClass object.

Value

a string describing the current version of the CyREST plugin.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cy <- CytoscapeConnection ()
print (pluginVersion (cy))
  # e.g. "v1"

## End(Not run)
```

`predictTimeToDisplayGraph` *predictTimeToDisplayGraph*

Description

Use simple heuristics and previously collected timing to predict the length of time that will be required to send the R graph across the CyREST wire to Cytoscape.

Usage

```
predictTimeToDisplayGraph(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

Time in seconds.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

CytoscapeWindow

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('predictTimeToDisplayGraph.test', graph=makeSimpleGraph(),
                       collectTimings=TRUE)
message (paste ('estimated time: ', predictTimeToDisplayGraph (cw)))
displayGraph (cw)
layoutNetwork(cw, 'force-directed')

## End(Not run)
```

`raiseWindow` *raiseWindow*

Description

Raise this window to the top on the Cytoscape desktop, so that it can be seen.

Usage

```
raiseWindow(obj, window.title=NA)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object, or its subclass, `CytoscapeWindowClass`.
`window.title`	a `string`.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

resizeWindow

Examples

```
## Not run:
# WARNING: Method RCy3::raiseWindow() is not implemented in RCy3!

# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('raiseWindow.test', graph=makeSimpleGraph())
cw2 <- CytoscapeWindow ('raiseWindow.test2', graph=makeSimpleGraph())
```

```
  raiseWindow (cw)

## End(Not run)
```

---

redraw *redraw*

Description

Ask Cytoscape to redraw all nodes and edges, applying the vizmap rules.

Usage

```
redraw(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

displayGraph layoutNetwork

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('redraw.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  redraw (cw)  # applies default vizmap (rendering) rules, plus any you
               # have specified

## End(Not run)
```

restoreLayout *restoreLayout*

Description

Restore the current layout (that is, node positions) from the information saved in the supplied filename.

Usage

```
restoreLayout(obj, filename)
```

Arguments

obj	a CytoscapeWindowClass object.
filename	a string

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

saveLayout

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('restoreLayout.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  saveLayout (cw, 'layout.RData')
  layoutNetwork(cw, 'isom')
  restoreLayout (cw, 'layout.RData')
  # you might need to adjust the zoom

## End(Not run)
```

saveImage *saveImage*

Description

Write an image of the specified type to the specified file. For image type 'png' there is an option to set the height of the image (see argument h). Note: the file is written to the file system of the computer upon which R is running, not Cytoscape – in those cases where they are different. It is saved to the working directory.

Usage

```
saveImage(obj, file.name, image.type, h = 600)
```

Arguments

obj	a CytoscapeWindowClass object.
file.name	a char object. Use an explicit, full path, or this file will be written into your home directory.
image.type	a char object. 'png', 'pdf', 'svg' are the only image types currently supported
h	a numeric object. The height of the image. Width will be automatically set based on the height. Option only available for 'png' image type. All other image types use the default of 600 for the height and it is not adjustable.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

selectNodes clearSelection

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('saveImage.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  redraw (cw)
  filename <- paste (getwd (), 'saveImageTest', sep='/')
  saveImage (cw, filename, 'svg')   # currently supports svg, pdf and png
  saveImage(cw, filename, 'png', 1600)

## End(Not run)
```

saveLayout *saveLayout*

Description

Save the current layout (that is, node positions) to the specified file.

Usage

```
saveLayout(obj, filename, timestamp.in.filename=FALSE)
```

Arguments

obj	a CytoscapeWindowClass object.
filename	a string.
timestamp.in.filename	logical.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

restoreLayout

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('saveLayout.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
saveLayout (cw, 'layout.RData')
layoutNetwork(cw, 'fruchterman-rheingold')
restoreLayout (cw, 'layout.RData')
# you might need to adjust the zoom
saveLayout (cw, 'layout2', timestamp.in.filename=TRUE)

## End(Not run)
```

saveNetwork *saveNetwork*

Description

Write a network of the specified type to the specified file, at the specified scaling factor. Note:the file is written to the file system of the computer upon which R is running, not Cytoscape – in those cases where they are different. It is saved to the working directory.

Usage

```
saveNetwork(obj, file.name, format='cys')
```

Arguments

obj	a CytoscapeWindowClass object.
file.name	a char object.
format	a char object. 'cys' is the only type currently supported

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

saveImage

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('saveNetwork.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  redraw (cw)
  filename <- paste (getwd (), 'saveNetworkTest', sep='/')
  saveNetwork (cw, filename) # overwrites files with the same name
  # check if the file exists
  file.exists (paste0(filename, '.cys'))

## End(Not run)
```

selectEdges *selectEdges*

Description

Select the specified edges.

Usage

```
selectEdges(obj, edge.names, preserve.current.selection=TRUE)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	a list of strings, the names of edges to select.
preserve.current.selection	a logical object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

clearSelection getSelectedEdgeCount getSelectedEdges hideSelectedEdges

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('selectEdges.test', graph=makeSimpleGraph())
  displayGraph (cw); layoutNetwork(cw); redraw (cw)
  clearSelection (cw)
  selectEdges (cw, c ("A (phosphorylates) B", "B (synthetic lethal) C"))
  getSelectedEdges (cw)

## End(Not run)
```

selectFirstNeighborsOfSelectedNodes
selectFirstNeighborsOfSelectedNodes

Description

Expand the selection by adding the first neighbors, in the Cytosape network, of the nodes currently selected (again, in the Cytoscape network). The R graph is unchanged.

Usage

```
selectFirstNeighborsOfSelectedNodes (obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

clearSelection getSelectedNodeCount getSelectedNodes hideSelectedNodes getFirstNeighbors

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('selectFirstNeighborsOfSelecteNodes.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw)
  clearSelection (cw)
  selectNodes (cw, 'A')
  selectFirstNeighborsOfSelectedNodes (cw)
  print (sort (getSelectedNodes (cw)))
    # [1] "A" "B" "C"

## End(Not run)
```

selectNodes *selectNodes*

Description

Select the specified nodes.

Usage

```
selectNodes(obj, node.names, preserve.current.selection=TRUE)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	a list of strings, the names of nodes to select.
preserve.current.selection	a logical object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

clearSelection getSelectedNodeCount getSelectedNodes hideSelectedNodes

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('selectNodes.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
clearSelection (cw)
selectNodes (cw, c ('A', 'B'))
getSelectedNodes (cw)
  # [1] "A" "B"

## End(Not run)
```

sendEdges *sendEdges*

Description

Transfer the edges of the R graph (found in obj@graph) to Cytoscape. This method is not recommended for the average user. It is called behind the scenes by displayGraph.

Usage

```
sendEdges(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

displayGraph sendNodes

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('sendEdges.test', graph=makeSimpleGraph())
  sendNodes (cw)
  sendEdges (cw)

## End(Not run)
```

sendNodes *sendNodes*

Description

Transfer the nodes of the R graph (found in obj@graph) to Cytoscape. This method is not recommended for the average user. It is called behind the scenes by displayGraph.

Usage

```
sendNodes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

displayGraph sendEdges

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('sendNodes.test', graph=makeSimpleGraph())
  sendNodes (cw)

## End(Not run)
```

---

setCenter *setCenter*

Description

This method can be used to pan and scroll the Cytoscape canvas, which is adjusted (moved) so that the specified x and y coordinates are at the center of the visible window.

Usage

```
setCenter(obj, x, y)
```

Arguments

obj	a CytoscapeWindowClass object.
x	a numeric object.
y	a numeric object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getCenter getZoom setZoom

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

window.title = 'setCenter demo'
cw <- CytoscapeWindow (window.title, graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
original.center <- getCenter (cw)  # named list, "x" and "y".
  # now pan the display to the left, by setting the the visual center
  # to increasing values of x, without changing the location of the
  # simple graph
setCenter (cw, 200, 90)
system ('sleep 0.1')
setCenter (cw, 100, 90)
system ('sleep 0.1')
setCenter (cw, 0, 90)
system ('sleep 0.1')
setCenter (cw, -100, 90)
system ('sleep 0.1')
  # and now pan back to the original position
setCenter (cw, -100, 0)
system ('sleep 0.1')
setCenter (cw, original.center$x, original.center$y)

## End(Not run)
```

`setDefaultBackgroundColor` *setDefaultBackgroundColor*

Description

Set the default color for the next CytoscapeWindow.

Usage

```
setDefaultBackgroundColor(obj, new.color, vizmap.style.name)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.
`new.color`	a `character` object, in quoted hexadecimal format
`vizmap.style.name`	a `character` object, 'default' by default

Value

A character string, a hexadecimal, e.g. #000000

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setDefaultBackgroundColor.test', graph=makeSimpleGraph())
setDefaultBackgroundColor (cw, '#CCCC00')
cw2 <- CytoscapeWindow ('setDefaultBackgroundColor.test2', graph=makeSimpleGraph())

## End(Not run)
```

---

setDefaultEdgeColor *setDefaultEdgeColor*

Description

Set the default edge color.

Usage

```
setDefaultEdgeColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object if this vizmap style needs to be distinguished from the default type.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setDefaultEdgeFontSize setEdgeColorRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setDefaultEdgeColor test', graph=makeSimpleGraph())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw, 'grid')
  setDefaultEdgeColor (cw, '#888888') # grey edges
  redraw (cw) # redraw is not required anymore

## End(Not run)
```

`setDefaultEdgeFontSize` *setDefaultEdgeFontSize*

Description

Set the default edge font size.

Usage

```
setDefaultEdgeFontSize(obj, new.size, vizmap.style.name = "default")
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.
`new.size`	an `integer`.
`vizmap.style.name`	a visual style.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeFontSize setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('test setDefaultEdgeFontSize', graph=makeSimpleGraph())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork(cw, 'force-directed')
  setEdgeLabelRule (cw, 'edgeType')
  setDefaultEdgeFontSize (cw, 20)
  setDefaultEdgeFontSize (cw, 1)
  #redraw (cw) # not required anymore

## End(Not run)
```

`setDefaultEdgeLineWidth` *setDefaultEdgeLineWidth*

Description

In the specified CytoscapeConnection, stipulate the default line width, in pixels for all edges.

Usage

```
setDefaultEdgeLineWidth(obj, new.width, vizmap.style.name = "default")
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.
`new.width`	an `integer` object, typically from 0 to 5.
`vizmap.style.name`	a `String` object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setEdgeColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setDefaultEdgeLineWidth.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
setDefaultEdgeLineWidth (cw, 5)
#redraw (cw)

## End(Not run)
```

`setDefaultEdgeReverseSelectionColor` *setDefaultEdgeReverseSelectionColor*

Description

Set the default color used to display selected edges.

Usage

```
setDefaultEdgeReverseSelectionColor(obj, new.color, vizmap.style.name)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.
`new.color`	a `character` object, in quoted hexadecimal format
`vizmap.style.name`	a `character` object, 'default' by default

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw = CytoscapeWindow ("demo.setDefaultEdgeReverseSelectionColor", graph=makeSimpleGraph ())
displayGraph (cw)
layoutNetwork (cw)
setDefaultEdgeReverseSelectionColor (cw, '#0000FF')
# redraw (cw) --> not required anymore
```

```
## End(Not run)
```

---

setDefaultEdgeSelectionColor *setDefaultEdgeSelectionColor*

Description

Set the default color used to display selected edges.

Usage

```
setDefaultEdgeSelectionColor(obj, new.color, vizmap.style.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
new.color	a character object, in quoted hexadecimal format
vizmap.style.name	a character object, 'default' by default

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw = CytoscapeWindow ("demo.setDefaultEdgeSelectionColor", graph=makeSimpleGraph ())
  displayGraph (cw)
  layoutNetwork (cw)
  setDefaultEdgeSelectionColor (cw, '#00FF00')
  # redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultEdgeSourceArrowColor *setDefaultEdgeSourceArrowColor*

Description

In the specified CytoscapeConnection, stipulate the default color for all edge source arrows other than those mentioned in an edge source arrow color rule.

Usage

```
setDefaultEdgeSourceArrowColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object, if this vizmap style needs to be distinguished from the default type.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setDefaultEdgeFontSize setEdgeColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setDefaultEdgeSourceArrowColor test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
setDefaultEdgeSourceArrowColor (cw, '#F0F0F0')
setEdgeSourceArrowShapeDirect (cw, c('A (phosphorylates) B', 'B (synthetic lethal) C'), 'DIAMOND')
# redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultEdgeTargetArrowColor
setDefaultEdgeTargetArrowColor

Description

In the specified CytoscapeConnection, stipulate the default color for all edge target arrows other than those mentioned in an edge target arrow color rule.

Usage

```
setDefaultEdgeTargetArrowColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object, if this vizmap style needs to be distinguished from the default type.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setDefaultEdgeFontSize setEdgeColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setDefaultEdgeTargetArrowColor test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
setDefaultEdgeTargetArrowColor (cw, '#0F0F0F')
setEdgeTargetArrowShapeDirect (cw, c('C (undefined) A', 'B (synthetic lethal) C'), 'ARROW_SHORT')
# redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultNodeBorderColor *setDefaultNodeBorderColor*

Description

In the specified CytoscapeConnection, stipulate the default color for all node borders other than those mentioned in a node border color rule.

Usage

```
setDefaultNodeBorderColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setDefaultNodeBorderColor.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
setDefaultNodeBorderWidth (cw, 4)
setDefaultNodeBorderColor (cw, '#000000') # white borders
# redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultNodeBorderWidth
setDefaultNodeBorderWidth

Description

In the specified CytoscapeConnection, stipulate the default border width for all node borders other than those mentioned in a node border width rule.

Usage

```
setDefaultNodeBorderWidth(obj, new.width, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.width	an integer.
vizmap.style.name	a String object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setDefaultNodeBorderWidth.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
setDefaultNodeBorderWidth (cw, 5)
# redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultNodeColor *setDefaultNodeColor*

Description

In the specified CytoscapeWindow, stipulate the default color for all nodes other than those mentioned in a node border color rule.

Usage

```
setDefaultNodeColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setDefaultNodeColor.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
setDefaultNodeColor (cw, '#FF8888') # light red
# redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultNodeFontSize *setDefaultNodeFontSize*

Description

In the specified CytoscapeWindow, stipulate the default font size for all nodes other than those mentioned in a node font size rule.

Usage

```
setDefaultNodeFontSize(obj, new.size, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.size	an integer.
vizmap.style.name	a String object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setDefaultNodeFontSize.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  setDefaultNodeFontSize (cw, 32)
  # redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultNodeLabelColor
setDefaultNodeLabelColor

Description

In the specified CytoscapeWindow, stipulate the default color for all node labels. There is, at present, no mapping rule for this trait.

Usage

```
setDefaultNodeLabelColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setDefaultNodeLabelColor.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setDefaultNodeLabelColor (cw, '#FFFFFF') # white node labels
  # redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultNodeReverseSelectionColor
setDefaultNodeReverseSelectionColor

Description

Set the default color used to display selected nodes.

Usage

```
setDefaultNodeReverseSelectionColor(obj, new.color, vizmap.style.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
new.color	a character object, in quoted hexadecimal format.
vizmap.style.name	a character object, 'default' by default.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

 cw = CytoscapeWindow ("demo.setDefaultNodeReverseSelectionColor", graph=makeSimpleGraph ())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setDefaultNodeReverseSelectionColor (cw, '#AA33FF') # purple
  # redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultNodeSelectionColor
setDefaultNodeSelectionColor

Description

Retrieve the default color used to display selected nodes.

Usage

```
setDefaultNodeSelectionColor(obj, new.color, vizmap.style.name)
```

Arguments

obj	a CytoscapeConnectionClass object.
new.color	a character object, in quoted hexadecimal format
vizmap.style.name	a character object, 'default' by default

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ("demo.setDefaultNodeSelectionColor", graph=makeSimpleGraph ())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
setDefaultNodeSelectionColor (cw, '##FFFF00') # yellow
# redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultNodeShape *setDefaultNodeShape*

Description

For all CytoscapeWindows, specify the default node shape.

Usage

```
setDefaultNodeShape(obj, new.shape, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.shape	a String object, one of the permissible values (see getNodeShapes).
vizmap.style.name	a String object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getNodeShapes setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNode-Color setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefault-NodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setDefaultNodeShape.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'grid')
legal.shapes <- getNodeShapes (cw)
#stopifnot ('DIAMOND'
setDefaultNodeShape (cw, 'DIAMOND')
# redraw (cw) --> not required anymore

## End(Not run)
```

setDefaultNodeSize *setDefaultNodeSize*

Description

In the specified CytoscapeConnection, stipulate the default size for all nodes other than those mentioned in a node size rule.

Usage

```
setDefaultNodeSize(obj, new.size, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeConnectionClass object.
new.size	a integer object, typically 20 to 100.
vizmap.style.name	a String object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setDefaultNodeSize.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'grid')
setDefaultNodeSize (cw, 60) # an intermediate value
# redraw (cw) --> not required anymore

## End(Not run)
```

setEdgeAttributes *setEdgeAttributes*

Description

Transfer the named edge attribute from the R graph (found in obj@graph) to Cytoscape.This method is typically called by displayGraph, which will suffice for most users' needs. It transfers the specified edge attributes, for all edges, from the cw@graph slot to Cytoscape.

Usage

```
setEdgeAttributes(obj, attribute.name)
```

Arguments

obj	a CytoscapeWindowClass object.
attribute.name	a string, one of the attributes defined on the edges.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeAttributesDirect setNodeAttributes setNodeAttributesDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeAttributes.test', graph=makeSimpleGraph())
attribute.names <- eda.names (cw@graph)
sendNodes (cw)
sendEdges (cw)

for (attribute.name in attribute.names){
  setEdgeAttributes (cw, attribute.name)
}

## End(Not run)
```

setEdgeAttributesDirect *setEdgeAttributesDirect*

Description

Transfer the named edge attributes to Cytoscape. This method is required, for instance, if you wish to run a 'movie.' For example, if you have a time course experiment with different values at successive time points of the 'phosphorylates' or 'binds' relationship between two nodes. With an edgeColor rule already specified, you can animate the display of the edges in the graph by pumping new values of the edge attributes, and then asking for a redraw.

Usage

```
setEdgeAttributesDirect(obj, attribute.name, attribute.type, edge.names, values)
```

Arguments

obj	a CytoscapeWindowClass object.
attribute.name	a string one of the attributes defined on the edges.
attribute.type	a string from one of these three groups: (floating, numeric, double), (integer, int), (string, char, character). This parameter is required because RCy3 cannot always infer the type of an attribute.
edge.names	a list of strings, edge names
values	a list of objects of the type specified by 'attribute.name', one per edge

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeAttributes setNodeAttributes setNodeAttributesDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeAttributesDirect.test', graph=makeSimpleGraph())
sendNodes(cw)
sendEdges(cw)
edge.names = as.character (cy2.edge.names (cw@graph))
stopifnot (length (edge.names) == 3)
edge.values = c ('alligator', 'hedgehog', 'anteater')
result = setEdgeAttributesDirect (cw, 'misc', 'string', edge.names, edge.values)
```

```
## End(Not run)
```

---

setEdgeColorDirect *setEdgeColorDirect*

Description

In the specified CytoscapeWindow, set the color of the specified edge or edges.

Usage

```
setEdgeColorDirect(obj, edge.names, new.value)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	one ore more String objects, cy2-style edge names.
new.value	a String object, a color in hex notation.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeColorDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeColorDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
edge.names <- as.character (cy2.edge.names (cw@graph))[1:2]
setEdgeColorDirect (cw, edge.names, '#F833AA')
# redraw (cw) --> not required anymore

## End(Not run)
```

setEdgeColorRule *setEdgeColorRule*

Description

Specify how data attributes – for the specified named attribute – is mapped to edge color.

Usage

```
setEdgeColorRule(obj, edge.attribute.name, control.points, colors, mode, default.color='#FFFFFF')
```

Arguments

obj	a CytoscapeWindowClass object.
edge.attribute.name	the edge attribute whose values will, when thiS rule is applied, determine the color of each edge.
control.points	a list of values, either numeric (for interpolate mode) or character strings (for 'lookup' mode).
colors	a list of colors, expressed as hexadecimal RGB, like this: '#FF0000' or '#FA8800'
mode	either 'interpolate' or 'lookup'.
default.color	a String object, expressed in hexadecimal RGB in this format: "#RRGGBB"

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeShapeRule (detailed example) setEdgeLineStyleRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeColorRule.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

edgeType.values = c ('phosphorylates', 'synthetic lethal', 'undefined')
colors = c ('#FF0000', '#FFFF00', '#00FF00')
setEdgeColorRule (cw, 'edgeType', edgeType.values, colors, mode='lookup')

score.values = c (-15, 0, 40);
colors = c ('#00FF00', '#FFFFFF', '#FF0000')
setEdgeColorRule (cw, 'score',  score.values, colors, mode='interpolate')
```

```
  # now swap the colors around
colors = c ('#FF0000', '#FFFFFF', '#00FF00')
setEdgeColorRule (cw, 'score',  score.values, colors, mode='interpolate')

# redraw (cw) --> not required anymore

## End(Not run)
```

---

setEdgeFontSizeDirect *setEdgeFontSizeDirect*

Description

In the specified CytoscapeWindow, set the font size of the specified edge or edges.

Usage

```
setEdgeFontSizeDirect(obj, edge.names, new.value)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	one or more String objects, cy2-style edge names.
new.value	an integer objects, specifying the font size in pixels.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeFontSizeDirect setEdgeLabelColorDirect setEdgeLabelDirect setEdgeLabelOpacityDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeFontSizeDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
edge.names = as.character (cy2.edge.names (cw@graph)) [1:2]
setEdgeLabelDirect (cw, edge.names, '250')
for (i in 8:30) {
  setEdgeFontSizeDirect (cw, edge.names, i)
  }
setEdgeFontSizeDirect (cw, edge.names, 12)
```

```
## End(Not run)
```

`setEdgeLabelColorDirect` *setEdgeLabelColorDirect*

Description

In the specified CytoscapeWindow, set the color of the label of the specified edge or edges.

Usage

```
setEdgeLabelColorDirect(obj, edge.names, new.value)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.names`	one or more `String` objects, cy2-style edge names.
`new.value`	a `String` object, an RGB color in '#RRGGBB' form.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeLabelColorDirect setEdgeLabelDirect setEdgeFontSizeDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setEdgeLabelColorDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  edge.names <- as.character (cy2.edge.names (cw@graph))
  setEdgeLabelDirect (cw, edge.names, 'some label')
  setEdgeLabelColorDirect (cw, edge.names, '#00FF00')
  setEdgeLabelColorDirect (cw, edge.names [1:2], '#FF0000')

## End(Not run)
```

setEdgeLabelDirect *setEdgeLabelDirect*

Description

In the specified CytoscapeWindow, set the edgeLabel of the specified edge or edges.

Usage

```
setEdgeLabelDirect(obj, edge.names, new.value)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	one or more String objects, cy2-style edge names.
new.value	a String object, the new label.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeLabelDirect setEdgeLabelColorDirect setEdgeFontSizeDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setEdgeLabelDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  edge.names <- as.character (cy2.edge.names (cw@graph)) [1:2]
  for (i in 1:10) {
    setEdgeLabelDirect (cw, edge.names, 255 - (i * 25))
    redraw (cw)
    }

## End(Not run)
```

setEdgeLabelOpacityDirect
setEdgeLabelOpacityDirect

Description

In the specified CytoscapeWindow, set the opacity of the specified edge or edges. Low numbers, near zero, are transparent. High numbers, near 255, are maximally opaque: they are fully visible.

Usage

```
setEdgeLabelOpacityDirect(obj, edge.names, new.value)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	one or more String objects, cy2-style edge names.
new.value	a numeric object, ranging from 0 to 255.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeLabelOpacityDirect setEdgeLabelColorDirect setEdgeLabelDirect setEdgeFontSizeDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setEdgeLabelOpacityDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  edge.names = as.character (cy2.edge.names (cw@graph)) [1:2]
  setEdgeLabelDirect (cw, edge.names, 'some lable')
  # fade out
  for (i in 1:10) {
    setEdgeLabelOpacityDirect (cw, edge.names, 255 - (i * 25))
    }
  # fade in
  for (i in 1:10) {
    setEdgeLabelOpacityDirect (cw, edge.names, i * 25)
    }

## End(Not run)
```

setEdgeLabelRule *setEdgeLabelRule*

Description

Specify the edge attribute to be used as the label displayed on each edge. Non-character attributes are converted to strings before they are used.

Usage

```
setEdgeLabelRule(obj, edge.attribute.name)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.attribute.name	the edge attribute whose values will determine the edge label on each edge, when this rule is applied.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorRule (detailed example) setEdgeColorRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setEdgeLabelRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setEdgeLabelRule (cw, 'edgeType')

## End(Not run)
```

setEdgeLineStyleDirect *setEdgeLineStyleDirect*

Description

In the specified CytoscapeWindow, set the line style of the specified edge or edges, bypassing all rule mapping. The getLineStyles method shows the possible values.

Usage

```
setEdgeLineStyleDirect(obj, edge.names, new.values)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	one or more String objects, cy2-style edge names.
new.values	one or more String object, from the supported set of line styles (see getLineStyles).

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeLineStyleRule setEdgeColorDirect setEdgeFontSizeDirect setEdgeLabelColorDirect setEdgeLabelDirect setEdgeLabelOpacityDirect setEdgeLabelWidthDirect setEdgeLineStyleDirect setEdgeLineWidthDirect setEdgeOpacityDirect setEdgeSourceArrowColorDirect setEdgeSourceArrowDirect setEdgeSourceArrowOpacityDirect setEdgeSourceArrowShapeDirect setEdgeTargetArrowColorDirect setEdgeTargetArrowDirect setEdgeTargetArrowOpacityDirect setEdgeTargetArrowShapeDirect setEdgeTooltipDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeLineStyleDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
edges.of.interest <- as.character (cy2.edge.names (cw@graph))
supported.styles <- getLineStyles (cw)

  # pass three edges and three styles
setEdgeLineStyleDirect (cw, edges.of.interest, supported.styles [5:7])

  # pass three edges and one style
```

```
  setEdgeLineStyleDirect (cw, edges.of.interest, supported.styles [8])

    # now loop through all of the styles
  for (style in supported.styles) {
    setEdgeLineStyleDirect (cw, edges.of.interest, style)
  }

     # restore the default
  setEdgeLineStyleDirect (cw, edges.of.interest, 'SOLID')

## End(Not run)
```

`setEdgeLineStyleRule` *specify the line styles to be used in drawing edges*

Description

Specify how data attributes – for the specified named attribute – are mapped to edge line style.

Usage

```
setEdgeLineStyleRule(obj, edge.attribute.name, attribute.values, line.styles, default.style='SOLI
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.attribute.name`	the edge attribute whose values will determine the line style of each edge when this rule is applied.
`attribute.values`	A list of scalar, discrete values. For instance, interaction types: 'phosphorylates', 'ubiquinates', 'represses', 'activates'
`line.styles`	One line style for each of the attribute.values
`default.style`	The style to use when an explicit mapping is not provided.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

`getLineStyles` `setNodeBorderColorRule` (detailed example)

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeLineStyleRule.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
line.styles <- c('SINEWAVE', 'DOT', 'PARALLEL_LINES')
edgeType.values <- c('phosphorylates', 'synthetic lethal', 'undefined')
setEdgeLineStyleRule (cw, 'edgeType', edgeType.values, line.styles)

## End(Not run)
```

`setEdgeLineWidthDirect`

setEdgeLineWidthDirect

Description

In the specified CytoscapeWindow, set the line width of the specified edge or edges. Width is measured in pixels.

Usage

```
setEdgeLineWidthDirect(obj, edge.names, new.value)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.names`	one or more `String` objects, cy2-style edge names.
`new.value`	an `integer` object, typically in the range of 0 to 10.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeLineWidthDirect setEdgeLineStyleRule setEdgeColorDirect setEdgeFontSizeDirect setEdgeLabelColorDirect setEdgeLabelDirect setEdgeLabelOpacityDirect setEdgeLabelWidthDirect setEdgeLineStyleDirect setEdgeOpacityDirect setEdgeSourceArrowColorDirect setEdgeSourceArrowDirect setEdgeSourceArrowOpacityDirect setEdgeSourceArrowShapeDirect setEdgeTargetArrowColorDirect setEdgeTargetArrowDirect setEdgeTargetArrowOpacityDirect setEdgeTargetArrowShapeDirect setEdgeTooltipDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeLineWidthDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
edge.names <- as.character (cy2.edge.names (cw@graph)) [1:2]
for (i in 1:10) {
  setEdgeLineWidthDirect (cw, edge.names, i)
  }

setEdgeLineWidthDirect (cw, edge.names, c(1,3))

## End(Not run)
```

`setEdgeLineWidthRule` *setEdgeLineWidthRule*

Description

Specify the edge attribute which controls the thickness of the edges displayed in the graph. This is currently only a lookup mapping. An interpolated mapping will be added in the future.

Usage

```
setEdgeLineWidthRule(obj, edge.attribute.name, attribute.values, line.widths, default.width)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.attribute.name`	the edge attribute whose values will, when this rule is applied, determine the edgeLineWidth on each edge.
`attribute.values`	observed values of the specified attribute on the edges.
`line.widths`	the corresponding widths.
`default.width`	use this where the rule fails to apply

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorRule (detailed example) setEdgeColorRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setEdgeLineWidthRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  edge.attribute.values <- c('phosphorylates', 'synthetic lethal', 'undefined')
  line.widths <- c(0, 8, 16)
  setEdgeLineWidthRule (cw, 'edgeType', edge.attribute.values, line.widths)

## End(Not run)
```

---

setEdgeOpacityDirect *setEdgeOpacityDirect*

Description

In the specified CytoscapeWindow, set the opacity of the specified edge or edges at the same time. Low numbers, near zero, are transparent. High numbers, near 255, are maximally opaque: they are fully visible.

Usage

```
setEdgeOpacityDirect(obj, edge.names, new.values)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	one or more String objects, cy2-style edge names.
new.values	a numeric object, ranging from 0 to 255.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeOpacityDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeOpacityDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
edge.names = as.character (cy2.edge.names (cw@graph)) [1:2]
for (i in 1:10) {
  setEdgeOpacityDirect (cw, edge.names, 255 - (i * 25))
  }

## End(Not run)
```

`setEdgeOpacityRule` *setEdgeOpacityRule*

Description

Specify how data attributes – for the specified named attribute – is mapped to edge opacity.

Usage

```
setEdgeOpacityRule(obj, edge.attribute.name, control.points, opacities, mode)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.attribute.name`	the edge attribute whose values will, when this rule is applied, determine the opacity of each edge.
`control.points`	a list of values, either numeric (for interpolate mode) or character strings (for 'lookup' mode).
`opacities`	a list of opacity values, integers between 0 (invisible) and 255 (completely visible)
`mode`	either 'interpolate' or 'lookup'.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorRule (detailed example) setEdgeColorRule setNodeShapeRule setEdgeLineStyleRule setNodeColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeOpacityRule.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

edgeType.values <- c ("phosphorylates", "synthetic lethal", "undefined")

  # want to see edges and both arrows, to check success of opacity rule
setEdgeTargetArrowRule (cw, 'edgeType', edgeType.values, rep ('ARROW', 3))
setEdgeSourceArrowRule (cw, 'edgeType', edgeType.values, rep ('ARROW', 3))
setDefaultEdgeLineWidth (cw, 5)

  # do the lookup rule
opacities <- c (25, 100, 255)
setEdgeOpacityRule (cw, 'edgeType',  edgeType.values, opacities, mode='lookup')

  # now do the interpolated version
opacities <- c (10, 125, 255)
control.points <- c (-12, 0, 35)
setEdgeOpacityRule (cw, 'score',  control.points, opacities, mode='interpolate')

## End(Not run)
```

`setEdgeSourceArrowColorDirect` *setEdgeSourceArrowColorDirect*

Description

In the specified CytoscapeWindow, set the edgeSourceArrowColor of the specified edge or edges.

Usage

```
setEdgeSourceArrowColorDirect(obj, edge.names, new.colors)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.names`	one or more `String` objects, edges in standard Cytoscape form.
`new.colors`	one or more `String` objects, representing a color in a '#RRGGBB' hex format.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeTargetArrowColorDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeSourceArrowColorDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layoutNetwork (cw, 'force-directed')

arrows <- c ('Arrow', 'Diamond', 'Circle')
edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')
setEdgeSourceArrowRule (cw, 'edgeType', edgeType.values, arrows)
setEdgeTargetArrowRule (cw, 'edgeType', edgeType.values, arrows)

colors.1 <- c ("#FFFFFF", "#FFFFFF", "#FFFFFF")
colors.2 <- c ("#AA00AA", "#00AAAA", "#0000AA")

edge.names <- as.character (cy2.edge.names (cw@graph)) [1:3]

for (i in 1:2) {
  setEdgeSourceArrowColorDirect (cw, edge.names, colors.1)
  Sys.sleep (0.3)
  setEdgeSourceArrowColorDirect (cw, edge.names, colors.2)
  Sys.sleep (0.3)
  } # for i

## End(Not run)
```

---

setEdgeSourceArrowColorRule

Specify Rule for the Source Arrow Color

Description

Specify how edge attributes – that is, data values of the specified edge attribute – control the color of the source arrow, found at the end of an edge, where it connects to the source node.

Usage

```
setEdgeSourceArrowColorRule(obj, edge.attribute.name, control.points, colors, mode="interpolate",
```

Arguments

obj	a CytoscapeWindowClass object.
edge.attribute.name	the edge attribute whose values will determine the color of the source arrow for each edge when this ColorRule is applied.

control.points	A list of scalar, discrete values. For instance, interaction types: 'phosphorylates', 'ubiquinates', 'represses', 'activates'
colors	A color for each of the attribute.values
mode	either 'interpolate' or 'lookup'.
default.color	The color to use when an explicit mapping is not provided.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeTargetArrowColorRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  # send and display graph
  cw <- CytoscapeWindow ('setEdgeSourceArrowColorRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  redraw (cw)
  layoutNetwork (cw, 'force-directed')

  colors <- c ("#AA00AA", "#AAAA00", "#AA0000")
  edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')

  # add edge arrows
  arrows <- c ('Arrow', 'Diamond', 'Circle')
  edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')
  setEdgeSourceArrowRule (cw, 'edgeType', edgeType.values, arrows)

  # set rule
  setEdgeSourceArrowColorRule (cw, 'edgeType', edgeType.values, colors, mode='lookup')

  # if not specified, the mode is interpolate
  colors <- c ("#FFFFFF", "#00FF00", "#00AA00", "#FF0000", "#AA0000")
  control.points <- c( -12.0, 35.0, 0.0 )
  setEdgeSourceArrowColorRule (cw, 'score', control.points, colors)

## End(Not run)
```

setEdgeSourceArrowOpacityDirect
setEdgeSourceArrowOpacityDirect

Description

In the specified CytoscapeWindow, set the opacity of the source arrow of the specified edge or edges. Opacity is an integer between 0 (invisible) and 255 (fully rendered).

Usage

```
setEdgeSourceArrowOpacityDirect(obj, edge.names, new.values)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	one or more cy2-style edge names, String objects.
new.values	one or more integer objects, between 0 and 255.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeTargetArrowOpacityDirect setDefaultEdgeSourceArrowColor setEdgeSourceArrowColorDirect setEdgeSourceArrowColorRule setEdgeSourceArrowRule setEdgeSourceArrowShapeDirect setEdgeTargetArrowColorDirect setEdgeTargetArrowColorRule setEdgeTargetArrowRule setEdgeTargetArrowShapeDirect setDefaultEdgeTargetArrowColor

Examples

```
## Not run:
# WARNING: Method RCy3::setEdgeSourceArrowOpacityDirect() is not implemented in RCy3!

# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeSourceArrowOpacityDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

edges.of.interest = as.character (cy2.edge.names (cw@graph))
  # make sure the source arrows are visible
setEdgeSourceArrowShapeDirect (cw, edges.of.interest, 'Circle')

  # first try passing three edges and three arrow opacity values
setEdgeSourceArrowOpacityDirect (cw, edges.of.interest, c (64, 128, 255))
```

```
  # now try passing three edges and one opacity value
setEdgeSourceArrowOpacityDirect (cw, edges.of.interest, 32)

  # restore the default
setEdgeSourceArrowOpacityDirect (cw, edges.of.interest, 255)

## End(Not run)
```

`setEdgeSourceArrowRule` *specify the arrow types to be used at the end of an edge at the 'source' node*

Description

Specify how data attributes – for the specified named attribute – are mapped to the source arrow type.

Usage

```
setEdgeSourceArrowRule(obj, edge.attribute.name, attribute.values, arrows, default='ARROW')
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.attribute.name`	the edge attribute whose values will determine the source arrow of each edge when this rule is applied.
`attribute.values`	A list of scalar, discrete values. For instance, interaction types:'phosphorylates', 'ubiquinates', 'represses', 'activates'
`arrows`	One arrow type for each of the attribute.values
`default`	The arrow type to use when an explicit mapping is not provided.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

`getArrowShapes` setNodeBorderColorRule (detailed example) setEdgeColorRule setNodeShapeRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setEdgeSourceArrowRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  arrows <- c('Arrow', 'Diamond', 'Circle')
  edgeType.values <- c('phosphorylates', 'synthetic lethal', 'undefined')
  setEdgeSourceArrowRule (cw, 'edgeType', edgeType.values, arrows)

## End(Not run)
```

`setEdgeSourceArrowShapeDirect` *setEdgeSourceArrowShapeDirect*

Description

In the specified CytoscapeWindow, set the source arrow shape of the specified edge or edges, using one of the supported shapes.

Usage

```
setEdgeSourceArrowShapeDirect(obj, edge.names, new.values)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.names`	one or more cy2-style edge names, `String` objects.
`new.values`	one or more `String` objects, from the supported set.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getArrowShapes setEdgeTargetArrowRule setDefaultEdgeSourceArrowColor setEdgeSourceArrowColorDirect setEdgeSourceArrowColorRule setEdgeSourceArrowRule setEdgeSourceArrowShapeDirect setEdgeTargetArrowColorDirect setEdgeTargetArrowColorRule setEdgeTargetArrowRule setEdgeTargetArrowShapeDirect setDefaultEdgeTargetArrowColor

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeSourceArrowShapeDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

edges.of.interest <- as.character (cy2.edge.names (cw@graph))
supported.arrow.shapes <- getArrowShapes (cw)

  # first try passing three edges and three arrow shapes
setEdgeSourceArrowShapeDirect (cw, edges.of.interest, supported.arrow.shapes [2:4])

  # now try passing three edges and one arrow.shapes
setEdgeSourceArrowShapeDirect (cw, edges.of.interest, supported.arrow.shapes [6])

  # restore the default
setEdgeSourceArrowShapeDirect (cw, edges.of.interest, 'NONE')

## End(Not run)
```

---

setEdgeTargetArrowColorDirect

setEdgeTargetArrowColorDirect

Description

In the specified CytoscapeWindow, set the edge target arrow color of the specified edge or edges.

Usage

```
setEdgeTargetArrowColorDirect(obj, edge.names, new.colors)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	one or more String objects, edges in standard Cytoscape form.
new.colors	one or more String object, representing a color in a '#RRGGBB' hex format.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeEdgeTargetArrowColorDirect setEdgeTargetArrowRule setDefaultEdgeSourceArrowColor setEdgeSourceArrowColorDirect setEdgeSourceArrowColorRule setEdgeSourceArrowRule setEdgeSourceArrowShapeDirect setEdgeTargetArrowColorRule setEdgeTargetArrowRule setEdgeTargetArrowShapeDirect setDefaultEdgeTargetArrowColor

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeTargetArrowColorDirect.test', graph=makeSimpleGraph ())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

arrows <- c ('Arrow', 'Diamond', 'Circle')
edgeType.values <- c('phosphorylates', 'synthetic lethal', 'undefined')
setEdgeTargetArrowRule (cw, 'edgeType', edgeType.values, arrows)
setEdgeTargetArrowRule (cw, 'edgeType', edgeType.values, arrows)

colors.1 <- c ("#FFFFFF", "#FFFFFF", "#FFFFFF")
colors.2 <- c ("#AA00AA", "#00AAAA", "#0000AA")

edge.names <- as.character (cy2.edge.names (cw@graph)) [1:3]

for (i in 1:2) {
  setEdgeTargetArrowColorDirect (cw, edge.names, colors.1)
  Sys.sleep (0.3)
  setEdgeTargetArrowColorDirect (cw, edge.names, colors.2)
  Sys.sleep (0.3)
  } # for i

## End(Not run)
```

---

setEdgeTargetArrowColorRule

Specify rule for the target arrow color

Description

Specify how edge attributes – that is, data values of the specified edge attribute – control the color of the target arrow, found at the end of an edge, where it connects to the target node.

Usage

```
setEdgeTargetArrowColorRule(obj, edge.attribute.name, control.points, colors, mode="interpolate",
```

Arguments

obj	a CytoscapeWindowClass object.
edge.attribute.name	the edge attribute whose values will determine the color of the target arrow of each edge when this rule is applied.
control.points	A list of scalar, discrete values. For instance, interaction types: 'phosphorylates', 'ubiquinates', 'represses', 'activates'
colors	A color for each of the attribute.values
mode	either 'interpolate' or 'lookup'.
default.color	The color to use when an explicit mapping is not provided.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorRule (detailed example) setEdgeSourceArrowColorRule setEdgeColorRule setNodeShapeRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

# send and display network
cw <- CytoscapeWindow ('setEdgeTargetArrowColorRule.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

# add edge arrows
arrows <- c ('CIRCLE', 'ARROW', 'DIAMOND')
edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')
setEdgeTargetArrowRule (cw, 'edgeType', edgeType.values, arrows)

colors <- c ("#AA00AA", "#AAAA00", "#AA0000")
edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')

# set rule
setEdgeTargetArrowColorRule (cw, 'edgeType', edgeType.values, colors, mode='lookup')

# if not specified, the mode is interpolate
colors <- c ("#FFFFFF", "#00FF00", "#00AA00", "#FF0000", "#AA0000")
control.points <- c( -12.0, 35.0, 0.0 )
setEdgeTargetArrowColorRule(cw, 'score', control.points, colors)

## End(Not run)
```

setEdgeTargetArrowOpacityDirect
setEdgeTargetArrowOpacityDirect

Description

In the specified CytoscapeWindow, set the opacity of the target arrow of the specified edge or edges. Opacity is an integer between 0 (invisible) and 255 (fully rendered).

Usage

```
setEdgeTargetArrowOpacityDirect(obj, edge.names, new.values)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.names	one or more cy2-style edge names, String objects.
new.values	one or more integer objects, between 0 and 255.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeTargetArrowOpacityDirect

Examples

```
## Not run:
# WARNING: Method RCy3::setEdgeTargetArrowOpacityDirect() is not implemented in RCy3!

# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeTargetArrowOpacityDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

edges.of.interest <- as.character (cy2.edge.names (cw@graph))

  # make sure the target arrows are visible
setEdgeTargetArrowShapeDirect (cw, edges.of.interest, 'Circle')

  # first try passing three edges and three arrow opacity values
setEdgeTargetArrowOpacityDirect (cw, edges.of.interest, c (64, 128, 255))

  # now try passing three edges and one opacity value
setEdgeTargetArrowOpacityDirect (cw, edges.of.interest, 32)
```

```
    # restore the default
  setEdgeTargetArrowOpacityDirect (cw, edges.of.interest, 255)

## End(Not run)
```

`setEdgeTargetArrowRule` *specify the arrow types to be used at the end of an edge, at the 'target' node*

Description

Specify how data attributes – for the specified named attribute – are mapped to the target arrow type.

Usage

```
setEdgeTargetArrowRule(obj, edge.attribute.name, attribute.values, arrows, default='ARROW')
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.attribute.name`	the edge attribute whose values will determine the target arrow of each edge when this rule is applied.
`attribute.values`	A list of scalar, discrete values. For instance, interaction types:'phosphorylates', 'ubiquinates', 'represses', 'activates'
`arrows`	One arrow type for each of the attribute.values
`default`	The arrow type to use when an explicit mapping is not provided.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

`getArrowShapes` setNodeBorderColorRule (detailed example) setEdgeColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeTargetArrowRule.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

arrows <- c('DIAMOND', 'ARROW', 'CIRCLE')
edgeType.values <- c('phosphorylates', 'synthetic lethal', 'undefined')

setEdgeTargetArrowRule (cw, 'edgeType', edgeType.values, arrows)

## End(Not run)
```

`setEdgeTargetArrowShapeDirect` *setEdgeTargetArrowShapeDirect*

Description

In the specified CytoscapeWindow, set the target arrow shape of the specified edge or edges, using one of the supported shapes.

Usage

```
setEdgeTargetArrowShapeDirect(obj, edge.names, new.values)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.names`	one or more cy2-style edge names, `String` objects.
`new.values`	one or more `String` objects, from the supported set.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getArrowShapes setDefaultEdgeSourceArrowColor setEdgeSourceArrowColorDirect setEdgeSourceArrowColorRule setEdgeSourceArrowRule setEdgeSourceArrowShapeDirect setEdgeTargetArrowColorDirect setEdgeTargetArrowColorRule setEdgeTargetArrowRule setEdgeTargetArrowShapeDirect setDefaultEdgeTargetArrowColor

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeTargetArrowShapeDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

edges.of.interest <- as.character (cy2.edge.names (cw@graph))
supported.arrow.shapes <- getArrowShapes (cw)

  # first try passing three edges and three arrow shapes
setEdgeTargetArrowShapeDirect (cw, edges.of.interest, supported.arrow.shapes [2:4])

  # now try passing three edges and one arrow.shapes
setEdgeTargetArrowShapeDirect (cw, edges.of.interest, supported.arrow.shapes [6])

  # restore the default
setEdgeTargetArrowShapeDirect (cw, edges.of.interest, 'NONE')

## End(Not run)
```

`setEdgeTooltipDirect` *setEdgeTooltipDirect*

Description

In the specified CytoscapeWindow, set the tooltips of the specified edge or edges.

Usage

```
setEdgeTooltipDirect(obj, edge.names, new.values)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.names`	one or more cy2-style edge names, `String` objects.
`new.values`	one or more `String` objects.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeTooltipRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setEdgeTooltipDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')

edges.of.interest <- as.character (cy2.edge.names (cw@graph))

  # first try passing three edges and three tooltips
setEdgeTooltipDirect (cw, edges.of.interest, c ('tooltip #1', 'tooltip #2', 'tooltip #3'))

  # now try passing three edges and one tooltip
setEdgeTooltipDirect (cw, edges.of.interest, 'a general purpose tooltip')

setEdgeTooltipDirect (cw, edges.of.interest, '')

## End(Not run)
```

---

setEdgeTooltipRule *setEdgeTooltipRule*

Description

Specify the edge attribute to be used as the tooltip for each edge. Non-character attributes are converted to strings before they are used as tooltips.

Usage

```
setEdgeTooltipRule(obj, edge.attribute.name)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.attribute.name	the edge attribute whose values will determine the tooltip on each edge when this rule is applied.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setEdgeTooltipDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setEdgeTooltipRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setEdgeTooltipRule (cw, 'edgeType')

## End(Not run)
```

---

setGraph *setGraph*

Description

Assigns the supplied graph object to the appropriate slot in the specified CytoscapeWindow object.

Usage

```
setGraph(obj, graph)
```

Arguments

obj	a CytoscapeWindowClass object.
graph	a graph object.

Value

The modified CytoscapeWindow object.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setGraph.test')  # an empty graph is created by default
  graph <- makeSimpleGraph ()
  setGraph (cw, graph)
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  print (length (nodes (getGraph (cw))))

## End(Not run)
```

setLayoutProperties *setLayoutProperties*

Description

Sets the specified properties for the specified layout. Unmentioned properties are left unchanged.

Usage

```
setLayoutProperties(obj, layout.name, properties.list)
```

Arguments

obj	a CytoscapeConnectionClass object.
layout.name	a string object.
properties.list	a a named list with as many entries as you wish to modify

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

layout getLayoutNames getLayoutNameMapping getLayoutPropertyNames getLayoutPropertyType getLayoutPropertyValue

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  prop.names <- getLayoutPropertyNames (cy, 'isom')
  print (prop.names)
    # [1] "maxEpoch"           "radiusConstantTime" "radius"             "minRadius"
    # [5] "initialAdaptation"  "minAdaptation"      "sizeFactor"         "coolingFactor"
    # [9] "singlePartition"
  print (getLayoutPropertyValue (cy, 'isom', 'radiusConstantTime'))
    # [1] 4

    # modify just two of the eight properties; the others are unchanged
  setLayoutProperties (cy, 'isom', list (radiusConstantTime=4, radius=20))

## End(Not run)
```

setNodeAttributes *setNodeAttributes*

Description

Transfer the named node attribute from the R graph (found in obj@graph) to Cytoscape. This method is typically called by displayGraph, which will suffice for most users' needs. It transfers the specified node attributes, for all nodes, from the cw@graph slot to Cytoscape.

Usage

```
setNodeAttributes(obj, attribute.name)
```

Arguments

obj	a CytoscapeWindowClass object.
attribute.name	a string one of the attributes defined on the nodes.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeAttributesDirect setEdgeAttributes setEdgeAttributesDirect sendEdges sendNodes displayGraph

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setNodeAttributes.test', graph=makeSimpleGraph())
attribute.names <- noa.names (cw@graph)
sendNodes (cw)
for (attribute.name in attribute.names){
  setNodeAttributes (cw, attribute.name)
}

## End(Not run)
```

setNodeAttributesDirect

setNodeAttributesDirect

Description

Transfer the named node attribute, for all named nodes, to Cytoscape. The attribute must be previously defined on the nodes of the graph: see nodeDataDefaults in the graph class.This method is useful if you wish to run a 'movie'. For example, if you have a timecourse experiment, with different values at successive time points of the 'lfc' (log fold change) measurements or 'pValue' of each node. With a node color and node size rule already specified, you can animate the display of the nodes across time in the graph by pumping new values of the attributes using this method.

Usage

```
setNodeAttributesDirect(obj, attribute.name, attribute.type, node.names, values)
```

Arguments

obj	a CytoscapeWindowClass object.
attribute.name	a string, one of the attributes defined on the nodes.
attribute.type	a string from one of these three groups: (floating, numeric, double), (integer, int), (string, char, character). This parameter is required because RCy3 cannot always infer the type of an attribute.
node.names	a list of strings, node names
values	a list of objects of the type specified by 'attribute.name', one per node

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeAttributes setEdgeAttributes setEdgeAttributesDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeAttributesDirect.test', graph=makeSimpleGraph())
  sendNodes(cw)
  setNodeAttributesDirect (cw, 'count', 'int', c ('A', 'B', 'C'), c (4, 8, 12))

## End(Not run)
```

setNodeBorderColorDirect
setNodeBorderColorDirect

Description

In the specified CytoscapeWindow, set the color of the border of the specified node.

Usage

```
setNodeBorderColorDirect(obj, node.names, new.colors)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.colors	a String object, in standard hex notation.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeSizeDirect setNodeBorderOpacityDirect setNodeBorderWidthDirect setNodeColorDirect setNodeFillOpacityDirect setNodeFontSizeDirect setNodeHeightDirect setNodeImageDirect setNodeLabelColorDirect setNodeLabelDirect setNodeLabelOpacityDirect setNodeOpacityDirect setNodeShapeDirect setNodeWidthDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setNodeBorderColorDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layoutNetwork (cw, 'force-directed')
setDefaultNodeBorderWidth(cw, 4)
setNodeBorderColorDirect (cw, 'A', '#FFFF00')
setNodeBorderColorDirect (cw, c('A', 'C'), c('#88FF00', '#880000'))
setNodeBorderColorDirect (cw, c('A', 'B'), '#FFFF00')

## End(Not run)
```

setNodeBorderColorRule

setNodeBorderColorRule

Description

Specify how data attributes – for the specified named attribute – are mapped to node color. There are two modes:'interpolate' and 'lookup'. In the former, you specify data values ('control points') and colors; when a node's corresponding data attribute value is exactly that of a control point, the specified color is used. If the node's data attribute falls between control points, then the color is interpolated. Note! In the 'interpolate' mode, you almost always want to provide two additional colors: one for node data values falling below the minimum control point, one for node data values falling above the maximum control point. If you provide an equal number of colors and control.points, the default.color is used to paint nodes above and below the specified range. A useful data exploration strategy would be to use `default.color <- '#000000'` causing all extreme nodes to be painted black. The 'lookup' mode provides no interpoloation, and is useful when you have a node attribute with a finite set of discrete values, each of which you you want to display in a specific color. For example: render all receptors in yellow, all transcription factors in blue, and all kinases in dark red.

Usage

```
setNodeBorderColorRule(obj, node.attribute.name, control.points, colors, mode, default.color='#00
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will determine the color of each node when this rule is applied.
`control.points`	a list of values. In the interpolate mode, a typical choice is the minimum, the maximum, some sensible midpoint.
`colors`	a list of colors, either two more than the number of control points (if mode='interpolate'), in which case the first color is used for all attributes values below the minimum, and the last color is used for those above the maximum. Or, if mode='lookup', the same number of colors as control.points are expected. Colors are expressed as quoted hexadecimal RGB strings, e.g. '#FF0000' or '#FA8800'
`mode`	'interpolate' or 'lookup'. This roughly corresponds to the visual mapping of continuously varying data (i.e., lfc or pValue), versus visual mapping of discrete data (i.e., molecule type, or phosphorylation status). With the interpolation mode, you must specify n+2 colors: adding a 'below' and an 'above' color. In lookup mode, specify exactly as many control.points as colors. If data attribute values are found on the nodes which do not appear inyour list, they will displayed in the default color.
`default.color`	'#000000' (black) by default, to catch your eye. Used primarily in mode=='lookup' and in mode='interpolate' if you fail to specify 'above' and 'below' values.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeShapeRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setNodeBorderColorRule.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
control.points <- c (-3.0, 0.0, 3.0)   # typical range of log-fold-change ratio values
  # paint negative values shades of green, positive values shades of
  # red, out-of-range low values are dark green; out-of-range high
  # values are dark red
colors <- c ("#00AA00", "#00FF00", "#FFFFFF", "#FF0000", "#AA0000")
setDefaultNodeBorderWidth (cw, 5)
setNodeBorderColorRule (cw, node.attribute.name='lfc', control.points, colors, mode='interpolate')
data.values <- c ("kinase", "transcription factor", "glycoprotein")
colors <- c ("#0000AA", "#FFFF00", "#00AAAA")
setNodeBorderColorRule (cw, node.attribute.name='type', data.values, colors, mode='lookup', default.color=

## End(Not run)
```

---

setNodeBorderOpacityDirect
setNodeBorderOpacityDirect

Description

In the specified CytoscapeWindow, set the opacity of the border of the specified node(s).

Usage

```
setNodeBorderOpacityDirect(obj, node.names, new.values)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.values	a numeric object, ranging from 0 to 255.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorRule (detailed example) setNodeBorderColorDirect setNodeBorderWidthDirect setNodeColorDirect setNodeFillOpacityDirect setNodeFontSizeDirect setNodeHeightDirect setNodeImageDirect setNodeLabelColorDirect setNodeLabelDirect setNodeLabelOpacityDirect setNodeOpacityDirect setNodeShapeDirect setNodeSizeDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeBorderOpacityDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  setDefaultNodeBorderWidth (cw, 5)
  setNodeBorderOpacityDirect (cw, 'A', 220)
  setNodeBorderOpacityDirect (cw, c('B', 'C'), c(22, 55))

## End(Not run)
```

`setNodeBorderWidthDirect`

setNodeBorderWidthDirect

Description

In the specified CytoscapeWindow, set the width of the border of the specified node(s).

Usage

```
setNodeBorderWidthDirect(obj, node.names, new.sizes)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.names`	one or more `String` objects, the node names.
`new.sizes`	an `integer`, in pixel units.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorDirect setNodeBorderOpacityDirect setNodeColorDirect setNodeFillOpacity-Direct setNodeFontSizeDirect setNodeHeightDirect setNodeImageDirect setNodeLabelColorDirect setNodeLabelDirect setNodeLabelOpacityDirect setNodeOpacityDirect setNodeShapeDirect setN-odeSizeDirect setNodeWidthDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeBorderWidthDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setNodeBorderWidthDirect (cw, 'A', 10)

## End(Not run)
```

`setNodeBorderWidthRule` *setNodeBorderWidthRule*

Description

Specify the node attribute which controls the thickness of the node borders displayed in the graph. This is currently only a lookup mapping. An interpolated mapping will be added in the future.

Usage

```
setNodeBorderWidthRule(obj, node.attribute.name, attribute.values, line.widths, default.width)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will determine the width of the node border on each node when this rule is applied.
`attribute.values`	observed values of the specified attribute on the nodes.
`line.widths`	the corresponding widths.
`default.width`	use this where the rule fails to apply

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorRule (detailed example)

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeBorderWidthRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  node.attribute.values <- c('kinase', 'transcription factor', 'glycoprotein')
  line.widths <- c(0, 8, 16)
  setNodeBorderWidthRule (cw, 'type', node.attribute.values, line.widths)

## End(Not run)
```

---

setNodeColorDirect *setNodeColorDirect*

Description

In the specified CytoscapeWindow, set the color of the specified node or nodes. This method bypasses the vizmap and excludes this node, for the duration of the current Cytoscape session, from further manipulation by vizmap color rules.

Usage

```
setNodeColorDirect(obj, node.names, new.colors)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	a String list object.
new.colors	an String object, using the standard hexadecimal form, eg, '#FF88AA'

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeColorRule setNodeBorderColorDirect setNodeBorderOpacityDirect setNodeBorderWidthDirect setNodeColorDirect setNodeFillOpacityDirect setNodeFontSizeDirect setNodeHeightDirect setNodeImageDirect setNodeLabelColorDirect setNodeLabelDirect setNodeLabelOpacityDirect setNodeOpacityDirect setNodeShapeDirect setNodeSizeDirect setNodeWidthDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setNodeColorDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
setNodeColorDirect (cw, 'A', '#880000')
setNodeColorDirect (cw, c('A', 'C'), c('#88FF00', '#880000'))

## End(Not run)
```

`setNodeColorRule` *setNodeColorRule*

Description

Specify how data attributes – for the specified named attribute – are mapped to node color. There are two modes: 'interpolate' and 'lookup'. In the former, you specify data values ('control points') and colors; when a node's corresponding data attribute value is exactly that of a control point, the specified color is used. If the node's data attribute falls between control points, then the color is interpolated. Note! In the 'interpolate' mode, you almost always want to provide two additional colors: one for node data values falling below the minimum control point, one for node data values falling above the maximum control point. If you provide an equal number of colors and control.points, the default.color is used to paint nodes above and below the specified range. A useful data exploration strategy would be to use `default.color <- '#000000'` causing all extreme nodes to be painted black. The 'lookup' mode provides no interpolation, and is useful when you have a node attribute with a finite set of discrete values, each of which you you want to display in a specific color.For example:render all receptors in yellow, all transcription factors in blue, and all kinases in dark red.

Usage

```
setNodeColorRule(obj, node.attribute.name, control.points, colors, mode, default.color='#FFFFFF')
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will determine the color of each node when this rule is applied.
`control.points`	a list of values. In the interpolate mode, a typical choice is the minimum, the maximum and some sensible midpoint.
`colors`	a list of colors, either two more than the number of control points (if mode='interpolate'), in which case the first color is used for all attributes values below the minimum, and the last color is used for those above the maximum. Or, if mode='lookup', the same number of colors as control.points are expected. Colors are expressed as quoted hexadecimal RGB strings, e.g., '#FF0000' or '#FA8800'

mode	'interpolate' or 'lookup'.This roughly corresponds to the visual mapping of continuously varying data (i.e., lfc or pValue), versus visual mapping of discrete data (i.e., molecule type, or phosphorylation status). With the interpolation mode, you must specify n+2 colors: adding a 'below' and an 'above' color. In lookup mode, specify exactly as many control.points as colors. If are data attribute values are found on the nodes which do not appear in your list, they will be displayed in the default color.
default.color	'#000000' (black) by default, to catch your eye. Used primarily in mode=='lookup' and in mode='interpolate' if you fail to specify 'above' and 'below' values.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorRule (detailed example) setNodeShapeRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeColorRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  control.points <- c(-3.0, 0.0, 3.0)   # typical range of log-fold-change ratio values
    # paint negative values shades of green, positive values shades of
    # red, out-of-range low values are dark green; out-of-range high
    # values are dark red
  node.colors <- c("#00AA00", "#00FF00", "#FFFFFF", "#FF0000", "#AA0000")
  setNodeColorRule (cw, node.attribute.name='lfc', control.points, node.colors, mode='interpolate')

    # lookup mode
  data.values <- c("kinase", "transcription factor", "glycoprotein")
  node.colors <- c("#0000AA", "#FFFF00", "#0000AA")
  setNodeColorRule (cw, node.attribute.name='type', data.values, node.colors, mode='lookup', default.color='

## End(Not run)
```

---

setNodeFillOpacityDirect

setNodeFillOpacityDirect

Description

In the specified CytoscapeWindow, set the opacity of the fill color of the specified node(s).

Usage

```
setNodeFillOpacityDirect(obj, node.names, new.values)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.values	a numeric object, ranging from 0 to 255.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorDirect setNodeBorderOpacityDirect setNodeBorderWidthDirect setNodeColorDirect setNodeFontSizeDirect setNodeHeightDirect setNodeImageDirect setNodeLabelColorDirect setNodeLabelDirect setNodeLabelOpacityDirect setNodeOpacityDirect setNodeShapeDirect setNodeSizeDirect setNodeWidthDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeFillOpacityDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setNodeFillOpacityDirect (cw, 'A', 20)
  setNodeFillOpacityDirect (cw, c('B', 'C'), c(122, 133))
  setNodeFillOpacityDirect (cw, c('B', 'C'), 1)

## End(Not run)
```

---

setNodeFontSizeDirect *setNodeFontSizeDirect*

Description

In the specified CytoscapeWindow, set the size of the font used in rendering the label of the specified node(s).

Usage

```
setNodeFontSizeDirect(obj, node.names, new.sizes)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.sizes	one or more integers, in pixel units.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorDirect setNodeBorderOpacityDirect setNodeBorderWidthDirect setNodeColorDirect setNodeFillOpacityDirect setNodeFontSizeDirect setNodeHeightDirect setNodeImageDirect setNodeLabelColorDirect setNodeLabelDirect setNodeLabelOpacityDirect setNodeOpacityDirect setNodeShapeDirect setNodeSizeDirect setNodeWidthDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeFontSizeDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  setNodeFontSizeDirect (cw, 'A', 32)
  setNodeFontSizeDirect (cw, c('B', 'C'), 32)
  setNodeFontSizeDirect (cw, c('A', 'B'), c(10, 15))

## End(Not run)
```

---

setNodeHeightDirect *setNodeHeightDirect*

Description

In the specified CytoscapeWindow, set the height of the specified node(s). Note that the node dimensions (height and width) must be unlocked for this to work. If they ARE locked, then node and height change together, as specified by a node size rule, or the setNodeSizeDirect method.

Usage

```
setNodeHeightDirect(obj, node.names, new.heights)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one ore more String objects.
new.heights	one or more integers, in pixel units.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeWidthDirect lockNodeDimensions setNodeSizeDirect setNodeHeightDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeHeightDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  lockNodeDimensions (cw, FALSE)
  setNodeHeightDirect (cw, 'A', 32)
  setNodeHeightDirect (cw, c('A', 'B'), 15)
  setNodeHeightDirect (cw, c('C', 'B'), c(15, 30))

## End(Not run)
```

---

setNodeImageDirect *setNodeImageDirect*

Description

In the specified CytoscapeWindow, set the images of the specified nodes.

Usage

```
setNodeImageDirect(obj, node.names, image.positions)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
image.positions	one or more int objects, specifying the position of the image in the Image Manager. If just one, then this is replicated for each of the supplied node.names.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeShapeDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setNodeImageDirect.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
# Before running the next function, upload your image to the Image Manager
# and determine the position of the image in the list (start counting from the bottom)
# Images are added at the top of the interface but at the end of the list
# There are currently 24 default images in the Image Manager
setNodeImageDirect (cw, 'C', 1 ) # last image in the Image Manager

## End(Not run)
```

---

setNodeLabelColorDirect

setNodeLabelColorDirect

Description

In the specified CytoscapeWindow, set the color of the font used in rendering the label of the specified node(s).

Usage

```
setNodeLabelColorDirect(obj, node.names, new.colors)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.colors	one or more String, specifying the color using standard hex notation.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorDirect setNodeBorderOpacityDirect setNodeBorderWidthDirect setNodeColorDirect setNodeFillOpacityDirect setNodeFontSizeDirect setNodeHeightDirect setNodeImageDirect setNodeLabelDirect setNodeLabelOpacityDirect setNodeOpacityDirect setNodeShapeDirect setNodeSizeDirect setNodeWidthDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeLabelColorDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setNodeFontSizeDirect (cw, getAllNodes(cw), 50)
  setNodeLabelColorDirect (cw, 'A', '#FFFF00')
  setNodeLabelColorDirect (cw, c('B', 'C'), '#AA00AA')
  setNodeLabelColorDirect (cw, c('B', 'C'), c('#FF0AAA', '#AAFFAA'))

## End(Not run)
```

---

setNodeLabelDirect *setNodeLabelDirect*

Description

In the specified CytoscapeWindow, set the labels of the specified node(s).

Usage

```
setNodeLabelDirect(obj, node.names, new.labels)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.labels	one or more String objects. If just one, then this is replicated for each of the supplied node.names.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeBorderColorDirect setNodeBorderOpacityDirect setNodeBorderWidthDirect setNodeColorDirect setNodeFillOpacityDirect setNodeFontSizeDirect setNodeHeightDirect setNodeImageDirect setNodeLabelColorDirect setNodeLabelOpacityDirect setNodeOpacityDirect setNodeShapeDirect setNodeSizeDirect setNodeWidthDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeLabelDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setNodeLabelDirect (cw, 'A', 'A new, very long label')
  setNodeLabelDirect (cw, c('B', 'C'), 'Some node')
  setNodeLabelDirect (cw, c('B', 'C'), c('node B', 'node C'))

## End(Not run)
```

---

setNodeLabelOpacityDirect

setNodeLabelOpacityDirect

Description

In the specified CytoscapeWindow, set the opacity of the label of the specified node(s).

Usage

```
setNodeLabelOpacityDirect(obj, node.names, new.values)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.values	one or more numeric objects, ranging from 0 to 255.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeFillOpacityDirect setNodeOpacityDirect setNodeBorderOpacityDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeLabelOpacityDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setNodeLabelOpacityDirect (cw, 'A', 210)
  setNodeLabelOpacityDirect (cw, c('B', 'C'), c(111, 133))
  setNodeLabelOpacityDirect (cw, c('B', 'C'), 11)

## End(Not run)
```

`setNodeLabelRule` *setNodeLabelRule*

Description

Specify the node attribute to be used as the label for each node. Non-character attributes are converted to strings before they are used as labels.

Usage

```
setNodeLabelRule(obj, node.attribute.name)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will determine the label on each node when this rule is applied.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeLabelRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setNodeLabelRule (cw, 'label')
  setNodeLabelRule (cw, 'type')
```

```
  setNodeLabelRule (cw, 'lfc')
  setNodeLabelRule (cw, 'count')

## End(Not run)
```

---

setNodeOpacityDirect *setNodeOpacityDirect*

Description

In the specified CytoscapeWindow, set the opacity of all aspectes of the specified node(s): fill color, border, label.

Usage

```
setNodeOpacityDirect(obj, node.names, new.values)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.values	one or more numeric objects, ranging from 0 to 255.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeFillOpacityDirect setNodeLabelOpacityDirect setNodeBorderOpacityDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeOpacityDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setDefaultNodeBorderWidth (cw, 4)
  setDefaultNodeBorderColor(cw, '#0000FF')

  setNodeOpacityDirect (cw, 'A', 20)
  setNodeOpacityDirect (cw, c('B', 'C'), c(111, 133))
  setNodeOpacityDirect (cw, c('B', 'C'), 10)

## End(Not run)
```

setNodeOpacityRule *setNodeOpacityRule*

Description

Specify how data attributes – for the specified named attribute – are mapped to node opacity. There are two modes: 'interpolate' and 'lookup'. In the former, you specify data values ('control points') and opacitys; when a node's corresponding data attribute value is exactly that of a control point, the specified opacity is used. If the node's data attribute falls between control points, then the opacity is interpolated. The 'lookup' mode provides no interpoloation, and is useful when you have a node attribute with a finite set of discrete values, each of which you you want to display in a specific opacity. For example: render all receptors with full brightness, all transcription factors faded by 50 and all kinases nearly invisible.

Usage

```
setNodeOpacityRule(obj, node.attribute.name, control.points, opacities, mode, aspect='all')
```

Arguments

obj	a CytoscapeWindowClass object.
node.attribute.name	the node attribute whose values will determine the opacity of each node when this rule is applied.
control.points	a list of values. In interpolate mode, a typical choice is the minimum, the maximum and some sensible midpoint.
opacities	a list of opacities, either two more than the number of control points (if mode='interpolate'), in which case the first opacity is used for all attributes values below the minimum, and the last opacity is used for those above the maximum. Or, if mode='lookup', the same number of opacities as control.points are expected. Opacities are expressed as integers in the range 0:255, from invisible to fully bright rendering.
mode	'interpolate' or 'lookup'. This roughly corresponds to the visual mapping of continuously varying data (i.e., lfc or pValue), versus visual mapping of discrete data (i.e., molecule type, or phosphorylation status). With the interpolation mode, you must specify n+2 opacitys: adding a 'below' and an 'above' opacity. In lookup mode, specify exactly as many control.points as opacities. If data attribute values are found on the nodes which do not appear in your list, they will be displayed in the default opacity.
aspect	a character string, with one or more of these values: 'border', 'label', 'fill', 'all'. The first three aspects describe elements of the displayed node: its border, its text label, and its body (or 'fill'). 'all' implies that all elements (border, label and fill) will be operated upon, equally, by this rule. If you want, for instance, the node label (its displayed name) to be visible even if the border and fill are dim, then use 'border, fill' as the aspect.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeColorRule, setNodeOpacityDirect

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setNodeOpacityRule.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'force-directed')
control.points <- c(-3.0, 0.0, 3.0)   # typical range of log-fold-change ratio values
opacities <- c(128, 80, 255)
setNodeOpacityRule (cw, node.attribute.name='lfc', control.points, opacities, mode='interpolate', aspect='
  # now restore full opacities
gene.types <- c("kinase", "transcription factor", "glycoprotein")
setNodeOpacityRule (cw, 'type', gene.types, c (255, 255, 255),  mode='lookup', aspect='all');
  # leaving node labels fully opaque -- fully visible -- change border and fill opacity
opacities <- c(10, 80, 255)
setNodeOpacityRule (cw, node.attribute.name='type', gene.types, opacities, mode='lookup', aspect='border,f

## End(Not run)
```

---

setNodePosition *setNodePosition*

Description

Set the position of the specified nodes on the CytoscapeWindow canvas. Use this for any hand-crafted layouts, or novel layout algorithms, you wish to use.

Usage

```
setNodePosition(obj, node.names, x.coords, y.coords)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	a list of strings, the names of nodes to select.
x.coords	a list of floating point numbers, one for each node in the node.names list.
y.coords	a list of floating point numbers, one for each node in the node.names list.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getNodePosition

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodePosition.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw)
  setNodePosition (cw, c ('A', 'B', 'C'), c (10.0, 20.0, 500), c (0.0,
              100.0, 3))

## End(Not run)
```

---

setNodeShapeDirect *setNodeShapeDirect*

Description

In the specified CytoscapeWindow, set the shape of the specified node(s).

Usage

```
setNodeShapeDirect(obj, node.names, new.shapes)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.shapes	one or more String objects, each of the allowed values returned by getNodeShape.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getNodeShapes

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeShapeDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'grid')
  getNodeShapes (cw)
  setNodeShapeDirect (cw, 'A', 'TRIANGLE')
  setNodeShapeDirect (cw, c('B', 'C'), getNodeShapes(cw)[2:3])
  setNodeShapeDirect (cw, c('A', 'B', 'C'), 'PARALLELOGRAM')

## End(Not run)
```

---

setNodeShapeRule *setNodeShapeRule*

Description

Specify how data attributes – the specified node attribute values – determine the node shape.

Usage

```
setNodeShapeRule (obj, node.attribute.name=, attribute.values,
node.shapes, default.shape)
```

Arguments

obj	a CytoscapeWindowClass object.
node.attribute.name	the node attribute whose values will determine the shape of each node when this rule is applied.
attribute.values	A list of scalar, discrete values. For instance, molecule types: 'transporter', 'receptor', 'kinase'
node.shapes	A list of nodes selected from among those supported.
default.shape	A single String, the shape used if no explicit mapping is provided.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeColorRule setNodeLabelRule

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeShapeRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  shapes <- c("TRIANGLE", "ROUND_RECTANGLE", "ELLIPSE")
  molecule.types <- c("kinase", "transcription factor", "glycoprotein")
  setNodeShapeRule (cw, node.attribute.name='type', molecule.types, shapes)

## End(Not run)
```

`setNodeSizeDirect` *setNodeSizeDirect*

Description

In the specified CytoscapeWindow, set the size of the specified node(s). Note that the node dimensions (height and width) must be unlocked for this to work. Node height and width change together.

Usage

```
setNodeSizeDirect(obj, node.names, new.sizes)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.names`	one or more `String` objects.
`new.sizes`	one or more `integers`, in pixel units.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

lockNodeDimensions setNodeWidthDirect setNodeHeightDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeSizeDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'grid')
  setNodeSizeDirect (cw, 'A', 32)
  setNodeSizeDirect (cw, getAllNodes(cw), 15)
  setNodeSizeDirect (cw, c('A', 'B'), c(30, 40))

## End(Not run)
```

`setNodeSizeRule` *setNodeSizeRule*

Description

Specify how data attributes – the specified node attribute values – determine the node size.

Usage

```
setNodeSizeRule (obj, node.attribute.name, control.points, node.sizes,mode, default.size=40)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will determine the size of each node.
`control.points`	A list of (currently, exactly 3) values, which specify the 'control points' to control the sizes of nodes.
`node.sizes`	The nodes sizes which correspond to the control points.
`mode`	'interpolate' or 'lookup'. This roughly corresponds to the visual mapping of continuously varying data (i.e., lfc or pValue), versus visual mapping of discrete data (i.e., molecule type, or phosphorylation status). With the interpolation mode, you must specify n+2 sizes: adding a 'below' and an 'above' size. In lookup mode, specify exactly as many control.points as sizes. If data attribute values are found on the nodes which do not appear in your list, they will displayed in the default size.
`default.size`	the size of nodes not otherwise specified.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeColorRule

Examples

```
## Not run:
# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

cw <- CytoscapeWindow ('setNodeSizeRule.test', graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork (cw, 'grid')
control.points <- c (10, 30, 80)
node.sizes  <- c (20, 50, 80)
node.attribute.name <- 'count'   # previously defined, has values which range between 2 and 100
  # remind yourself of the values of count on each of the three nodes
print (noa (getGraph (cw), 'count'))
  #  A   B   C
  #  2  30 100
setNodeSizeRule (cw, node.attribute.name, control.points, node.sizes, mode='interpolate')
  # a warning is issued; below and above sizes must be inferred

  # now make a new rule. Explicitly specify below and above sizes
node.sizes  <- c(1, 20, 50, 80, 200)
  # anything below 20 will have size of 1; anything above 80 will be 200.
setNodeSizeRule (cw, node.attribute.name, control.points, node.sizes, mode='interpolate')
  # a warning is issued; below and above sizes must be inferred

  # now use a mode='lookup' rule. Specify a size for two of the molecule types
  # look to see that the third type, glycoprotein, gets the tiny default.size of 5

molecule.types <- c('kinase', 'transcription factor')
node.sizes  <- c(60, 80)
setNodeSizeRule (cw, 'type', molecule.types,  node.sizes, default.size= 5, mode='lookup')

## End(Not run)
```

---

setNodeTooltipRule *setNodeTooltipRule*

Description

Specify the node attribute to be used as the tooltip for each node. Non-character attributes are converted to strings before they are used as tooltips.

Usage

```
setNodeTooltipRule(obj, node.attribute.name)
```

Arguments

obj	a CytoscapeWindowClass object.
node.attribute.name	the node attribute whose values will determine the tooltip on each node when this rule is applied.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeTooltipRule.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  setNodeTooltipRule (cw, 'type')
  setNodeTooltipRule (cw, 'lfc')
  setNodeTooltipRule (cw, 'count')

## End(Not run)
```

---

setNodeWidthDirect *setNodeWidthDirect*

Description

In the specified CytoscapeWindow, set the width of the specified node(s). Note that the node dimensions (width and width) must be unlocked for this to work. If they are locked, then node and width change together, as specified by a node size rule, or the setNodeSize method.

Usage

```
setNodeWidthDirect(obj, node.names, new.widths)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	one or more String objects.
new.widths	one or more integer objects, in pixel units.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setNodeWidthRule lockNodeDimensions setNodeSizeDirect setNodeHeightDirect

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('setNodeWidthDirect.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork (cw, 'force-directed')
  # lockNodeDimensions (cw, 'default', FALSE) # --> not required anymore.
  # setNodeWidthDirect does this for you.
  setNodeWidthDirect (cw, 'A', 32)
  setNodeWidthDirect (cw, getAllNodes(cw), 15)
  setNodeWidthDirect (cw, c('A', 'B'), c(30, 40))

## End(Not run)
```

---

setTooltipDismissDelay

setTooltipDismissDelay

Description

Specify the number of milliseconds before the tooltip (a small lightweight window) disappears over a node or edge after appearing.

Usage

```
setTooltipDismissDelay(obj, msecs)
```

Arguments

obj	a CytoscapeConnectionClass object.
msecs	an integer.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setTooltipInitialDelay, setTooltip, setNodeTooltipRule, setEdgeTooltipRule, setNodeTooltipDirect, setEdgeTooltipDirect

Examples

```
  ## Not run:
  # WARNING: Method RCy3::setTooltipDismissDelay() is not implemented in RCy3!

  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.title = 'setTooltipDismissDelay demo'
  cw <- CytoscapeWindow (window.title, graph=makeSimpleGraph())
    # use node type as the tooltip
  setNodeTooltipRule (cw, 'type')
    # and edgeType
  setEdgeTooltipRule (cw, 'edgeType')
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
     # have the tooltips popup after 200 milliseconds, and then
     # disappear after 3000 (3 seconds)
  setTooltipInitialDelay (cw, 200)
  setTooltipDismissDelay (cw, 3000)

## End(Not run)
```

`setTooltipInitialDelay`

setTooltipInitialDelay

Description

Specify the number of milliseconds before the tooltip (a small lightweight window) pops up over a node or edge.

Usage

```
setTooltipInitialDelay(obj, msecs)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.
`msecs`	an `integer`.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

setTooltipDismissDelay, setTooltip, setNodeTooltipRule, setEdgeTooltipRule, setNodeTooltipDirect, setEdgeTooltipDirect

Examples

```
## Not run:
# WARNING: Method RCy3::setTooltipInitialDelay() is not implemented in RCy3!

# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

window.title = 'setTooltipInitialDelay.demo'
cw <- CytoscapeWindow (window.title, graph=makeSimpleGraph())
  # use node type as the tooltip
setNodeTooltipRule (cw, 'type')
  # and edgeType
setEdgeTooltipRule (cw, 'edgeType')
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
   # have the tooltips popup right away, as soon as the mouse hovers
   # over a node or edge, and then stay up as long as the mouse
   # remains on top of that node or edge
setTooltipInitialDelay (cw, 0)
setTooltipDismissDelay (cw, 0)

## End(Not run)
```

`setVisualStyle` *setVisualStyle*

Description

Cytoscape provides a number of canned visual styles. You can also create your own. Use this method to establish an (already-defined) visual style as the style which governs the display of a network in the specified CytoscapeWindow object.

Usage

```
setVisualStyle(obj, new.style.name)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`new.style.name`	a `character string` specifying the name of an existing style you wish to use.

Value

Nothing.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getVisualStyleNames copyVisualStyle

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.name <- 'demo.setVisualStyle'
  cw <- CytoscapeWindow (window.name, graph=makeSimpleGraph ())
  displayGraph (cw)
  layoutNetwork(cw)

  styles <- getVisualStyleNames (cw)
  setVisualStyle (cw, styles[5])

## End(Not run)
```

---

setWindowSize *setWindowSize*

Description

Control the size of the CytoscapeWindow by specifying a width and height. On a typical screen, there may be 1200 pixels in the width of a full-size window, and 800 pixels in height.

Usage

```
setWindowSize(obj, width, height)
```

Arguments

obj	a CytoscapeWindowClass object.
width	a numeric object.
height	a numeric object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getZoom setZoom getCenter setCenter getViewCoordinates fitContent

Examples

```
## Not run:
# WARNING: Method RCy3::setWindowSize() is not implemented in RCy3!

# first, delete existing windows to save memory:
deleteAllWindows(CytoscapeConnection())

window.title = 'setWindowSize demo'
cw <- CytoscapeWindow (window.title, graph=makeSimpleGraph())
displayGraph (cw)
layoutNetwork(cw, 'force-directed')
setWindowSize (cw, 1200, 800)
fitContent (cw)
setWindowSize (cw, 120, 80)
fitContent (cw)
setWindowSize (cw, 600, 400)
fitContent (cw)

## End(Not run)
```

`setZoom` *setZoom*

Description

This method expands or contracts the relative size of the objects (the graph) displayed in the CytoscapeWindow. A value of 1.0 typically renders the graph with an ample margin. A call to fitContent produces a zoom level of about 1.5.

Usage

```
setZoom(obj, new.level)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`new.level`	a `numeric` object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

getZoom getCenter setCenter getViewCoordinates fitContent

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  window.title = 'setZoom demo'
  cw <- CytoscapeWindow (window.title, graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  setZoom (cw, 0.3)
  system ('sleep 0.3')
  setZoom (cw, 3.0)
  system ('sleep 0.3')
  setZoom (cw, 1.0)

## End(Not run)
```

`showGraphicsDetails` *showGraphicsDetails*

Description

For all windows, and regardless of the current zoom level, display or hide graphics details – of which node labels are the most obvious example.

Usage

```
showGraphicsDetails(obj, new.value)
```

Arguments

`obj`	a `CytoscapeConnectionClass` object.
`new.value`	a `boolean`, if the details are shown

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cy <- CytoscapeConnection ()
  showGraphicsDetails (cy, new.value)

## End(Not run)
```

unhideAll *unhideAll*

Description

Currently not implemented yet

Usage

```
unhideAll(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Tanja Muetze, Georgi Kolishovski, Paul Shannon

See Also

selectNodes clearSelection

Examples

```
  ## Not run:
  # first, delete existing windows to save memory:
  deleteAllWindows(CytoscapeConnection())

  cw <- CytoscapeWindow ('unhideAll.test', graph=makeSimpleGraph())
  displayGraph (cw)
  layoutNetwork(cw, 'force-directed')
  redraw (cw)
  clearSelection (cw)
  selectNodes (cw, 'A')
  hideSelectedNodes (cw)
  system ('sleep 0.5')
  unhideAll (cw)

## End(Not run)
```

Index