

NAME

ined – Access and manipulate tkined objects.

DESCRIPTION

The **Tnm** extension can be used to write applications for the tkined(1) network editor. The **ined** command described in this man page allows to access and manipulate the objects shown inside of the editor. The communication between the tkined(1) editor and the **Tnm** process is encapsulated by the **ined** command. The **ined** command also allows to make use of some standard dialogs provided by the tkined(1) editor.

TCL OBJECT FORMAT

The tkined(1) editor distinguishes between various object types. Every object has a unique identifier, called its id, which is used to access and manipulate this object. Exchanging the id between tkined(1) and the **Tnm** extension would be sufficient to implement all of the commands below. However, this would not be very efficient because an application often needs to access a set of standard object attributes. To reduce communication costs, an external object representation of the various object types is used which allows to cache some information in the **Tnm** extension.

The external object format is a Tcl list. The first element in this list contains the object type and the second element the id of the object. The following list elements depend on the type of the object as described below.

To extract elements out of the external object format, you should always use one of the **ined** commands. Although not stated explicitly, all the commands that return information which is included in the external object format accept the external format as a single argument and return the requested value as extracted from the external format. By using these commands, you never need to access the external format directly. This makes your scripts usually more readable and makes sure that your scripts continue to work even if the format changes in a future version.

NETWORK

Network objects are used to represent networks (Ethernet, Token Ring, FDDI, ATM). A network object is represented by a Tcl list containing six elements:

```
{ NETWORK <id> <name> <address> <oid> <list of links> }
```

The first element defines that the list represents a network object. The <id> element is the unique identifier for the network object. The value of the name and the address attributes of the network object are contained in the <name> and <address> elements. The <oid> element contains an external object identifier which is stored with the object and may be used by applications to link external information (e.g. a database) to a tkined object. The last field contains a list of link objects that are connected to this network. The list contains the ids of the links objects.

NODE

Node objects are used to represent devices connected to a network. Nodes may be used to model complex devices such as network server or simple devices such as a transceiver, depending on the level of detail shown in a map. A node object is represented by a Tcl list containing six elements:

```
{ NODE <id> <name> <address> <oid> <list of links> }
```

The meaning of these six elements is similar to the network object described above.

LINK

Link objects represent connections between node objects or a node object and a network object. Note that a link can not connect two network objects directly. You always have to model the device that connects two networks by a node object. A link object is represented by a Tcl list containing four elements:

```
{ LINK <id> <src> <dst> }
```

The first element defines that the list represents a link object. The <id> element is the unique identifier for the link object. The <src> and <dst> elements contain the ids of the node or the network objects that are connected by this link.

GROUP

Group objects are container objects. They can contain arbitrary tkined objects including other group objects. This allows to build hierarchical structures. A group object is represented by a Tcl list containing five elements:

```
{ GROUP <id> <name> <oid> <list of objects> }
```

The first element defines that the list represents a group object. The <id> element is the unique identifier for the group object. The value of the name attribute of the group object is contained in the <name> element. The <oid> element contains an external object identifier which is stored with the object and may be used by applications to link external information (e.g. a database) to a tkined object. The <list of objects> element contains a list of object ids which are contained in this group object.

TEXT

Text objects allow to display some text on the map. They are used to make annotations or comments. A text object is represented by a Tcl list containing three elements:

```
{ TEXT <id> <text> }
```

The first element defines that the list represents a text object. The <id> element is the unique identifier for the text object. The text displayed by the text object is contained in the <text> element.

IMAGE

Image object allow to create and manipulate images. For example, image objects can be used to put a floor plan or a geographical map in the background of a map. An image object is represented by a Tcl list containing three elements:

```
{ IMAGE <id> <filename> }
```

The first element defines that the list represents an image object. The <id> element is the unique identifier for the image object. The image that is displayed by this image object is given by the local file name contained in the <filename> element. The currently supported image format is the X11 bitmap format.

MENU

Menu objects are used to create new menus in the menu bar. An application usually creates one or more menu objects which contain commands provided by this application. A menu object is represented by a Tcl list containing four elements:

```
{ MENU <id> <name> <list of commands> }
```

The first element defines that the list represents a menu object. The <id> element is the unique identifier for the menu object. The <name> element contains the name of the menu as shown in the menu bar and the <list of commands> element lists the commands that are exported by this menu. These commands correspond to Tcl procedures provided by the application. The procedures are called with the list of selected objects.

INTERPRETER

Interpreter objects represent external processes which provide additional functionality. Interpreter objects allow to start and control additional applications from an already running application. They can also be used to exchange messages between applications. An interpreter object is represented by a Tcl list containing three elements:

```
{ INTERPRETER <id> <name> }
```

The first element defines that the list represents an interpreter object. The <id> element is the unique identifier for the interpreter object. The <name> element contains the script or program name that was started when the interpreter object was created.

LOG

Log objects represent output windows and provide a way for applications to write texts to the user interface. A log object has a menubar which allows the user to load, save, print or email the contents of the text window. A log object is represented by a Tcl list containing four elements:

```
{ LOG <id> <name> <address> }
```

The first element defines that the list represents a log object. The <id> element is the unique identifier for the log object. The <name> element contains the name of the output window and the <address> element contains the default email address.

REFERENCE

Reference objects are pointers (similar to hypertext links) to other tkined maps. They can be used to tie logical and physical maps for the same network together or they can be used to split up large maps into smaller pieces. A reference object is represented by a Tcl list containing four elements:

```
{ REFERENCE <id> <name> <address> }
```

The first element defines that the list represents a reference object. The <id> element is the unique identifier for the reference object. The <name> element contains the value of the name attribute and the <address> element contains the pointer. The pointer can either be a simple path to a file which contains a tkined map or a FTP URL like `ftp://ftp.ibr.cs.tu-bs.de/pub/local/tkined/maps/ibr.tki`. In the later case, the file will be retrieved via anonymous ftp. This feature allows you to let different people maintain their private network maps while sharing the whole picture globally.

STRIPCHART

Stripchart objects are used to display measured data. They show values in a X-Y diagram which is scaled automatically. Scale lines indicate the current scale value. A stripchart object is represented by a Tcl list containing four elements:

```
{ STRIPCHART <id> <name> <address> }
```

The first element defines that the list represents a stripchart object. The <id> element is the unique identifier

for the stripchart object. The value of the name and the address attributes of the stripchart object are contained in the <name> and <address> elements.

BARCHART

Barchart objects are used to display measured data. They show a set of values in a barchart diagram. Scale lines indicate the current scale value. A barchart object is represented by a Tcl list containing four elements:

```
{ BARCHART <id> <name> <address> }
```

The first element defines that the list represents a barchart object. The <id> element is the unique identifier for the barchart object. The value of the name and the address attributes of the barchart object are contained in the <name> and <address> elements.

GRAPH

Graph objects are used to display measured data. Data is again shown in a X-Y diagram. Graph objects use their own top-level window to display the data and are not embedded into the map. Graph objects are implemented using the BLT Tcl extension. Note, graph objects will be mapped to stripchart objects if there is no BLT extension available. A graph object is represented by a Tcl list containing four elements:

```
{ GRAPH <id> <name> <address> }
```

The first element defines that the list represents a graph object. The <id> element is the unique identifier for the graph object. The value of the name and the address attributes of the graph object are contained in the <name> and <address> elements.

INED OBJECT COMMANDS

This section describes the **ined** commands which can be used to manipulate tkined objects.

ined create NODE

ined create NETWORK [*x1 y1 x2 y2 ...*]

ined create LINK *id1 id2* [*x1 y1 ...*]

ined create GROUP [*ida idb ...*]

ined create TEXT *string*

ined create IMAGE *filename*

ined create MENU *name command1* [*command2 ...*]

ined create INTERPRETER *name*

ined create LOG

ined create REFERENCE

ined create STRIPCHART

ined create BARCHART

ined create GRAPH

The **ined create** command allows to create new tkined objects. The **ined create NODE** command creates a new node object and does not require any arguments. The node will appear with the built-in default icon and the default name. Its initial position is the upper left corner.

The **ined create NETWORK** command creates a new network object. Coordinates for the fixed points of the network object can be defined by providing the optional coordinates. Tkined will use the default coordinates 0 0 130 0 if you do not specify any coordinates.

The **ined create LINK** command creates a new link object. The mandatory parameters define the

two objects which are connected by the new link object. Link objects find their position based on the position of the nodes connected by this link. It is possible to specify additional fixed points that will be respected by the link positioning algorithm.

The **ined create GROUP** command creates a new group object. Optional parameters define the ids of all objects that are becoming members of the new group object. The position of the group object will be determined by the position of its members. The position is the upper left corner if there are no members.

The **ined create TEXT** command creates a new text object. Text objects are always created at the upper left corner and simply display the string given in the mandatory parameter. Note, the string may contain newline characters if they are written as `\n`.

The **ined create IMAGE** command creates background images which are read from the file name given in the mandatory argument. Images are positioned at the upper left corner. The file name must point to a valid X11 bitmap file.

The **ined create MENU** command creates a new menu object. The menu object appears in the menu bar. The first mandatory argument defines the name of the menu as shown in the menu bar. The command arguments will pop up when the menu is opened and correspond to Tcl procedures that are called when a menu entry is selected.

The **ined create INTERPRETER** command creates a new interpreter object which starts an additional application. The mandatory argument specifies which program or script is to be started.

The **ined create LOG** command creates a new log object which can be used to display texts. The new output window automatically pops up on the screen.

The **ined create REFERENCE** command creates a reference object which points to other tkined maps. The address attribute can point to local map files or remote map files by using FTP URLs.

The **ined create STRIPCHART**, **ined create BARCHART** and **ined create GRAPH** commands create stripchart, barchart or graph objects. They can be used by applications to display status and monitoring information. They behave much like node objects but they are not allowed to have links.

ined delete *id*

The **ined delete** command deletes the object given by *id*. This command is understood by all tkined object types. If the object has associated link objects, then these links are also deleted.

ined type *id*

The **ined type** command returns the type of the object given by *id*. The type name is returned in uppercase letters.

ined id *id*

The **ined id** command returns the unique identifier for the object given by *id*. This command is especially useful to extract the object id out of the external object representation. It can also be used to test if the object given by *id* still exists.

ined name *id* [*string*]

ined address *id* [*string*]

ined oid *id* [*number*]

ined attribute *id attribute* [*string*]

The **ined name** command allows to get or set the name attribute of the object given by *id*. The **ined address** command allows to get or set the address attribute of the object given by *id*. The **ined oid** command allows to get or set the oid attribute of the object given by *id*. All three commands return the current value of the attribute or an empty string if the object does not support this attribute. The **ined attribute** command allows to get or set arbitrary user-defined attributes of the object given by *id*. The current value of the *attribute* is returned. An empty string is returned if the *attribute* does not exist.

ined select

ined select *id*

ined unselect *id*

ined selected *id*

The **ined select** command without any arguments returns the list of currently selected objects. It is possible to add an object to the current selection by using the **ined select** command with an argument which contains the object *id*. The **ined unselect** command removes the object given by *id* from the current selection. The **ined selected** command can be used to test if an object given by *id* is currently selected or not. The **ined selected** command returns a boolean value.

ined retrieve [*id*]

The **ined retrieve** command returns a list containing the external object representations of all objects associated with the current view of the tkined editor. The **ined retrieve** command returns the object representation of the object with the given *id* if this optional argument is present.

ined icon *id* [*name*]

The **ined icon** command allows to get or change the name of the icon associated with the object given by *id*. This command is ignored by objects that don't support icons.

ined label *id*

ined label *id* clear

ined label *id* name

ined label *id* address

ined label *id attribute*

The **ined label** command allows an application to query, modify or delete the text displayed in the label below an icon. The first version of the **ined label** command returns the current settings while the second version clears the current label. The **ined label** *id* name command tells tkined to use the value of the name attribute as a label and the **ined label** *id* address command tells tkined to use the value of the address attribute. The **ined label** *id attribute* command tells tkined to use the value found in a user defined *attribute*.

ined font *id* [*fontname*]

The **ined font** command allows to get or change the font used by the object given by *id*. This command is ignored by objects that don't support fonts.

ined color *id* [*colorname*]

The **ined color** command allows to get or change the color which is used to draw the object given by *id*. This command is ignored by objects that don't support color.

ined flash *id seconds*

The **ined flash** command allows to flash the icon associated with the object given by *id*. This command is ignored by objects that can't flash.

ined move *id [x y]*

The **ined move** command allows to move an object and to retrieve the coordinates of an object. The command always returns the current position of the object. The optional *x* and *y* parameters move the object relative to the current position.

ined size *id*

This **ined size** command returns the size of the object given by *id*. The size is returned as the coordinates of the bounding box, that is the upper left and the lower right corners of the bounding box.

ined text *id [text]*

The **ined text** command allows to get or change the text displayed by a text object given by *id*.

ined links *id*

The **ined links** command returns the links currently connected to the object given by *id*. This command is only understood by objects that can be connected to links, like nodes and networks.

ined parent *id*

The **ined parent** command returns the unique identifier of the group which contains the object given by *id*. An empty string is returned if the object given by *id* is not a member of a group object.

ined members *id [list]*

The **ined members** command allows to get or change the members of the group object given by *id*. The command always returns the list of members currently contained in the group.

ined collapse *id*

The **ined collapse** command tells tkined to show the group as a single icon. All objects contained in the group are invisible.

ined expand *id*

The **ined expand** command expands a collapsed group which makes all the members of the group visible. The group itself is shown as a rectangle encompassing the members of the group.

ined collapsed *id*

The **ined collapsed** command allows to check whether a group is collapsed or expanded. It returns a boolean value which is true if the group objects given by *id* is currently collapsed.

ined clear *id*

The **ined clear** command is used to clear all values contained in a graph, bar- or stripchart given by *id*. The **ined clear** can also be used to remove all text from a log object if the object given by *id* is a log object.

ined append *id text*

The **ined append** command appends some *text* to the log object given by *id*.

ined hyperlink *id cmd text*

The **ined hyperlink** command appends the *text* to the log object given by *id* and associates the command *cmd* with it. A button press on *text* will send *cmd* to the interpreter which created the log object given by *id*.

ined send *id cmd*

The **ined send** command is understood by interpreter objects. The command *cmd* is send to the interpreter given by *id* for execution. This allows scripts to use and control other scripts.

ined values *id [number ...]*

The **ined values** command is used to write new data to a barchart, stripchart or graph object given by *id*. If called without the optional arguments, the command returns all data currently stored in the barchart, stripchart or graph object.

ined scale *id [value]*

The **ined scale** command allows to set the scaling factor of a graph, barchart or stripchart object given by *id*. The default scale value is 100. Applications can use this command to adjust the default if they know a better value. The command returns the actual value.

ined jump *id [number]*

The **ined jump** command sets or retrieves the amount of pixels that are scrolled when the stripchart given by *id* reaches the right border.

ined dump *id*

The **ined dump** command returns a command string which can be used to recreate the object given by *id*. The contents of the command string depends on the object type of *id*.

INED DIALOG COMMANDS

This section describes the **ined** commands which can be used to create tkined dialogs on behalf of an application.

ined acknowledge *line [line ...]*

The **ined acknowledge** command displays some text lines and waits until the user clicks on the dismiss button. This dialog is usually used to inform the user about some error situations.

ined confirm *line [line ...] buttonlist*

The **ined confirm** command issues a dialog which displays some text lines and waits until the user clicks on one of the buttons given in the *buttonlist*. The name of the pressed button is returned by the **ined confirm** command.

ined fileselect *title [directory] [file]*

The **ined fileselect** dialog allows to select a local file name. The title of the fileselect dialog is given by the *title* argument. The optional parameters define the default *directory* in the file system and the default *file* name.

ined list *title list buttonlist*

The **ined list** dialog displays a listbox which allows to select one of the elements given in the *list* argument. The *title* parameter defines the title displayed above the listbox. The *buttonlist* contains a set of buttons which the user can press to end the dialog. The result of the dialog is a Tcl list containing two elements. The first element is the button pressed by the user and the second element is the selected list object.

ined browse *title text*

The **ined browse** command allows to display some text that can be browsed by the user. The text can be scrolled by pressing the middle mouse button.

ined request *title requestlist buttonlist*

The **ined request** command allows to request information from the user. The displayed input form has a *title* at the top and a list of buttons as defined in the *buttonlist* at the bottom. Every element in the *requestlist* specifies an input field. The specification on an input field is again a Tcl list. The first element of this list is the label for the input field. The second element is the default value for the input field. The third element defines which input method is used. Recognized input methods are:

scale The scale input method uses a slider to read the value. Additional arguments in the list specify the minimum and the maximum value.

radio The radio input method uses radiobuttons to select one or multiple options. Additional arguments in the list define the legal options.

check The check input method uses checkbuttons to select one or more options. Additional arguments in the list define the legal options.

option The option input method uses an option menu to select one option out of a potentially large set of options. Additional arguments in the list define the legal options.

request

The request input method displays an entry field where the user can type and edit some text. The argument in the list specifies the length of the entry box.

The **ined request** command returns a Tcl list. The first element of this list is the button pressed by the user to end the dialog. The following elements correspond to the elements in the *requestlist* and contain the value as selected by the user.

INED GENERAL COMMANDS

This section describes some commands that manipulate the editor as a whole and are not associated with particular objects.

ined size

The **ined size** command returns the overall size of the map. The size is returned as the coordinates of the bounding box, that is the upper left and the lower right corners of the bounding box. The upper left corner always has the coordinates 0 0.

ined page [*size* [*orientation*]]

The **ined page** command can be used to get or set the page size. The **ined page** command always returns a string with the current page size and its orientation. Legal *size* values are Letter, Legal, DINA0, DINA1, DINA2, DINA3 and DINA4. An unknown *size* value is mapped to DINA4. The *orientation* may be portrait or landscape.

ined trace *callback*

The **ined trace** command tells tkined to trace all operations performed on its objects. For every operation that modifies an object state, the *callback* procedure is called. It is expected to have two arguments. The first argument contains the ined command and the second one the result returned by the ined command. Applications can use this trace for debugging purposes.

ined restart [*command*]

The **ined restart** command allows an application to save some additional information in a tkined map file. This is usually used to automatically restart monitoring jobs. When a new job is created, a Tcl command string is assembled by the application that will restart this job. This *command* string is then send to tkined using the **ined restart** command. Tkined will save this command string together with the map file. If the map is loaded later, the interpreter object is re-created and the command string send to it. The **ined restart** command always returns the current command stored for the current interpreter object.

BUGS

INTERPRETER objects should be named APPLICATION objects.
LOG objects should be named OUTPUT objects.

SEE ALSO

scotty(1), tkined(1), Tnm(n), Tcl(n), Tk(n)

AUTHORS

Juergen Schoenwaelder <schoenw@cs.utwente.nl>