TKdoTEXT

XFCN
No. 4401
Version: 5.0 (ONLY for Registered owners of ToolKit™)

TKdoTEXT allows fast retrieval of TEXT stored as TEXT resources in the stack eliminating the overhead of hiding and showing different fields. You can store formatted TEXT, item lists for menus, preferences, compare lists, action lists, and even scripts. Another advantage is that all the TEXT resources are global to the stack.

∂

Examples - Using Styled Text

ÓÊget TKdoTEXT("myText","styl","List")
ÊÊ›- display "myText" with style in cd fld "List"

ÊÊ›get TKdoTEXT("myText","styl,bg","List")
ÊÊ›- display "myText" with style in BkGnd fld "List"

ÊÊ›get TKdoTEXT("myText","sav,styl","List")
ÊÊ›- save cd fld "List" with styling as resource "myText"

ÊÊ›get TKdoTEXT("myText","sav,styl,bg","List")
ÊÊ›- save bkgnd fld "List" with styling as resource "myText"

Using Styled Text

get TKdoTEXT("myText","styl","List")

This will fetch the 'TEXT' resource "myText" and put it into the (default) card field "List" with styling. Notice we get TKdoTEXT and we do NOT put TKdoTEXT because there is no normal target to the put command (the message window would pop up). This is because the XCMD must display the text into the HyperCard™ field in order to apply the 'STYL' resource to it. In this case TKdoTEXT assumes the existence of a CARD field named "List" because the BG specifier is not included in parameter 2.

„ f the target field does not exist TKdoTEXT will beep and display the appropriate error message in the message window.

Í ou may be wondering why we simply cannot put cd fld "List" into the third parameter. That would pass the contents of cd fld "List" and not a description of the target field. The XFCN could have been written to accept the description in quotes: "cd fld List", however, this would not work if the name of the field were longer than one word. Multiple word names must, themselves, be in quotes which would create a parameter with double quotes in it: "cd fld "my List""

get TKdoTEXT("myText","styl,bg","List")

This is similar to the example above. The only difference is we are specifying a BACKGROUND field be used by including the BG specifier in parameter 2. It will fetch the 'TEXT' resource "myText" and put it into BACKGROUND field "List" WITH styling. Notice we get TKdoTEXT and we do NOT put TKdoTEXT. This is because the XCMD must display the text into the HyperCard™ field in order to apply the 'STYL' resource and style it.

get TKdoTEXT("myText","sav,styl","List")

This will fetch the contents of card field "List" and save it as a 'TEXT' resource named "myText". It will also preserve the styling of the text in a 'STYL' resource of the same name. Notice we get TKdoTEXT and we do NOT put TKdoTEXT since we are only interested in saving the text, not displaying it. The Short Name of the field is in parameter 3. It defaults to a CARD field because no BG specifier is used in

parameter 2. Any 'TEXT' or 'STYL' resource that may pre-exist this save is overwritten and cannot be retrieved.

get TKdoTEXT("myText","sav,styl,bg","List")

This will fetch the contents of BACKGROUND field "List" and save it as a 'TEXT' resource named "myText" because we have included the BG specifier in parameter 2. It will also preserve the styling of the text in a 'STYL' resource of the same name. Notice we get TKdoTEXT and we do NOT put TKdoTEXT since we are only interested in saving the text, not displaying it. The Short Name of the field is in parameter 3. Any 'TEXT' or 'STYL' resource that may pre-exist this save is overwritten and cannot be retrieved.

Examples - without styling

ÓÊget TKdoTEXT("myText")
ÊÊ›- fetch 'TEXT' resource "myText"

ÊÊ›put TKdoTEXT("myText") into cd fld "List"
ÊÊ›- display "myText" without style in cd fld "List"

ÊÊ›get TKdoTEXT("myText","sav",cd fld "List")
ÊÊ›- save cd fld "List" without styling as resource "myText"

ÊÊ›get TKdoTEXT("myText","del")
ÊÊ›- delete "myText" 'TEXT' and 'STYL' resources

Using Text Without Styling

get TKdoTEXT("myText")

will fetch the 'TEXT' resource "myText" and hold it in the HyperCard™ global "it". No styling is applied. See "More Information" below for hints on this usage.

put TKdoTEXT("myText") into cd fld "List"

will fetch the 'TEXT' resource "myText" and put it into card field "List". No styling is applied.

get TKdoTEXT("myText","sav",cd fld "List")

This will fetch the contents of card field "List", as specified by parameter 3, and save it as a 'TEXT' resource named "myText". Notice we get TKdoTEXT and we do NOT put TKdoTEXT since we are only interested in saving the text, not displaying it. No styling is preserved and any 'TEXT' or 'STYL' resource that may pre-exist this save is destroyed and cannot be retrieved.

get TKdoTEXT("myText","del")

This will delete any 'TEXT' and 'STYL' resource named "myText". It cannot be undone!

Parameters

Parameter 1

rName: This MUST be the first item and always refers to the name of the TEXT resource to be used by TKdoTEXT.

Parameter 2 (items are separated by commas)

save: This first removes any existing 'TEXT' and 'STYL' resource named rName. It then saves the text designated by parameter 2 to a 'TEXT' resource named rName. No 'STYL' resource is created unless

the STYL command specifier is also used (see below).

style: This command specifier is used to SAVe or Display a TEXT resource.

If the SAVe specifier is used then a 'STYL' resource is created for the text. This assumes the source of the text is a HyperCard™ field. HyperCard™ containers holding text do NOT maintain styling.

If the SAVe specifier is NOT used then it is assumed you are attempting to display a 'TEXT' resource in a HyperCard™ field and the short name of the field must exist (in quotes) as parameter 3 for TKdoTEXT. The default is a CARD field. If you are attempting to display into a BACKGROUND field then the BG specifier (below) must also be included in the parameter 2 list. See the examples for more detail.

bg: is ONLY used with STYL and specifies that a BACKGROUND field is the target for text to be displayed with styling.

delete: will remove the 'TEXT' and 'STYL' resource named rName.

Parameter 3

Short name: used ONLY with STYL when displaying a 'TEXT' with styling into a HyperCard™ field.

- or -

container name: passed for SAVing a HyperCard™ container or field to a 'TEXT' resource WITHOUT styling.

Requirements

Ó one

Return Values

Returns the TEXT resource if SAVE or DELETE are NOT used, otherwise, nothing is returned.

Í If you are trying to retrieve a 'TEXT' resource and it does NOT exist you will get a beep!

Related Topics

Ó ee TKpopMenu to use TEXT resources as item lists.

Special Features

TKdoTEXT is excellent for storing TEXT resources in your stacks as item lists for popUp menus using TKpopMenu. This is very fast and allows the item list to be global to the stack. TKdoTEXT allows FAST retrieval so the item list can be altered before displaying the menu to the user.

TKdoTEXT is also a clean way to store TEXT as an alternative to having different blocks of TEXT in several fields and then hiding or showing the fields as desired. Since the text is NOT compressed in any way the space required for the text itself is the same, however, the overhead for having and maintaining several fields is eliminated.

More Information

TKdoTEXT can be used to store "Action Lists". As an example, if a menu had "Help" in it and the user selected "Help" you could retrieve the menu action list and direct the action according to the second item

in the selection line.   This can reduce scripting tremendously by eliminating all the conditional statements for each item of the menu.   For example, if a menu had 10 items in it it would normally require ten conditional statements:

```
get TKpopMenu("myMenu")

if it = "Open…" then           -- item   1
  doOpen
else if it = "Close…" then -- item   2
  doClose
    •
    •                              -- items 3-9
    •
else if it = "Help" then     -- item 10
  go stack "Help"
end if
```

However, if the popUp menu XCMD returns the name of the item selected then you can fetch an action list for that menu and perform handler specified by the list.   The action list might look similar to this:

```
______________________
Open…,doOpen
Close…,doClose
  •
  •
  •
Help,go stack "Help"
______________________
```

```
on doPopMenu mName,L,T
  get TKpopMenu(mName,L,T)
  if it = "" then exit doPopMenu

  get TKdoTEXT(mName & ".action")
  repeat with i = 1 to number of lines in it
    if item 1 of line i of it = what then
      do (item 2 of line i of it)
      exit doMenu
    end if
  end repeat
end doPopMenu
```

This makes it really easy to update the action list and makes it simple to create a single handler for any menu simply by passing the name of the 'TEXT' resource to it.

The drawback to this is maintaining the code.   It is not apparent what you are trying to do unless you have access to the action list.

°°°