

wb

COLLABORATORS

	<i>TITLE :</i> wb		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 23, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	wb	1
1.1	wb.doc	1
1.2	workbench.library/AddAppIconA	1
1.3	workbench.library/AddAppMenuItemA	3
1.4	workbench.library/AddAppWindowA	4
1.5	workbench.library/RemoveAppIcon	5
1.6	workbench.library/RemoveAppMenuItem	6
1.7	workbench.library/RemoveAppWindow	7
1.8	workbench.library/WBInfo	7

Chapter 1

wb

1.1 wb.doc

```
AddAppIconA ()
AddAppMenuItemA ()
AddAppWindowA ()
RemoveAppIcon ()
RemoveAppMenuItem ()
RemoveAppWindow ()
WBInfo ()
```

1.2 workbench.library/AddAppIconA

NAME
AddAppIconA - add an icon to workbench's list of appicons. (V36)

SYNOPSIS
AppIcon = AddAppIconA(id, userdata, text, msgport,

D0 D0 D1 A0 A1

lock, diskobj, taglist)
A2 A3 A4

```
struct AppIcon *AddAppIconA(ULONG, ULONG, char *,
    struct MsgPort *, struct FileLock *, struct DiskObject *,
    struct TagItem *);
```

Alternate, varargs version:

```
struct AppIcon *AddAppIcon(ULONG, ULONG, char *,
    struct MsgPort *, struct FileLock *,
    struct DiskObject *,
    tag1, data1,
    tag2, data2,
    ...
    TAG_END );
```

FUNCTION
Attempt to add an icon to workbench's list of appicons. If

successful, the icon is displayed on the workbench (the same place disk icons are displayed).

This call is provided to allow applications to be notified when a graphical object (non necessarily associated with a file) gets 'manipulated'. (explained later).

The notification consists of an AppMessage (found in workbench.h/i) of type 'MTYPE_APPICON' arriving at the message port you specified. The types of 'manipulation' that can occur are:

1. Double-clicking on the icon. am_NumArgs will be zero and am_ArgList will be NULL.
2. Dropping an icon or icons on your appicon. am_NumArgs will be the number of icons dropped on your app icon plus one. am_ArgList will be an array of ptrs to WBArg structures. Refer to the 'WBStartup Message' section of the RKM for more info.
3. Dropping your appicon on another icon. NOT SUPPORTED.

INPUTS

id - this variable is strictly for your own use and is ignored by workbench. Typical uses in C are in switch and case statements, and in assembly language table lookup.

userdata - this variable is strictly for your own use and is ignored by workbench.

text - name of icon (char *)

lock - NULL (Currently unused)

msgport - pointer to message port workbench will use to send you an AppMessage message of type 'MTYPE_APPICON' when your icon gets 'manipulated' (explained above).

diskobj - pointer to a DiskObject structure filled in as follows:

do_Magic - NULL

do_Version - NULL

do_Gadget - a gadget structure filled in as follows:

NextGadget - NULL

LeftEdge - NULL

TopEdge - NULL

Width - width of icon hit-box

Height - height of icon hit-box

Flags - NULL or GADGHIMAGE

Activation - NULL

GadgetType - NULL

GadgetRender - pointer to Image structure filled in as follows:

LeftEdge - NULL

TopEdge - NULL

Width - width of image (must be <= Width of hit box)

Height - height of image (must be <= Height of hit box)

Depth - # of bit-planes in image

ImageData - pointer to actual word aligned bits (CHIP MEM)

PlanePick - Plane mask ((1 << depth) - 1)

PlaneOnOff - 0

NextImage - NULL

SelectRender - pointer to alternate Image struct or NULL

GadgetText - NULL

MutualExclude - NULL

SpecialInfo - NULL

GadgetID - NULL

UserData - NULL

do_Type - NULL

do_DefaultTool - NULL

```

do_ToolTypes - NULL
do_CurrentX - NO_ICON_POSITION (recommended)
do_CurrentY - NO_ICON_POSITION (recommended)
do_DrawerData - NULL
do_ToolWindow - NULL
do_StackSize - NULL

```

(an easy way to create one of these (a DiskObject) is to create an icon with the V2.0 icon editor and save it out. Your application can then call GetDiskObject on it and pass that to AddAppIcon.)

taglist - ptr to a list of tag items. Must be NULL for V2.0.

RESULTS

AppIcon - a pointer to an appicon structure which you pass to RemoveAppIcon when you want to remove the icon from workbench's list of appicons. NULL if workbench was unable to add your icon; typically happens when workbench is not running or under low memory conditions.

EXAMPLE

You could design a print-spooler icon and add it to the workbench. Any file dropped on the print spooler would be printed. If the user double-clicked (opened) your printer-spooler icon, you could open a window showing the status of the print spool, allow changes to print priorities, allow deletions, etc. If you registered this window as an 'appwindow' (explained in workbench.library AddAppWindow) files could also be dropped in the window and added to the spool.

SEE ALSO

RemoveAppIcon()

BUGS

Currently Info cannot be obtained on appicons.

1.3 workbench.library/AddAppMenuItemA

NAME

AddAppMenuItemA - add a menuitem to workbench's list of appmenuitems. (V36)

SYNOPSIS

```

AppMenuItem = AddAppMenuItemA(id, userdata, text, msgport, taglist)
D0           D0       D1       A0       A1       A2

```

```

struct AppMenuItem *AddAppMenuItemA(ULONG, ULONG, char *,
    struct MsgPort *,
    struct TagItem *);

```

Alternate, varargs version:

```

struct AppMenuItem *AddAppMenuItem(ULONG, ULONG, char *,
    struct MsgPort *,
    tag1, data1,
    tag2, data2,

```

```
...
TAG_END );
```

FUNCTION

Attempt to add the text as a menuitem to workbench's list of appmenuitems (the 'Tools' menu strip).

INPUTS

id - this variable is strictly for your own use and is ignored by workbench. Typical uses in C are in switch and case statements, and in assembly language table lookup.

userdata - this variable is strictly for your own use and is ignored by workbench.

text - text for the menuitem (char *)

msgport - pointer to message port workbench will use to send you an AppMessage message of type 'MTYPE_APPMENUITEM' when your menuitem gets selected.

taglist - ptr to a list of tag items. Must be NULL for V2.0.

RESULTS

AppMenuItem - a pointer to an appmenuitem structure which you pass to RemoveAppMenuItem when you want to remove the menuitem from workbench's list of appmenuitems. NULL if workbench was unable to add your menuitem; typically happens when workbench is not running or under low memory conditions.

SEE ALSO

RemoveAppMenuItem()

BUGS

Currently does not limit the system to 63 menu items... Any menu items after the 63rd will not be selectable.

1.4 workbench.library/AddAppWindowA

NAME

AddAppWindowA - add a window to workbench's list of appwindows. (V36)

SYNOPSIS

AppWindow = AddAppWindowA(id, userdata, window, msgport, taglist)

D0 D0 D1 A0 A1 A2

```
struct AppWindow *AddAppWindowA(ULONG, ULONG, struct Window *,
                                struct MsgPort *, struct TagItem *);
```

Alternate, varargs version:

```
struct AppWindow *AddAppWindow(ULONG, ULONG, struct Window *,
                                struct MsgPort *
                                tag1, data1,
                                tag2, data2,
                                ...
                                TAG_END );
```

FUNCTION

Attempt to add the window to workbench's list of appwindows. Normally non-workbench windows (those not opened by workbench) cannot have icons dropped in them. This call is provided to allow applications to be notified when an icon or icons get dropped inside a window that they have registered with workbench. The notification consists of an AppMessage (found in workbench.h/i) of type 'MTYPE_APPWINDOW' arriving at the message port you specified. What you do with the list of icons (pointed to by am_ArgList) is up to you, but generally you would want to call GetDiskObjectNew on them.

INPUTS

id - this variable is strictly for your own use and is ignored by workbench. Typical uses in C are in switch and case statements, and in assembly language table lookup.
 userdata - this variable is strictly for your own use and is ignored by workbench.
 window - pointer to window to add.
 msgport - pointer to message port workbench will use to send you an AppMessage message of type 'MTYPE_APPWINDOW' when your window gets an icon or icons dropped in it.
 taglist - ptr to a list of tag items. Must be NULL for V2.0.

RESULTS

AppWindow - a pointer to an appwindow structure which you pass to RemoveAppWindow when you want to remove the window from workbench's list of appwindows. NULL if workbench was unable to add your window; typically happens when workbench is not running or under low memory conditions.

SEE ALSO

RemoveAppWindow()

NOTES

The V2.0 icon editor is an example of an app window. Note that app window applications generally want to call GetDiskObjectNew (as opposed to GetDiskObject) to get the disk object for the icon dropped in the window.

BUGS

None

1.5 workbench.library/RemoveAppIcon

NAME

RemoveAppIcon - remove an icon from workbench's list of appicons. (V36)

SYNOPSIS

```
error = RemoveAppIcon(AppIcon)
      D0                      A0
BOOL RemoveAppIcon(struct AppIcon *);
```

FUNCTION

Attempt to remove an appicon from workbench's list of appicons.

INPUTS

AppIcon - pointer to an AppIcon structure returned by AddAppIcon.

RESULTS

error - Currently always TRUE...

NOTES

As with anything that deals with async operation, you will need to do a final check for messages on your App message port for messages that may have come in between the last time you checked and the call to removed the App.

SEE ALSO

AddAppIconA()

BUGS

None

1.6 workbench.library/RemoveAppMenuItem

NAME

RemoveAppMenuItem - remove a menuitem from workbench's list of appmenuitems. (V36)

SYNOPSIS

```
error = RemoveAppMenuItem(AppMenuItem)
      D0                                A0
BOOL RemoveAppMenuItem(struct AppMenuItem *);
```

FUNCTION

Attempt to remove an appmenuitem from workbench's list of appmenuitems.

INPUTS

AppMenuItem - pointer to an AppMenuItem structure returned by AddAppMenuItem.

RESULTS

error - Currently always TRUE...

NOTES

As with anything that deals with async operation, you will need to do a final check for messages on your App message port for messages that may have come in between the last time you checked and the call to removed the App.

SEE ALSO

AddAppMenuItemA()

BUGS

None

1.7 workbench.library/RemoveAppWindow

NAME

RemoveAppWindow - remove a window from workbench's list of appwindows. (V36)

SYNOPSIS

```
error = RemoveAppWindow(AppWindow)
      D0                      A0
BOOL RemoveAppWindow(struct AppWindow *);
```

FUNCTION

Attempt to remove an appwindow from workbench's list of appwindows.

INPUTS

AppWindow - pointer to an AppWindow structure returned by AddAppWindow.

RESULTS

error - Currently always TRUE...

NOTES

As with anything that deals with async operation, you will need to do a final check for messages on your App message port for messages that may have come in between the last time you checked and the call to removed the App.

SEE ALSO

AddAppWindowA()

BUGS

None

1.8 workbench.library/WBInfo

NAME

WBInfo - Bring up the Information requester (V39)

SYNOPSIS

```
worked = WBInfo(lock, name, screen)
d0      a0      a1      a2

ULONG WBInfo(BPTR, STRPTR, struct Screen *);
```

FUNCTION

This is the LVO that Workbench calls to bring up the Icon Information requester. External applications may also call this requester. In addition, if someone were to wish to replace this requester with another one, they could do so via a SetFunction.

INPUTS

lock - A lock on the parent directory
name - The name of the icon contained within above directory
screen - A screen pointer on which the requester is to show up

RESULTS

worked - Returns TRUE if the requester came up, FALSE if it did not.

NOTE

Note that this LVO may be called many times by different tasks before other calls return. Thus, the code must be 100% re-entrant.

SEE ALSO

icon.library

BUGS
