AccessWare Function Reference

Optegra® Release 6

DOC40183-007

Parametric Technology Corporation

Parametric Technology Corporation, 140 Kendrick Street, Needham, MA 02494-2714
8 January 2001

Table of Contents

Preface

Introduction to AccessWare Procedure Calls

Retrieving Item Values and Attributes

Setting Item Values and Attributes

Updating Item Appearance

Working with Lists

Working with Tree Nodes

Miscellaneous Functions

Procedure Call Summary

Function Cross-Reference Table

Preface

AccessWare Function Reference describes the procedure calls that allow you to access and manipulate your interface. These procedures enable you to get and put values into an item defined in your resource files. It also allows you to alter the visual appearance of items.

This document explains AccessWare Functions, their parameters and usage with regard to the items to which they apply.

Related Documents

The following documents may be helpful as you use *AccessWare Function Reference*:

- *AccessWare User Guide*
- *AccessWare Quick Reference*
- *Customizing EPD.Connect*

Book Conventions

The following table illustrates and explains conventions used in writing about Optegra applications.

Convention	Example	Explanation
EPD_HOME	`cd` $EPD_HOME`/install` (UNIX) `cd` %EPD_HOME%`\install` (Windows)	Represents the default path where the current version of the product is installed.
Menu selections	Vault > Check Out > Lock	Indicates a command that you can choose from a menu.
Command buttons and options	Mandatory check box, Add button, Description text box	Names selectable items from dialog boxes: options, buttons, toggles, text boxes, and switches.

User input and code	`Wheel_Assy_details`
	
	`-xvf /dev/rst0`
	
	`Enter command>` **`plot_config`**
		Enter the text in a text box or on a command line.
		
		Where system output and user input are mixed, user input is in bold.

System output	`CT_struct.aename`	Indicates system responses.
Parameter and variable names	`tar -cvf /dev/rst0` filename	Supply an appropriate substitute for each parameter or variable; for example, replace filename with an actual file name.
Commands and keywords	The `ciaddobj` command creates an instance of a binder.	Shows command syntax.
Text string	`"SRFGROUPA"` or `'SRFGROUPA'`	Shows text strings. Enclose text strings with single or double quotation marks.
Integer	n	Supply an integer for *n*.
Real number	x	Supply a real number for *x*.
#	`# mkdir /cdrom`	Indicates the root (superuser) prompt on command lines.
%	`%` `rlogin` remote_system_name `-l root`	Indicates the C shell prompt on command lines.
$	`$` `rlogin` remote_system_name `-l root`	Indicates the Bourne shell prompt on command lines.
>	`> copy` filename	Indicates the MS-DOS prompt on command lines.
Keystrokes	`Return` or `Control-g`	Indicates the keys to press on a keyboard.

Online User Documentation

Online documentation for each Optegra book is provided in HTML if the documentation CD-ROM is installed. You can view the online documentation from an HTML browser or from the HELP command.

You can also view the online documentation directly from the CD-ROM without installing it.

From an HTML Browser:

1. Navigate to the directory where the documents are installed. For example,

   $EPD_HOME`/data/html/htmldoc/` (UNIX)

   %EPD_HOME%`\data\html\htmldoc\` (Windows NT)
2. Click `mainmenu.html`. A list of available Optegra documentation appears.
3. Click the book title you want to view.

From the HELP Command:

To view the online documentation for your specific application, click HELP. (Consult the documentation specific to your application for more information.)

From the Documentation CD-ROM:

1. Mount the documentation CD-ROM.
2. Point your browser to:

   CDROM_mount_point`/htmldoc/mainmenu.html` (UNIX)

   CDROM_Drive:`\htmldoc\mainmenu.html` (Windows NT)

Printing Documentation

A PDF (Portable Document Format) file is included on the CD-ROM for each online book. See the first page of each online book for the document number referenced in the PDF file name. Check with your system administrator if you need more information.

You must have Acrobat Reader installed to view and print PDF files.

The default documentation directories are:

- $EPD_HOME`/data/html/pdf/`doc_number`.pdf` (UNIX)
- %EPD_HOME%`\data\html\pdf\`doc_number`.pdf` (Windows NT)

Resources and Services

For resources and services to help you with PTC (Parametric Technology Corporation) software products, see the *PTC Customer Service Guide*. It includes instructions for using the World Wide Web or fax transmissions for customer support.

Documentation Comments

PTC welcomes your suggestions and comments. You can send feedback in the following ways:

- Send comments electronically to `doc-webhelp@ptc.com`.
- Fill out and mail the PTC Documentation Survey located in the *PTC Customer Service Guide.*

Chapter 1

Introduction to AccessWare Procedure Calls

This chapter provides an introduction to using AccessWare functions. It also describes the functions that initiate and exit AccessWare.

- Introduction
- aw_init_windows ()
- aw_exit_windows ()

Introduction

The AccessWare Programmatic Interface enables you to invoke your own programs from actions in an AccessWare interface. It includes a suite of C procedures that enable the your programs to access items in the interface and manipulate their contents. This book describes all the procedure calls available in AccessWare.

AccessWare programmatic `ACTIONs` can be associated with any item in a window. They form the link between the user interface and the application code. They are initiated as a result of some activity in the interface, such as the clicking of a button, and result in a piece of code being executed in the application.

To use the C Programmatic Interface, the `ACTION` must simply specify an integer value. This number identifies a section of C code to be executed when the user clicks the item concerned.

Example:

```
ACTION,1000,
```

The mechanism used to action the code is as follows:

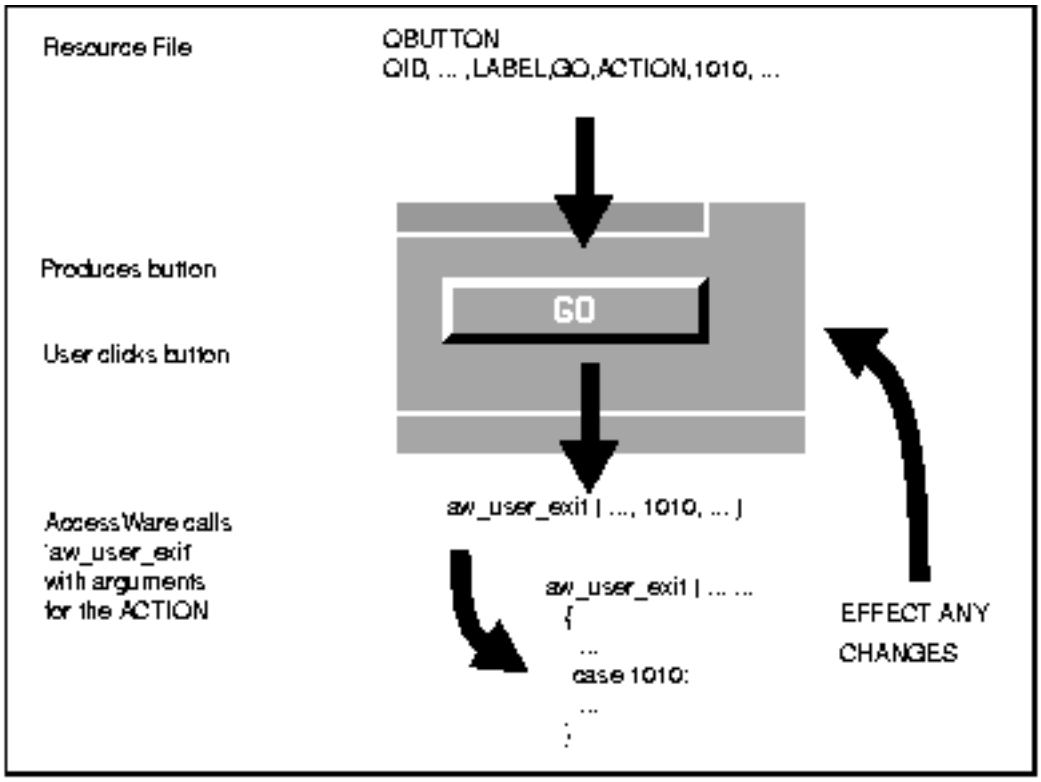
- AccessWare calls the procedure `aw_user_exit` and passes the `ACTION` number, an item identifier, the item value, and the mouse state to it as arguments.
- The `aw_user_exit` procedure comprises a switch statement that contains a list of case statements, each identifying by a label the appropriate section of code for the action required. The statements following the specified label are executed.
- The `aw_user_exit` procedure is called each time the mouse is clicked in an item, or when a window is created or removed.

You must write, compile, and link your own `aw_user_exit` procedure to create your application. The structure of the `aw_user_exit` procedure to build an application using the Programmatic Interface is described in Chapter 6 of the *AccessWare User Guide*.

Please note: Action numbers `0-500` are reserved for AccessWare use. Consequently, all action numbers specified by the application programmer must be greater than `500`.

The following is a diagrammatic representation of how the AccessWare Programmatic Interface works.

Figure 1-1 The Programmatic Interface



Using Libraries

When using AccessWare procedure calls and/or CMOM APIs you need to link to the following CMOM libraires:

- Windows NT:
  - `%EPD_HOME%\lib\optmdll.lib`
  - `%EPD_HOME%\lib\optmlib.lib`
- Unix:

  `$EPD_HOME/lib/libcvoptmsg.so`

aw_init_windows ()

Purpose

`aw_init_windows` passes control to AccessWare from your main procedure.

Syntax

```
#include"access.h"
int   aw_init_windows ( resource_file )
char  *resource_file;
```

Description

This function passes the main program control to AccessWare. Call this function from your main procedure so that all user input enters through the AccessWare user interface.

AccessWare initializes the window specified by `resource_file`.

Diagnostics

The function returns 0 (zero) on success and non zero on failure.

See Also

`aw_exit_windows ()`

aw_exit_windows ()

Purpose

`aw_exit_windows` exits cleanly from the application.

Syntax

```
int   aw_exit_windows ( status )
int   status;
```

Description

This function exits cleanly from the application back to the operating system, and returns the status value.

Use this function instead of the standard `exit()` function so as to enable AccessWare to tidy up properly before exiting. The application exits with the status code passed to this function.

This function exits directly to the operating system and does not pass control back to the application code.

Diagnostics

This function never returns any value.

See Also

`aw_init_windows ()`

Chapter 2

Retrieving Item Values and Attributes

This chapter describes functions that retrieve item values and attributes.

- aw_get_active_id ()
- aw_get_item_action ()
- aw_get_item_active ()
- aw_get_item_backcolor ()
- aw_get_item_bit ()
- aw_get_item_caret ()
- aw_get_item_cols ()
- aw_get_item_depth ()
- aw_get_item_dicon ()
- aw_get_item_dimension ()
- aw_get_item_etch ()
- aw_get_item_font ()
- aw_get_item_greycolor ()
- aw_get_item_height ()
- aw_get_item_helpfile ()
- aw_get_item_icon ()
- aw_get_item_integer ()
- aw_get_item_invert ()
- aw_get_item_itemcolor ()
- aw_get_item_label ()
- aw_get_item_lowercase ()

- aw_get_item_mixedcase ()
- aw_get_item_multiselect ()
- aw_get_item_position ()
- aw_get_item_prompt ()
- aw_get_item_readonly ()
- aw_get_item_real ()
- aw_get_item_rows ()
- aw_get_item_textcolor ()
- aw_get_item_uppercase ()
- aw_get_item_value ()
- aw_get_item_visible ()
- aw_get_item_width ()
- aw_get_item_xposition ()
- aw_get_item_yposition ()
- aw_get_menuitem_state ()
- aw_get_state ()
- aw_get_text_integer ()
- aw_get_text_lowercase ()
- aw_get_item_mixedcase ()
- aw_get_text_readonly ()
- aw_get_text_real ()
- aw_get_text_uppercase ()
- aw_get_udata ()
- aw_get_window_dimension ()
- aw_get_window_font ()
- aw_get_window_height ()
- aw_get_window_position ()
- aw_get_window_resource ()
- aw_get_window_stripe ()
- aw_get_window_width ()
- aw_is_item_active ()
- aw_is_item_bit_set ()

- aw_is_item_multiselect ()
- aw_is_item_readonly ()
- aw_is_item_visible ()

aw_get_active_id ()

Purpose

`*aw_get_active_id` returns the name of the item that initiated the current action.

Syntax

```
#include"access.h"
char  *aw_get_active_id (  )
```

Description

This function returns the name of the currently active item, that is, the item that initiated the current action. The item is identified by its `QID` specified in the resource file.

Diagnostics

The function returns a `NULL` pointer both on failure or if there is no `QID` to return.

See Also

`aw_get_item_action ()`
`aw_set_item_action ()`

aw_get_item_action ()

Purpose

`aw_get_item_action` returns the action associated with an item.

Syntax

```
int   aw_get_item_action ( qid )
char  *qid;
```

Description

This function returns the programmatic action code associated with an item. The item is identified by its `QID` specified in the resource file.

The value returned is the action code of the item. The action is returned whether or not the item is visible. This function applies to any items that have a valid AccessWare action.

Diagnostics

The function returns 0 (zero) for an invalid `QID`, or the action code for a valid `QID`.

See Also

`aw_set_item_action ()`
`aw_get_active_id ()`

aw_get_item_active ()

Purpose

`aw_get_item_active` returns the active status of an item.

Syntax

```
int   aw_get_item_active ( qid )
char  *qid;
```

Description

This function checks if the item, identified by its `QID` as specified in the resource file, is active, that is, whether the user is able to select it.

The function returns the following values:

```
1    ACTIVE, that is, the item is selectable
0    INACTIVE
```

This function applies to items that can be selected.

Diagnostics

None.

See Also

`aw_set_item_active ()`

aw_get_item_backcolor ()

Purpose

`aw_get_item_backcolor` returns the background color of an item.

Syntax

```
int   aw_get_backcolor ( qid )
char  *qid;
```

Description

This function returns the color used as the background color for an item, for example, the color displayed when a button is selected. The item is identified by its `QID` specified in the resource file.

The color is returned as an integer from 0 through 15, which represents the color allocated from the color palette file.

Diagnostics

The function returns 0 (zero) for an invalid `QID` or when a color is the default background color. A positive value indicates the background color.

See Also

```
aw_set_item_backcolor ()
aw_get_item_itemcolor ()
aw_get_item_textcolor ()
aw_get_item_greycolor ()
```

aw_get_item_bit ()

Purpose

aw_get_item_bit gets an option setting or a row number.

Syntax

```
int   aw_get_item_bit ( qid, bit )
char  *qid;
int   bit;
```

Description

This function gets the bit setting for the specified bit of an item specified by QID. The bit can be a row in a QLIST item or a button in a QCHOICE list. It is used for multiple choice items such as QCHOICE and QLIST when they are defined as MODE,MULTI. The item is identified by its QID specified in the resource file.

QLIST: For list items, the argument refers to a line in the list.

QCHOICE: For choice type items, the bit argument must be an integer from 1 to the number of valid options. The options are numbered 1 through n in the order specified in the LABEL definition. The function returns the following values:

```
1    the option/row bit is set
0    the option/row bit is not set
```

Diagnostics

None.

See Also

```
aw_is_item_bit_set ()
aw_set_item_bit ()
aw_reset_item_bit ()
aw_set_item_value ()
aw_get_item_value ()
```

aw_get_item_caret ()

Purpose

`*aw_get_item_caret` returns the name of the item containing the text input caret.

Syntax

```
#include"access.h"
char  *aw_get_item_caret (  )
```

Description

This function returns the name of the item containing the text input caret, that is, the vertical bar that indicates where the user will next enter text. The name of the item returned is its `QID` as specified in the resource file.

Diagnostics

The function returns a `NULL` pointer both on failure or if there is no `QID` to return.

See Also

`aw_set_item_caret ()`

aw_get_item_cols ()

Purpose

`aw_get_item_cols` returns the width of an item defined by the `COLS` attribute.

Syntax

```
int   aw_get_item_cols ( qid )
char  *qid;
```

Description

This function returns the width of an item in character columns. The item is identified by its `QID` specified in the resource file.

The value returned is the value of the `COLS` attribute of the item in character columns. The width is returned whether or not the item is visible. This function applies to all items for which the `COLS` attribute is valid.

Diagnostics

None.

See Also

```
aw_set_item_cols ()
aw_get_item_width ()
aw_get_item_height ()
aw_get_item_real ()
aw_get_item_dimension ()
```

aw_get_item_depth ()

Purpose

`aw_get_item_depth` returns the depth of an item.

Syntax

```
int   aw_get_item_depth ( qid )
char  *qid;
```

Description

This function returns the depth of an item. The item is identified by its `QID` specified in the resource file.

The value returned is the value of the `DEPTH` attribute of the item in pixels. The depth is returned whether or not the item is visible. This function applies to all items for which the `DEPTH` attribute is valid.

Diagnostics

None.

See Also

`aw_set_item_depth ()`
`aw_get_item_invert ()`
`aw_get_item_etch ()`

aw_get_item_dicon ()

Purpose

`*aw_get_item_dicon` returns the name of the icon defined by `DICON` attribute of an item.

Syntax

```
#include"access.h"
char  *aw_get_item_dicon ( qid )
char  *qid;
```

Description

This function returns the name of the icon defined by the item's `DICON` attribute. It is the icon that is displayed when the item is selected. The item is identified by its `QID` specified in the resource file.

The format of the returned icon name is exactly as specified in the resource file.

Diagnostics

The function returns a `NULL` pointer both on failure or if no `DICON` for the item is defined.

See Also

```
aw_set_item_dicon ()
aw_get_item_icon ()
aw_set_item_icon ()
```

aw_get_item_dimension ()

Purpose

`aw_get_item_dimension` returns the dimension of an item.

Syntax

```
int   aw_get_item_dimension ( qid, width, height )
char  *qid;
int   *width, *height;
```

Description

This function returns the size of an item. The item is identified by its `QID` specified in the resource file.

The values returned are the width and height of the item in pixels. This function does not apply to `QWINDOW` and `QPOPUP` items; instead, use the function `aw_get_window_dimension`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_height ()`
`aw_get_item_width ()`
`aw_set_item_height ()`
`aw_set_item_width ()`

aw_get_item_etch ()

Purpose

`aw_get_item_etch` returns the etch setting of an item.

Syntax

```
int   aw_get_item_etch ( qid )
char  *qid;
```

Description

This function returns the `ETCH` setting of an item. The item is identified by its `QID` specified in the resource file.

The value returned is the value of the `ETCH` attribute of the item. The value is returned whether or not the item is visible.

The function returns the following values:

```
1    the item is shown as etched
0    the item is not shown as etched.
```

This function applies to all items for which the `ETCH` attribute is valid.

The `ETCH` attribute is applied to items to maintain consistency with the MOTIF style of interface.

Diagnostics

None.

See Also

```
aw_set_item_etch ()
aw_get_item_depth ()
```

aw_get_item_font ()

Purpose

`*aw_get_item_font` returns the name of the text font of an item.

Syntax

```
#include"access.h"
char  *aw_get_item_font ( qid )
char  *qid;
```

Description

This function returns the name of the font defined for the item. The item is identified by its `QID` specified in the resource file. The font returned is as specified in the resource file for that item, defined by the `FONT` attribute of the item.

The font which is applied to an item can be defined by a `FONT` attribute of the item, by a `FONT` attribute of its parent window, or by a `FONT` attribute of the interface `main` window.

Diagnostics

The function returns a `NULL` pointer both on failure or if no font is defined for the item.

See Also

`aw_set_item_font ()`

aw_get_item_greycolor ()

Purpose

`aw_get_item_greycolor` returns the grey-out color of an item.

Syntax

```
int   aw_get_item_greycolor ( qid )
char  *qid;
```

Description

This function returns the color defined as the `GREYOUTCOL` color for an item, that is, the color used to merge the item into the background when the item is greyed out. The item is identified by its `QID` specified in the resource file.

The color is returned as an integer from 0 through 15, which represents the color allocated from the color palette file.

Diagnostics

The function returns 0 (zero) for an invalid `QID` or when a color is in the default background color. A positive value delivers the grey-out color.

See Also

```
aw_set_item_greycolor ()
aw_get_item_backcolor ()
aw_get_item_itemcolor ()
aw_get_item_textcolor ()
```

aw_get_item_height ()

Purpose

`aw_get_item_height` returns the height of an item.

Syntax

```
int   aw_get_item_height ( qid )
char  *qid;
```

Description

This function returns the height of an item. The item is identified by its `QID` specified in the resource file. The value returned is the height of the item in pixels. The height is returned whether or not the item is visible.

This function does not apply to `QWINDOW` and `QPOPUP` items.

Diagnostics

The function returns 0 (zero) for an invalid `QID`, or the item height for a valid `QID`.

See Also

`aw_set_item_height ()`
`aw_get_item_dimension ()`
`aw_get_item_width ()`

aw_get_item_helpfile ()

Purpose

`*aw_get_item_helpfile` returns the name of the helpfile referenced by an item.

Syntax

```
#include"access.h"
char  *aw_get_item_helpfile ( qid )
char  *qid;
```

Description

This function returns the name of the helpfile referenced by the item. The item is identified by its `QID` specified in the resource file.

The format of the returned file name is the full path name for the file.

Diagnostics

The function returns a `NULL` pointer both on failure or if no helpfile for the item exists.

See Also

`aw_set_item_helpfile ()`

aw_get_item_icon ()

Purpose

`*aw_get_item_icon` returns the name of the icon defined for an item.

Syntax

```
#include"access.h"
char  *aw_get_item_icon ( qid )
char  *qid;
```

Description

This function returns the name of the icon defined by the item's `ICON` attribute. The item is identified by its `QID` specified in the resource file.

The format of the returned icon name is as specified in the resource file.

Diagnostics

The function returns a `NULL` pointer both on failure or if no icon for the item is defined.

See Also

`aw_set_item_icon ()`
`aw_get_item_dicon ()`
`aw_set_item_dicon ()`

aw_get_item_integer ()

Purpose

`aw_get_item_integer` returns the status of integer validation of a text item.

Syntax

```
int   aw_get_item_integer ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE, INTEGER` attribute is assigned for a text item. This attribute validates the input string for an integer number. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1    validation of input string for an integer is specified
0    validation of input string for an integer is not specified
```

Diagnostics

None.

See Also

```
aw_set_item_integer ()
aw_get_text_integer ()
aw_get_item_real ()
```

aw_get_item_invert ()

Purpose

`aw_get_item_invert` returns the invert status of an item.

Syntax

```
int   aw_get_item_invert ( qid )
char  *qid;
```

Description

This function returns the invert status of an item. The item is identified by its `QID` specified in the resource file.

The value returned is the setting of the `INVERT` attribute of the item. The value is returned whether or not the item is visible.

The function returns the following values:

```
1    the item is shown as inverted
0    the item is not shown as inverted.
```

This function applies to all items for which the `INVERT` attribute is valid.

Diagnostics

None.

See Also

```
aw_set_item_invert ()
aw_get_item_depth ()
```

aw_get_item_itemcolor ()

Purpose

`aw_get_item_itemcolor` returns the item foreground color of an item.

Syntax

```
int   aw_get_item_itemcolor ( qid )
char  *qid;
```

Description

This function returns the color used as the item foreground color for an item, for example, the color displayed when a button is not selected. The item is identified by its `QID` specified in the resource file.

The color is returned as an integer from 0 through 15, which represents the color allocated from the color palette file.

Diagnostics

The function returns 0 (zero) for an invalid `QID` or when a color is in the default background color. A positive value indicates the item foreground color.

See Also

```
aw_set_item_itemcolor ()
aw_get_item_backcolor ()
aw_get_item_textcolor ()
aw_get_item_greycolor ()
```

aw_get_item_label ()

Purpose

`*aw_get_item_label` returns the `LABEL` of an item.

Syntax

```
#include"access.h"
char  *aw_get_item_label ( qid )
char  *qid;
```

Description

This function returns the text string defined by the item's `LABEL` attribute. The item is identified by its `QID` specified in the resource file. The format of the returned label is as specified in the resource file.

Diagnostics

The function returns a `NULL` pointer both on failure or if no label for the item exists.

See Also

`aw_set_item_label ()`
`aw_get_item_lowercase ()`

aw_get_item_lowercase ()

Purpose

`aw_get_item_lowercase` returns the status of the text case mode of an item.

Syntax

```
int   aw_get_item_lowercase ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,LOWER` attribute is assigned for an item. The item is identified by its `QID` specified in the resource file.

If the attribute `MODE,LOWER` is specified then the text is converted to lowercase characters on input.

The function returns the following values:

```
1   LOWER CASE mode specified
0   LOWER CASE mode not specified
```

Diagnostics

None.

See Also

```
aw_set_item_lowercase ()
aw_get_text_lowercase ()
aw_get_item_uppercase ()
aw_get_item_mixedcase ()
```

aw_get_item_mixedcase ()

Purpose

`aw_get_item_mixedcase` returns the status of the text case mode of an item.

Syntax

```
int   aw_get_item_mixedcase ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,LOWER` or `MODE,UPPER` attribute is assigned for an item. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1   MIXED CASE mode specified
0   MIXED CASE mode not specified
```

`MIXED CASE` mode not specified implies that either a `MODE,UPPER` or a `MODE,LOWER` attribute is set for the item

Diagnostics

None.

See Also

```
aw_set_item_mixedcase ()
aw_get_text_mixedcase ()
aw_get_item_lowercase ()
aw_get_item_uppercase ()
```

aw_get_item_multiselect ()

Purpose

`aw_get_item_multiselect` returns the multiple selection status of an item.

Syntax

```
int   aw_get_item_multiselect ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,MULTI` attribute is assigned for an item. The item is identified by its `QID` specified in the resource file.

The `MODE,MULTI` attribute applies to `QLIST` and `QCHOICE` items and determines whether multiple item selection is permitted.

The function returns the following values:

```
1   MULTI selection mode specified
0   MULTI selection mode not specified
```

Diagnostics

None.

See Also

`aw_set_item_multiselect ()`

aw_get_item_position ()

Purpose

`aw_get_item_position` returns the position of an item.

Syntax

```
int   aw_get_item_position ( qid, xpos, ypos )
char  *qid;
int   *xpos, *ypos;
```

Description

This function identifies the position of an item. The item is identified by its `QID` specified in the resource file.

The position coordinates `xpos ypos` returned are the number of pixels between the top left corner of the item and the top left corner of the window. The position is returned whether or not the item is visible.

This function does not apply to `QWINDOW` and `QPOPUP` items. Instead, use the function `aw_get_window_position`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_dimension ()`
`aw_get_item_xposition ()`
`aw_get_item_yposition ()`

aw_get_item_prompt ()

Purpose

*aw_get_item_prompt returns the text string for the prompt defined for an item.

Syntax

```
#include"access.h"
char  *aw_get_item_prompt ( qid )
char  *qid;
```

Description

This function returns the text string defined by the item's PROMPT attribute. The item is identified by its QID specified in the resource file.

The prompt string is the text displayed in the interface prompt field, that is, the QPANEL item with an ACTION,PR attribute.

Diagnostics

The function returns a NULL pointer both on failure or if no PROMPT string for the item is defined.

See Also

aw_set_item_prompt ()

aw_get_item_readonly ()

Purpose

`aw_get_item_readonly` returns the read/write status of the item.

Syntax

```
int   aw_get_item_readonly ( qid )
char  *qid;
```

Description

This function checks whether or not the `STATE,READONLY` attribute is assigned for an item. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1    READONLY is set true, i.e. the item is set to read only
0    READONLY is set false, i.e. the item is set to allow
     read/write
```

Diagnostics

The function returns 0 (zero) for an item with read/write status, or for an invalid `QID`.

See Also

```
aw_set_item_readonly ()
aw_is_item_readonly ()
```

aw_get_item_real ()

Purpose

`aw_get_item_real` returns the status of real number validation of a text item.

Syntax

```
int   aw_get_item_real ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,REAL` attribute is assigned for a text item. This attribute validates an input text string for an real number. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1    validation of input string for an real number is specified
0   validation of input string for an real number is not specified
```

Diagnostics

None.

See Also

```
aw_set_item_real ()
aw_get_text_real ()
aw_get_item_integer ()
```

aw_get_item_rows ()

Purpose

`aw_get_item_rows` returns the height of an item as defined by the `ROWS` attribute.

Syntax

```
int   aw_get_item_rows ( qid )
char  *qid;
```

Description

This function returns the height of an item in terms of rows of text. The item is identified by its `QID` specified in the resource file.

The value returned is the value of the `ROWS` attribute of the item. The height is returned whether or not the item is visible.

This function applies to all items for which the `ROWS` attribute is valid.

Diagnostics

None.

See Also

`aw_set_item_rows ()`
`aw_get_item_width ()`
`aw_get_item_height ()`
`aw_get_item_cols ()`
`aw_get_item_dimension ()`

aw_get_item_textcolor ()

Purpose

`aw_get_item_textcolor` returns the text color of an item.

Syntax

```
int   aw_get_item_textcolor ( qid )
char  *qid;
```

Description

This function returns the color used as the item text color for an item, for example, the color in which the text is displayed when a button is not selected. The item is identified by its `QID` specified in the resource file.

The color is returned as an integer from 0 through 15, which represents the color allocated from the color palette file.

Diagnostics

The function returns 0 (zero) for an invalid `QID` or when a color is in the default background color. A positive value indicates the item text color.

See Also

```
aw_set_item_textcolor ()
aw_get_item_backcolor ()
aw_get_item_itemcolor ()
aw_get_item_greycolor ()
```

aw_get_item_uppercase ()

Purpose

`aw_get_item_uppercase` returns the status of the text case mode of the item.

Syntax

```
int   aw_get_item_uppercase ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,UPPER` attribute is assigned for an item. The item is identified by its `QID` specified in the resource file.

If the `MODE,UPPER` attribute is specified then the text is converted to uppercase characters on input.

The function returns the following values:

```
1   UPPER CASE mode specified
0   UPPER CASE mode not specified
```

Diagnostics

None.

See Also

```
aw_set_item_uppercase ()
aw_get_text_uppercase ()
aw_get_item_lowercase ()
aw_get_item_mixedcase ()
```

aw_get_item_value ()

Purpose

*aw_get_item_value returns the VALUE of an item.

Syntax

```
#include"access.h"
char  *aw_get_item_value ( qid )
char  *qid;
```

Description

This function returns as a string the value defined by the item's VALUE attribute. The item is identified by its QID specified in the resource file.

QBUTTON: For button type items, the value returned is 1 or 0 according to the state of the button:

```
1           button pressed
0           button not pressed
```

QCHOICE: For choice type items, the value returned indicates which option the user selected. The options are numbered 1 through n in the order specified in the LABEL definition. This function returns the number corresponding to the selected option.

QLIST: For list items, the value returned is the line number in the selected file.

Please note: The value returned is in a string format. Use the standard C procedure call atoi to convert the string to integer format.

Diagnostics

The function returns a NULL pointer on failure.

See Also

```
aw_set_item_value ()
aw_reset_item_bit ()
aw_is_item_bit_set ()
```

aw_get_item_visible ()

Purpose

`aw_get_item_visible` returns the visibility setting associated with an item.

Syntax

```
int   aw_get_item_visible ( qid )
char  *qid;
```

Description

This function checks if the item is visible. Thus, this function checks whether the item, identified by its `QID,` as specified in the resource file, is currently displayed to the user. The visibility of an item is specified by the `VISIBLE` attribute of the item.

The function returns the following values:

```
1    VISIBLE, i.e. the item is displayed.
0    INVISIBLE, i.e. the item is not displayed.
```

Diagnostics

None.

See Also

```
aw_set_item_visible ()
aw_show_item ()
aw_hide_item ()
```

aw_get_item_width ()

Purpose

`aw_get_item_width` returns the width of an item.

Syntax

```
int   aw_get_item_width ( qid )
char  *qid;
```

Description

This function returns the width of an item. The item is identified by its `QID` specified in the resource file. The width returned is the number of pixels occupied by the item. The width is returned whether or not the item is visible. This function does not apply to `QWINDOW` and `QPOPUP` items.

Diagnostics

The function returns 0 (zero) for an invalid `QID`, or the item width for a valid `QID`.

See Also

`aw_set_item_width ()`
`aw_get_item_dimension ()`
`aw_get_item_height ()`

aw_get_item_xposition ()

Purpose

`aw_get_item_xposition` returns the value of the x location of an item.

Syntax

```
int   aw_get_item_xposition ( qid )
char  *qid;
```

Description

This function returns the `x` location of an item. The item is identified by its `QID` specified in the resource file. The value returned is the value of the `x` location in pixels. The value is returned whether or not the item is visible.

Diagnostics

None.

See Also

`aw_set_item_xposition ()`
`aw_get_item_yposition ()`
`aw_get_item_position ()`

aw_get_item_yposition ()

Purpose

`aw_get_item_yposition` returns the value of the y location of an item.

Syntax

```
int   aw_get_item_yposition ( qid )
char  *qid;
```

Description

This function returns the y location of an item. The item is identified by its `QID` specified in the resource file.

The value returned is the value of the `y` location in pixels. The value is returned whether or not the item is visible.

Diagnostics

None.

See Also

`aw_set_item_yposition ()`
`aw_get_item_xposition ()`
`aw_get_item_position ()`

aw_get_menuitem_state ()

Purpose

`aw_get_menuitem_state` gets the state of a menu item.

Syntax

```
int aw_get_menuitem_state (char *id )
```

Description

This function allows the user to get the state of a menu item. A menu item can have one of three states:

- visible and active
- visible and inactive
- invisible

Diagnostics

The function returns values as follows:

- 0 on invisible
- 1 on visible and active
- 2 on visible and inactive

See Also

`aw_get_state ()`

aw_get_state ()

Purpose

`aw_get_state` returns the value of a control state item.

Syntax

```
int   aw_get_state ( char *qid )
```

Description

This function retrieves the value of a control state, identified by its `QID` as specified in the resource file.

The function return values indicate the following:

`1` `ENABLED/SHOWN`

This means that the menu item which uses this control state in it's `ACTIVEDEP` or `VISIBLEDEP` fields is selectable

`0` `DISABLED/HIDDEN`

Diagnostics

The function returns the state value on success and -1 on failure.

See Also

`aw_set_state ()`
`aw_get_udata ()`

aw_get_text_integer ()

Purpose

`aw_get_text_integer` returns the status of integer validation of a text item.

Syntax

```
int   aw_get_text_integer ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE, INTEGER` attribute is assigned for a text item. This attribute validates the input string for an integer number. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1    validation of input string for an integer is specified
0    validation of input string for an integer is not specified
```

Diagnostics

None.

See Also

```
aw_set_text_integer ()
aw_get_text_real ()
aw_set_item_integer ()
```

aw_get_text_lowercase ()

Purpose

`aw_get_text_lowercase` returns the status of the text case mode of a text item.

Syntax

```
int   aw_get_text_lowercase ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,LOWER` attribute is assigned for a text item. The item is identified by its `QID` specified in the resource file.

If the attribute `MODE,LOWER` is specified then the text is converted to lowercase characters on input.

The function returns the following values:

```
1   LOWER CASE mode specified
0   LOWER CASE mode not specified
```

Diagnostics

None.

See Also

```
aw_set_text_lowercase ()
aw_get_text_uppercase ()
aw_get_text_mixedcase ()
aw_get_item_lowercase ()
```

aw_get_text_mixedcase ()

Purpose

`aw_get_text_mixedcase` returns the status of the text case mode of a text item.

Syntax

```
int   aw_get_text_mixedcase ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,LOWER` or `MODE,UPPER` attribute is assigned for a text item. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1   MIXED CASE mode specified
0   MIXED CASE mode not specified
```

`MIXED CASE` mode not specified implies that either a `MODE,UPPER` or a `MODE,LOWER` attribute is set for the item.

Diagnostics

None.

See Also

```
aw_set_text_mixedcase ()
aw_get_text_uppercase ()
aw_get_text_lowercase ()
aw_get_item_mixedcase ()
```

aw_get_text_readonly ()

Purpose

`aw_get_text_readonly` returns the read/write status of a text item.

Syntax

```
int   aw_get_text_readonly ( qid )
char  *qid;
```

Description

This function checks whether or not the `STATE,READONLY` attribute is assigned for a text item. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1    READONLY is set true, i.e. the item is set to read only.
0    READONLY is set false, i.e. the item is set to allow
     read/write.
```

Diagnostics

The function returns 0 (zero) for an item with read/write status, or for an invalid `QID`.

See Also

```
aw_set_text_readonly ()
aw_is_item_readonly ()
```

aw_get_text_real ()

Purpose

`aw_get_text_real` returns the status of real number validation of a text item.

Syntax

```
int   aw_get_text_real ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,REAL` attribute is assigned for a text item. This attribute validates the input text string for a real number. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1    validation of input string for an real number is specified
0    validation of input string for an real number is not specified
```

Diagnostics

None.

See Also

```
aw_set_text_real ()
aw_get_text_integer ()
aw_set_item_real ()
```

aw_get_text_uppercase ()

Purpose

`aw_get_text_uppercase` returns the status of the text case mode of a text item.

Syntax

```
int   aw_get_text_uppercase ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,UPPER` attribute is assigned for a text item. The item is identified by its `QID` specified in the resource file.

If the attribute `MODE,UPPER` is specified then the text is converted to uppercase characters on input.

The function returns the following values:

```
1   UPPER CASE mode specified
0   UPPER CASE mode not specified
```

Diagnostics

None.

See Also

```
aw_set_item_uppercase ()
aw_get_text_lowercase ()
aw_get_text_mixedcase ()
aw_get_item_uppercase ()
```

aw_get_udata ()

Purpose

`aw_get_udata` returns the string specified by the `UDATA` attribute for a menu item.

Syntax

```
char  *aw_get_udata ( char *qid )
```

Description

This function retrieves the user's data in the definition of a menu item under `UDATA`. This will typically be used for `QMENUITEM` arguments.

Diagnostics

The function returns the string specified under `UDATA` or `NULL` if not defined. This function is applicable only to `QMENUITEM`.

See Also

`aw_set_state ()`
`aw_get_state ()`

aw_get_window_dimension ()

Purpose

`aw_get_window_dimension` gets the size of the window.

Syntax

```
int   aw_get_window_dimension ( qid, width, height )
char  *qid;
int   *width, *height;
```

Description

This function returns the current size of the window in pixels. The window is identified by its `QID` specified in the resource file.

The size in pixels is returned as width and height whether or not the window is visible.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_window_position ()`

aw_get_window_font ()

Purpose

`*aw_get_window_font` returns the name of the text font of a window.

Syntax

```
#include"access.h"
char  *aw_get_window_font ( qid )
char  *qid;
```

Description

This function returns the name of the font defined for a window. The window is identified by the `QID` of the `QPOPUP` item referencing the window as specified in the resource file.

If the main window of the application is to be specified, use `mainwin` as the `QID`.

Diagnostics

The function returns a `NULL` pointer both on failure or if no font is defined.

See Also

None.

aw_get_window_height ()

Purpose

`aw_get_window_height` returns the height of a window.

Syntax

```
int   aw_get_window_height ( qid )
char  *qid;
```

Description

This function returns the height of a `QWINDOW` item, identified by the `QID` of its `QPOPUP` specified in the resource file. If the main window of the application is to be specified, use `mainwin` as the `QID`. The value returned is the height of the window in pixels. The height is returned whether or not the window is visible.

Diagnostics

None.

See Also

`aw_set_window_height ()`
`aw_get_window_width ()`

aw_get_window_position ()

Purpose

`aw_get_window_position` gets the position of the window.

Syntax

```
int   aw_get_window_position ( qid, xpos, ypos )
char  *qid;
int   *xpos, *ypos;
```

Description

This function reports the current position of the window in pixels from the top left corner of the screen. The window is identified by its `QID` of its `QPOPUP` specified in the resource file.

The position coordinates `xpos` `ypos` are returned in pixels whether or not the window is visible. If the main window of the application is to be specified, use `mainwin` as the `QID`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_window_dimension ()`

aw_get_window_resource ()

Purpose

`*aw_get_window_resource` returns the name of the resource file of a window.

Syntax

```
#include"access.h"
char  *aw_get_window_resource ( qid )
char  *qid;
```

Description

This function returns the name of the resource file defined for a window. The window is identified by the `QID` of the `QPOPUP` item referencing the window as specified in the resource file.

If the main window of the application is to be specified, use `mainwin` as the `QID`. The format of the returned file name is the full path name for the file.

Diagnostics

The function returns a `NULL` pointer both on failure or if no font is defined.

See Also

`aw_set_window_resource ()`

aw_get_window_stripe ()

Purpose

`aw_get_window_stripe` returns the window stripe setting of an item.

Syntax

```
int   aw_get_window_stripe ( qid )
char  *qid;
```

Description

This function returns the stripe setting of an item. The stripe is the horizontal bar placed across the top of the window by the windowing system. The item is identified by the `QID` of its `QPOPUP` specified in the resource file.

If the main window of the application is to be specified, use `mainwin` as the `QID`. The value is returned whether or not the item is visible.

The function returns the following values:

```
1    the window stripe is displayed
0    the window is displayed without a stripe.
```

Diagnostics

None.

See Also

None.

aw_get_window_width ()

Purpose

`aw_get_window_width` returns the width of a window.

Syntax

```
int   aw_get_window_width ( qid )
char  *qid;
```

Description

This function returns the width of a `QWINDOW` item, identified by the `QID` of its `QPOPUP` specified in the resource file.

If the main window of the application is to be specified, use `mainwin` as the `QID`.

The value returned is the width of the window in pixels. The width is returned whether or not the item is visible.

Diagnostics

None.

See Also

`aw_set_window_width ()`
`aw_get_window_height ()`

aw_is_item_active ()

Purpose

`aw_is_item_active` returns the active status of the item.

Syntax

```
int   aw_is_item_active ( qid )
char  *qid;
```

Description

This function interrogates the status of an item. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1     TRUE  for activated
0     FALSE for deactivated.
```

Diagnostics

The function returns 0 (zero) for a deactivated item, or for an invalid `QID`.

See Also

```
aw_activate_item ()
aw_deactivate_item ()
```

aw_is_item_bit_set ()

Purpose

`aw_is_item_bit_set` checks if a specified option or row has been selected.

Syntax

```
int   aw_is_item_bit_set ( qid, option )
char  *qid;
int   option;
```

Description

This function checks which options or rows are selected by the user. It is used for multiple choice items, such as `QCHOICE` and `QLIST` when they are defined as `MODE,MULTI`. The item is identified by its `QID` specified in the resource file.

The function returns `TRUE 1` or `FALSE 0` to indicate whether or not the specified option is set.

QCHOICE: For choice type items, the option must be an integer from 1 to the number of valid options. The options are numbered 1 through n in the order specified in the `LABEL` definition. The function returns 0 (zero) to indicate that the option is not selected, and `1` when it is selected.

QLIST: For list items, the option refers to a row number in the list. The function returns 0 (zero) to indicate that the row is not selected, and `1` when it is selected.

Usage

The following extracts of code checks which `QCHOICE` items are selected from an item containing 4 options with a `QID` of `mychoice`:

```
int i;
int selected[4];
for ( i=1; i <= 4; i++)
      if ( aw_is_item_bit_set ( ”mychoice”, i ) )
            selected[i-1] == 1;
            else
            selected[i-1] == 0;
```

Diagnostics

The function returns 0 (zero) for an item with the option not set, or for an invalid `QID`.

See Also

`aw_set_item_value ()`
`aw_get_item_value ()`
`aw_reset_item_bit ()`

aw_is_item_multiselect ()

Purpose

`aw_is_item_multiselect` returns the single/multi mode of the item.

Syntax

```
int   aw_is_item_multiselect ( qid )
char  *qid;
```

Description

This function checks whether or not the `MODE,MULTI` attribute is assigned for an item. The item is identified by its `QID` specified in the resource file. Item types must be `QLIST` or `QCHOICE`.

The function returns the following values:

```
1   Multiple-selection mode specified
0   Multiple-selection mode not specified
```

Diagnostics

The function returns 0 (zero) for an item with single-selection mode, or for an invalid `QID`.

See Also

`aw_set_item_multiselect ()`

aw_is_item_readonly ()

Purpose

`aw_is_item_readonly` returns the read/write status of the item.

Syntax

```
int   aw_is_item_readonly ( qid )
char  *qid;
```

Description

This function checks whether or not the `STATE,READONLY` attribute is assigned for an item. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1   READONLY state specified
0   READONLY state not specified
```

Diagnostics

The function returns 0 (zero) for an item with read/write status, or for an invalid `QID`.

See Also

`aw_set_item_readonly ()`

aw_is_item_visible ()

Purpose

`aw_is_item_visible` returns the visibility status of the item.

Syntax

```
int   aw_is_item_visible ( qid )
char  *qid;
```

Description

This function interrogates the visibility status of an item. The item is identified by its `QID` specified in the resource file.

The function returns the following values:

```
1   VISIBLE
0   HIDDEN
```

Diagnostics

The function returns 0 (zero) for a hidden item or for an invalid `QID`.

See Also

```
aw_hide_item ()
aw_show_item ()
```

Chapter 3

Setting Item Values and Attributes

This chapter describes functions that set item values and attributes.

- aw_clear_prompt ()
- aw_clear_message ()
- aw_message ()
- aw_prompt ()
- aw_set_item_action ()
- aw_set_item_active ()
- aw_set_item_backcolor ()
- aw_set_item_bit ()
- aw_set_item_caret ()
- aw_set_item_cols ()
- aw_set_item_depth ()
- aw_set_item_dicon ()
- aw_set_item_etch ()
- aw_set_item_font ()
- aw_set_item_greycolor ()
- aw_set_item_height ()
- aw_set_item_helpfile ()
- aw_set_item_icon ()
- aw_set_item_integer ()
- aw_set_item_invert ()
- aw_set_item_itemcolor ()

- aw_set_item_label ()
- aw_set_item_lowercase ()
- aw_set_item_mixedcase ()
- aw_set_item_multiselect ()
- aw_set_item_position ()
- aw_set_item_prompt ()
- aw_set_item_readonly ()
- aw_set_item_real ()
- aw_set_item_rows ()
- aw_set_item_selectcolor ()
- aw_set_item_textcolor ()
- aw_set_item_uppercase ()
- aw_set_item_value ()
- aw_set_item_visible ()
- aw_set_item_width ()
- aw_set_item_xposition ()
- aw_set_item_yposition ()
- aw_reset_item_bit ()
- aw_set_state ()
- aw_set_text_integer ()
- aw_set_text_lowercase ()
- aw_set_text_mixedcase ()
- aw_set_text_readonly ()
- aw_set_text_real ()
- aw_win_message ()
- aw_win_prompt ()
- aw_set_window_height ()
- aw_set_window_position ()
- aw_set_window_width ()

aw_clear_prompt ()

Purpose

`aw_clear_prompt` clears the current prompt field.

Syntax

```
int   aw_clear_prompt (  )
```

Description

This function clears the prompt field on the currently active window. The message field is identified in the resource file as the `QPANEL` item to which the `ACTION,PR` is assigned.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_set_item_value ()`
`aw_prompt ()`
`aw_win_prompt ()`

aw_clear_message ()

Purpose

`aw_clear_message` clears the current message field.

Syntax

```
int   aw_clear_message ( )
```

Description

This function clears the message field on the currently active window. The message field is identified in the resource file as the `QPANEL` item to which the `ACTION, ME` is assigned.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_set_item_value ()`
`aw_message ()`
`aw_win_message ()`

aw_message ()

Purpose

`aw_message` displays a message to the user.

Syntax

```
int   aw_message ( message )
char  *message;
```

Description

This function displays the specified `message` string to the user. The message appears in the message field of the currently active window.

The message field is identified in the resource file as a `QPANEL` item to which the `ACTION,ME` attribute is assigned. If the message string exceeds the length of the message field, the string is truncated for display and the full string is output on the standard output stream.

The message arguments are specified in the same way as the arguments to a standard `printf` function. For example,

```
aw_message ( ” Error opening file %s after %d attempts”, filename,
no_tries);
```

Diagnostics

None.

See Also

`aw_set_item_value ()`
`aw_prompt ()`

aw_prompt ()

Purpose

`aw_prompt` displays a prompt to the user.

Syntax

```
int   aw_prompt ( prompt )
char  *prompt;
```

Description

This function displays the specified `PROMPT` string to the user. The prompt appears on the prompt field of the currently active window.

The prompt field is identified in the resource file as a `QPANEL` item to which the `ACTION,PR` attribute is assigned. If the prompt string exceeds the length of the prompt field, the string is truncated for display and the full string is output on the standard output stream.

The prompt arguments are specified in the same way as the arguments to a standard `printf` function.

```
For example:
aw_prompt ( " Enter printer to output file %s, filename);
```

Diagnostics

None.

See Also

`aw_set_item_value ()`
`aw_message ()`

aw_set_item_action ()

Purpose

`aw_set_item_action` sets the specified action for the item.

Syntax

```
int   aw_set_item_action ( qid, action )
char  *qid;
int   action;
```

Description

This function sets a specified action number against the specified item. This overrides the action originally allocated to the item by the `ACTION` attribute in the resource file. The item is identified by its `QID` specified in the resource file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_action ()`
`aw_get_active_id ()`

aw_set_item_active ()

Purpose

`aw_set_item_active` sets the active state of an item.

Syntax

```
int   aw_set_item_active ( qid, state )
char  *qid;
int   state;
```

Description

This function resets the active status of the specified item.,that is whether the user is able to select it.

The item is identified by its `QID` specified in the resource file.

The `state` is set as follows:

```
1    ACTIVE
0    INACTIVE, i.e. not selectable
```

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_active ()`

aw_set_item_backcolor ()

Purpose

`aw_set_item_backcolor` sets the background color for the specified item.

Syntax

```
int   aw_set_item_backcolor ( qid, color )
char  *qid;
int   color;
```

Description

This function sets the background color for a selected item to the value specified as color. The item is identified by its `QID` specified in the resource file.

The color is specified as an integer from 0 through 15, which represents a color from the color palette file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_item_backcolor ()
aw_set_item_itemcolor ()
aw_set_item_selectcolor ()
aw_set_item_textcolor ()
aw_set_item_greycolor ()
```

aw_set_item_bit ()

Purpose

`aw_set_item_bit` sets the specified option or row.

Syntax

```
int   aw_set_item_bit ( qid, option )
char  *qid;
int   option;
```

Description

This function sets a specified option, which can also be a row in a `QLIST` item. It is used for multiple choice items such as `QCHOICE` and `QLIST` when they are defined as `MODE,MULTI`. The item is identified by its `QID` specified in the resource file.

QCHOICE: For choice type items, the option argument must be an integer from 1 to the number of valid options. The options are numbered 1 through n in the order specified in the `LABEL` definition.

QLIST: For list items, the option argument refers to a line number in the list.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_set_item_value ()
aw_get_item_value ()
aw_is_item_bit_set ()
aw_reset_item_bit ()
```

aw_set_item_caret ()

Purpose

`aw_set_item_caret` places the text input caret in an item.

Syntax

```
int   aw_set_item_caret ( qid )
char  *qid;
```

Description

This function places the text input `caret` into the item specified. The item is identified by its `QID` specified in the resource file.

If the item already contains text, the caret is positioned after it.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_caret ()`
`aw_cursor_into_item ()`

aw_set_item_cols ()

Purpose

`aw_set_item_cols` resets the width of an item.

Syntax

```
int   aw_set_item_cols ( qid, cols )
char  *qid;
int   cols;
```

Description

This function resets the width of the specified item. The item is identified by its `QID` specified in the resource file.

The parameter `cols` specifies the number of character columns for the item to redefine its width, based on the current font for the item.

If the item is currently displayed, it is redrawn immediately.

This function does not apply to `QWINDOW` and `QPOPUP` items; instead use the function `aw_set_window_width`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_cols ()`
`aw_set_item_width ()`
`aw_set_item_rows ()`
`aw_set_item_height ()`

aw_set_item_depth ()

Purpose

`aw_set_item_depth` resets the depth of an item.

Syntax

```
int   aw_set_item_depth ( qid, depth )
char  *qid;
int   depth;
```

Description

This function resets the depth of the specified item. The item is identified by its `QID` specified in the resource file. It sets or resets the `DEPTH` attribute of the item.

The argument depth specifies the number of pixels to redefine its depth of the item. If the item is currently displayed, it is redrawn immediately. This function applies to all items for which the `DEPTH` attribute is valid.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_depth ()`
`aw_set_item_invert ()`
`aw_set_item_etch ()`

aw_set_item_dicon ()

Purpose

`aw_set_item_dicon` specifies or respecifies the selected icon for an item

Syntax

```
int   aw_set_item_dicon ( qid, dicon )
char  *qid, *dicon;
```

Description

This function allocates or reallocates a specified icon file to the item. The item is identified by its `QID` specified in the resource file.

The name specified by `DICON` can be either a full or relative pathname.

Please note: In order to display `QBUTTON` items as icons, the `STATE,ICONIC` attribute must be assigned to them in the resource file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_dicon ()`
`aw_set_item_icon ()`

aw_set_item_etch ()

Purpose

`aw_set_item_etch` sets the etch of the specified item.

Syntax

```
int   aw_set_item_etch ( qid, etch )
char  *qid;
int   etch;
```

Description

This function resets the etch of the specified item. The item is identified by its `QID` specified in the resource file.

The `ETCH` is the number of pixels used to create a border around the item. If the item is currently displayed, it is redrawn immediately.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_etch ()`
`aw_set_item_depth ()`
`aw_set_item_invert ()`

aw_set_item_font ()

Purpose

`aw_set_item_font` specifies or respecifies a font associated with an item.

Syntax

```
int   aw_set_item_font ( qid, fontname )
char  *qid, *fontname;
```

Description

This function assigns a specified font to the item. The item is identified by its `QID` specified in the resource file.

The font to be assigned is specified by fontname.

The function sets or resets the `FONT` attribute of the item, which will affect an item, a single window or the complete interface depending on whether it is applied to the item, a window, or the main window.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_font ()`

aw_set_item_greycolor ()

Purpose

`aw_set_item_greycolor` sets the item greyout color of an item.

Syntax

```
int   aw_set_item_greycolor ( qid, color )
char  *qid;
int   color;
```

Description

This function sets the `GREYOUTCOL` attribute for an item, that is, the color used to merge the item into the background when the item is greyed out.

The item is identified by its `QID` specified in the resource file.

The color is specified as an integer from 0 through 15, which represents a color from the color palette file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_item_greycolor ()
aw_set_item_backcolor ()
aw_set_item_itemcolor ()
aw_set_item_selectcolor ()
aw_set_item_textcolor ()
```

aw_set_item_height ()

Purpose

`aw_set_item_height` sets the height of the specified item.

Syntax

```
int   aw_set_item_height ( qid, height )
char  *qid;
int   height;
```

Description

This function resets the height of the specified item. The item is identified by its `QID` specified in the resource file.

The height is specified as the number of pixels for the item. If the item is currently displayed, it is redrawn immediately.

This function does not apply to `QWINDOW` and `QPOPUP` items; instead use the function `aw_set_window_height`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_height ()`
`aw_set_item_width ()`
`aw_set_window_height ()`
`aw_set_window_width ()`

aw_set_item_helpfile ()

Purpose

`aw_set_item_helpfile` assigns or reassigns the associated help file for an item.

Syntax

```
int   aw_set_item_helpfile ( qid, helpfile )
char  *qid, *helpfile;
```

Description

This function assigns or reassigns a specified help file to the item. The item is identified by its `QID` specified in the resource file.

The help file name specified by `HELPFILE` can be either a full or relative pathname. The help file is defined by the `HELPFILE` attribute. It is displayed in the defined Help window when Help is toggled on and the item is selected.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_helpfile ()`

aw_set_item_icon ()

Purpose

`aw_set_item_icon` sets the icon for the specified item.

Syntax

```
int   aw_set_item_icon ( qid, icon )
char  *qid, *icon;
```

Description

This function allocates a specified icon file to the item. The item is identified by its `QID` specified in the resource file.

The icon name can specify either a full or relative pathname.

Please note: In order to display `QBUTTON` items as icons, the `STATE,ICONIC` attribute must be assigned to them in the resource file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_icon ()`
`aw_close_window ()`
`aw_open_window ()`

aw_set_item_integer ()

Purpose

`aw_set_item_integer` sets the mode to validate input for an integer number.

Syntax

```
int   aw_set_item_integer ( qid )
char  *qid;
```

Description

This function sets the `MODE` attribute of a `QTEXT` item to `MODE,INTEGER`. The `QTEXT` item is identified by its `QID` specified in the resource file.

Any text entered into the text field is validated as an integer number.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_item_integer ()
aw_set_item_real ()
aw_set_text_integer ()
```

aw_set_item_invert ()

Purpose

`aw_set_item_invert` sets or resets the invert state of an item.

Syntax

```
int   aw_set_item_invert ( qid, state )
char  *qid;
int   state;
```

Description

This function sets or resets the invert status of the specified item.,that is, whether the `DEPTH` attribute is applied negatively to give the appearance of a recessed item.

The item is identified by its `QID` specified in the resource file.

The state is set as follows:

```
1    INVERT is set on, i.e. the item appears recessed
0    INVERT is not set, i.e. the item is not inverted
```

If the item is currently displayed, it is redrawn immediately.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_invert ()`
`aw_set_item_depth ()`

aw_set_item_itemcolor ()

Purpose

`aw_set_item_itemcolor` sets the item color of the specified item.

Syntax

```
int   aw_set_item_itemcolor ( qid, color )
char  *qid;
int   color;
```

Description

This function sets the main foreground color for a deselected item to the value specified as `color`. The item is identified by its `QID` specified in the resource file.

The `color` is specified as an integer from 0 through 15, which represents a color from the color palette file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_item_itemcolor ()
aw_set_item_backcolor ()
aw_set_item_selectcolor ()
aw_set_item_textcolor ()
aw_set_item_greycolor ()
```

aw_set_item_label ()

Purpose

`aw_set_item_label` defines the label of an item.

Syntax

```
int   aw_set_item_label ( qid, label )
char  *qid, *label;
```

Description

This function resets a `label` defined by the `LABEL` attribute for an item. The item is identified by its `QID` specified in the resource file.

The string specified by `label` resets the information defined by the `LABEL` attribute, and replaces any previous string.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_label ()`

aw_set_item_lowercase ()

Purpose

`aw_set_item_lowercase` sets the `MODE` to convert input to lowercase characters.

Syntax

```
int   aw_set_item_lowercase ( qid )
char  *qid;
```

Description

This function sets the `MODE` attribute of a `QTEXT` item to `MODE,LOWER`. The `QTEXT` item is identified by its `QID` specified in the resource file.

Any text already displayed in the field remains unchanged. All text subsequently entered into the field is converted to lowercase characters.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_set_text_lowercase ()
aw_get_item_lowercase ()
aw_set_item_uppercase ()
aw_set_item_mixedcase ()
```

aw_set_item_mixedcase ()

Purpose

`aw_set_item_mixedcase` sets the `MODE` to convert input to mixed case characters.

Syntax

```
int   aw_set_item_mixedcase ( qid )
char  *qid;
```

Description

This function resets the `MODE` attribute of a `QTEXT` item to its default state allowing text entry in mixed case characters.

The `QTEXT` item is identified by its `QID` specified in the resource file.

Any text already displayed in the field remains unchanged. All text subsequently entered into the field is accepted as entered, and not converted.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_item_mixedcase ()
aw_set_text_mixedcase ()
aw_set_item_uppercase ()
aw_set_item_lowercase ()
```

aw_set_item_multiselect ()

Purpose

`aw_set_item_multiselect` sets the `MODE` of an item to single or multi selection.

Syntax

```
int   aw_set_item_multiselect ( qid, state )
char  *qid;
int   *state;
```

Description

This function sets the `MODE` attribute of an item to its `MODE,SINGLE` default for single selection mode, or to `MODE,MULTI` for multi selection mode.

The item is identified by its `QID` specified in the resource file.

The item must be of type `QLIST` or `QCHOICE`.

Specify the required state as follows:

```
1     TRUE  Set the item to multi-selection mode
0     FALSE Set the item to single-selection mode
```

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_multiselect ()`

aw_set_item_position ()

Purpose

`aw_set_item_position` resets the x and y location of an item.

Syntax

```
int   aw_set_item_position ( qid, xposition, yposition )
char  *qid;
int   xposition, yposition;
```

Description

This function resets the location of the specified item. The item is identified by its `QID` specified in the resource file.

The argument `xposition` specifies the `x` location of the item in pixels. The parameter `yposition` specifies the `y` location of the item in pixels.

If the item is currently displayed, it is redrawn immediately.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_position ()`
`aw_set_item_xposition ()`
`aw_set_item_yposition ()`

aw_set_item_prompt ()

Purpose

`aw_set_item_prompt` sets or resets the prompt text string for an item.

Syntax

```
int   aw_set_item_prompt ( qid, prompt )
char  *qid, *prompt;
```

Description

This function sets or resets the prompt string for an item. The item is identified by its `QID` specified in the resource file.

The prompt string is specified by `PROMPT`.

The new prompt text is displayed whenever the cursor moves into the area encompassed by an item.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_prompt ()`

aw_set_item_readonly ()

Purpose

aw_set_item_readonly sets the item to readonly state.

Syntax

```
int   aw_set_item_readonly ( qid, state )
char  *qid;
int   state;
```

Description

This function sets the STATE attribute of a QTEXT item to STATE,READONLY or cancels STATE,READONLY. When set, STATE,READONLY prevents the user modifying the value or contents of the field.

The item is identified by its QID specified in the resource file.

Specify the required state as follows:

```
1     TRUE  Set the item to read only state
0     FALSE Set the item to allow the text to be modified
```

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

aw_is_item_readonly ()

aw_set_item_real ()

Purpose

`aw_set_item_real` sets the `MODE` to validate input for a real number.

Syntax

```
int   aw_set_item_real ( qid )
char  *qid;
```

Description

This function sets the `MODE` attribute of a `QTEXT` item to `MODE,REAL`. The `QTEXT` item is identified by its `QID` specified in the resource file.

Any text entered in the text field will be validated for a real number.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_real ()`
`aw_set_item_integer ()`
`aw_set_text_real ()`

aw_set_item_rows ()

Purpose

`aw_set_item_rows` resets the height of an item.

Syntax

```
int   aw_set_item_rows ( qid, rows )
char  *qid;
int   rows;
```

Description

This function resets the height of the specified item. The item is identified by its `QID` specified in the resource file.

The argument rows specifies the number of lines for the item to redefine its height, based on the current font for the item.

If the item is currently displayed, it is redrawn immediately.

This function does not apply to `QWINDOW` and `QPOPUP` items; instead use the function `aw_set_window_height`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_rows ()`
`aw_set_item_height ()`
`aw_set_item_cols ()`
`aw_set_item_width ()`

aw_set_item_selectcolor ()

Purpose

`aw_set_item_selectcolor` sets the text color of a selected item.

Syntax

```
int   aw_set_item_selectcolor ( qid, color )
char  *qid;
int   color;
```

Description

This function sets the color of the text for a selected item to the value specified as `color`. The item is identified by its `QID` specified in the resource file.

The color is specified as an integer from 0 through 15, which represents a color from the color palette file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_set_item_itemcolor ()
aw_set_item_backcolor ()
aw_set_item_textcolor ()
aw_set_item_greycolor ()
```

aw_set_item_textcolor ()

Purpose

`aw_set_item_textcolor` sets the text color of a deselected item.

Syntax

```
int   aw_set_item_textcolor ( qid, color )
char  *qid;
int   color;
```

Description

This function sets the color of the text for a deselected item to the value specified as `color`. The item is identified by its `QID` specified in the resource file.

The color is specified as an integer from 0 through 15, which represents a color from the color palette file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_item_textcolor ()
aw_set_item_itemcolor ()
aw_set_item_backcolor ()
aw_set_item_selectcolor ()
aw_set_item_greycolor ()
```

aw_set_item_uppercase ()

Purpose

`aw_set_item_uppercase` sets the `MODE` to convert input to uppercase characters.

Syntax

```
int   aw_set_item_uppercase ( qid )
char  *qid;
```

Description

This function sets the `MODE` attribute of a `QTEXT` item to `MODE,UPPER`. The `QTEXT` item is identified by its `QID` specified in the resource file.

Any text already displayed in the field remains unchanged. All text subsequently entered into the field is converted to uppercase characters.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_item_uppercase ()
aw_set_item_lowercase ()
aw_set_item_mixedcase ()
```

aw_set_item_value ()

Purpose

aw_set_item_value sets the VALUE attribute of an item.

Syntax

```
int   aw_set_item_value ( qid, value )
char  *qid, *value;
```

Description

This function sets the parameter of the VALUE attribute of an item. The item is identified by its QID specified in the resource file.

The string variable value sets the value as follows:

QBUTTON: For QBUTTON items, the value represents the state of the button:

```
1           button pressed
0           button not pressed
```

QCHOICE: For QCHOICE items, the value denotes which option is selected as an initial default. The options are numbered 1 through n1 through n in the order specified in the LABEL attribute definition.

> The number in string format of value corresponds to the required option. A value of 0 denotes no options selected.

QPANEL: The VALUE attribute applies only to a QPANEL item configured as a scroll bar by an ACTION,VS or ACTION,HS attribute. The value represents the number of rows or character columns to scroll.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_item_value ()
aw_reset_item_bit ()
aw_is_item_bit_set ()
```

aw_set_item_visible ()

Purpose

`aw_set_item_visible` resets the visibility state of an item.

Syntax

```
int   aw_set_item_visible ( qid, state )
char  *qid;
int   state;
```

Description

This function resets the visibility status of the specified item, that is, whether the item is displayed in the window. It sets or resets the `VISIBLE` attribute of the item.

The item is identified by its `QID` specified in the resource file.

The state is set as follows:

```
1    VISIBILITY is set on, i.e. item will be visible
0    VISIBILITY is set off, i.e. item not visible
```

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_item_visible ()
aw_show_item ()
aw_hide_item ()
```

aw_set_item_width ()

Purpose

`aw_set_item_width` sets the width of the specified item.

Syntax

```
int   aw_set_item_width ( qid, width )
char  *qid;
int   width;
```

Description

This function resets the width of the specified item.

The width is defined as the number of pixels for the item.

If the item is currently displayed, it is redrawn immediately.

This function does not apply to `QWINDOW` and `QPOPUP` items; instead use the function `aw_set_window_width`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_width ()`
`aw_set_item_height ()`
`aw_set_window_height ()`
`aw_set_window_width ()`

aw_set_item_xposition ()

Purpose

`aw_set_item_xposition` resets the x location of an item.

Syntax

```
int   aw_set_item_xposition ( qid, xposition )
char  *qid;
int   xposition;
```

Description

This function resets the `x` value of the location of the specified item. The item is identified by its `QID` specified in the resource file.

The argument `xposition` specifies the `x` location of the item in pixels.

If the item is currently displayed, it is redrawn immediately.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_xposition ()`
`aw_set_item_position ()`
`aw_set_item_yposition ()`

aw_set_item_yposition ()

Purpose

`aw_set_item_yposition` resets the y location of an item

Syntax

```
int   aw_set_item_yposition ( qid, yposition )
char  *qid;
int   yposition;
```

Description

This function resets the `y` value of the location of the specified item. The item is identified by its `QID` specified in the resource file.

The argument `yposition` specifies the `y` location of the item in pixels.

If the item is currently displayed, it is redrawn immediately.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_yposition ()`
`aw_set_item_position ()`
`aw_set_item_yposition ()`

aw_reset_item_bit ()

Purpose

`aw_reset_item_bit` resets the specified option or row.

Syntax

```
int   aw_reset_item_bit ( qid, option )
char  *qid;
int   option;
```

Description

This function resets the specified option, which can also be a row in a `QLIST` item. It is used for multiple choice items such as `QCHOICE` and `QLIST` when they are defined as `MODE,MULTI`. The item is identified by its `QID` specified in the resource file.

QCHOICE: For choice type items, the `option` argument must be an integer from 1 to the number of valid options. The options are numbered 1 through n in the order specified in the `LABEL` definition.

QLIST: For list items, the `option` argument refers to a line number in the list.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_set_item_value ()
aw_get_item_value ()
aw_is_item_bit_set ()
aw_set_item_bit ()
```

aw_set_state ()

Purpose

`aw_set_state` sets the state of a menu item.

Syntax

```
int aw_set_state (char *id, int state)
```

Description

This function allows the user to set the state of a menu item by which it can be enabled, disabled, hidden, or shown.

Diagnostics

The function returns 0 (zero) on success and 1 on failure.

See Also

`aw_get_state ()`
`aw_get_udata ()`

aw_set_text_integer ()

Purpose

`aw_set_text_integer` sets the `MODE` to validate input for an integer number.

Syntax

```
int   aw_set_text_integer ( qid )
char  *qid;
```

Description

This function sets the `MODE` attribute of a `QTEXT` item to `MODE,INTEGER`. The `QTEXT` item is identified by its `QID` specified in the resource file.

Any text entered into the text field is validated as an integer number.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_set_item_integer ()
aw_set_text_real ()
aw_get_text_integer ()
```

aw_set_text_lowercase ()

Purpose

`aw_set_text_lowercase` sets the `MODE` to convert input to lowercase characters.

Syntax

```
int   aw_set_text_lowercase ( qid )
char  *qid;
```

Description

This function sets the `MODE` attribute of a `QTEXT` item to `MODE,LOWER`. The `QTEXT` item is identified by its `QID` specified in the resource file.

Any text already displayed in the field remains unchanged. All text subsequently entered into the field is converted to lowercase characters.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_text_lowercase ()`
`aw_set_item_lowercase ()`
`aw_set_text_mixedcase ()`

aw_set_text_mixedcase ()

Purpose

`aw_set_text_mixedcase` sets the `MODE` to convert input to mixed case characters.

Syntax

```
int   aw_set_text_mixedcase ( qid )
char  *qid;
```

Description

This function sets the `MODE` attribute of a `QTEXT` item to `MODE,MIXED`. The `QTEXT` item is identified by its `QID` specified in the resource file.

Any text already displayed in the field remains unchanged. All text subsequently entered into the field does not undergo case conversion.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_text_mixedcase ()`
`aw_set_text_lowercase ()`
`aw_set_item_mixedcase ()`

aw_set_text_readonly ()

Purpose

`aw_set_text_readonly` sets the write state of a text field.

Syntax

```
int   aw_set_text_readonly ( qid, state )
char  *qid;
int   state
```

Description

This function sets the `STATE` attribute of a `QTEXT` item to `STATE,READONLY` or cancels `STATE,READONLY`. When set, `STATE,READONLY` prevents the user modifying the value or contents of the field. The `QTEXT` item is identified by its `QID` specified in the resource file.

Specify the required state as follows:

```
1    READONLY is set true; the item is read-only.
0    READONLY is set false; the item is readable and writable.
```

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_text_readonly ()
aw_set_item_readonly ()
```

aw_set_text_real ()

Purpose

`aw_set_text_real` sets the `MODE` to validate input for a real number.

Syntax

```
int   aw_set_text_real ( qid )
char  *qid;
```

Description

This function sets the `MODE` attribute of a `QTEXT` item to `MODE,REAL`. The `QTEXT` item is identified by its `QID` specified in the resource file.

Any text entered in the text field is validated as a real number.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_text_real ()
aw_set_text_integer ()
aw_set_item_real ()
```

aw_win_message ()

Purpose

`aw_win_message` displays a message to the user.

Syntax

```
int   aw_win_message ( qid, message )
char  *message;
```

Description

This function displays the specified message in the message field identified by the `QID` of the specified window.

The message field is identified in the appropriate resource file as a `QPANEL` item to which the `ACTION,ME` attribute is assigned. If the message string exceeds the length of the message field, the string is truncated for display and the full string is output on the standard output stream.

Please note: If the window is not currently displayed, the message is not visible.

The message arguments are specified in the same way as the arguments to a standard `printf` function. For example,

```
aw_message ( ” Error opening file %s after %d attempts”, filename,
no_tries);
```

Diagnostics

None.

See Also

`aw_message ()`

aw_win_prompt ()

Purpose

`aw_win_prompt` displays a prompt to the user.

Syntax

```
int   aw_win_prompt ( qid, prompt )
char  *prompt;
```

Description

This function displays the specified `prompt` in the prompt field identified by `QID` of the specified window.

The prompt field is identified in the appropriate resource file as a `QPANEL` item to which the `ACTION,PR` attribute is assigned. If the prompt string exceeds the length of the prompt field, the string is truncated for display and the full string is output on the standard output stream.

Please note: If the window is not currently displayed, the prompt is not visible.

The `prompt` arguments are specified in the same way as the arguments to a standard `printf` function. For example,

```
aw_prompt ( " Enter printer to output file %s, filename);
```

Diagnostics

None.

See Also

`aw_prompt ()`

aw_set_window_height ()

Purpose

`aw_set_window_height` sets the height of the specified window.

Syntax

```
int   aw_set_window_height ( qid, height )
char  *qid;
int   height;
```

Description

This function resets the `height` of a `QWINDOW` item, identified by its `QPOPUP QID` specified in the resource file.

Specify the height as the number of pixels for the window.

If the window is currently displayed, it is redrawn immediately.

The item must be a `QPOPUP` item. To reset the main window of the application, use `mainwin` as the `QID`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_set_item_height ()`
`aw_set_item_width ()`
`aw_set_window_width ()`

aw_set_window_position ()

Purpose

`aw_set_window_position` sets the position of the specified window.

Syntax

```
int   aw_set_window_width ( qid, xpos, ypos )
char  *qid;
int   xpos, ypos;
```

Description

This function resets the position of a `QWINDOW` item, identified by its `QPOPUP QID` specified in the resource file.

Specify the position coordinates `xpos ypos` as the number of pixels from the top left corner of the screen.

If the window is currently displayed, it is redrawn immediately.

The item must be a `QPOPUP` item. To reset the main window of the application, use `mainwin` as the `QID`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_set_window_height ()`
`aw_set_window_width ()`
`aw_get_window_position ()`

aw_set_window_width ()

Purpose

`aw_set_window_width` sets the width of the specified window.

Syntax

```
int   aw_set_window_width ( qid, width )
char  *qid;
int   width;
```

Description

This function resets the `width` of a `QWINDOW` item, identified by its `QPOPUP QID` specified in the resource file.

Specify the width as the number of pixels for the window.

If the window is currently displayed, it is redrawn immediately.

The item must be a `QPOPUP` item. If resetting the main window of the application, use `mainwin` as the `QID`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_set_item_height ()`
`aw_set_item_width ()`
`aw_set_window_height ()`

Chapter 4

Updating Item Appearance

This chapter descries functions that update item appearance.

- aw_activate_item ()
- aw_deactivate_item ()
- aw_show_item ()
- aw_show_item_relative ()
- aw_hide_all_popups ()
- aw_hide_item ()
- aw_open_window ()
- aw_close_window ()
- aw_refresh_window ()
- aw_set_window_resource ()

aw_activate_item ()

Purpose

`aw_activate_item` activates an item on the screen.

Syntax

```
int   aw_activate_item ( qid )
char  *qid;
```

Description

This function activates the specified window item. The item is identified by its `QID` specified in the resource file.

Use this function to activate items currently greyed-out or deactivated. An item already displayed is immediately repainted. For a hidden item, the relevant area of the window is immediately repainted to display the active version of the item.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_hide_item ()
aw_show_item ()
aw_deactivate_item ()
aw_get_item_active ()
aw_set_item_active ()
```

aw_deactivate_item ()

Purpose

`aw_deactivate_item` deactivates and greys out an item on the screen.

Syntax

```
#include"access.h"
int   aw_deactivate_item ( qid )
char  *qid;
```

Description

This function deactivates the specified window item. The item is identified by its `QID` specified in the resource file.

Use this function to deactivate currently activated items and show them greyed-out on the screen.

The relevant area of the window is immediately repainted with the color defined in `GREYOUTCOL`. The item remains visible on the screen unless `GREYOUTCOL` is the same as the background color behind the item, which would render it invisible.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_hide_item ()
aw_show_item ()
aw_activate_item ()
aw_get_item_active ()
aw_set_item_active ()
```

aw_show_item ()

Purpose

`aw_show_item` shows an item on the screen.

Syntax

```
int   aw_show_item ( qid )
char  *qid;
```

Description

This function displays the specified item on the screen. The item is identified by its `QID` specified in the resource file.

The item may be a window or an item within a window.

If the item is already displayed, the item is repainted immediately.

When hiding overlapping items and showing others, make all the `aw_hide_item` calls first, then make the `aw_show_item` calls. Otherwise, the newly painted background areas of the `aw_hide_item` calls may obscure items from the `aw_show_item` calls.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_hide_item ()`
`aw_activate_item ()`
`aw_deactivate_item ()`
`aw_get_item_visible ()`
`aw_set_item_visible ()`

aw_show_item_relative ()

Purpose

`aw_show_item_relative` shows an item on the screen relative to another item.

Syntax

```
int   aw_show_item_relative ( qid, relative_qid, xoff, yoff )
char  *qid, *relative_qid;
int   xoff, yoff;
```

Description

This function shows the specified item on the screen relative to another item. The items are identified by their `QID` and `relative_qid`, respectively, as specified in the resource file.

The item to be made visible may be a window or an item within a window.

If the item is already displayed, the item is repainted immediately.

Specify the coordinate values `xoff` `yoff` as the number of pixels that the top left corner of the item `QID` is offset from the top left corner of the item `relative_qid`.

Positive `x` direction is across the screen. Positive `y` direction is down the screen.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_show_item ()`

aw_hide_all_popups ()

Purpose

`aw_hide_all_popups` hides all popup windows.

Syntax

```
int   aw_hide_all_popups (  )
```

Description

This function hides all currently visible `QPOPUP` windows.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_show_item ()
aw_hide_item ()
aw_get_item_visible ()
aw_set_item_visible ()
```

aw_hide_item ()

Purpose

`aw_hide_item` hides an item on the screen.

Syntax

```
int   aw_hide_item ( qid )
char  *qid;
```

Description

This function removes the specified item from the display. The item is identified by its `QID` specified in the resource file. The item may be a window or an item within a window.

When the item is within a window, the area occupied by the item is repainted in the window background color. A call to this function immediately affects the display.

When hiding overlapping items and showing others, make all the `aw_hide_item` calls first, then make the `aw_show_item` calls. Otherwise, the newly painted background areas of the `aw_hide_item` calls may obscure items from the `aw_show_item` calls.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_show_item ()
aw_activate_item ()
aw_deactivate_item ()
aw_get_item_visible ()
aw_set_item_visible ()
```

aw_open_window ()

Purpose

`aw_open_window` opens the window from an icon.

Syntax

```
int   aw_open_window ( qid )
char  *qid;
```

Description

This function opens a window from its closed icon. The window is identified by its `QID` specified in the resource file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_close_window ()`

aw_close_window ()

Purpose

`aw_close_window` reduces the specified window to an icon.

Syntax

```
int   aw_close_window ( qid )
char  *qid;
```

Description

This function reduces the window identified by its `QID` specified in the resource file to an icon.

Use this function for a currently visible window. The default AccessWare icon is used, unless an alternative icon is specified in the resource file for `QWINDOW`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_open_window ()`
`aw_hide_item ()`
`aw_hide_all_popups ()`

aw_refresh_window ()

Purpose

`aw_refresh_window` updates the window from its resource file.

Syntax

```
int   aw_refresh_window ( qid )
char  *qid;
```

Description

This function forces AccessWare to read the resource file associated with this window and repaint the window accordingly. The window is identified by its `QID` specified in the resource file.

This function reloads the resource file and is usually called following a call to the function `aw_set_window_resource`, which replaces the original resource file for a window with another resource file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_set_window_resource ()`

aw_set_window_resource ()

Purpose

`aw_set_window_resource` allocates a new resource file to the window.

Syntax

```
int   aw_set_window_resource ( qid, resource_file )
char  *qid, *resource_file;
```

Description

This function allocates a new resource file to a `QWINDOW` item, identified by its `QPOPUP` `QID` specified in the resource file.

Specify a full or relative pathname for `resource_file` as the name of the new resource file allocated to the window.

The item must be a `QPOPUP` item. To reset the main window of the application, use `mainwin` as the `QID`.

This function does not automatically update the window. To do so, use the function `aw_refresh_window`.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_refresh_window ()`

Chapter 5

Working with Lists

This chapter describes functions associated with lists.

- aw_add_line_to_list ()
- aw_delete_line_from_list ()
- aw_get_line ()
- aw_get_list_file ()
- aw_get_list_heading ()
- aw_get_list_length ()
- aw_get_list_line ()
- aw_get_list_selection ()
- aw_get_list_width ()
- aw_get_scroll_char ()
- aw_get_scroll_line ()
- aw_mark_list_line ()
- aw_refresh_list ()
- aw_reset_list ()
- aw_save_listfile ()
- aw_scroll_to_char ()
- aw_scroll_to_line ()
- aw_set_item_heading ()
- aw_set_list_file ()

- aw_set_list_heading ()
- aw_switch_list_font ()
- aw_update_list ()

aw_add_line_to_list ()

Purpose

`aw_add_line_to_list` adds a row into an existing list.

Syntax

```
int   aw_add_line_to_list ( qid, line, row )
char  *qid, *line;
int   row;
```

Description

This function adds the text string line into the contents of the list item at row number row. The item is identified by its `QID` specified in the resource file.

The size of the displayed list item is not affected. An additional row is added into the list contents file. Any text occupying the specified row position is moved down one row. It is not overwritten. If the specified new line position exceeds the current last line, the new line is appended immediately at the end of the list. Positions of zero or less return an error.

A list already displayed is immediately repainted.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_delete_line_from_list ()`
`aw_get_line ()`
`aw_get_list_line ()`
`aw_get_list_length ()`

aw_delete_line_from_list ()

Purpose

`aw_delete_line_from_list` deletes a row from an existing list.

Syntax

```
int   aw_delete_line_from_list ( qid, row )
char  *qid,
int   row;
```

Description

This function deletes the text string line from the contents of the list item at row number row. The item is identified by its `QID` specified in the resource file.

The size of the displayed list item is not affected. A row is removed from the list contents. Any text following the specified row position is moved up and the list contains one less row.

A list already displayed is immediately repainted. If the specified position exceeds the end of the current list, an error is returned.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_add_line_to_list ()`
`aw_get_line ()`
`aw_get_list_line ()`
`aw_get_list_length ()`

aw_get_line ()

Purpose

`*aw_get_line` returns the specified line from the list.

Syntax

```
#include"access.h"
char  *aw_get_line ( qid, lineno )
char  *qid;
int   lineno;
```

Description

This function returns as a string the text contents of the line specified by `lineno` from the specified list. The `QLIST` item is identified by its `QID` specified in the resource file.

The returned string contains the text of the line, exactly as held in the text file identified by the `LISTFILE` attribute of the `QLIST`.

Diagnostics

The function returns a `NULL` pointer on failure.

See Also

`aw_get_list_line ()`
`aw_is_item_bit_set ()`

aw_get_list_file ()

Purpose

`*aw_get_list_file` returns the pathname of the file for a specified list.

Syntax

```
#include"access.h"
char  *aw_get_list_file ( qid )
char  *qid;
```

Description

This function returns the name of the file identified by the `QLIST` item's `LISTFILE` attribute. The `QLIST` item is identified by its `QID` specified in the resource file. The returned string contains the full pathname of the list file.

Diagnostics

The function returns a `NULL` pointer on failure.

See Also

`aw_set_list_file ()`
`aw_get_list_heading ()`

aw_get_list_heading ()

Purpose

`*aw_get_list_heading` returns the heading associated with a list item.

Syntax

```
#include"access.h"
char  *aw_get_list_heading ( qid )
char  *qid;
```

Description

This function returns the heading of the `QLIST` item. The item is identified by its `QID` specified in the resource file.

The heading is read from the file associated with the `QLIST`'s list file, that is, it is the file in the same directory as the list file, but with a suffix `.hdg`.

Diagnostics

The function returns a `NULL` pointer both on failure or if no heading for the item exists.

See Also

`aw_set_list_heading ()`
`aw_get_list_file ()`

aw_get_list_length ()

Purpose

`aw_get_list_length` returns the number of rows in the list.

Syntax

```
int   aw_get_list_length ( qid )
char  *qid;
```

Description

This function returns the number of rows in a list. The `QLIST` item is identified by its `QID` specified in the resource file.

When the list is currently displayed, the returned integer represents the number of rows in the list.

When not currently displayed, the value represents the number of rows when last displayed. If not previously displayed, the function returns 0 (zero).

This procedure is often used in conjunction with the function `aw_is_item_bit_set` to check each row in turn to ascertain whether it is selected.

Diagnostics

The function returns 0 (zero) on failure.

See Also

`aw_is_item_bit_set ()`

aw_get_list_line ()

Purpose

`*aw_get_list_line` returns the currently selected line from the list.

Syntax

```
#include"access.h"
char  *aw_get_list_line ( qid )
char  *qid;
```

Description

This function returns the text string of the currently-selected line in the specified list. The `QLIST` item is identified by its `QID` specified in the resource file.

The returned string contains the text of the line, exactly as held in the text file identified by the `LISTFILE` attribute of the `QLIST`, including any markers.

The row number of the selected line can be obtained via a call to the function `aw_get_item_value()`.

Diagnostics

The function returns a `NULL` pointer on failure.

See Also

`aw_get_item_value ()`
`aw_get_line ()`

aw_get_list_selection ()

Purpose

`*aw_get_list_selection` returns the row numbers that are selected from the list.

Syntax

```
int *aw_get_list_selection (qid)
char *qid;
```

Description

This function gives the row numbers that are selected from the list. It returns an array of integers, each element containing the line number of the selected line.

The last value is 0 (zero).

Please note: Ensure that you free the memory.

Diagnostics

This function returns a 0 (zero) in the first array element if no entries are selected in the list. The row numbers are displayed if an entry is selected in the list or the tree.

The function returns a `NULL` value, if the `QID` is not a `TREE` or a `LIST ID`.

aw_get_list_width ()

Purpose

`aw_get_list_width` returns the width of a list.

Syntax

```
int   aw_get_list_width ( qid )
char  *qid;
```

Description

This function returns the width of the contents from the text file identified by the `LISTFILE` attribute of the `QLIST` item. The `QLIST` item is identified by its `QID` specified in the resource file.

The width returned is the maximum number of characters in any row in the list file. The width is returned whether or not the item is visible.

Diagnostics

The function returns 0 (zero) for an invalid `QID`, or the list width for a valid `QID`.

See Also

`aw_get_item_dimension ()`
`aw_get_list_length ()`

aw_get_scroll_char ()

Purpose

`aw_get_scroll_char` returns the number of characters by which a list was scrolled.

Syntax

```
int   aw_get_scroll_char ( qid )
char  *qid;
```

Description

This function returns the number of characters by which a list was scrolled horizontally. The `QLIST` item is identified by its `QID` specified in the resource file.

The scroll count is returned whether or not the item is visible.

Diagnostics

The function returns 0 (zero) both for an invalid `QID` or for a list that is not scrolled. A positive value indicates the scroll count.

See Also

`aw_get_scroll_line ()`
`aw_scroll_to_char ()`

aw_get_scroll_line ()

Purpose

`aw_get_scroll_line` returns the number of rows by which a list was scrolled.

Syntax

```
int   aw_get_scroll_line ( qid )
char  *qid;
```

Description

This function returns the number of rows by which a list was scrolled. The `QLIST` item is identified by its `QID` specified in the resource file.

The scroll count is returned whether or not the item is visible.

Diagnostics

The function returns 0 (zero) both for an invalid `QID` or for a list that is not scrolled. A positive value indicates the scroll count.

See Also

`aw_get_scroll_char ()`
`aw_scroll_to_line ()`

aw_mark_list_line ()

Purpose

`aw_mark_list_line` displays the specified marker in the list at the specified row and column.

Syntax

```
int   aw_mark_list_line ( qid, row, col, mark )
char  *qid;
int   row, col;
char  *mark;
```

Description

This function writes the characters specified in the `mark` argument into the list at the specified character position column and row-number row. The `QLIST` item is identified by its `QID` specified in the resource file.

A currently displayed list is immediately affected.

The function overwrites any text previously displayed at that position.

Any calls to obtain the contents of the list, for example, `aw_get_list_line()` or `aw_get_line()`, receive the mark characters in the returned string.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_list_line ()`

aw_refresh_list ()

Purpose

`aw_refresh_list` updates the displayed list contents from the list contents held in memory.

Syntax

```
int   aw_refresh_list ( qid )
char  *qid;
```

Description

This function refreshes a `QLIST` item from the list contents held in memory. The item is identified by its `QID` specified in the resource file.

Any previously selected rows are reset.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_update_list ()`
`aw_set_item_value ()`

aw_reset_list ()

Purpose

`aw_reset_list` clears the specified list and associated file.

Syntax

```
int   aw_reset_list ( qid )
char  *qid;
```

Description

This function clears the list contents identified by the `LISTFILE` attribute displayed on the screen, and deletes the disk file associated with the list. The `QLIST` item is identified by its `QID` specified in the resource file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

None.

aw_save_listfile ()

Purpose

`aw_save_listfile` saves the current contents of a list to file.

Syntax

```
int   aw_save_listfile ( qid, filename )
char  *qid, *filename;
```

Description

This function writes the current contents of the specified list to the file. The `QLIST` item is identified by its `QID` specified in the resource file.

The filename argument can contain either a full or relative pathname.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_list_file ()`
`aw_set_list_file ()`

aw_scroll_to_char ()

Purpose

`aw_scroll_to_char` scrolls the list to the specified character number.

Syntax

```
int   aw_scroll_to_char ( qid, col )
char  *qid;
int   col;
```

Description

This function scrolls the list contents horizontally, such that the left hand side of the displayed list starts at the specified character column number col. The `QLIST` item is identified by its `QID` specified in the resource file.

The list is only scrolled horizontally. This call does not affect the vertical scrolling.

Please note: This function counts only columns of characters. It is not concerned with columns of data that may exist in the `LISTFILE`.

For example, if your `LISTFILE` contains 3 columns of data, scrolling to column 3 finds the third character in each row not the third column of data. The left hand side of the redisplayed list now starts at the third column of characters.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_set_item_value ()`
`aw_scroll_to_line ()`

aw_scroll_to_line ()

Purpose

`aw_scroll_to_line` scrolls the list to the specified line number.

Syntax

```
#include"access.h"
int   aw_scroll_to_line ( qid, row )
char  *qid;
int   row;
```

Description

This functions scrolls the list contents vertically, such that the top of the displayed list starts at the line specified by row. The `QLIST` item is identified by its `QID` specified in the resource file.

The list is only scrolled vertically. This call does not affect the horizontal scrolling.

The list name is identified by the `QID` argument.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_set_item_value ()`
`aw_scroll_to_char ()`

aw_set_item_heading ()

Purpose

`aw_set_item_heading` sets the heading of the specified item.

Syntax

```
int   aw_set_item_heading ( qid, heading )
char  *qid, *heading;
```

Description

This function sets or resets the heading text string associated with a `QLIST` item. The header associated with the `QLIST` is a `QPANEL` item identified by its `QID` specified in the resource file.

Please note: This is the `QID` of the scrolling header `QPANEL` item, not of the `QLIST` item.

An `ACTION,HEAD` attribute assigned to a `QPANEL` item identifies it as a panel item configured as scrolling header bar, and the `list_qid` argument to this `ACTION` attribute identifies the `QLIST` item with which it is associated.

If the list header item is currently displayed, it is redrawn immediately.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_list_heading ()`
`aw_set_list_heading ()`
`aw_set_item_value ()`
`aw_set_item_label ()`

aw_set_list_file ()

Purpose

aw_set_list_file resets the name of the file referenced by the LISTFILE of the item.

Syntax

```
int   aw_set_list_file ( qid, listfile )
char  *qid, *listfile;
```

Description

This function resets the name of the file identified by the item's LISTFILE attribute. The item is identified by its QID specified in the resource file.

The LISTFILE specifies a full or relative pathname for the file, which can list the contents of either a QLIST or QMENU item.

If the item is currently displayed, the screen contents are not immediately updated.

To refresh the screen and display the contents of the new LISTFILE, use the function aw_show_item or aw_update_list.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

aw_get_list_file ()
aw_set_item_value ()

aw_set_list_heading ()

Purpose

`aw_set_list_heading` sets or resets the heading of a list item.

Syntax

```
int   aw_set_list_heading ( qid, heading )
char  *qid, *heading;
```

Description

This function sets or resets the heading item associated with a `QLIST` item. The item is identified by its `QID` specified in the resource file.

The text string specified by heading is written to the file associated with the `QLIST`'s list file, that is, it is the file in the same directory as the list file, but with a suffix `.hdg`.

The heading is read from the file whenever the `QLIST` item is updated from its `LISTFILE` file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_list_heading ()`

aw_switch_list_font ()

Purpose

`aw_switch_list_font` sets the font in which a list is displayed.

Syntax

```
int   aw_switch_list_font ( qid, fontname )
char  *qid, *fontname;
```

Description

This function resets the font in which a list displayed in a `QLIST` item is displayed. The item is identified by its `QID` specified in the resource file.

The font in which the list is to be displayed is specified by fontname.

The function sets the `FONT` attribute of the `QLIST` item.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_item_font ()`
`aw_set_item_font ()`

aw_update_list ()

Purpose

`aw_update_list` updates the displayed list contents according to the contents of the file identified by the `LISTFILE` attribute.

Syntax

```
int   aw_update_list ( qid )
char  *qid;
```

Description

This function refreshes a `QLIST` item from the contents of the contents file identified by the `LISTFILE` attribute. The item is identified by its `QID` specified in the resource file.

The contents file is identified by the `LISTFILE` attribute defined for the item in the resource file. This function reads the contents of the file and enters it into the specified `QLIST` item.

Any previously selected rows are reset.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_refresh_list ()`
`aw_set_item_value ()`

Chapter 6

Working with Tree Nodes

This chapter describes functions associated with tree nodes.

- aw_get_node_access ()
- aw_get_node_highlight ()
- aw_get_node_linkcolor ()
- aw_get_node_selected ()
- aw_get_node_textcolor ()
- aw_get_node_visible ()
- aw_is_node_highlighted ()
- aw_is_node_selected ()
- aw_is_node_visible ()
- aw_set_node_access ()
- aw_set_node_highlight ()
- aw_set_node_linkcolor ()
- aw_set_node_selected ()
- aw_set_node_textcolor ()
- aw_set_node_visible ()
- aw_create_cgm_file ()

aw_get_node_access ()

Purpose

`aw_get_node_access` returns the access indicator color of a tree node.

Syntax

```
int   aw_get_node_access ( qid, row )
char  *qid;
int   row;
```

Description

This function returns the color of the access indicator flag for the node of a tree that represents the specified row of a `QLIST` item. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of linked nodes. The access indicator is shown as a small square symbol at the right end of each node.

The color is returned as an integer from 0 through 15, which represents the color allocated from the color palette file.

Diagnostics

The function returns 0 (zero) for an invalid `QID` or when the access indicator is in the default background color. A positive value indicates the access indicator color.

See Also

`aw_set_node_access ()`

aw_get_node_highlight ()

Purpose

`aw_get_node_highlight` returns the highlight color of a tree node.

Syntax

```
int   aw_get_node_highlight ( qid, row )
char  *qid;
int   row;
```

Description

This function returns the color of the highlighting used for the node of a tree that represents the specified row of a `QLIST` item. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of linked nodes, with the highlighting shown as a colored outline around the node.

The color is returned as an integer from 0 through 15, which represents the color allocated from the color palette file.

Diagnostics

The function returns 0 (zero) for an invalid `QID` or when the node highlighting is in the default background color. A positive value indicates the highlight color.

See Also

`aw_set_node_highlight ()`
`aw_is_node_highlighted ()`

aw_get_node_linkcolor ()

Purpose

`aw_get_node_linkcolor` returns the link-line color of a tree node.

Syntax

```
int   aw_get_node_linkcolor ( qid, row )
char  *qid;
int   row;
```

Description

This function returns the color used to show the link between the nodes of a tree, representing the specified row of a `QLIST` item, and its parent node. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of nodes linked by colored lines.

The color is returned as an integer from 0 through 15, which represents the color allocated from the color palette file.

Diagnostics

The function returns 0 (zero) for an invalid `QID` or when a node link is in the default background color. A positive value indicates the link color.

See Also

`aw_set_node_linkcolor ()`

aw_get_node_selected ()

Purpose

`aw_get_node_selected` checks if a specified tree node is selected.

Syntax

```
int   aw_get_node_selected ( qid, row )
char  *qid;
int   row;
```

Description

This function checks if the node of a tree, representing the specified row of a `QLIST` item, is in the selected state. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of linked nodes, with nodes selected shown in the selected color.

The function returns the following values:

```
1     SELECTED, i.e. the node is 'selected'
0     NOT SELECTED
```

Diagnostics

None.

See Also

`aw_is_node_selected ()`
`aw_set_node_selected ()`

aw_get_node_textcolor ()

Purpose

`aw_get_node_textcolor` returns the color of the text of a tree node.

Syntax

```
int   aw_get_node_textcolor ( qid, row )
char  *qid;
int   row;
```

Description

This function returns the color used to display the text of the node of a tree. The `QLIST` item is identified by its `QID` specified in the resource file. The argument row identifies a line in the `LISTFILE` file associated with the `QLIST` which defines the node to be referenced.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

The color is returned as an integer from 0 through 15, which represents the color allocated from the color palette file.

Diagnostics

The function returns 0 (zero) for an invalid `QID` or when a node text is in the default background color. A positive value indicates the node text color.

See Also

`aw_set_node_textcolor ()`

aw_get_node_visible ()

Purpose

`aw_get_node_visible` checks if the specified tree node is visible.

Syntax

```
int   aw_get_node_visible ( qid, row )
char  *qid;
int   row;
```

Description

This function checks if the node of a tree, representing the specified row of a `QLIST` item, is visible on the screen. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute. In this form, the list is displayed in a hierarchical tree format as a series of linked nodes.

The function returns the following values:

```
1     VISIBLE
0     HIDDEN, i.e. the node is not currently displayed
```

Diagnostics

None.

See Also

```
aw_is_node_visible ()
aw_set_node_textcolor ()
aw_set_node_visible ()
```

aw_is_node_highlighted ()

Purpose

`aw_is_node_highlighted` checks if a specified tree node is highlighted.

Syntax

```
int   aw_is_node_highlighted ( qid, row )
char  *qid;
int   row;
```

Description

This function checks if the node of a tree, represented by the specified row of a `QLIST` item, is shown highlighted on the screen. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of linked nodes, with the highlighting shown as a colored outline around the node.

The function returns the following values:

```
1 to 15     HIGHLIGHTED
0           NOT HIGHLIGHTED
```

A positive value represents the number of the color from the color palette file used for the highlighting.

Diagnostics

None.

See Also

`aw_get_node_highlight ()`
`aw_set_node_highlight ()`

aw_is_node_selected ()

Purpose

`aw_is_node_selected` checks if a specified tree node is selected.

Syntax

```
int   aw_is_node_selected ( qid, row )
char  *qid;
int   row;
```

Description

This function checks if the node of a tree, represented by the specified row of a `QLIST` item, is in the selected state. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of linked nodes, with nodes selected shown in a select color.

The function returns the following values:

```
1     SELECTED
0     NOT SELECTED
```

Diagnostics

None.

See Also

`aw_get_node_selected ()`
`aw_set_node_selected ()`

aw_is_node_visible ()

Purpose

`aw_is_node_visible` checks if a specified tree node is visible.

Syntax

```
int   aw_is_node_visible ( qid, row )
char  *qid;
int   row;
```

Description

This function checks if the node of a tree, represented by the specified row of a `QLIST` item, is visible on the screen. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute. In this form, the list is displayed in a hierarchical tree format as a series of linked nodes.

The function returns the following values:

```
1     VISIBLE
0     HIDDEN
```

Diagnostics

None.

See Also

```
aw_get_node_visible ()
aw_set_node_textcolor ()
aw_set_node_visible ()
```

aw_set_node_access ()

Purpose

`aw_set_node_access` sets the access indicator color of a tree node.

Syntax

```
int   aw_set_node_access ( qid, row, color )
char  *qid;
int   row, color;
```

Description

This function sets the color of the access indicator flag for the node of a tree that represents the specified row of a `QLIST` item. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of linked nodes. The access indicator is shown as a small square symbol at the right end of each node.

Specify the color as an integer from 0 through 15 to represent the required color from the color palette file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_node_highlight ()`

aw_set_node_highlight ()

Purpose

`aw_set_node_highlight` highlights a tree node in a specified color.

Syntax

```
int   aw_set_node_highlight ( qid, row, color )
char  *qid;
int   row, color;
```

Description

This function highlights in a specified color the node of a tree representing the specified row of a `QLIST` item. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of linked nodes, with the highlighting shown as a colored outline around the node.

Specify the color as an integer from 0 through 15 to represent the required color from the color palette file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_node_highlight ()`
`aw_is_node_highlighted ()`

aw_set_node_linkcolor ()

Purpose

`aw_set_node_linkcolor` sets the link-line color of a tree node.

Syntax

```
int   aw_set_node_linkcolor ( qid, row, color )
char  *qid;
int   row, color;
```

Description

This function sets the color used to show the link between the node of a tree, representing the specified row of a `QLIST` item, and its parent node. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of nodes linked by colored lines.

In this form, the list is displayed in tree format as a series of rectangular nodes, with links shown between each node and its parent.

Specify the color as an integer from 0 through 15 to represent the required color from the color palette file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_node_linkcolor ()`

aw_set_node_selected ()

Purpose

`aw_set_node_selected` sets a tree node to the selected state.

Syntax

```
int   aw_set_node_selected ( qid, row, state )
char  *qid;
int   row, state;
```

Description

This function sets the node of a tree, represented by the specified row of a `QLIST` item, to the selected or deselected state. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute.

In this form, the list is displayed in a hierarchical tree format as a series of linked nodes, with nodes selected shown in a select color.

Set the state as follows:

```
1     SELECTED
0     NOT SELECTED
```

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_node_selected ()
aw_is_node_selected ()
```

aw_set_node_textcolor ()

Purpose

`aw_set_node_textcolor` sets the color of the text of a tree node.

Syntax

```
int   aw_set_node_textcolor ( qid, row, color)
char  *qid;
int   row;
int   color;
```

Description

This function sets the color used to display the text of the node of a tree. The `QLIST` item is identified by its `QID` specified in the resource file. The argument row identifies a line in the `LISTFILE` file associated with the `QLIST` which defines the node to be referenced.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the QLIST item must be defined with the `STATE,TREE` attribute.

Specify the color as an integer from 0 through 15, which represents the color allocated from the color palette file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_node_textcolor ()`

aw_set_node_visible ()

Purpose

`aw_set_node_visible` sets the visibility state of a tree node.

Syntax

```
int   aw_set_node_visible ( qid, row, state, repaint )
char  *qid;
int   row, state, repaint;
```

Description

This function sets the visibility state for the node of a tree representing the specified row of a `QLIST` item, that is, sets the node visible or hidden on the screen. The `QLIST` item is identified by its `QID` specified in the resource file.

To display the rows of a `QLIST` item as nodes of a tree, its list contents file must be specified in the correct form and the `QLIST` item must be defined with the `STATE,TREE` attribute. In this form, the list is displayed in a hierarchical tree format as a series of linked nodes.

Set the state as follows:

```
1     MAKE NODE VISIBLE
0     HIDE NODE
```

Set the value of repaint as follows:

```
1     REPAINT THE SCREEN
0     DO NOT REPAINT THE SCREEN
```

To perform a series of hide and show operations, set the repaint value to `0` for all except the last in the sequence. For the last operation, set the value of repaint to `1`. Otherwise, unnecessary time is taken to repaint the screen after each operation.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

```
aw_get_node_visible ()
aw_is_node_visible ()
```

aw_create_cgm_file ()

Purpose

`aw_create_cgm_file` creates binary CGM format files of a tree.

Please note: This function is available only on UNIX platforms.

Syntax

```
int   aw_create_cgm_file (tree_qid, cgm_filename, scale, paper_size,
is_all_tiers, tier_from, tier_to, is_multiple_cgm_pages)
char *tree_qid;
char *cgm_filename;
double scale;
char *paper_size;
int is_all_tiers;
int tier_from;
int tier_to;
int is_multiple_cgm_pages;
```

where:

- tree_qid — A character string with a maximum length of 64 characters.
- cgm_filename — A character string with a maximum length of 255 characters, including any file extension. This string also includes the path to the file. If no default directory is provided, an error message is displayed. A file extension is not mandatory.
- scale — Double data type. Setting this to a value less than or equal to 0.0 returns an error message. The drawing size increases proportionately for values greater than 1.0.
- paper_size — Options include A, B, C, D, E, A0, A1, A2, A3, and A4.
- is_all_tiers — Set this parameter to 1 to select all tiers in a tree.
- tier_from — Beginning of the range of tiers in Tree. The root node of a given tree is called the zero tier.
- tier_to — End of the range of tiers in the Tree.
- is_multiple_cgm_pages — Set this to 1 to create multiple CGM pages, of a given paper size, for a large tree. Otherwise, paper sizes are ignored and it creates a single CGM file.

  When the value is set to 1, additional pages created are saved as files in the same directory. These files are named as follows:

<cgmfilename_with_no_fileextension><pagerow_pagecolumn>.<fileextension>

For example, the subsequent pages of the CGM file `mytree.cgm` are named `mytree1_2.cgm, mytree2_3.cgm`, and so on.

Description

This function creates a binary CGM format file that represents the tree as displayed in the application. Hidden elements and other attributes in the tree are not represented in the resulting CGM file.

Diagnostics

The function returns the total number of pages when the CGM binary files are successfully created and nonzero on failure.

See Also

None

Chapter 7

Miscellaneous Functions

This chapter describes functions that do not fit into any other category.

- aw_add_item_to_group ()
- aw_beep ()
- aw_cursor_into_item ()
- aw_del_item_from_group ()
- aw_exit_macro ()
- aw_get_cursor_position ()
- aw_get_global ()
- aw_perl_exec ()
- aw_register_repository ()
- aw_register_repository_latest ()
- aw_search_path ()
- aw_set_global ()
- aw_start_macro ()
- aw_system ()
- aw_to_cadds ()

aw_add_item_to_group ()

Purpose

`aw_add_item_to_group` adds an item into an existing group.

Syntax

```
int   aw_add_item_to_group ( group, qid )
char  *group, *qid;
```

Description

This function adds another item to an existing group. The item to be added is identified by its `QID` specified in the resource file.

The group to which the item is to be added is identified by group which is the `QID` of the `QGROUP` item in the resource file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_del_item_from_group ()`

aw_beep ()

Purpose

`aw_beep` sounds the local PC or workstation bell.

Syntax

```
void  aw_beep (  )
```

Description

This function sounds the bell on the local PC or workstation.

Diagnostics

None.

See Also

None

aw_cursor_into_item ()

Purpose

`aw_cursor_into_item` positions the mouse cursor over the item.

Syntax

```
int   aw_cursor_into_item ( qid )
char  *qid;
```

Description

This function positions the mouse cursor over the specified window item. The item is identified by its `QID` specified in the resource file.

The item must be currently visible. This function has no affect on the position of the text input caret.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_get_cursor_position ()`
`aw_set_item_caret ()`

aw_del_item_from_group ()

Purpose

`aw_del_item_from_group` deletes an item from a group.

Syntax

```
int   aw_del_item_from_group ( group, qid )
char  *group, *qid;
```

Description

This function deletes an item from a defined group of items. The item is identified by its `QID` specified in the resource file.

The group from which the item is to be deleted is identified by group which is the `QID` of the `QGROUP` item in the resource file.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_add_item_to_group ()`

aw_exit_macro ()

Purpose

`aw_exit_macro` returns the application to normal operating mode (exits the macro process). Mouse cursor returns to normal.

Syntax

```
int aw_exit_macro ( )
```

Description

This function exits the launched script and returns the application to normal operating mode.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_start_macro ()`

aw_get_cursor_position ()

Purpose

`aw_get_cursor_position` gets the mouse cursor position.

Syntax

```
int   aw_get_cursor_position ( xpos, ypos )
int   *xpos, *ypos;
```

Description

This function obtains the position of the mouse cursor or pointer.

The position coordinates `xpos` `ypos` returned are the number of pixels from the top left corner of the AccessWare window in which the last event occurred.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_cursor_into_item ()`

aw_get_global ()

Purpose

aw_get_global gets the value of the defined global variable.

Syntax

```
int   aw_get_global ( global, value )
char  *global, *value;
```

Description

This function obtains the value of the global variable defined by global. The value of the global variable is set either by an operating system environment variable or by the file referenced by the GLOBVARS attribute.

The GLOBVARS file is read at system startup and the values stored in memory. aw_get_global first searches this memory for the specified global variable. If it finds no match, it interrogates the operating system for the variable.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

aw_set_global ()

aw_perl_exec ()

Purpose

`aw_perl_exec` executes the specified script.

Syntax

```
int   aw_perl_exec ( char *script )
```

Description

This function enables the application to pass a given `perl` script directly to the operating system. The environment variable `AW_PERL_EXEC` needs to be set to use this. Operating system command formed is:

```
$AW_PERL_EXEC script
```

Diagnostics

The function returns the `status` value passed by the operating system.

See Also

`aw_system ()`

aw_register_repository ()

Purpose

`aw_register_repository` registers a file as a repository other than the `GLOBVARS` files.

Syntax

```
int   aw_register_repository ( char* filename )
```

Description

This function allows you to register a file as a repository of environment variable definitions other than those specified by you in the `GLOBVARS` files.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_register_repository_latest ()`
`aw_get_global ()`
`aw_set_global ()`

aw_register_repository_latest ()

Purpose

`aw_register_repository_latest` registers a file as the latest repository of environment variable definitions.

Syntax

```
int aw_register_repository_latest ( char* filename )
```

Description

This function allows you to register a file as the latest repository of environment variable definitions. It overrides the `GLOBVARS` repository file of environment variables.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_register_repository ()`
`aw_get_global ()`
`aw_set_global ()`

aw_search_path ()

Purpose

`*aw_search_path` finds the pathname of the specified file in the current search path.

Syntax

```
char  *aw_search_path ( file )
char  *file;
```

Description

This function searches through the application's search path to find the first occurrence of the specified file named in the argument. The function returns the full pathname for this file.

For example, if the file is the UNIX list directory command `ls`, the function returns the string `/bin/ls`.

Diagnostics

The function returns a `NULL` pointer on failure when the file does not exist.

See Also

None.

aw_set_global ()

Purpose

`aw_set_global` sets the value of the defined global variable.

Syntax

```
int     aw_set_global ( global, string )
char    *global, *string;
```

Description

This function sets the value of the global variable defined by a string. The set value can be retrieved using the function `aw_get_global`.

Diagnostics

The function returns 0 (zero) on success and non-zero (1) on failure.

See Also

`aw_get_global ()`
`aw_perl_exec ()`

aw_start_macro ()

Purpose

`aw_start_macro` starts a keyword macro process. This function changes mouse cursor to busy mode.

Syntax

```
int aw_start_macro ()
```

Description

This function launches a script in the background of the application.

Diagnostics

The function returns 0 (zero) on success and non-zero on failure.

See Also

`aw_exit_macro ()`

aw_system ()

Purpose

`aw_system` passes a command to the operating system.

Syntax

```
int   aw_system ( char* command )
```

Description

This function enables the application to pass a given `command`, which can be an operating system command or a script, directly to the operating system.

The call does not return until the command or script file has been processed.

Please note: When a script is launched via `aw_system()` or `aw_perl_exec()`, the application will background the spawned process and release control back to the event processor.

Diagnostics

The function returns the `status` value passed by the operating system.

See Also

`aw_perl_exec ()`

aw_to_cadds ()

Purpose

`aw_to_cadds` passes a command to CADDS 5i.

Syntax

```
int   aw_to_cadds ( command )
char  *command;
```

Description

This function enables the application to pass a given `command` directly to the CADDS input stream for processing.

The call does not return until the command has been processed.

Diagnostics

The function returns the `status` value passed by CADDS.

See Also

None.

Appendix A

Procedure Call Summary

This chapter summarizes the functions of the supplied AccessWare procedure calls.

- Initiating and Exiting AccessWare
- Retrieving Item Values and Attributes
- Setting Item Values and Attributes
- Updating Item Appearance
- Procedures Associated with Lists
- Procedures Associated with Tree Nodes
- Miscellaneous Procedures

Initiating and Exiting AccessWare

The functions in this section initiate and exit AccessWare.

Procedure Call	Description
int aw_init_windows (resource_file)	Pass control to the AccessWare environment
int aw_exit_windows (status)	Abort the application where a serious error occurs

Retrieving Item Values and Attributes

The functions in this section retrieve item values and attributes.

Procedure Call	Description
int aw_get_item_position (qid, x, y)	Get the position of an item in pixels
int aw_get_item_xposition (qid)	Return the value of the x location of an item.
int aw_get_item_yposition (qid)	Return the value of the y location of an item.
int aw_get_item_dimension (qid, width, height)	Get the size of an item in pixels
int aw_get_item_width (qid)	Get the width of an item in pixels
int aw_get_item_height (qid)	Get the height of an item in pixels
int aw_get_item_cols (qid)	Return the width of an item defined by the COLS attribute
int aw_get_item_rows (qid)	Return the height of an item as defined by the ROWS attribute
int aw_get_window_position (qid, x, y)	Get the position of a window in pixels
int aw_get_window_dimension (qid, width, height)	Get the size of a window in pixels
int aw_get_window_width (qid)	Return the width of a window
int aw_get_window_height (qid)	Return the height of a window
int aw_get_item_depth (qid)	Return the depth of an item
int aw_get_item_invert (qid)	Return the invert status of an item
int aw_get_item_etch (qid)	Return the etch setting of an item
int aw_get_item_itemcolor (qid)	Return the item foreground color of an item
int aw_get_item_backcolor (qid)	Return the background color of an item
int aw_get_item_textcolor (qid)	Return the text color of an item
int aw_get_item_greycolor (qid)	Return the greyout color of an item
char *aw_get_item_label (qid)	Get the current label string of an item
char *aw_get_item_icon (qid)	Return the name of the icon defined for an item
char *aw_get_item_dicon (qid)	Return the name of the selected icon of an item
char *aw_get_item_font (qid)	Return the name of the text font of an item
char *aw_get_window_font (qid)	Return the name of the text font of a window
int aw_get_item_lowercase (qid)	Return the status of the text case mode of an item
int aw_get_item_uppercase (qid)	Return the status of the text case mode of the item
int aw_get_item_mixedcase (qid)	Return the status of the text case mode of an item
int aw_get_item_integer (qid)	Return the status of integer validation of a text item
int aw_get_item_real (qid)	Return the status of real number validation of a text item
int aw_get_item_readonly (qid)	Return the read/write status of the item
int aw_is_item_readonly (qid)	Check whether an item is in a read only state
int aw_get_text_lowercase (qid)	Return the status of the text case mode of a text item
int aw_get_text_uppercase (qid)	Return the status of the text case mode of a text item
int aw_get_text_mixedcase (qid)	Return the status of the text case mode of a text item

Procedure Call	Description
int aw_get_text_integer (qid)	Return the status of integer validation of a text item
int aw_get_text_real (qid)	Return the status of real number validation of a text item
int aw_get_text_readonly (qid)	Return the read/write status of a text item
int aw_get_item_multiselect (qid)	Return the multiple selection status of an item
int aw_is_item_multiselect (qid)	Return the single/multi mode of the item
int aw_get_item_action (qid)	Return the action associated with an item
int aw_get_item_visible (qid)	Return the visibility setting associated with an item
int aw_is_item_visible (qid)	Check whether an item is visible
int aw_get_item_active (qid)	Return the active status of an item
int aw_is_item_active (qid)	Check whether an item is in an active state
char *aw_get_active_id ( )	Return the name of the item that initiated the current action
char *aw_get_item_caret ( )	Return the name of the item containing the text input caret
char *aw_get_item_value (qid)	Get the current value of an item number of row/option for list/choice items in character format
int aw_get_item_bit (qid, bit)	Get an option setting or a row number
int aw_is_item_bit_set (qid, number)	Check if a row `QLIST` item or option `QCHOICE` item is selected
char *aw_get_item_prompt (qid)	Return the text string for the prompt defined for an item
char *aw_get_item_helpfile (qid)	Return the name of the helpfile referenced by an item
int aw_get_window_stripe (qid)	Return the window stripe setting of an item a window
char *aw_get_window_resource (qid)	Return the name of the resource file of a window
int aw_get_state (char *qid)	Return the value of a control state.
char *aw_get_udata (char *qid)	Return user data.
int aw_get_menuitem_state (char *id)	Gets the state of a menu item.

Setting Item Values and Attributes

The functions in this section set item values and attributes.

Procedure Call	Description
int aw_set_item_position (qid, x, y)	Reset the x and y location of an item.
int aw_set_item_xposition (qid, xvalue)	Reset the x location of an item
int aw_set_item_yposition (qid, yvalue)	Reset the y location of an item
int aw_set_window_position (qid, x, y)	Set the position of a window
int aw_set_item_width (qid, width)	Set the width of an item not a window
int aw_set_item_height (qid, height)	Set the height of an item not a window
int aw_set_item_cols (qid, cols)	Reset the width of an item
int aw_set_item_rows (qid, rows)	Reset the height of an item
int aw_set_window_width (qid, width)	Set the width of a window in pixels
int aw_set_window_height (qid, height)	Set the height of a window in pixels
int aw_set_item_depth (qid, depth)	Reset the depth of an item
int aw_set_item_invert (qid, state)	Set/reset the invert state of an item
int aw_set_item_etch (qid, value)	Set the etch value for an item to a specified number of pixels
int aw_set_item_itemcolor (qid, color)	Set the color of an item
int aw_set_item_backcolor (qid, color)	Set the background color of an item
int aw_set_item_textcolor (qid, color)	Set the text color of an item
int aw_set_item_selectcolor (qid, color)	Set the select color of an item
int aw_set_item_greycolor (qid, color)	Set the item greyout color of an item
int aw_set_item_label (qid, new_label)	Set the label of an item
int aw_set_item_icon (qid, icon_name)	Set the icon associated with an item. The item must be visible and have a STATE, ICONIC attribute for the display to be updated
int aw_set_item_dicon (qid, icon_name)	Specify/respecify the selected icon for an item
int aw_set_item_font (qid, fontname)	Specify/respecify a font associated with an item
int aw_set_item_lowercase (qid)	Set the text case mode of an item to convert input to lower case
int aw_set_item_uppercase (qid)	Set the text case mode of an item to convert input to upper case
int aw_set_item_mixedcase (qid)	Set the text case mode of the item to permit mixed case text
int aw_set_item_integer (qid)	Set/reset the mode to validate input for an integer number
int aw_set_item_real (qid)	Set the MODE to validate input for a real number
int aw_set_item_readonly (qid, state)	Set an item mode to allow or restrict modification of the contents
int aw_set_text_lowercase (qid)	Set the MODE to convert input to lower case characters
int aw_set_text_mixedcase (qid)	Set the MODE to convert input to mixed case characters
int aw_set_text_integer (qid)	Set the MODE to validate input for an integer number
int aw_set_text_real (qid)	Set the MODE to validate input for a real number
int aw_set_text_readonly (qid, state)	Set/reset the read/write state of a text field

Procedure Call	Description
int aw_set_item_multiselect (qid, on/off)	Set a list or choice item to single value or multi value selection
int aw_set_item_action (qid, action_no)	Set the action number of an item
int aw_set_item_visible (qid, state)	Reset the visibility state of an item
int aw_set_item_active (qid, state)	Set/reset the active state of an item
int aw_set_item_caret (qid)	Place the text input caret in a item
int aw_set_item_value (qid, new_value)	Set the value of an item
int aw_set_item_bit (qid, value)	Select a row QLIST item or option QCHOICE item
int aw_reset_item_bit (qid, value)	Deselect a row QLIST item or option QCHOICE item
int aw_set_item_helpfile (qid, filename)	Assign/reassign the associated helpfile for an item
int aw_set_item_prompt (qid, string)	Set/reset the prompt text string for an item
int aw_prompt (prompt_string)	Display a prompt in the prompt field of the current window
int aw_win_prompt (qid, prompt_string)	Display a prompt in the nominated prompt field of the window
int aw_clear_prompt ( )	Remove any prompt text from the currently active prompt field
int aw_message (message_string)	Display a message in the message field of the current window
int aw_win_message (qid, message_string)	Display a message in the nominated message field of a window
int aw_clear_message ( )	Remove any message from the currently active message field
int aw_set_state (qid, state_value)	Sets the state of a menu item.

Updating Item Appearance

The functions in this section update item appearance.

Procedure Call	Description
int aw_activate_item (qid)	Activate an item cancel greyed-out state
int aw_deactivate_item (qid)	Deactivate a item grey-out the item
int aw_show_item (qid)	Show an item an entire window or specified items in a window
int aw_show_item_relative (qid, relative_id, xoffset, yoffset)	Show an item relative to another item
int aw_hide_item (qid)	Hide an item
int aw_hide_all_popups ( )	Hide all popup windows from the screen
int aw_open_window (qid)	Open a window from an icon
int aw_close_window (qid)	Close a window to an icon
int aw_refresh_window (qid)	Force AccessWare to read the resource file for a window and refresh its content.
int aw_set_window_resource (qid, resource_file)	Redefine a window from the contents of a new resource file

Procedures Associated with Lists

The functions in this section control procedures associated with lists.

Procedure Call	Description
int aw_get_list_width (qid)	Get the maximum number of columns in a list contents file
int aw_get_list_length (qid)	Get the number of rows in a list
int aw_mark_list_line (qid, line_no, char_pos, mark)	Write character `mark` at character position `char_pos`
char *aw_get_list_line (qid)	Get the text of the currently selected row of a list item
char *aw_get_line (qid, lineno)	Get the row text at `lineno` of an list item
int aw_get_scroll_char (qid)	Get the number of columns by which the list has been scrolled
int aw_get_scroll_line (qid)	Get the number of rows by which the list has been scrolled
int aw_scroll_to_line (qid, line_no)	Vertically scroll a list item to the given line number
int aw_scroll_to_char (qid, char_no)	Horizontally scroll a list item to the character column number
int aw_add_line_to_list (qid, line, pos)	Add a new line into a list item
int aw_delete_line_from_list (qid, line, pos)	Delete an existing line from a list item at a given row position
char *aw_get_list_file (qid)	Get the name of the file associated with a list
char *aw_get_list_heading (qid)	Return the heading associated with a list item
int aw_set_item_heading (qid, heading)	Set or reset the text for the header item associated with a list item
int aw_set_list_heading (qid, string)	Set/rest the heading of a list item
int aw_refresh_list (qid)	Force an update of a displayed list from contents held in memory
int aw_update_list (qid)	Force AccessWare to reread the listfile associated with a list item
int aw_save_listfile (qid, listfile)	Save the contents of a list in a file.
int aw_reset_list (qid)	Clear the list contents on the screen and delete from the disk the disk file associated with the list
int aw_set_list_file (qid, listfile)	Reset the pathname of the listfile to be used in an item this could be a list contents or a menu contents
int aw_switch_list_font (qid, fontname)	Reset the font in which a list is displayed
int aw_get_list_selection (qid)	Returns the row numbers that are selected from the list.

Procedures Associated with Tree Nodes

The functions in this section control procedures associated with tree nodes.

Procedure Call	Description
int aw_get_node_selected (qid, line_no_of_listfile)	Check if a tree node is selected
int aw_get_node_highlight (qid, line_no_of_listfile)	Get the highlight color of a tree node
int aw_get_node_visible (qid, line_no_of_listfile)	Check if a tree node is visible
int aw_is_node_selected (qid, line_no_of_listfile)	Check if a tree node is selected
int aw_is_node_highlighted (qid, line_no_of_listfile)	Check if a tree node is highlighted
int aw_is_node_visible (qid, line_no_of_listfile)	Check if a tree node is visible
int aw_get_node_textcolor (qid, line_number)	Return the color of the text of a tree node
int aw_get_node_linkcolor (qid, line_no_of_listfile)	Get the link color of a tree node
int aw_get_node_access (qid, line_no_of_listfile)	Get the color of the access indicator of a tree node
int aw_set_node_selected (qid, line_no_of_listfile, sel_state)	Set a tree node to the selected state
int aw_set_node_highlight (qid, line_no_of_listfile, color)	Highlight a tree node in a specified color
int aw_set_node_visible (qid, line_no_of_listfile, state, repaint)	Set a tree node to visible or hidden
int aw_set_node_textcolor (qid, line_no_of_listfile, color)	Set the color of the text of a tree node.
int aw_set_node_linkcolor (qid, line_no_of_listfile, color)	Set the color of the link between a tree node and its parent
int aw_set_node_access (qid, line_no_of_listfile, color)	Set the access indicator color of a tree node
aw_create_cgm_file (tree_qid, cgm_filename, scale, paper_size, all_tiers, tier_from, tier_to, is_multiple_cgm_pages)	Creates a binary CGM format file of a tree.

Miscellaneous Procedures

The functions in this section do not fit into any other category and are therefore placed here.

Procedure Call	Description
int aw_add_item_to_group (group, qid)	Add an item into an existing group
int aw_del_item_from_group (group, qid)	Delete an item from a group
int aw_get_cursor_position (x, y)	Return the position of the mouse cursor from the top left corner of the AccessWare window in which the last event occurred
int aw_cursor_into_item (qid)	Move the mouse cursor into an item
int aw_start_macro ( )	Start a macro process and mouse cursor changes to busy.
int aw_exit_macro ( )	Return the application to normal operating mode. Mouse cursor returns to normal mode.
int aw_get_global (global, string)	Return the expansion of the named global variable
int aw_set_global (global, string)	Sets the value of the defined global variable
char *aw_search_path (file)	Search through the search path to find the first occurrence of a file
int aw_perl_exec ( )	Execute a specified `perl` script.
int aw_system (call)	Pass the string to the native operating system for processing
int aw_to_cadds (cmd)	Pass the string to CADDS for processing
void aw_beep ( )	Sound the bell on the PC or workstation being used
int aw_register_repository (char* filename)	Registers a file as a repository other than the GLOBVARS files
int aw_register_repository_latest (file)	Registers the latest message file.

Appendix B

Function Cross-Reference Table

This chapter of the manual provides an alphabetical list of the AccessWare procedure calls and indicates the item types to which they apply.

- Function Cross-Reference Table

Function Cross-Reference Table

Table B-1 Functions Cross-referenced to Items

	QPANEL	QLIST	QBUTTON	QCHOICE	QMESSAGE	QTEXT	QMENU	QMENUITEM	QCONTROLSTATE	QEXPRESSION	QPOPUP	QGROUP	QTABCONTROL	QSLIDER	QMENUBAR	QSEPARATOR
aw_activate_item	Y	Y	Y	Y	Y	Y						Y	Y	Y		Y
aw_add_item_to_group	Y	Y	Y	Y	Y	Y										
aw_add_line_to_list		Y														
aw_beep																
aw_clear_message																
aw_clear_prompt																
aw_close_window											Y					
aw_create_cgm_file		Y														
aw_cursor_into_item	Y	Y	Y	Y	Y	Y							Y	Y		Y
aw_deactivate_item	Y	Y	Y	Y	Y	Y						Y	Y	Y		Y
aw_delete_line_from_list		Y														
aw_del_item_from_group	Y	Y	Y	Y	Y	Y										
aw_exit_macro																
aw_exit_windows																
*aw_get_active_id																
aw_get_cursor_position																
aw_get_global																
aw_get_item_action	Y	Y	Y	Y	Y	Y	Y	Y					Y	Y		Y
aw_get_item_active	Y	Y	Y	Y	Y	Y	Y						Y	Y		Y
aw_get_item_backcolor	Y	Y	Y	Y	Y	Y	Y						Y	Y		Y
aw_get_item_bit		Y		Y												
*aw_get_item_caret						Y										
aw_get_item_cols	Y	Y	Y	Y	Y	Y	Y									
aw_get_item_depth	Y	Y	Y	Y		Y	Y									
*aw_get_item_dicon			Y													
aw_get_item_dimension	Y	Y	Y	Y	Y	Y							Y	Y	Y	Y
aw_get_item_etch	Y	Y	Y													

Table B-1 Functions Cross-referenced to Items

	QPANEL	QLIST	QBUTTON	QCHOICE	QMESSAGE	QTEXT	QMENU	QMENUITEM	QCONTROLSTATE	QEXPRESSION	QPOPUP	QGROUP	QTABCONTROL	QSLIDER	QMENUBAR	QSEPARATOR
*aw_get_item_font	Y	Y	Y	Y	Y	Y	Y						Y	Y		Y
aw_get_item_greycolor	Y	Y	Y	Y	Y	Y										
aw_get_item_height	Y	Y	Y	Y	Y								Y	Y		Y
*aw_get_item_helpfile	Y	Y	Y	Y	Y	Y							Y	Y		Y
*aw_get_item_icon			Y													
aw_get_item_integer						Y										
aw_get_item_invert	Y	Y	Y			Y										Y
aw_get_item_itemcolor	Y	Y	Y	Y	Y	Y	Y									
*aw_get_item_label	Y		Y	Y	Y	Y		Y			Y		Y	Y		
aw_get_item_lowercase						Y										
aw_get_item_mixedcase						Y										
aw_get_item_multiselect		Y		Y												
aw_get_item_position	Y	Y	Y	Y	Y	Y							Y	Y		Y
*aw_get_item_prompt	Y	Y	Y	Y	Y	Y							Y	Y		Y
aw_get_item_readonly						Y										
aw_get_item_real						Y										
aw_get_item_rows	Y	Y				Y										
aw_get_item_textcolor	Y	Y	Y	Y	Y	Y	Y									
aw_get_item_uppercase						Y										
*aw_get_item_value	Y		Y	Y		Y			Y	Y			Y	Y		Y
aw_get_item_visible	Y	Y	Y	Y	Y	Y					Y		Y	Y		Y
aw_get_item_width	Y	Y	Y	Y	Y	Y	Y						Y	Y		Y
aw_get_item_xposition	Y	Y	Y	Y	Y	Y							Y	Y		Y
aw_get_item_yposition	Y	Y	Y	Y	Y	Y							Y	Y		Y
*aw_get_line		Y					Y									
*aw_get_list_file		Y					Y								Y	
*aw_get_list_heading		Y														
aw_get_list_length		Y					Y									
*aw_get_list_line		Y					Y									

Table B-1 Functions Cross-referenced to Items

	QPANEL	QLIST	QBUTTON	QCHOICE	QMESSAGE	QTEXT	QMENU	QMENUITEM	QCONTROLSTATE	QEXPRESSION	QPOPUP	QGROUP	QTABCONTROL	QSLIDER	QMENUBAR	QSEPARATOR
aw_get_list_width		Y					Y									
*aw_get_list_selection		Y														
aw_get_menuitem_state								Y								
aw_get_node_access		Y														
aw_get_node_highlight		Y														
aw_get_node_linkcolor		Y														
aw_get_node_selected		Y														
aw_get_node_textcolor		Y														
aw_get_node_visible		Y														
aw_get_scroll_char		Y														
aw_get_scroll_line		Y														
aw_get_state									Y							
aw_get_text_integer						Y										
aw_get_text_lowercase						Y										
aw_get_text_mixedcase						Y										
aw_get_text_readonly						Y										
aw_get_text_real						Y										
aw_get_text_uppercase						Y										
*aw_get_udata								Y								
aw_get_window_dimension											Y					
*aw_get_window_font											Y					
aw_get_window_height											Y					
aw_get_window_position											Y					
*aw_get_window_resource											Y					
aw_get_window_stripe											Y					
aw_get_window_width											Y					
aw_hide_all_popups											Y		Y	Y		Y
aw_hide_item	Y	Y	Y	Y	Y		Y				Y	Y	Y	Y		Y
aw_init_windows																

Table B-1 Functions Cross-referenced to Items

	QPANEL	QLIST	QBUTTON	QCHOICE	QMESSAGE	QTEXT	QMENU	QMENUITEM	QCONTROLSTATE	QEXPRESSION	QPOPUP	QGROUP	QTABCONTROL	QSLIDER	QMENUBAR	QSEPARATOR
aw_is_item_active	Y	Y	Y	Y	Y	Y	Y						Y	Y		Y
aw_is_item_bit_set		Y		Y												
aw_is_item_multiselect		Y		Y		Y										
aw_is_item_readonly						Y										
aw_is_item_visible	Y	Y	Y	Y	Y	Y					Y		Y	Y	Y	Y
aw_is_node_highlighted		Y														
aw_is_node_selected		Y														
aw_is_node_visible		Y														
aw_mark_list_line		Y														
aw_message																
aw_open_window											Y					
aw_perl_exec																
aw_prompt																
aw_refresh_list		Y														
aw_refresh_window											Y					
aw_register_repository																
aw_register_repository_latest																
aw_reset_item_bit		Y		Y												
aw_reset_list		Y														
aw_save_listfile		Y														
aw_scroll_to_char		Y														
aw_scroll_to_line		Y														
*aw_search_path																
aw_set_global																
aw_set_item_action	Y	Y	Y	Y	Y	Y		Y			Y		Y	Y		Y
aw_set_item_active	Y	Y	Y	Y	Y	Y	Y						Y	Y		Y
aw_set_item_backcolor	Y	Y	Y	Y	Y	Y	Y									
aw_set_item_bit		Y		Y												
aw_set_item_caret						Y										

Table B-1 Functions Cross-referenced to Items

	QPANEL	QLIST	QBUTTON	QCHOICE	QMESSAGE	QTEXT	QMENU	QMENUITEM	QCONTROLSTATE	QEXPRESSION	QPOPUP	QGROUP	QTABCONTROL	QSLIDER	QMENUBAR	QSEPARATOR
aw_set_item_cols	Y	Y	Y	Y	Y	Y	Y									
aw_set_item_depth	Y	Y	Y	Y		Y	Y									
aw_set_item_dicon			Y										Y	Y		Y
aw_set_item_etch	Y	Y	Y										Y	Y		Y
aw_set_item_font	Y	Y	Y	Y	Y	Y	Y									
aw_set_item_greycolor	Y	Y	Y	Y	Y	Y							Y	Y		Y
aw_set_item_heading		Y											Y	Y		Y
aw_set_item_height	Y	Y	Y	Y		Y										
aw_set_item_helpfile	Y	Y	Y	Y	Y	Y										
aw_set_item_icon			Y													
aw_set_item_integer						Y										
aw_set_item_invert	Y	Y	Y			Y							Y	Y		Y
aw_set_item_itemcolor	Y	Y	Y	Y	Y	Y	Y									
aw_set_item_label	Y		Y	Y	Y	Y					Y		Y	Y		Y
aw_set_item_lowercase						Y										
aw_set_item_mixedcase						Y										
aw_set_item_multiselect		Y		Y												
aw_set_item_position	Y	Y	Y	Y	Y	Y							Y	Y		Y
aw_set_item_prompt	Y	Y	Y	Y	Y	Y							Y	Y		Y
aw_set_item_readonly						Y							Y	Y		Y
aw_set_item_real						Y							Y	Y		Y
aw_set_item_rows	Y	Y				Y										
aw_set_item_selectcolor		Y	Y	Y			Y						Y	Y		Y
aw_set_item_textcolor	Y	Y	Y	Y	Y	Y	Y						Y	Y		Y
aw_set_item_uppercase						Y							Y	Y		Y
aw_set_item_value	Y		Y	Y		Y							Y	Y		Y
aw_set_item_visible	Y	Y	Y	Y	Y	Y					Y		Y	Y		Y
aw_set_item_width	Y	Y	Y	Y		Y										
aw_set_item_xposition	Y	Y	Y	Y	Y	Y										

Table B-1 Functions Cross-referenced to Items

	QPANEL	QLIST	QBUTTON	QCHOICE	QMESSAGE	QTEXT	QMENU	QMENUITEM	QCONTROLSTATE	QEXPRESSION	QPOPUP	QGROUP	QTABCONTROL	QSLIDER	QMENUBAR	QSEPARATOR
aw_set_item_yposition	Y	Y	Y	Y	Y	Y										
aw_set_list_file		Y					Y								Y	
aw_set_list_heading		Y														
aw_set_node_access		Y														
aw_set_node_highlight		Y														
aw_set_node_linkcolor		Y														
aw_set_node_selected		Y														
aw_set_node_textcolor		Y														
aw_set_node_visible		Y														
aw_set_state									Y							
aw_set_text_integer						Y										
aw_set_text_lowercase						Y										
aw_set_text_mixedcase						Y										
aw_set_text_readonly						Y										
aw_set_text_real						Y										
aw_set_window_height											Y					
aw_set_window_position											Y					
aw_set_window_resource											Y					
aw_set_window_width											Y					
aw_show_item	Y	Y	Y	Y	Y	Y					Y	Y	Y	Y		Y
aw_show_item_relative	Y		Y	Y	Y	Y					Y		Y	Y		Y
aw_start_macro																
aw_switch_list_font		Y														
aw_system																
aw_to_cadds																
aw_update_list		Y														
aw_win_message											Y					
aw_win_prompt											Y					

Please note that functions not cross-referenced to any item are for general usage in the application.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Y