

# AccessWare Function Reference

---

Optegra® Release 6

DOC40183-007

---

**Copyright © 2001 Parametric Technology Corporation. All Rights Reserved.**

User documentation from Parametric Technology Corporation (PTC) is subject to copyright laws of the United States and other countries and is provided under a license agreement, which restricts copying, disclosure, and use of such documentation. PTC hereby grants to the licensed user the right to make copies in printed form of PTC user documentation provided on software or documentation media, but only for internal, noncommercial use by the licensed user in accordance with the license agreement under which the applicable software and documentation are licensed. Any copy made hereunder shall include the Parametric Technology Corporation copyright notice and any other proprietary notice provided by PTC. User documentation may not be disclosed, transferred, or modified without the prior written consent of PTC and no authorization is granted to make copies for such purposes.

Information described in this document is furnished for general information only, is subject to change without notice, and should not be construed as a warranty or commitment by PTC. PTC assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

The software described in this document is provided under written license agreement, contains valuable trade secrets and proprietary information, and is protected by the copyright laws of the United States and other countries. UNAUTHORIZED USE OF SOFTWARE OR ITS DOCUMENTATION CAN RESULT IN CIVIL DAMAGES AND CRIMINAL PROSECUTION.

**Registered Trademarks of Parametric Technology Corporation or a Subsidiary**

Advanced Surface Design, CADD5, CADDShade, Computervision, Computervision Services, Electronic Product Definition, EPD, HARNESSDESIGN, Info\*Engine, InPart, MEDUSA, Optegra, Parametric Technology, Parametric Technology Corporation, Pro/ENGINEER, Pro/HELP, Pro/INTRALINK, Pro/MECHANICA, Pro/TOOLKIT, PTC, PT/Products, Windchill, InPart logo, and PTC logo.

**Trademarks of Parametric Technology Corporation or a Subsidiary**

3DPAINT, Associative Topology Bus, Behavioral Modeler, BOMBOT, CDRS, CounterPart, CV, CVact, CVaec, CVdesign, CV-DORS, CVMAC, CVNC, CVToolmaker, DesignSuite, DIMENSION III, DIVISION, DVSAFEWORK, DVS, e-Series, EDE, e/ENGINEER, Electrical Design Entry, Expert Machinist, Expert Toolmaker, Flexible Engineering, *i*-Series, ICEM, Import Data Doctor, Information for Innovation, ISSM, MEDEA, ModelCHECK, NC Builder, Nitidus, PARTBOT, PartSpeak, Pro/ANIMATE, Pro/ASSEMBLY, Pro/CABLING, Pro/CASTING, Pro/CDT, Pro/CMM, Pro/COMPOSITE, Pro/CONVERT, Pro/DATA for PDGS, Pro/DESIGNER, Pro/DESKTOP, Pro/DETAIL, Pro/DIAGRAM, Pro/DIEFACE, Pro/DRAW, Pro/ECAD, Pro/ENGINE, Pro/FEATURE, Pro/FEM-POST, Pro/FLY-THROUGH, Pro/HARNESS-MFG, Pro/INTERFACE for CADD5, Pro/INTERFACE for CATIA, Pro/LANGUAGE, Pro/LEGACY, Pro/LIBRARYACCESS, Pro/MESH, Pro/Model.View, Pro/MOLDESIGN, Pro/NC-ADVANCED, Pro/NC-CHECK, Pro/NC-MILL, Pro/NC-SHEETMETAL, Pro/NC-TURN, Pro/NC-WEDM, Pro/NC-Wire EDM, Pro/NCPOST, Pro/NETWORK ANIMATOR, Pro/NOTEBOOK, Pro/PDM, Pro/PHOTORENDER, Pro/PHOTORENDER TEXTURE LIBRARY, Pro/PIPING, Pro/PLASTIC ADVISOR, Pro/PLOT, Pro/POWER DESIGN, Pro/PROCESS, Pro/REPORT, Pro/REVIEW, Pro/SCAN-TOOLS, Pro/SHEETMETAL, Pro/SURFACE, Pro/VERIFY, Pro/Web.Link, Pro/Web.Publish, Pro/WELDING, Product Structure Navigator, PTC *i*-Series, Shaping Innovation, Shrinkwrap, The Product Development Company, Virtual Design Environment, Windchill e-Series, CV-Computervision logo, DIVISION logo, and ICEM logo.

**Third-Party Trademarks**

Oracle is a registered trademark of Oracle Corporation. Windows and Windows NT are registered trademarks of Microsoft Corporation. Java and all Java based marks are trademarks or registered trademarks of Sun Microsystems, Inc. CATIA is a registered trademark of Dassault Systems. PDGS is a registered trademark of Ford Motor Company. SAP and R/3 are registered trademarks of SAP AG Germany. FLEX/m is a registered trademark of GLOBEtrouter Software, Inc. VisTools library is copyrighted software of Visual Kinematics, Inc. (VKI) containing confidential trade secret information belonging to VKI. HOOPS graphics system is a proprietary software product of, and copyrighted by, Tech Soft America, Inc. All other brand or product names are trademarks or registered trademarks of their respective holders.

---

---

**UNITED STATES GOVERNMENT RESTRICTED RIGHTS LEGEND**

This document and the software described herein are Commercial Computer Documentation and Software, pursuant to FAR 12.212(a)-(b) or DFARS 227.7202-1(a) and 227.7202-3(a), and are provided to the Government under a limited commercial license only. For procurements predating the above clauses, use, duplication, or disclosure by the Government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or Commercial Computer Software-Restricted Rights at FAR 52.227-19, as applicable.

**Parametric Technology Corporation, 140 Kendrick Street, Needham, MA 02494-2714**

**8 January 2001**

---



---

# Table of Contents

---

## Preface

Related Documents .....	xxxv
Book Conventions .....	xxxvi
Online User Documentation .....	xxxvii
Printing Documentation .....	xxxvii
Resources and Services .....	xxxviii
Documentation Comments .....	xxxviii

## Introduction to AccessWare Procedure Calls

Introduction .....	1-2
Using Libraries .....	1-3
aw_init_windows () .....	1-4
Purpose .....	1-4
Syntax .....	1-4
Description .....	1-4
Diagnostics .....	1-4
See Also .....	1-4
aw_exit_windows () .....	1-5
Purpose .....	1-5
Syntax .....	1-5
Description .....	1-5
Diagnostics .....	1-5
See Also .....	1-5

---

## Retrieving Item Values and Attributes

aw_get_active_id ()	2-4
Purpose	2-4
Syntax	2-4
Description	2-4
Diagnostics	2-4
See Also	2-4
aw_get_item_action ()	2-5
Purpose	2-5
Syntax	2-5
Description	2-5
Diagnostics	2-5
See Also	2-5
aw_get_item_active ()	2-6
Purpose	2-6
Syntax	2-6
Description	2-6
Diagnostics	2-6
See Also	2-6
aw_get_item_backcolor ()	2-7
Purpose	2-7
Syntax	2-7
Description	2-7
Diagnostics	2-7
See Also	2-7
aw_get_item_bit ()	2-8
Purpose	2-8
Syntax	2-8
Description	2-8
Diagnostics	2-8
See Also	2-8

---

aw_get_item_caret ()	2-9
Purpose	2-9
Syntax	2-9
Description	2-9
Diagnostics	2-9
See Also	2-9
aw_get_item_cols ()	2-10
Purpose	2-10
Syntax	2-10
Description	2-10
Diagnostics	2-10
See Also	2-10
aw_get_item_depth ()	2-11
Purpose	2-11
Syntax	2-11
Description	2-11
Diagnostics	2-11
See Also	2-11
aw_get_item_dicon ()	2-12
Purpose	2-12
Syntax	2-12
Description	2-12
Diagnostics	2-12
See Also	2-12
aw_get_item_dimension ()	2-13
Purpose	2-13
Syntax	2-13
Description	2-13
Diagnostics	2-13
See Also	2-13
aw_get_item_etch ()	2-14
Purpose	2-14
Syntax	2-14
Description	2-14
Diagnostics	2-14
See Also	2-14

---

aw_get_item_font ()	2-15
Purpose	2-15
Syntax	2-15
Description	2-15
Diagnostics	2-15
See Also	2-15
aw_get_item_greycolor ()	2-16
Purpose	2-16
Syntax	2-16
Description	2-16
Diagnostics	2-16
See Also	2-16
aw_get_item_height ()	2-17
Purpose	2-17
Syntax	2-17
Description	2-17
Diagnostics	2-17
See Also	2-17
aw_get_item_helpfile ()	2-18
Purpose	2-18
Syntax	2-18
Description	2-18
Diagnostics	2-18
See Also	2-18
aw_get_item_icon ()	2-19
Purpose	2-19
Syntax	2-19
Description	2-19
Diagnostics	2-19
See Also	2-19
aw_get_item_integer ()	2-20
Purpose	2-20
Syntax	2-20
Description	2-20
Diagnostics	2-20
See Also	2-20



---

aw_get_item_invert () _____	2-21
Purpose _____	2-21
Syntax _____	2-21
Description _____	2-21
Diagnostics _____	2-21
See Also _____	2-21
aw_get_item_itemcolor () _____	2-22
Purpose _____	2-22
Syntax _____	2-22
Description _____	2-22
Diagnostics _____	2-22
See Also _____	2-22
aw_get_item_label () _____	2-23
Purpose _____	2-23
Syntax _____	2-23
Description _____	2-23
Diagnostics _____	2-23
See Also _____	2-23
aw_get_item_lowercase () _____	2-24
Purpose _____	2-24
Syntax _____	2-24
Description _____	2-24
Diagnostics _____	2-24
See Also _____	2-24
aw_get_item_mixedcase () _____	2-25
Purpose _____	2-25
Syntax _____	2-25
Description _____	2-25
Diagnostics _____	2-25
See Also _____	2-25
aw_get_item_multiselect () _____	2-26
Purpose _____	2-26
Syntax _____	2-26
Description _____	2-26
Diagnostics _____	2-26
See Also _____	2-26

---

aw_get_item_position () _____	2-27
Purpose _____	2-27
Syntax _____	2-27
Description _____	2-27
Diagnostics _____	2-27
See Also _____	2-27
aw_get_item_prompt () _____	2-28
Purpose _____	2-28
Syntax _____	2-28
Description _____	2-28
Diagnostics _____	2-28
See Also _____	2-28
aw_get_item_readonly () _____	2-29
Purpose _____	2-29
Syntax _____	2-29
Description _____	2-29
Diagnostics _____	2-29
See Also _____	2-29
aw_get_item_real () _____	2-30
Purpose _____	2-30
Syntax _____	2-30
Description _____	2-30
Diagnostics _____	2-30
See Also _____	2-30
aw_get_item_rows () _____	2-31
Purpose _____	2-31
Syntax _____	2-31
Description _____	2-31
Diagnostics _____	2-31
See Also _____	2-31
aw_get_item_textcolor () _____	2-32
Purpose _____	2-32
Syntax _____	2-32
Description _____	2-32
Diagnostics _____	2-32
See Also _____	2-32

---

aw_get_item_uppercase () _____	2-33
Purpose _____	2-33
Syntax _____	2-33
Description _____	2-33
Diagnostics _____	2-33
See Also _____	2-33
aw_get_item_value () _____	2-34
Purpose _____	2-34
Syntax _____	2-34
Description _____	2-34
Diagnostics _____	2-34
See Also _____	2-34
aw_get_item_visible () _____	2-35
Purpose _____	2-35
Syntax _____	2-35
Description _____	2-35
Diagnostics _____	2-35
See Also _____	2-35
aw_get_item_width () _____	2-36
Purpose _____	2-36
Syntax _____	2-36
Description _____	2-36
Diagnostics _____	2-36
See Also _____	2-36
aw_get_item_xposition () _____	2-37
Purpose _____	2-37
Syntax _____	2-37
Description _____	2-37
Diagnostics _____	2-37
See Also _____	2-37
aw_get_item_yposition () _____	2-38
Purpose _____	2-38
Syntax _____	2-38
Description _____	2-38
Diagnostics _____	2-38
See Also _____	2-38

---

aw_get_menuitem_state ()	2-39
Purpose	2-39
Syntax	2-39
Description	2-39
Diagnostics	2-39
See Also	2-39
aw_get_state ()	2-40
Purpose	2-40
Syntax	2-40
Description	2-40
Diagnostics	2-40
See Also	2-40
aw_get_text_integer ()	2-41
Purpose	2-41
Syntax	2-41
Description	2-41
Diagnostics	2-41
See Also	2-41
aw_get_text_lowercase ()	2-42
Purpose	2-42
Syntax	2-42
Description	2-42
Diagnostics	2-42
See Also	2-42
aw_get_text_mixedcase ()	2-43
Purpose	2-43
Syntax	2-43
Description	2-43
Diagnostics	2-43
See Also	2-43
aw_get_text_readonly ()	2-44
Purpose	2-44
Syntax	2-44
Description	2-44
Diagnostics	2-44
See Also	2-44

---

aw_get_text_real ()	2-45
Purpose	2-45
Syntax	2-45
Description	2-45
Diagnostics	2-45
See Also	2-45
aw_get_text_uppercase ()	2-46
Purpose	2-46
Syntax	2-46
Description	2-46
Diagnostics	2-46
See Also	2-46
aw_get_udata ()	2-47
Purpose	2-47
Syntax	2-47
Description	2-47
Diagnostics	2-47
See Also	2-47
aw_get_window_dimension ()	2-48
Purpose	2-48
Syntax	2-48
Description	2-48
Diagnostics	2-48
See Also	2-48
aw_get_window_font ()	2-49
Purpose	2-49
Syntax	2-49
Description	2-49
Diagnostics	2-49
See Also	2-49
aw_get_window_height ()	2-50
Purpose	2-50
Syntax	2-50
Description	2-50
Diagnostics	2-50
See Also	2-50

---

aw_get_window_position ()	2-51
Purpose	2-51
Syntax	2-51
Description	2-51
Diagnostics	2-51
See Also	2-51
aw_get_window_resource ()	2-52
Purpose	2-52
Syntax	2-52
Description	2-52
Diagnostics	2-52
See Also	2-52
aw_get_window_stripe ()	2-53
Purpose	2-53
Syntax	2-53
Description	2-53
Diagnostics	2-53
See Also	2-53
aw_get_window_width ()	2-54
Purpose	2-54
Syntax	2-54
Description	2-54
Diagnostics	2-54
See Also	2-54
aw_is_item_active ()	2-55
Purpose	2-55
Syntax	2-55
Description	2-55
Diagnostics	2-55
See Also	2-55
aw_is_item_bit_set ()	2-56
Purpose	2-56
Syntax	2-56
Description	2-56
Usage	2-56
Diagnostics	2-57
See Also	2-57

---

aw_is_item_multiselect () _____	2-58
Purpose _____	2-58
Syntax _____	2-58
Description _____	2-58
Diagnostics _____	2-58
See Also _____	2-58
aw_is_item_readonly () _____	2-59
Purpose _____	2-59
Syntax _____	2-59
Description _____	2-59
Diagnostics _____	2-59
See Also _____	2-59
aw_is_item_visible () _____	2-60
Purpose _____	2-60
Syntax _____	2-60
Description _____	2-60
Diagnostics _____	2-60
See Also _____	2-60

## Setting Item Values and Attributes

aw_clear_prompt () _____	3-3
Purpose _____	3-3
Syntax _____	3-3
Description _____	3-3
Diagnostics _____	3-3
See Also _____	3-3
aw_clear_message () _____	3-4
Purpose _____	3-4
Syntax _____	3-4
Description _____	3-4
Diagnostics _____	3-4
See Also _____	3-4

---

aw_message ()	3-5
Purpose	3-5
Syntax	3-5
Description	3-5
Diagnostics	3-5
See Also	3-5
aw_prompt ()	3-6
Purpose	3-6
Syntax	3-6
Description	3-6
Diagnostics	3-6
See Also	3-6
aw_set_item_action ()	3-7
Purpose	3-7
Syntax	3-7
Description	3-7
Diagnostics	3-7
See Also	3-7
aw_set_item_active ()	3-8
Purpose	3-8
Syntax	3-8
Description	3-8
Diagnostics	3-8
See Also	3-8
aw_set_item_backcolor ()	3-9
Purpose	3-9
Syntax	3-9
Description	3-9
Diagnostics	3-9
See Also	3-9
aw_set_item_bit ()	3-10
Purpose	3-10
Syntax	3-10
Description	3-10
Diagnostics	3-10
See Also	3-10



---

aw_set_item_caret () _____	3-11
Purpose _____	3-11
Syntax _____	3-11
Description _____	3-11
Diagnostics _____	3-11
See Also _____	3-11
aw_set_item_cols () _____	3-12
Purpose _____	3-12
Syntax _____	3-12
Description _____	3-12
Diagnostics _____	3-12
See Also _____	3-12
aw_set_item_depth () _____	3-13
Purpose _____	3-13
Syntax _____	3-13
Description _____	3-13
Diagnostics _____	3-13
See Also _____	3-13
aw_set_item_dicon () _____	3-14
Purpose _____	3-14
Syntax _____	3-14
Description _____	3-14
Diagnostics _____	3-14
See Also _____	3-14
aw_set_item_etch () _____	3-15
Purpose _____	3-15
Syntax _____	3-15
Description _____	3-15
Diagnostics _____	3-15
See Also _____	3-15
aw_set_item_font () _____	3-16
Purpose _____	3-16
Syntax _____	3-16
Description _____	3-16
Diagnostics _____	3-16
See Also _____	3-16

---

aw_set_item_greycolor () _____	3-17
Purpose _____	3-17
Syntax _____	3-17
Description _____	3-17
Diagnostics _____	3-17
See Also _____	3-17
aw_set_item_height () _____	3-18
Purpose _____	3-18
Syntax _____	3-18
Description _____	3-18
Diagnostics _____	3-18
See Also _____	3-18
aw_set_item_helpfile () _____	3-19
Purpose _____	3-19
Syntax _____	3-19
Description _____	3-19
Diagnostics _____	3-19
See Also _____	3-19
aw_set_item_icon () _____	3-20
Purpose _____	3-20
Syntax _____	3-20
Description _____	3-20
Diagnostics _____	3-20
See Also _____	3-20
aw_set_item_integer () _____	3-21
Purpose _____	3-21
Syntax _____	3-21
Description _____	3-21
Diagnostics _____	3-21
See Also _____	3-21
aw_set_item_invert () _____	3-22
Purpose _____	3-22
Syntax _____	3-22
Description _____	3-22
Diagnostics _____	3-22
See Also _____	3-22

---

aw_set_item_itemcolor () _____	3-23
Purpose _____	3-23
Syntax _____	3-23
Description _____	3-23
Diagnostics _____	3-23
See Also _____	3-23
aw_set_item_label () _____	3-24
Purpose _____	3-24
Syntax _____	3-24
Description _____	3-24
Diagnostics _____	3-24
See Also _____	3-24
aw_set_item_lowercase () _____	3-25
Purpose _____	3-25
Syntax _____	3-25
Description _____	3-25
Diagnostics _____	3-25
See Also _____	3-25
aw_set_item_mixedcase () _____	3-26
Purpose _____	3-26
Syntax _____	3-26
Description _____	3-26
Diagnostics _____	3-26
See Also _____	3-26
aw_set_item_multiselect () _____	3-27
Purpose _____	3-27
Syntax _____	3-27
Description _____	3-27
Diagnostics _____	3-27
See Also _____	3-27
aw_set_item_position () _____	3-28
Purpose _____	3-28
Syntax _____	3-28
Description _____	3-28
Diagnostics _____	3-28
See Also _____	3-28

---

aw_set_item_prompt () _____	3-29
Purpose _____	3-29
Syntax _____	3-29
Description _____	3-29
Diagnostics _____	3-29
See Also _____	3-29
aw_set_item_readonly () _____	3-30
Purpose _____	3-30
Syntax _____	3-30
Description _____	3-30
Diagnostics _____	3-30
See Also _____	3-30
aw_set_item_real () _____	3-31
Purpose _____	3-31
Syntax _____	3-31
Description _____	3-31
Diagnostics _____	3-31
See Also _____	3-31
aw_set_item_rows () _____	3-32
Purpose _____	3-32
Syntax _____	3-32
Description _____	3-32
Diagnostics _____	3-32
See Also _____	3-32
aw_set_item_selectcolor () _____	3-33
Purpose _____	3-33
Syntax _____	3-33
Description _____	3-33
Diagnostics _____	3-33
See Also _____	3-33
aw_set_item_textcolor () _____	3-34
Purpose _____	3-34
Syntax _____	3-34
Description _____	3-34
Diagnostics _____	3-34
See Also _____	3-34

---

aw_set_item_uppercase ()	3-35
Purpose	3-35
Syntax	3-35
Description	3-35
Diagnostics	3-35
See Also	3-35
aw_set_item_value ()	3-36
Purpose	3-36
Syntax	3-36
Description	3-36
Diagnostics	3-36
See Also	3-36
aw_set_item_visible ()	3-37
Purpose	3-37
Syntax	3-37
Description	3-37
Diagnostics	3-37
See Also	3-37
aw_set_item_width ()	3-38
Purpose	3-38
Syntax	3-38
Description	3-38
Diagnostics	3-38
See Also	3-38
aw_set_item_xposition ()	3-39
Purpose	3-39
Syntax	3-39
Description	3-39
Diagnostics	3-39
See Also	3-39
aw_set_item_yposition ()	3-40
Purpose	3-40
Syntax	3-40
Description	3-40
Diagnostics	3-40
See Also	3-40

---

aw_reset_item_bit ()	3-41
Purpose	3-41
Syntax	3-41
Description	3-41
Diagnostics	3-41
See Also	3-41
aw_set_state ()	3-42
Purpose	3-42
Syntax	3-42
Description	3-42
Diagnostics	3-42
See Also	3-42
aw_set_text_integer ()	3-43
Purpose	3-43
Syntax	3-43
Description	3-43
Diagnostics	3-43
See Also	3-43
aw_set_text_lowercase ()	3-44
Purpose	3-44
Syntax	3-44
Description	3-44
Diagnostics	3-44
See Also	3-44
aw_set_text_mixedcase ()	3-45
Purpose	3-45
Syntax	3-45
Description	3-45
Diagnostics	3-45
See Also	3-45
aw_set_text_readonly ()	3-46
Purpose	3-46
Syntax	3-46
Description	3-46
Diagnostics	3-46
See Also	3-46

aw_set_text_real () _____	3-47
Purpose _____	3-47
Syntax _____	3-47
Description _____	3-47
Diagnostics _____	3-47
See Also _____	3-47
aw_win_message () _____	3-48
Purpose _____	3-48
Syntax _____	3-48
Description _____	3-48
Diagnostics _____	3-48
See Also _____	3-48
aw_win_prompt () _____	3-49
Purpose _____	3-49
Syntax _____	3-49
Description _____	3-49
Diagnostics _____	3-49
See Also _____	3-49
aw_set_window_height () _____	3-50
Purpose _____	3-50
Syntax _____	3-50
Description _____	3-50
Diagnostics _____	3-50
See Also _____	3-50
aw_set_window_position () _____	3-51
Purpose _____	3-51
Syntax _____	3-51
Description _____	3-51
Diagnostics _____	3-51
See Also _____	3-51
aw_set_window_width () _____	3-52
Purpose _____	3-52
Syntax _____	3-52
Description _____	3-52
Diagnostics _____	3-52
See Also _____	3-52

---

## Updating Item Appearance

aw_activate_item ()	4-2
Purpose	4-2
Syntax	4-2
Description	4-2
Diagnostics	4-2
See Also	4-2
aw_deactivate_item ()	4-3
Purpose	4-3
Syntax	4-3
Description	4-3
Diagnostics	4-3
See Also	4-3
aw_show_item ()	4-4
Purpose	4-4
Syntax	4-4
Description	4-4
Diagnostics	4-4
See Also	4-4
aw_show_item_relative ()	4-5
Purpose	4-5
Syntax	4-5
Description	4-5
Diagnostics	4-5
See Also	4-5
aw_hide_all_popups ()	4-6
Purpose	4-6
Syntax	4-6
Description	4-6
Diagnostics	4-6
See Also	4-6



---

aw_hide_item ()	4-7
Purpose	4-7
Syntax	4-7
Description	4-7
Diagnostics	4-7
See Also	4-7
aw_open_window ()	4-8
Purpose	4-8
Syntax	4-8
Description	4-8
Diagnostics	4-8
See Also	4-8
aw_close_window ()	4-9
Purpose	4-9
Syntax	4-9
Description	4-9
Diagnostics	4-9
See Also	4-9
aw_refresh_window ()	4-10
Purpose	4-10
Syntax	4-10
Description	4-10
Diagnostics	4-10
See Also	4-10
aw_set_window_resource ()	4-11
Purpose	4-11
Syntax	4-11
Description	4-11
Diagnostics	4-11
See Also	4-11

---

## Working with Lists

aw_add_line_to_list ()	5-3
Purpose	5-3
Syntax	5-3
Description	5-3
Diagnostics	5-3
See Also	5-3
aw_delete_line_from_list ()	5-4
Purpose	5-4
Syntax	5-4
Description	5-4
Diagnostics	5-4
See Also	5-4
aw_get_line ()	5-5
Purpose	5-5
Syntax	5-5
Description	5-5
Diagnostics	5-5
See Also	5-5
aw_get_list_file ()	5-6
Purpose	5-6
Syntax	5-6
Description	5-6
Diagnostics	5-6
See Also	5-6
aw_get_list_heading ()	5-7
Purpose	5-7
Syntax	5-7
Description	5-7
Diagnostics	5-7
See Also	5-7

---

aw_get_list_length () _____	5-8
Purpose _____	5-8
Syntax _____	5-8
Description _____	5-8
Diagnostics _____	5-8
See Also _____	5-8
aw_get_list_line () _____	5-9
Purpose _____	5-9
Syntax _____	5-9
Description _____	5-9
Diagnostics _____	5-9
See Also _____	5-9
aw_get_list_selection () _____	5-10
Purpose _____	5-10
Syntax _____	5-10
Description _____	5-10
Diagnostics _____	5-10
aw_get_list_width () _____	5-11
Purpose _____	5-11
Syntax _____	5-11
Description _____	5-11
Diagnostics _____	5-11
See Also _____	5-11
aw_get_scroll_char () _____	5-12
Purpose _____	5-12
Syntax _____	5-12
Description _____	5-12
Diagnostics _____	5-12
See Also _____	5-12
aw_get_scroll_line () _____	5-13
Purpose _____	5-13
Syntax _____	5-13
Description _____	5-13
Diagnostics _____	5-13
See Also _____	5-13

---

aw_mark_list_line ()	5-14
Purpose	5-14
Syntax	5-14
Description	5-14
Diagnostics	5-14
See Also	5-14
aw_refresh_list ()	5-15
Purpose	5-15
Syntax	5-15
Description	5-15
Diagnostics	5-15
See Also	5-15
aw_reset_list ()	5-16
Purpose	5-16
Syntax	5-16
Description	5-16
Diagnostics	5-16
See Also	5-16
aw_save_listfile ()	5-17
Purpose	5-17
Syntax	5-17
Description	5-17
Diagnostics	5-17
See Also	5-17
aw_scroll_to_char ()	5-18
Purpose	5-18
Syntax	5-18
Description	5-18
Diagnostics	5-18
See Also	5-18
aw_scroll_to_line ()	5-19
Purpose	5-19
Syntax	5-19
Description	5-19
Diagnostics	5-19
See Also	5-19

---

aw_set_item_heading ()	5-20
Purpose	5-20
Syntax	5-20
Description	5-20
Diagnostics	5-20
See Also	5-20
aw_set_list_file ()	5-21
Purpose	5-21
Syntax	5-21
Description	5-21
Diagnostics	5-21
See Also	5-21
aw_set_list_heading ()	5-22
Purpose	5-22
Syntax	5-22
Description	5-22
Diagnostics	5-22
See Also	5-22
aw_switch_list_font ()	5-23
Purpose	5-23
Syntax	5-23
Description	5-23
Diagnostics	5-23
See Also	5-23
aw_update_list ()	5-24
Purpose	5-24
Syntax	5-24
Description	5-24
Diagnostics	5-24
See Also	5-24

---

## Working with Tree Nodes

aw_get_node_access ()	6-2
Purpose	6-2
Syntax	6-2
Description	6-2
Diagnostics	6-2
See Also	6-2
aw_get_node_highlight ()	6-3
Purpose	6-3
Syntax	6-3
Description	6-3
Diagnostics	6-3
See Also	6-3
aw_get_node_linkcolor ()	6-4
Purpose	6-4
Syntax	6-4
Description	6-4
Diagnostics	6-4
See Also	6-4
aw_get_node_selected ()	6-5
Purpose	6-5
Syntax	6-5
Description	6-5
Diagnostics	6-5
See Also	6-5
aw_get_node_textcolor ()	6-6
Purpose	6-6
Syntax	6-6
Description	6-6
Diagnostics	6-6
See Also	6-6

---

aw_get_node_visible ()	6-7
Purpose	6-7
Syntax	6-7
Description	6-7
Diagnostics	6-7
See Also	6-7
aw_is_node_highlighted ()	6-8
Purpose	6-8
Syntax	6-8
Description	6-8
Diagnostics	6-8
See Also	6-8
aw_is_node_selected ()	6-9
Purpose	6-9
Syntax	6-9
Description	6-9
Diagnostics	6-9
See Also	6-9
aw_is_node_visible ()	6-10
Purpose	6-10
Syntax	6-10
Description	6-10
Diagnostics	6-10
See Also	6-10
aw_set_node_access ()	6-11
Purpose	6-11
Syntax	6-11
Description	6-11
Diagnostics	6-11
See Also	6-11
aw_set_node_highlight ()	6-12
Purpose	6-12
Syntax	6-12
Description	6-12
Diagnostics	6-12
See Also	6-12

---

aw_set_node_linkcolor ()	6-13
Purpose	6-13
Syntax	6-13
Description	6-13
Diagnostics	6-13
See Also	6-13
aw_set_node_selected ()	6-14
Purpose	6-14
Syntax	6-14
Description	6-14
Diagnostics	6-14
See Also	6-14
aw_set_node_textcolor ()	6-15
Purpose	6-15
Syntax	6-15
Description	6-15
Diagnostics	6-15
See Also	6-15
aw_set_node_visible ()	6-16
Purpose	6-16
Syntax	6-16
Description	6-16
Diagnostics	6-16
See Also	6-16
aw_create_cgm_file ()	6-17
Purpose	6-17
Syntax	6-17
Description	6-18
Diagnostics	6-18
See Also	6-18



---

## Miscellaneous Functions

aw_add_item_to_group ()	7-2
Purpose	7-2
Syntax	7-2
Description	7-2
Diagnostics	7-2
See Also	7-2
aw_bEEP ()	7-3
Purpose	7-3
Syntax	7-3
Description	7-3
Diagnostics	7-3
See Also	7-3
aw_cursor_into_item ()	7-4
Purpose	7-4
Syntax	7-4
Description	7-4
Diagnostics	7-4
See Also	7-4
aw_del_item_from_group ()	7-5
Purpose	7-5
Syntax	7-5
Description	7-5
Diagnostics	7-5
See Also	7-5
aw_exit_macro ()	7-6
Purpose	7-6
Syntax	7-6
Description	7-6
Diagnostics	7-6
See Also	7-6

---

aw_get_cursor_position ()	7-7
Purpose	7-7
Syntax	7-7
Description	7-7
Diagnostics	7-7
See Also	7-7
aw_get_global ()	7-8
Purpose	7-8
Syntax	7-8
Description	7-8
Diagnostics	7-8
See Also	7-8
aw_perl_exec ()	7-9
Purpose	7-9
Syntax	7-9
Description	7-9
Diagnostics	7-9
See Also	7-9
aw_register_repository ()	7-10
Purpose	7-10
Syntax	7-10
Description	7-10
Diagnostics	7-10
See Also	7-10
aw_register_repository_latest ()	7-11
Purpose	7-11
Syntax	7-11
Description	7-11
Diagnostics	7-11
See Also	7-11
aw_search_path ()	7-12
Purpose	7-12
Syntax	7-12
Description	7-12
Diagnostics	7-12
See Also	7-12

---

aw_set_global () _____	7-13
Purpose _____	7-13
Syntax _____	7-13
Description _____	7-13
Diagnostics _____	7-13
See Also _____	7-13
aw_start_macro () _____	7-14
Purpose _____	7-14
Syntax _____	7-14
Description _____	7-14
Diagnostics _____	7-14
See Also _____	7-14
aw_system () _____	7-15
Purpose _____	7-15
Syntax _____	7-15
Description _____	7-15
Diagnostics _____	7-15
See Also _____	7-15
aw_to_cadds () _____	7-16
Purpose _____	7-16
Syntax _____	7-16
Description _____	7-16
Diagnostics _____	7-16
See Also _____	7-16

## Procedure Call Summary

Initiating and Exiting AccessWare _____	A-2
Retrieving Item Values and Attributes _____	A-3
Setting Item Values and Attributes _____	A-5
Updating Item Appearance _____	A-7
Procedures Associated with Lists _____	A-8
Procedures Associated with Tree Nodes _____	A-9
Miscellaneous Procedures _____	A-10

---

# Function Cross-Reference Table

Function Cross-Reference Table \_\_\_\_\_ B-2

---

# 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> <b>plot_config</b>	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 <b>filename</b> 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 <i>n</i> .
Real number	x	Supply a real number for <i>x</i> .
#	# 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](mailto:doc-webhelp@ptc.com).
- Fill out and mail the PTC Documentation Survey located in the *PTC Customer Service Guide*.



# 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:

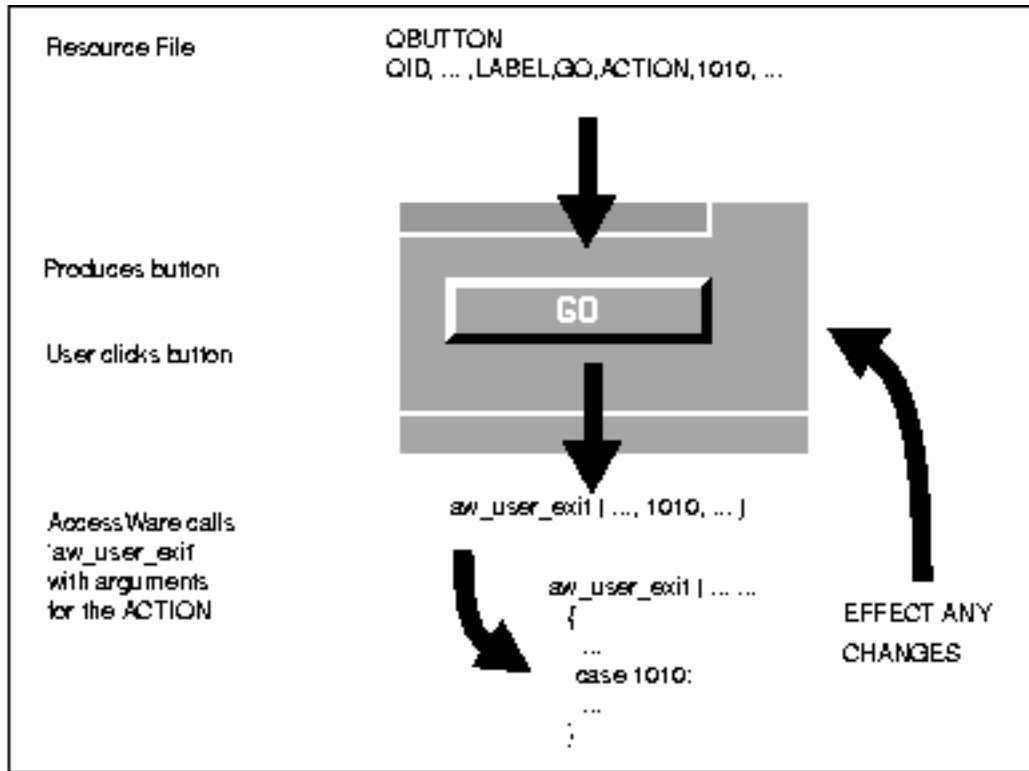
- AccessWare calls the procedure `aw_user_exit` and passes the `ACTION` number, an item identifier, the item value, and the mouse state to it as arguments.
- The `aw_user_exit` procedure comprises a switch statement that contains a list of case statements, each identifying by a label the appropriate 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 ()



# 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_adata ()`
- `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 QPOPOP 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\_ufile ()

# 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 QPOPOPUP 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 or 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 ()
```

# 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 `Q_PANEL` 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\_uadata ()

## 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 `Q_PANEL` 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 ()
```

# Updating Item Appearance

---

This chapter describes 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 ()`



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 ()
```

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

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 CADD5 5i.

### Syntax

```
int    aw_to_cadds ( command )  
char  *command;
```

### Description

This function enables the application to pass a given `command` directly to the CADD5 input stream for processing.

The call does not return until the command has been processed.

### Diagnostics

The function returns the `status` value passed by CADD5.

### See Also

None.

# 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.

<b>Procedure Call</b>	<b>Description</b>
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

<b>Procedure Call</b>	<b>Description</b>
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

<b>Procedure Call</b>	<b>Description</b>
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.

<b>Procedure Call</b>	<b>Description</b>
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.

<b>Procedure Call</b>	<b>Description</b>
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 <code>mark</code> at character position <code>char_pos</code>
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 <code>lineno</code> 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/reset 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.

<b>Procedure Call</b>	<b>Description</b>
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 <code>perl</code> 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.

# 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**

	Q P A N E L	Q L I S T	Q B U T T O N	Q C H O I C E	Q M E S S A G E	Q T E X T	Q M E N U	Q M E N U I T E M	Q C O N T R O L S T A T E	Q E X P R E S S I O N	Q P O P U P	Q G R O U P	Q T A B C O N T R O L	Q S L I D E R	Q M E N U B A R	Q S E P A R A T O R
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**

	Q P A N E L	Q L I S T	Q B U T T O N	Q C H O I C E	Q M E S S A G E	Q T E X T	Q M E N U	Q M E N U I T E M	Q C O N T R O L S T A T E	Q E X P R E S S I O N	Q P O P U P	Q G R O U P	Q T A B C O N T R O L	Q S L I D E R	Q M E N U B A R	Q S E P A R A T O R
*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**

	Q P A N E L	Q L I S T	Q B U T T O N	Q C H O I C E	Q M E S S A G E	Q T E X T	Q M E N U	Q M E N U I T E M	Q C O N T R O L S T A T E	Q E X P R E S S I O N	Q P O P U P	Q G R O U P	Q T A B C O N T R O L	Q S L I D E R	Q M E N U B A R	Q S E P A R A T O R
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**

	Q P A N E L	Q L I S T	Q B U T T O N	Q C H O I C E	Q M E S S A G E	Q T E X T	Q M E N U	Q M E N U I T E M	Q C O N T R O L S T A T E	Q E X P R E S S I O N	Q P O P U P	Q G R O U P	Q T A B C O N T R O L	Q S L I D E R	Q M E N U B A R	Q S E P A R A T O R
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**

	Q P A N E L	Q L I S T	Q B U T T O N	Q C H O I C E	Q M E S S A G E	Q T E X T	Q M E N U	Q M E N U I T E M	Q C O N T R O L S T A T E	Q E X P R E S S I O N	Q P O P U P	Q G R O U P	Q T A B C O N T R O L	Q S L I D E R	Q M E N U B A R	Q S E P A R A T O R
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**

	Q P A N E L	Q L I S T	Q B U T T O N	Q C H O I C E	Q M E S S A G E	Q T E X T	Q M E N U	Q M E N U I T E M	Q C O N T R O L S T A T E	Q E X P R E S S I O N	Q P O P U P	Q G R O U P	Q T A B C O N T R O L	Q S L I D E R	Q M E N U B A R	Q S E P A R A T O R
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

- access indicator color of tree node,  
    returning 6-2
- AccessWare Programmatic Interface 1-2
- action associated with item, returning 2-5
- action for item, setting 3-7
- activating an item 4-2
- active status of item, returning 2-6, 2-55
- adding item to existing group 7-2
- adding row to existing list 5-3
- allocating a new resource file to the window 4-11
- Applying a function 1-2
- assigning help file for an item 3-19
- aw\_activate\_item function 4-2
- aw\_add\_item\_to\_group function 7-2
- aw\_add\_line\_to\_list function 5-3, 7-3
- aw\_beep function 7-3
- aw\_clear\_message function 3-4
- aw\_clear\_prompt function 3-3, 4-9
- aw\_close\_window function 4-9
- aw\_create\_cgm\_file function 6-17
- aw\_cursor\_into\_item function 7-4
- aw\_del\_item\_from\_group function 7-6
- aw\_delete\_line\_from\_list function 5-4
- aw\_exit\_macro function 7-6
- aw\_exit\_windows function 1-5
- aw\_get\_active\_id function 2-4
- aw\_get\_cursor\_position function 7-7
- aw\_get\_global function 7-8
- aw\_get\_item\_action function 2-5
- aw\_get\_item\_active function 2-6
- aw\_get\_item\_backcolor function 2-7
- aw\_get\_item\_bit function 2-8
- aw\_get\_item\_caret function 2-9
- aw\_get\_item\_cols function 2-10
- aw\_get\_item\_depth function 2-11
- aw\_get\_item\_dicon function 2-12
- aw\_get\_item\_dimension function 2-13
- aw\_get\_item\_etch function 2-14
- aw\_get\_item\_font function 2-15
- aw\_get\_item\_greycolor function 2-16
- aw\_get\_item\_height function 2-17
- aw\_get\_item\_helpfile function 2-18
- aw\_get\_item\_icon function 2-19
- aw\_get\_item\_integer function 2-20
- aw\_get\_item\_invert function 2-21
- aw\_get\_item\_itemcolor function 2-22
- aw\_get\_item\_label function 2-23
- aw\_get\_item\_lowercase function 2-24
- aw\_get\_item\_mixedcase function 2-25
- aw\_get\_item\_multiselect function 2-26
- aw\_get\_item\_position function 2-27
- aw\_get\_item\_prompt function 2-28
- aw\_get\_item\_readonly function 2-29
- aw\_get\_item\_real function 2-30
- aw\_get\_item\_rows function 2-31
- aw\_get\_item\_textcolor function 2-32
- aw\_get\_item\_uppercase function 2-33
- aw\_get\_item\_value function 2-34
- aw\_get\_item\_visible function 2-35
- aw\_get\_item\_width function 2-36
- aw\_get\_item\_xposition function 2-37
- aw\_get\_item\_yposition function 2-38
- aw\_get\_line function 5-5
- aw\_get\_list\_file function 5-6
- aw\_get\_list\_heading function 5-7
- aw\_get\_list\_length function 5-8
- aw\_get\_list\_line function 5-9
- aw\_get\_list\_width function 5-11

aw\_get\_menuitem\_state function 2-39  
aw\_get\_node\_access function 6-2, 6-3  
aw\_get\_node\_highlight function 6-3  
aw\_get\_node\_linkcolor function 6-4  
aw\_get\_node\_selected function 6-5  
aw\_get\_node\_textcolor function 6-6  
aw\_get\_node\_visible function 6-7  
aw\_get\_scroll\_char function 5-12  
aw\_get\_scroll\_line function 5-13  
aw\_get\_state function 2-40  
aw\_get\_text\_integer function 2-41  
aw\_get\_text\_lowercase function 2-42  
aw\_get\_text\_mixedcase function 2-43  
aw\_get\_text\_readonly function 2-44  
aw\_get\_text\_real function 2-45  
aw\_get\_text\_uppercase function 2-46  
aw\_get\_udata function 2-47  
aw\_get\_window\_dimension function 2-48  
aw\_get\_window\_font function 2-49  
aw\_get\_window\_height function 2-50  
aw\_get\_window\_position function 2-51  
aw\_get\_window\_resource function 2-52  
aw\_get\_window\_stripe function 2-53  
aw\_get\_window\_width function 2-54  
aw\_hide\_all\_popups function 4-6  
aw\_hide\_item function 4-7  
aw\_init\_windows function 1-4, 1-5  
aw\_is\_item\_active function 2-55  
aw\_is\_item\_bit\_set function 2-56  
aw\_is\_item\_multiselect function 2-58  
aw\_is\_item\_readonly function 2-59  
aw\_is\_item\_visible function 2-60  
aw\_is\_node\_highlighted function 6-8  
aw\_is\_node\_selected function 6-9  
aw\_is\_node\_visible function 6-10  
aw\_mark\_list\_line function 5-14, 7-9  
aw\_message function 3-5, 4-8  
aw\_open\_window function 4-8  
aw\_perl\_exec function 7-9  
aw\_prompt function 3-6, 5-15  
aw\_refresh\_list function 5-15  
aw\_refresh\_window function 7-10  
aw\_register\_repository function 7-10  
aw\_reset\_item\_bit function 5-16  
aw\_reset\_list function 5-16  
aw\_save\_listfile 5-17  
aw\_save\_listfile function 7-12  
aw\_scroll\_to\_char function 5-18, 7-12  
aw\_scroll\_to\_line function 5-19  
aw\_search\_path function 7-12  
aw\_set\_global function 7-13  
aw\_set\_item\_action function 3-50  
aw\_set\_item\_active function 3-8  
aw\_set\_item\_backcolor function 3-9  
aw\_set\_item\_bit function 3-10  
aw\_set\_item\_caret function 3-11  
aw\_set\_item\_cols function 3-12  
aw\_set\_item\_depth function 3-13  
aw\_set\_item\_dicon function 3-14  
aw\_set\_item\_etch function 3-15  
aw\_set\_item\_font function 3-16  
aw\_set\_item\_greycolor function 3-17  
aw\_set\_item\_heading function 5-20  
aw\_set\_item\_height function 3-18  
aw\_set\_item\_helpfile function 3-19  
aw\_set\_item\_icon function 3-20  
aw\_set\_item\_integer function 3-21  
aw\_set\_item\_invert function 3-22  
aw\_set\_item\_itemcolor function 3-23  
aw\_set\_item\_label function 3-24  
aw\_set\_item\_lowercase function 3-25  
aw\_set\_item\_mixedcase function 3-26  
aw\_set\_item\_multiselect function 3-27  
aw\_set\_item\_position function 3-28  
aw\_set\_item\_prompt function 3-29  
aw\_set\_item\_readonly function 3-30  
aw\_set\_item\_real function 3-31  
aw\_set\_item\_rows function 3-32  
aw\_set\_item\_selectcolor function 3-33  
aw\_set\_item\_textcolor function 3-34  
aw\_set\_item\_uppercase function 3-35  
aw\_set\_item\_value function 3-36  
aw\_set\_item\_visible function 3-37  
aw\_set\_item\_width function 3-38  
aw\_set\_item\_xposition function 3-39  
aw\_set\_item\_yposition function 3-40  
aw\_set\_list\_file function 5-21  
aw\_set\_list\_heading function 5-22  
aw\_set\_node\_access function 6-11  
aw\_set\_node\_highlight function 6-12  
aw\_set\_node\_linkcolor function 6-13  
aw\_set\_node\_selected function 6-14  
aw\_set\_node\_visible function 6-15, 6-16  
aw\_set\_state function 3-42  
aw\_set\_text\_lowercase function 3-44  
aw\_set\_text\_mixedcase function 3-45  
aw\_set\_text\_readonly function 3-46  
aw\_set\_text\_real function 3-47  
aw\_set\_window\_height function 3-50  
aw\_set\_window\_position function 3-51



aw\_set\_window\_resource function 4-11  
 aw\_set\_window\_width function 3-52  
 aw\_show\_item function 7-14  
 aw\_show\_item\_relative function 4-5  
 aw\_start\_macro function 7-14  
 aw\_switch\_list\_font function 5-23  
 aw\_system function 7-15  
 aw\_to\_cadds function 7-16  
 aw\_update\_list function 5-24  
 aw\_win\_message function 3-48  
 aw\_win\_prompt function 3-49

## B

background color of item, returning 2-7  
 bell, sounding 7-3

## C

CADDS, passing a command to 7-16  
 CGM file, creating 6-17, A-9  
 checking if specified option or row is selected 2-56  
 checking if specified tree node is highlighted 6-8  
 checking if specified tree node is selected 6-5, 6-9  
 checking if specified tree node is visible 6-7, 6-10  
 clearing current message field 3-4  
 clearing current prompt field 3-3  
 clearing specified list and associated file 5-16  
 closing a window 4-9  
 color of specified item, setting 3-23  
 command, passing to CADDS 7-16  
 command, passing to the operating system 7-15  
 creating a CGM file 6-17, A-9

## D

deactivating item 4-3  
 defining label of an item 3-24  
 deleting item from a group 7-5  
 deleting row from existing list 5-4  
 depth of an item, returning 2-11  
 dimension of an item, returning 2-13  
 displaying a message to the user 3-5, 3-48

displaying a prompt to the user 3-6, 3-49  
 displaying specified marker in list at specified row and column 5-14  
 Documentation, printing from Portable Document Format (PDF) file xxxvii

## E

etch of the specified item, setting 3-15  
 etch setting of item, returning 2-14  
 executing a perl script 7-9  
 exiting 1-5  
 exiting a macro script 7-6

## F

file referenced by LISTFILE, resetting name 5-21  
 file, finding the pathname 7-12  
 finding the pathname of the specified file 7-12  
 font associated with an item, specifying 3-16  
 foreground color of item, returning 2-22  
 functions, applying 1-2

## G

getting menu item state 2-39  
 getting mouse cursor position 7-7  
 getting option row number 2-8  
 getting option setting 2-8  
 getting the position of the window 2-51  
 getting the size of the window 2-48  
 getting value of global variable 7-8  
 global variable, getting value 7-8  
 global variable, setting 7-13  
 grey-out color of item, returning 2-16  
 greycolor of item, setting 3-17

## H

heading associated with list item, returning 5-7  
 height of a window, returning 2-50  
 height of an item, resetting 3-32  
 height of an item, returning 2-17  
 height of item defined by ROWS attribute,

- returning 2-31
- help file for an item, assigning 3-19
- helpfile referenced by item, returning name 2-18
- hiding all popup windows 4-6
- hiding an item on the screen 4-7
- highlight color of tree node, returning 6-3
- highlighting a tree node in a specified color 6-12

## I

- icon for an item, specifying 3-14
- icon for item, setting 3-20
- Initiating and Exiting Access Ware A-2
- input for a real number, setting MODE to validate 3-47
- input for an integer number, setting MODE to validate 3-43
- input of real, setting MODE to validate 3-31
- input to lowercase, setting MODE to convert 3-44
- input to mixed case, setting the MODE to convert 3-45
- input to uppercase, setting MODE to convert 3-35
- input validation for integer 3-21
- input, converting to lowercase 3-25
- input, converting to mixed case 3-26
- invert state of an item, setting 3-22
- invert status of item, returning 2-21
- item active state, setting 3-8
- item background color, setting 3-9
- item depth, resetting 3-13
- item heading, setting 5-20
- item height, setting 3-18
- item initiating current action, returning 2-4
- item label, defining 3-24
- item on the screen, hiding 4-7
- item prompt text string, returning 2-28
- item width, resetting 3-12
- item width, setting 3-38
- item, activating 4-2
- item, adding to existing group 7-2
- item, deactivating 4-3
- item, deleting from a group 7-5
- item, showing on the screen 4-4
- items, showing on screen relative to each other 4-5

## L

- LABEL of an item, returning 2-23
- libraries, linking 1-3
  - UNIX 1-3
  - Windows 95 1-3
  - Windows NT 1-3
- line selected from list, returning 5-9
- linking to libraries 1-3
  - UNIX 1-3
  - Windows 95 1-3
  - Windows NT 1-3
- link-line color of tree node, returning 6-4
- list and associated file, clearing 5-16
- list contents, saving to file 5-17
- list contents, updating from memory 5-15
- list contents, updating with contents of file identified by LISTFILE attribute 5-24
- list font, setting 5-23
- list item heading, setting 5-22
- list scrolling, returning characters 5-12
- list, returning pathname of file 5-6
- list, scrolling to specified character number 5-18
- list, scrolling to the specified line number 5-19

## M

- macro script, exiting 7-6
- macro script, starting 7-14
- marker in list, displaying 5-14
- menu item state, getting 2-39
- menu item state, setting 3-42
- message field, clearing the current 3-4
- message, displaying to the user 3-48
- message, displaying to user 3-5
- minimizing a window 4-9
- Miscellaneous Procedures A-10
- mouse cursor position, getting 7-7
- mouse cursor, positioning over item 7-4
- multiple selection status of item, returning 2-26

## N

name of icon defined by DICON attribute, returning 2-12  
 name of icon defined for item, returning 2-19  
 name of the item with text input caret, returning 2-9  
 number of rows in list, returning 5-8

## O

opening the window from an icon 4-8  
 operating system, passing a command to 7-15  
 option setting, getting 2-8  
 option, checking if specified is selected 2-56  
 option, resetting 3-41

## P

passing a command to CADD5 7-16  
 passing a command to the operating system 7-15  
 passing control to AccessWare from main procedure 1-4  
 perl script, executing 7-9  
 placing text input caret in item 3-11  
 popup windows, hiding all 4-6  
 position of item, returning 2-27  
 position of specified window, setting 3-51  
 positioning mouse cursor over item 7-4  
 Printing documentation from Portable Document Format (PDF) file xxxvii  
 Procedures Associated with Lists A-8  
 Procedures Associated with Tree Nodes A-9  
 Programmatic Interface 1-2  
 prompt field, clearing current 3-3  
 prompt text string, resetting 3-29  
 prompt, displaying to the user 3-6, 3-49

## R

read/write status of item, returning 2-29, 2-59  
 readonly state, setting item 3-30  
 real number validation of text item, returning status 2-30

registering an additional repository 7-10, 7-11  
 repository, registering 7-10, 7-11  
 resetting name of the file referenced by LISTFILE 5-21  
 resetting the depth of an item 3-13  
 resetting the height of an item 3-32  
 resetting the prompt text string for an item 3-29  
 resetting the specified option or row 3-41  
 resetting the width of an item 3-12  
 resetting the x and y location of an item 3-28  
 resetting the x location of an item 3-39  
 resetting the y location of an item 3-40  
 resetting visibility state of an item 3-37  
 resource file for window, allocating new 4-11  
 resource file of window, returning name 2-52  
 Retrieving Item Values and Attributes A-3  
 returning access indicator color of tree node 6-2  
 returning action associated with item 2-5  
 returning active status of item 2-6, 2-55  
 returning background color of item 2-7  
 returning characters by which list scrolled 5-12  
 returning color of text of tree node 6-6  
 returning currently selected line from list 5-9  
 returning depth of an item 2-11  
 returning dimension of an item 2-13  
 returning etch setting of item 2-14  
 returning foreground color of item 2-22  
 returning grey-out color of item 2-16  
 returning heading associated with list item 5-7  
 returning height of an item 2-17  
 returning highlight color of tree node 6-3  
 returning invert status of item 2-21  
 returning item initiating current action 2-4  
 returning LABEL of an item 2-23  
 returning link-line color of tree node 6-4  
 returning multiple selection status an item 2-26  
 returning name of helpfile referenced by item 2-18  
 returning name of icon defined by DICON attribute 2-12  
 returning name of icon defined for item 2-19  
 returning name of resource file of window 2-52  
 returning name of text font of window 2-49  
 returning name of the item with text input caret 2-9  
 returning number of rows in list 5-8  
 returning pathname of file for specified list 5-6  
 returning position of item 2-27  
 returning read/write status of item 2-29, 2-59

returning read/write status of text item 2-44  
returning rows by which a list scrolled 5-13  
returning specified line from list 5-5  
returning status of integer validation 2-20, 2-41  
returning status of real number validation of text item 2-30, 2-45  
returning status of text case 2-24, 2-25, 2-33  
returning status of text case mode of text item 2-42, 2-43, 2-46  
returning string specified by the UDATA attribute 2-47  
returning text color of item, returning 2-32  
returning text font of item 2-15  
returning text string for item prompt 2-28  
returning the height of a window 2-50  
returning the single/multi mode of the item 2-58  
returning the value of a control state item 2-40  
returning the VALUE of an item 2-34  
returning the width of a window 2-54  
returning the width of an item 2-36  
returning value of x location of item 2-37  
returning value of y location of item 2-38  
returning visibility setting associated with item 2-35  
returning visibility status of the item 2-60  
returning width of item defined by COLS attribute 2-10  
returning width of list 5-11  
returning window stripe setting of item 2-53  
returns height of item defined by ROWS attribute 2-31  
row number, getting 2-8  
row, adding to existing list 5-3  
row, checking if specified is selected 2-56  
row, deleting from existing list 5-4  
row, resetting 3-41  
rows by which a list scrolled, returning 5-13

## S

saving the current contents of a list to file 5-17  
scrolling list to specified character number 5-18  
scrolling list to the specified line number 5-19  
selection, single or multi, setting 3-27  
setting active state of an item 3-8  
setting background color for item 3-9  
setting color of text of a tree node 6-15  
setting font in which list is displayed 5-23  
setting global variable 7-13

setting greyout color of item 3-17  
setting heading of a list item 5-22  
setting heading of specified item 5-20  
setting height of specified item 3-18  
setting height of specified window 3-50  
setting icon for item 3-20  
setting invert state of an item 3-22  
setting item color of specified item 3-23  
setting item to readonly state 3-30  
Setting Item Values and Attributes A-5  
setting MODE to convert input to lowercase 3-44  
setting MODE to convert input to mixed case 3-26  
setting MODE to convert input to uppercase 3-35  
setting MODE to validate input for a real number 3-47  
setting MODE to validate input for an integer number 3-43  
setting mode to validate input integer 3-21  
setting MODE to validate input of real 3-31  
setting option 3-10  
setting position of specified window 3-51  
setting row 3-10  
setting state of a menu item 3-42  
setting text color of deselected item 3-34  
setting text color of selected item 3-33  
setting the etch of the specified item 3-15  
setting the link-line color of a tree node 6-13  
setting the MODE of an item to single or multi selection 3-27  
setting the MODE to convert input to lowercase 3-25  
setting the MODE to convert input to mixed case 3-45  
setting the specified action for the item 3-7  
setting the specified option or setting 3-10  
setting the width of the specified item 3-38  
setting tree node to selected state 6-14  
setting VALUE attribute of an item 3-36  
setting visibility state of a tree node 6-16  
setting width of specified window 3-52  
setting write state of text field 3-46  
showing an item on the screen 4-4  
showing item on screen relative to another 4-5  
single/multi mode of the item, returning 2-58  
size of the window, getting 2-48  
sounding bell 7-3  
specified line from list, returning 5-5  
specifying font associated with an item 3-16  
specifying selected icon for an item 3-14  
starting a macro script 7-14

status of integer validation, returning 2-20, 2-41  
 status of text case, returning 2-33  
 string specified by the UDATA attribute,  
 returning 2-47

## T

text case mode of text item, returning 2-42, 2-43,  
 2-46  
 text case, returning status 2-24, 2-25  
 text color of deselected item, setting 3-34  
 text color of selected item, setting 3-33  
 text field write state, setting 3-46  
 text font name of window, returning 2-49  
 text font of item, returning 2-15  
 text input caret, placing in item 3-11  
 text item read/write status, returning 2-44  
 text item real number validation, returning  
 status 2-45  
 tree node link-line color, setting 6-13  
 tree node text, setting color 6-15  
 tree node visibility state, setting 6-16  
 tree node, checking if specified is  
 highlighted 6-8  
 tree node, checking if specified is selected 6-5,  
 6-9  
 tree node, checking if specified is visible 6-7, 6-10  
 tree node, color of text, returning 6-6  
 tree node, highlighting in a specified color 6-12  
 tree node, setting to selected state 6-14

## U

updating displayed list contents from  
 memory 5-15  
 updating list contents with contents of file  
 identified by LISTFILE attribute 5-24  
 updating window from its resource file 4-10

## V

VALUE attribute of an item, setting 3-36  
 value of a control state item, returning 2-40  
 VALUE of an item, returning 2-34  
 visibility setting associated with item,  
 returning 2-35

visibility state of an item, resetting 3-37  
 visibility status of the item, returning 2-60

## W

width of a window, returning 2-54  
 width of an item, returning 2-36  
 width of item defined by COLS attribute,  
 returning 2-10  
 width of list, returning 5-11  
 window position, getting 2-51  
 window resource file, allocating new 4-11  
 window stripe setting of item, returning 2-53  
 window, minimizing 4-9  
 window, opening from an icon 4-8  
 window, setting height 3-50  
 window, setting width of specified 3-52  
 window, updating from its resource file 4-10

## X

x and y location of an item, resetting 3-28  
 x location of an item, resetting 3-39  
 x location of item, returning value 2-37

## Y

y location of an item, resetting 3-40  
 y location of item, returning 2-38

