

AREALIST™ PRO 8.2

Last Update: April 29, 2009

Automated Solutions Group
16742 Gothard Street, Suite 210
Huntington Beach, CA 92647
714.375.4252
714.848.0382 FAX
<http://www.asgsoft.com>
<mailto:sales@asgsoft.com.com>

Release History

REQUIREMENTS:

- Arealist Pro 8.2 requires 4th Dimension 2003 or greater.

Version 8.2 - April 29, 2009
Version 8.1.1 - October 24, 2008
Version 8.1 - October 8, 2008
Version 8.0.1 - February 7, 2008
Version 8.0 - October 16, 2007 — First Release (Universal Binary and .Bundle Format)

Important Upgrade Information

ALP and Enterability

If you are upgrading an existing application which uses data entry with ALP, you should be aware of the following important items:

- Field Display - don't change selection
- Adding or Deleting rows from form button
- Properly Configuring Entry / Exit Callbacks
- Initiating Data Entry - click and hold

Field Display

If you are using ALP to display fields and are performing custom actions via ALPs exit callback (see AL_SetCallback) you should no longer be executing code which will change the current selection as ALP now maintains the current record status when the exit callback method is invoked.

For example, with previous versions of 4D you had to do something like the following in your exit callback code:

```
AL_GetCurrCell($1;$row;$col)
GOTO SELECTED RECORD((Customers);$row)
```

This is no longer required and if the code is still making this type of call that will change the current selection, you will lose your edits.

Note: When inside the callback method, you can obtain the previous table value by using the 4D Command **Old**.

```
$prevValue:=Old((Customers)Zip)
If ($prevValue#(Customers)Zip)
  $ret:=ZC_LookupZip((Customers)Zip;->[(Customers)City;[(Customers)State]
End if
```

Adding or Deleting Rows from form button

When using ALP as an included layout, it is common to have buttons in the input form to Add or Delete rows from the ALP area. If you have code that performs this type of action, you will need to make a small change to your code so that records are correctly added/deleted when using enterability. There three different methods you can use to handle this issue.

1. Option one will be to modify the call to **AL_SetEntryOpts** to instruct 4D to commit the current cell when focus is lost

```
AL_SetEntryOpts(eArea;n+8;...) `where n is the desired data entry trigger
```

2. Option two will be to call **AL_ExitCell** in the script of the add or delete button on each method which performs a similar action.

```
AL_ExitCell(eCustomers)
```

3. Place a call in each of the exit callback methods to instruct ALP to exit the cell

```
AL_ExitCell($1)
```

Properly Configuring Exit Callback

When using an exit callback method, ALP now requires the existence of a result value (C_BOOLEAN) and failure to have the callback method incorrectly declared will produce an error in compiled applications (see screenshot.) Your callback method should be declared as

```
`PM: ALP_CB
C_BOOLEAN($0) `required ALP 7.9 or greater
C_LONGINT($1;$AL_AREA)
C_LONGINT($2;$AL_CAUSE)
```

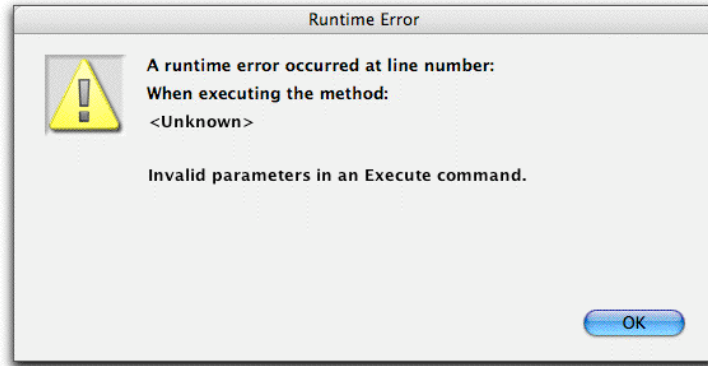


Figure 1 - Compiled runtime error when \$0 not declared

Initiating Data Entry - click and hold

ALP now provides the ability to initiate data entry by clicking and holding the mouse button down in the cell where you wish to perform data entry. Using this interface, users are still able to select lines via Single Click and Double click. If you wish to initiate data entry using this new method, use the **AL_SetEntryOpts** routine as follows:

```
AL_SetInterface (eALP;-1;-1;-1;-1;60) `initiate data entry after 1 sec of holding  
AL_SetEntryOpts (eALP;7;0) `initiate data entry via Control-DoubleClick
```

Data Entry will be initiated whenever any modifier+click action is defined as the data entry initiator (value 2..7). If the user moves the mouse during the click and hold action, ALP may interpret that as a drag action when ALP dragging actions are active.

Backwards Compatibility

If you are using AreaList Pro in an existing application, please be aware of the following changes. Failure to follow this information will result in a -9939 error (see Figure 2) when using ALP in heterogeneous applications (using with both Mac and Win clients).

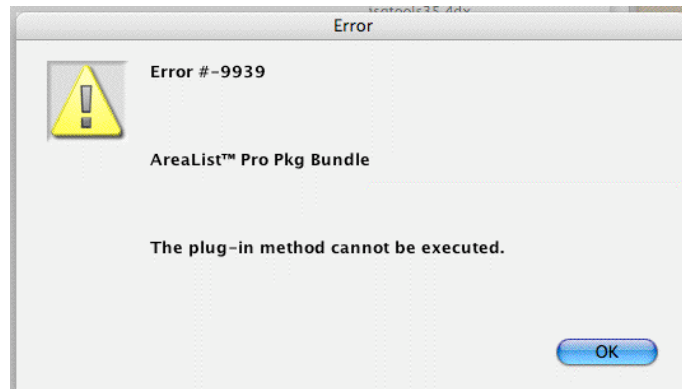


Figure 2 - Error when mismatch versions exist in Mac4dx and Win4dx

Introduction

AreaList Pro 8.2 contains many new enhanced features and bug fixes from previous 7.8.x releases. If you are using an existing version of AreaList Pro™, you will need to upgrade and obtain a new registration number as your existing registration number will not function with ALP 8.2.

For more information about upgrading to AreaList Pro, please contact our sales department.

E-Mail: sales@asgsoft.com
Phone: 800.375.4274

Registration

AreaList Pro requires a registration number to "unlock" the product making it a full working version. Call the **AL_Register** command (see the manual for syntax) in the On Startup procedure. Without the serial number, AreaList Pro will operate in demonstration mode.

When deploying applications which include AreaList Pro, you must purchase a valid server license or runtime license as the developer version is not intended for deployment purposes. For more information about the various deployment options, please contact our sales department.

AreaList Pro 8.2 requires new registration numbers, your existing ALP registration numbers will not work with ALP 8.2. Please contact Automated Solutions Group for upgrade information.

Documentation

The manual is in Adobe Acrobat PDF format, and requires Acrobat Reader v5 or greater. The Acrobat Reader application is included on the 4D Products CD-ROM, and is also available for free download from the Adobe Web site (<http://www.adobe.com/>).

Open Items

The following items are still under research and will be cured in a future release of AreaList Pro.

Data Entry Filters

We are working on supporting the same data entry filters used by 4th Dimension within ALP. Currently, you can use entry filters, but they do not support the placeholder characters available when using native 4D entry objects.

Change History

The following section outlines the changes which have been made to AreaList Pro 7.8.3 since the initial release.

- Red items indicate a bug fix
- Green items indicate a new feature
- Blue items indicate a new or modified routine name

Note: For a complete description of each of these routines, please refer to the **New / Modified AreaList Pro Routines** section below.

Version 8.2 – April 29, 2009

Corrected "Select All" bug when using 4D v11.3/11.4

Corrected a crashing bug which would occur when using Select All with 4D 11.3hf2

Corrected crashing bug AL_GetCellValue and AL_SetCellValue

Corrected numerous crashing issues when using AL_GetCellValue and AL_SetCellValue with both text and picture data

Corrected issue with AL_SetScroll

Corrected issue when using AL_SetScroll when procedurally hiding/showing scrollbar at runtime.

Corrected issue with default column size when using integer columns

Implemented required code change from modifications made in 4D 2004.7/4D v11.3hf2 which changes the way in which ALP calculates the default column width for integer columns.

Spurious Syntax Error Dialog - Enterable Cells - 4D v11.3 or higher ONLY

Fixed a bug which would cause 4D v11.3 (problem only occurs with 4D 11.3) to display a spurious error alert when tabbing out of enterable fields.

Crashing when copying data in enterable cell

Fixed an issue when using data entry mode and copying cell data using 4D v11.3

4D 11.3/11.4 Compatibility

Fixed a number of issues with 4D v11.3/11.4 attributed to the change of internal use and access of 4D resources. 4D v11.3/11.4 changed the way in which plugins would utilize internal 4D resources for use with internal code execute, thus breaking a number of places where internal code execute occurred.

Some of the changes include

- Modern date popup control not display at all
- Callback events display "invalid" and spurious alert messages

Popup Control - Memory Leak

Fixed a memory leak (4D v11.3 only) which would occur when accessing ALP popup controls

Version 8.1.1 – October 24, 2008

Offscreen Drawing

Fixed a bug which would cause ALP to draw a portion of its contents offscreen when display 4D dialogs in a sheet window via exit callbacks.

Version 8.1 – October 8, 2008

Moved Date/Time Popup Icons to PICT resource

Moved the date/time popup icons to PICT resources so that can be more easily customized by developers (previous method was using 'cicn' resources which are difficult to edit) and cicn resource editors no longer exist for MacOS X

Corrected Date/Time Popup Icon Drawing

Adjusted the position of date/time popup icons

Corrected date popup control crash when icon is close to vertical scrollbar

Corrected a crashing bug which would occur when date column is next to vertical scrollbar and user clicked date popup control

Revised drawing of date / time popup icons

Adjusted the physical location where the date / time popup icons are displayed.

Corrected Sort Editor Issues

Adjusted a few issues reported in the ALP Sort Editor

- drag/drop sort order list causes error on 4D v11
- field identifier icons not drawn on Windows - 4D v11

Corrected AL_GetCellValue Bug

Corrected a bug when calling AL_GetCellValue which would cause ALP to crash if data object was >255 characters

Drag & Drop Issue 4D v11.2 - Windows

Corrected a bug which would not allow ALP row to be reordered when using 4D v11.2 on Windows

ALP Drop Area Interference

Corrected a bug when using an ALP Drop Area behind another 4D object, causing interference. The ALP Drop Area should now always be transparent and not cause any interface interference.

Updated Time Control

Updated the ALP time control to use similar appearance (font and size) as date control

Enhanced Date Control

Enhanced date control to use native platform date control. Therefore, the date control will appear differently between MacOS and Windows.

Added New Events (Header and Footer clicks)

Added two new events (AL Column control click event and AL Footer click event)

- 10: AL Column click event
- 11: AL Column control click event
- 12: AL Footer click event

In the case of header click, AL Column click event will ONLY fire if the column has auto sort disabled. If user contextual click on header, the column will not be sorted.

Fixed ALP Popup bug (when source is Real array)

Fixed a bug when using a Real array as the source of column popup. Previously, the array would be displayed with empty data.

Fixed issue when display Date control on 4D sheet Window

Fixed a bug would cause problems display the ALP date popup control when ALP area is in a form displayed in 4D Sheet Window (4D 2004 or 4D v11)

Prompted to Version 8.1

There have been a large number of changes in ALP since the initial 8.0.x release and in order to assure users are using the latest build, we have promoted to version 8.1. If you are using ALP 8.0.x, it is strongly recommended that you upgrade to 8.1 (upgrades to 8.1 are free of charge for all 8.0.x users).

Added .4cx support

ALP 8.1 now includes a .4cx version, providing continued support for 4D 2003 (previous beta builds of 8.0.x were only available in .bundle format)

Added New Event - AL Column click event

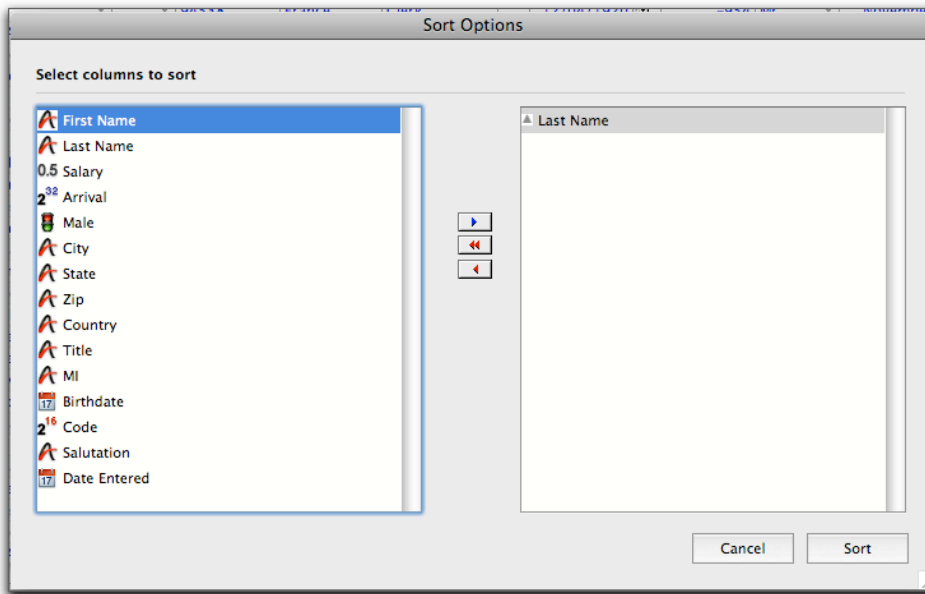
ALP 8.1 now includes a new event AL Column click event (-10) which will be triggered when a column header is clicked and auto sort is not activated.

Fixed Recursive Event Calls

Fixed a bug where ALP would get caught in a loop and recursive call event update code

Enhanced Sort Editor - 4D v11

Enhanced Sort Editor display when used with 4D v11 to display new 4D v11 style icons



New Routine - AL_IsValidArea

Added new routine AL_IsValidArea which provides the ability to determine if a given area is a valid ALP reference.

Established New Font Defaults

AreaList Pro has been modified to establish new font styles, providing consistent modern font styles. If you wish to override the ALP default font settings, you can use the new **AL_SetDefaultStyle** routine.

Macintosh Defaults

Header: Lucida Grande, 13 point, 0 plain
List: Lucida Grande, 11 point, 0 plain
Footer: Lucida Grande, 11 point, 0 plain

Windows Defaults

Header: Tahoma, 13 point, 0 plain
List: Tahoma, 11 point, 0 plain
Footer: Tahoma, 11 point, 0 plain

New Routine - AL_SetDefaultStyle

Added a new routine **AL_SetDefaultStyle** which provides the ability to set default font settings (List, Header and Footer) thus overriding the internal ALP defaults (See **New / Modified Routines** below for complete details)

New Routine - AL_SetDefaultFormat

Added a new routine **AL_SetDefaultFormat** which provides the ability to set default display formats for the various ALP data types (See **New / Modified Routines** below for complete details)

Fixed Display of 2-dimension picture arrays (4D v11 Only)

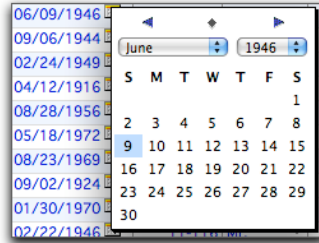
Fixed a bug which would cause 4D to crash when displaying 2D picture arrays when using 4D v11. This problem does not exist when using 4D 2003 / 4D 2004.

Fixed Sort Editor

Corrected cosmetic issues with ALP Sort Editor (introduced in ALP 8.0.1b11 release)

New Date Popup Control

ALP now includes a new Date Popup Control, providing a more modern interface. The new date popup control can be activated using the AL_SetInterface routine (newInterface parameter)



Expanded AL_SetInterface to support newInterface (date popup control)

AL_SetInterface provides the ability to customize various default actions when using ALP. You can now control the date popup control interface (now a dialog).

AL_SetInterface(area:L;appearance:L;sortIndicator:L;useElipsis:L;ignoreMenuMeta:L;clickDelay:L;allowPartialRow:L;dateControl:L)

dateControl - Instructs ALP to use the old or new date popup control

- 0 - Use original date popup (default)
- 1 - Use new date popup control interface

Drag and Drop Issues

Performed a number of fixes in the Drag & Drop routines. There is a new version of the W4D_DRAG.DLL contained in the Win4DX directory, please make sure you update this as well when using the new ALP 7.9.2b2

Navigation Issues

Corrected issues when navigating from page to page, which would randomly cause 4D to crash.

Fixed Sort Issues

Corrected issues when sorting alphanumeric columns and maintaining column synchronization with other columns.

Corrected bug with AL_SetSort

Corrected a bug when using AL_SetSort in the 'On load' phase (specifically to alphanumeric columns)

Corrected Data Entry Issues

Corrected a few issues related to data entry in enterable ALP cells

Corrected intermittent crashing bug when displaying fields

Corrected a few intermittent crashing bugs which occurred when using ALP to display fields

Version 8.0 – October 16, 2007

4D v11 SQL Native

AreaList Pro is now available in Universal Binary format, working natively with 4D v11 SQL

Corrected bug in AL_GotoCell

Corrected a bug in AL_GotoCell which was not working correctly with 4D 2004.6 (works fine with 4D 2003)

Corrected isolated crashing issue

Corrected an isolated crashing issue when using ALP scroll wheel on Windows with partial row enabled.

Corrected scrollbar slider drag and update issue

Corrected a bug when dragging the scrollbar slider up or down and not correctly updating the screen until mouseUp event. The area is now correctly updated when when you drag the slider up or down.

Added two new routines (for use in callbacks)

There have been two (2) routines added to this release to provide greater control during cell editing. For complete details, please refer to the updated Command Reference below

- AL_GetCellText(area:L;outText:L;inFlag:L)
- AL_SetCellText(area:L;inText:L;inFlag:L)

Enhanced AL_GetCellHigh and AL_SetCellHigh

Enhanced AL_GetCellHigh and AL_SetCellHigh to work during data entry (cell editing). Previously, these routines only worked during entry and exit callbacks.

Provided more detailed documentation for AL_SetEditMenuCallback Routine

ALP 8.0 introduced a new routine AL_SetEditMenuCallback which provides the developer with complete hook to working with the Edit menu. This release includes complete documentation for routine, along with a sample database for better explanation.

ALP Edit Menu Callback Example Database

The following example database demonstrates the various ways in which the ALP Edit Menu Callback can be used in your applications

4D 2003 Mac Example
http://www.asgsoft.com/ftp/private_beta/alp/AL_EditMenuCallbackExample.zip

Corrected field callback issue

Corrected an issue when using enterability with fields and ALP was incorrectly unloading the record during paste operation.

Corrected display of cell popup controls

Corrected drawing issues of cell popup arrows when cell has been defined as enterable. This problem was introduced in fc1

Changed action (and name) of AL Cell deselect action to AL Cell Validate

In previous versions of ALP, there were two constants: **AL_ExitCell** action and **AL_Cell Deselect action**. From the code it seems that AL ExitCell action was triggered when cell editing had to end, while AL Cell Deselect action would mean callback can refuse the end of editing, but this was not implemented consistently, and ALP manual does not clearly state the difference between the events either. Manual states that returning False from exit callback will mean area refuses to end editing, but ALP often ignored the return value.

ALP 8.0 changes meaning and handling of these events as follows:

AL_ExitCell

AL_ExitCell action now means that some event occurred that requires end of editing session. We tried to ensure that the return values will be honored and editing will not end if callback returns false - for example it means that if, during editing, user tries to scroll the area, and exit callback returns false, editing will not be terminated and area will not scroll.

AL_Cell Validate

AL_Cell validate action now means that 4D requested validation of area data, because it is going to execute some script or method, but 4D does not want to move focus from ALP area and editing will not end. The return value is not important, as editing will not end whatever is returned.

Note that this event will occur if user clicks on non-focusable object, including buttons, popups etc. If the executed script requires that editing ends before the script is executed, it have to call AL_ExitCell at its beginning.

Corrected slow scrolling issues on Mac

Corrected some slow scrolling issues that were introduced in b33

Corrected many scrolling issues on Windows

Corrected a number of scrolling issues which have existed in previous beta releases, including the recently reported issues with live scrolling (reported in b34).

Corrected issues with AL_SetCellStyle and AL_SetCellColor when working with >32k rows

Corrected issues with AL_SetCellStyle and AL_SetCellColor which were not working beyond 32k rows. This bug was introduced with the new support of displaying >32 rows.

Corrected bug when using calculated column callbacks

Corrected an issue (which may lead to crashing) when using AL calculated columns (and associated callbacks). Among the issues fixed were duplicate execution of callback and some other issues related to using enterable cells in same areas as calculated columns.

Corrected alternate row coloring with incomplete lines

Corrected an issue when default alternate row color is active for unused rows (those at bottom of area) when the new "show incomplete line" feature is activate. ALP was not correctly drawing the next alternate row color if it was a partial (bottom of area) row.

Adjust Vista column borders and fill

Performed some updated UI elements when using the Vista interface to provide better consistency with existing applications. The current Vista interface will no longer have extra vertical lines when the first column is selected.

Fixed descending sort arrow on Vista

Fixed a drawing issue when clicking on columns to sort in descending order. In addition, if ALP does not have enough space (at least 13 pixels) to display the header, it will be automatically resized to fit. This should not affect any installed applications as the Vista interface is new to ALP 8.0 and you would have to use a very small header font to make this problem occur.

Fixed interface display issues when using with Windows 2003 (Citrix)

Corrected column display when using ALP with W2K3 and Citrix. In the past, when used on W2K3 or Citrix, it would display the old style headers. ALP will now correctly display the appropriate platform columns.

Fixed popup arrows for enterable cells (Vista and W2K3)

Modified the default sort arrow for enterable cells, to use the correct platform default.

Expanded AL_SetInterface to support forcing platforms

AL_SetInterface provides the ability to customize various default actions when using ALP. One of those options is providing the ability to override the default platform interface based on the current client (machine) using ALP.

For example, if you wish to force the Vista interface while using MacOSX, you can use the *appearance* parameter of AL_SetInterface. The AL_SetInterface routine has been enhanced as follows

```
AL_SetInterface(area:L;appearance:L;sortIndicator:L;useElipsis:L;ignoreMenuMeta:L;clickDelay:L;allowPartialRow:L)
```

appearance - Instructs ALP to use the defined appearance setting, regardless of the current OS platform.

- 0 - Automatic appearance (default)
- 1 - force Platinum interface (MacOS 9)
- 2 - force Aqua interface (MacOS X)
- 3 - force XP interface
- 4 - force Vista interface

Fixed ALP Header display issues when using Vista

Fixed drawing of headers when using ALP with Vista. In addition, the sort arrow has been moved to be centered above column header, thus you need to take care to make sure header height is high enough to display sort arrow.

Fixed small offset drawing issues

Corrected some very minor offset issues, including:

- The left edge draws one pixel to the left of its boundary.
- The top edge draws one pixel above its boundary.

ALP Area drawn in actual size

There is a new option (see **AL_SetInterface**) whereby you can configure ALP areas to draw in the exact same height as defined in the form. In the past, ALP would resize the height of the area to match the font attributes and assure that only a complete row was visible (see Figure 3). This had an adverse affect that the area may be resized to be shorter than that which was drawn in the form, causing some UI inconsistencies.

This interface is controlled using a new parameter in AL_SetInterface:

```
AL_SetInterface(area:L;appearance:L;sortIndicator:L;useElipsis:L;ignoreMenuMeta:L;clickDelay:L;allowPartialRow:L)
```

allowPartialRow - Instructs ALP to display partial rows, thus causing ALP to draw in the exact areaRect defined in form.

- 0 - Don't display partial row, area will be resized vertically to only show full rows [default]
- 1 - Allow display of partial rows (area will not be resized)

First Name	Last Name
Bailey	Erickson
Joelle	Erickson
Kira	Erickson
Brady	Erickson
Trevor	Erickson

Figure 3 - ALP Area auto sized (see gap at bottom)

First Name	Last Name
Bailey	Erickson
Joelle	Erickson
Kira	Erickson
Brady	Erickson
Trevor	Erickson

Figure 4 - ALP drawn in actual size of form object

Corrected issue when using MenuPack connection with ALP .bundle

Corrected a bug which would cause the ALP/MenuPack connection to not function properly when using the .bundle (plugin) version of ALP.

AL_GetColumn and AL_GetRow Warning

AL_GetColumn and AL_GetRow routines should not be used in entry or exit callback methods as it reports where the user has clicked, not where the cursor may reside. If you wish to get the current row and column within exit callback, you should use **AL_GetCurrCell**.

Corrected screen flickering issues with Windows Vista

Corrected a number of interface issues related to Windows Vista, including a consistent screen flicker

Corrected crashing issue when selecting multiple lines from bottom of list

Corrected a very rare bug which would cause ALP to crash when you select the last line in an ALP area then select other lines above it (shift click or control click). This bug only occurred on Windows and very specific steps.

ALP Screen Border lighter grey

Modified the display of an ALP area to use a lighter (platform specific) shade of grey when the ALP area is not selected. In the past, this border was black, it has been modified to match that of the OS in which it is running.

New Routine - AL_SetEditMenuCallback

AL_SetEditMenuCallback will install a callback method which will be called when any Edit menu action occurs. You will have the option of overriding an 4D edit action for a given ALP area, providing an extensive customization interface when using Edit menu.

Some examples of how the edit menu callback interface can be used:

- You can modify the data which is placed on the clipboard to include an additional information you wish, such as header or footer data, using the **AL_GetHeaders** and **AL_GetFooters** routines, followed by calling 4D native clipboard routines.
- You can also trap information pasted to an ALP area, providing an interface which might take column row data copied from Excel and paste into ALP area, updating existing data (inserting or removing rows as necessary).

Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_GetVersion

AL_GetVersion will return the version of the ALP plug-in. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

Modified Routine - AL_SetMiscRGBColor Definition

The AL_SetMiscRGBColor routine was omitted from ALP 7.8 Addendum. This routine is similar to **AL_SetMiscColor** routine, except it provides ability to set color using RGB values. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

Enhanced functionality of event callback (see AL_SetEventCallback)

The return value from callback has been extended by possibility to add 2 to result value, which instructs ALP to NOT call 4D to update variables

- <0 - Reserved (will not update variables and will not force script execution)
- 0 - do not execute area script, update 4D variables
- 1 - execute area script, update 4D variables
- 2 - do not execute area script, do not update variables
- 3 - execute area script, no do not update 4D variables

New Routine - AL_GetFooters

Added new routine AL_GetFooters which will return footer information (see AL_GetHeaders). Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

Corrected issue when calling AL_ExitCell from event callback

Corrected bug when calling AL_ExitCell from within event callback. In previous implementations, this command was not functioning correctly.

Corrected drag/drop issues when using bundle version on Windows

Restored drag/drop interface when using .bundle version of ALP plugin (The W4DRAG.DLL was inadvertently omitted from bundle plugin resources).

Fixed crashing when calling AL_SetAltRowClr

Fixed a bug which would cause ALP to crash when calling the routine AL_SetAltRowClr.

Fixed issues with drag / drop on Windows

Fixed a bug introduced in the b24 release which caused ALP to incorrectly handle drag/drop operations (interfering with new enterable option).

Resizable Sort Editor

The ALP default sort editor is now displayed as a resizable window (and default window is larger).

Corrected Type Ahead Issues

Corrected issues when using Type Ahead, problems were introduced in b24

Corrected .bundle errors on 4D Server 2004

Corrected a major issue which kept ALP .bundle format from working correctly under 4D Server 2004 (produced an error that Cache could not be created).

Increased Sort Editor window size

Increased the size of the ALP Sort Editor so it will display the full contents of larger column headers.

Updated ALP Bundle - Added W4D_DRAG.DLL

Updated windows bundle version (Plugins) to include the W4DRAG.DLL, correcting drag/drop issues on Windows when using bundle version.

Fixed AL Event generation when using keyboard during multi-row select

Fixed a bug which would not correctly fire an associated event (single click ALProEvt = 1) when the user presses the up/down arrow during multi-row selection.

Enhanced ALP Data Entry Initiation

Add a new data entry initiation option which will initiate cell entry when the user clicks and holds down the mouse button for the developer determined period of time. When you have configured ALP to allow data entry (using **AL_SetEntryOpts**) with an modifier-click (Control, Command, etc.) data entry will be automatically initiated when the user hold down the mouse button.

This interface will allow you to create an interface whereby users can Single Click or Double Click on cells and initiate data entry without requiring the defined keyboard modifier.

Example:

The following example has data entry configured as Control-DoubleClick, however, it will also be activated when the users has held down the mouse button in an enterable cell for one second (60 ticks) as configured in **AL_SetInterface**.

```
AL_SetInterface (eALP;-1;-1;-1;-1;60) `initiate data entry after 1 sec of holding  
AL_SetEntryOpts (eALP;7;0)
```

Enhanced ALP Enterable Popups - Meta Character support

You can now optionally disable meta characters in ALP Enterable Popup controls, enabling you to use special characters such as "/" or "|" in menu items. See **New AreaList Pro Routines** for updated parameter reference.

There are two different methods for disabling Meta Characters. The first method will be using a new parameter in **AL_SetInterface** routine, while the second method will be using new options in the **AL_SetEnterable** routine.

Enterable and 4D 2004

Removed the require to customize the exit callback when enterability is active with the "New Menu Architecture" active. You no longer need to trap and conditionally respond to softDeselect (evtCode 8).

The behavior of deselect event - now, by default, ALP should handle correctly Edit menu events if 'New Edit menu behavior' is set. However, this may break compatibility with existing code. If '8' is added to first parameter to **AL_SetEntryOpts** call, the ALP behavior would switch to previous behavior.

The difference between the two behaviors is as follows:

4D send two kinds of deselect events to external area, real (hard) deselect and validating (soft) deselect. The first means user clicked on other focusable object (like edit field) and focus is going to pass from ALP area to the new object, second means that user clicked on some non-focusable object and focus will stay with ALP area.

However, difference between focusable and non-focusable objects are not obvious (for users), so ALP till 7.8. does not handled the two events differently. Now, it has to do, as 4D sends the soft deselect to ALP before it sends edit menu commands.

Soft deselect is passed to exit method routine with selector
deselect_EndEntry = 8,

When exit callback is called with this selector, it means the editing will not be finished in fact, just the value should be validated because 4D is going to execute some action. It should not cause much problems in existing applications, but we will see. The developers can always force former behavior (no discrimination between deselections) by using **AL_SetEntryOpts** call.

Sort Header Icon positioning

Adjusted the drawing location of the new sort header icon interface added in 8.0 (see **AL_SetHeaderOptions**).

AL_GetCellColor

Fixed a bug when calling **AL_GetCellColor** on Windows to correctly return the index color.

Last column resizable

The last column in an ALP area is now resizable

Header Text Color

Fixed issues when customizing the header text color when platform default interface is used. In previous versions, ALP would not use custom header text attributes, only using the standard font attributes (size, style)

New Routine - Custom Header Actions

Added a new routine which provides the ability to override the header action (sort button), providing an interface to use custom icons and callbacks.

Note: ALP v8.0 or greater requires that you recompile your applications and you must make sure you have matching versions across Mac/Win clients if you are using multi-platform deployments. ALP 8.0 is NOT drop in compatible, if you drop in ALP 8.0 or greater into an existing compiled application, you will receive the following error dialog.

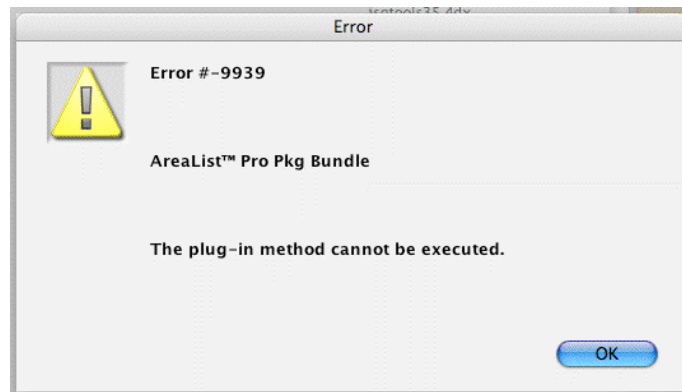


Figure 5 - Error when mismatch versions exist in Mac4dx and Win4dx

When ALP has been configured to allow multiple row selection, pressing the up or down arrow keys, ALP would not correctly respond and move accordingly. The following conditions have now been set

- When the up arrow key is pressed, the row prior to the first highlighted row will be selected and will be the new active row
- When the down arrow key is pressed, the row after the last highlighted row will be selected and will be the new active row

This interface is off by default (for backwards compatibility with current applications) and may be activated using the **AL_SetCellOpts** routine (second parameter, activation).

Example:

The following parameter will activate the new keyboard scrolling options when using multi-row selection option

AL_SetCellOpts(eArea;3;...) `turn on enhanced arrow key support (see also **AL_SetDefaults** [parameter 7])

Header Foreground Colors

Corrected an issue when using the **AL_SetForeRGBColor** or **AL_SetForeColor** routines, setting the foreground text color for headers with the new platform default interface, the header text was not correctly set (it was always black).

New Event Callback Interface

AreaList Pro contains a new event management interface which can be used in place of the current 'During' (On plugin area) event. When using the callback interface, 4D will no longer have the interference currently plagued by the various 4D revisions. For example, when 4D released 2003, they introduced a new event menu mechanism for Edit menus. This new interface caused a series of unexpected behaviors when using ALP.

With the release of ALP 8.0, the callback interface is the recommended method for responding to ALP events and will be expanded to support greater event control in future versions and the current 'During' method will be deprecated and new events will no longer be passed to this interface (and ALProEvt)

The method for utilizing the new event callback interface will be pass the name of the callback method to the **AL_SetEventCallback** method.

Example:

The following example will define the callback method for the ALP area

Case of

```
:(Form event=On_Load)
  $ret:=AL_SetEventCallback (eProdList;"REG_RegGen_CB")
```

End case

The event callback method requires the following declarations

```
`PM: ALP_EventCallback(area:L;event:L;eventMod:L;col:L;row:L;modifiers:L;tip:S) -> result:L
```

```
C_LONGINT($0) `status, return 0
C_LONGINT($1) `ALP area
C_LONGINT($2) `alp event
C_LONGINT($3) `event modifier
C_LONGINT($4) `column - last clicked column
C_LONGINT($5) `row - last clicked row
C_LONGINT($6) `modifiers
C_STRING(255;$7) `tip string
C_STRING(255;$8) `area name
```

alpArea - ALP Area Reference

alpEvent - ALP Event (See ALProEvt for current list of supported events)

eventModifier - event modifier (currently unused, but will be exposed in future release)

lastColumn - last clicked column

lastRow - last clicked row

keyModifier - keyboard modifier

256 - Command Key
512 - Shift Key
1024 - Caps Lock
2048 - Option Key
4096 - Control Key

tipString - alp tool tip string If you would like to display a tool tip for a given cell with custom text, this can be done in the **mouseMove** event (18)

```
:(($alpEvent=18) `mouse event
  $0:=0 `mouse event - processed here

  $AL_Area:=$1
  $AL_Event:=$2
  $AL_LastColumn:=$4
  $AL_LastRow:=$5
  $AL_AreaName:=$8

  If (Macintosh option down)
    AL_GetCellValue ($AL_Area;$AL_LastRow;$AL_LastColumn;$value)
    $tip:="This cell contains the value of "+$value+"!"
    $7:=$tip
  End if
```

areaName - text area name (see **AL_SetAreaName**)

This property will enable you to create a generic callback method for all ALP areas, then using the AreaName for extend customization.

New Events (Event Callback Only)

AreaList Pro has expanded the events which can be triggered in an ALP area. The following event codes are ONLY available when using the new Event Callback interface (it is recommended that all applications move to the new event callback for compatibility with future versions of 4th Dimension).

<u>Event Code</u>	<u>Description</u>
-------------------	--------------------

Enterable Time Cells

Corrected a few issues when using data entry with cells display TIME values.

Ghosting Scroll Bars

When using ALP on multi-page forms, ALP would sometimes "ghost" the scrollbars from previous pages when switching to a new page. This bug should now be fixed and calling AL_SetScroll should now properly permanently disable scrollbars.

Example:

The following routine will disable the horizontal and vertical scrollbars

```
AL_SetScroll(eArea;-3;-3)
```

Entry / Exit Callbacks

Fixed a bug in entry and exit callbacks which does not execute SAVE RECORD before validation so that you can reference previous values and responding accordingly.

Enhanced Routine - AL_SetEntryOpts

Added a new option to ignore soft deselect events (e.g. clicking on non-focusable button) add 8 to entrySelect parameter.

Example:

The following example will configure area to ignore soft deselect events

```
AL_SetEntryOpts(eArea;n+8;...)
```

Fixed redraw issues when modifying 4D objects in callback

Fixed a bug which would filter update events and would not properly redraw form when other form objects were updated in ALP callback.

Native ALP Area Focus

ALP focus border now use native appearance, matching other form objects.

New Entry Cell Border

ALP 8.0 now uses the native cell border when performing data entry (see Figure 6).

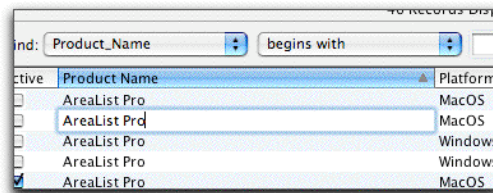


Figure 6 - New ALP Cell Border

New Routine - AL_SetAreaName

Added a new routine to provide a more descriptive name for referencing ALP areas. This area name is also available in the new ALP Event Callback (parameter 8). Pass a null string to clear the area name. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

Example:

The following example will define the area name as "myArea" for the eArea plug-in area

```
AL_SetAreaName(eArea;"myName")
```

New Routine - AL_GetAreaName

Added a new routine obtain the area name referenced by area reference. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

Example:

The following example will store in sAreaName the name of the ALP area referenced by eArea

```
sAreaName:=""
AL_GetAreaName(eArea;sAreaName)
```

New Keyboard Scrolling Options

ALP 8.0 has a new scrolling option available when using the up/down arrows during multi-row selections. Using the AL_SetCellOpts routine, you can activate the new scrolling options when multiple rows are selected

Example:

The following parameter will activate the new keyboard scrolling options when using multi-row selection option

```
AL_SetCellOpts(eArea;3;...)
```

AL_SetCellOpts has been enhanced to provide additional cell selection options. Using the cellSelection parameter, you now have the option of using ALPs new cell selection interface. When multiRow selection is active, the following will occur:

- If the user presses the up arrow during multiRow select, the row before the first highlighted row will be selected
- If the user presses the down arrow during multiRow select, the row after the last highlighted row will be selected

area - ALP area

cellSelection - cell selection options

0 - row selection is enabled according to the MultiLine option of **AL_SetRowOpts** (page69) (default)

1 - only one cell at a time may be selected (single cell selection)

2 - many cells may be selected, contiguous or discontiguous (multiple cell selection)

3 - handle new multiRow selection interface

Removed 32k Row Display Limit

ALP 8.0 now supports 4 million rows, increasing the previous limit of 32,000 rows. In order to effectively use more than 32k rows and the support routines for getting or setting the selection, you must now pass a LONGINT array instead of the previous format of INTEGER array.

Example:

The following example will return the number of selected rows (multi-line select).

```
ARRAY LONGINT(aiSelectedRows;0) `previously used INTEGER array
$ret:=AL_GetSelect(eArea;aiSelectedRows)
```

Modified Routine - AL_SetSelect

AL_SetSelect now supports ARRAY INTEGER or ARRAY LONGINT as result array for setting the current selection of rows. It is recommended that you modify your existing code to use LONGINT arrays to support ALPs support for >32k row selections. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

Modified Routine - AL_GetSelect

AL_GetSelect now supports ARRAY INTEGER or ARRAY LONGINT as result array for selected rows. It is recommended that you modify your existing code to use LONGINT arrays to support ALPs support for >32k row selections. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_GetCellOpts

AL_GetCellOpts will return the current settings configured using **AL_SetCellOpts**. For complete details about return values, please see the **AL_SetCellOpts** routine for possible configuration values. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_GetCopyOpts

AL_GetCopyOpts will return the current settings configured using **AL_SetCopyOpts**. For complete details about return values, please see the **AL_SetCopyOpts** routine for possible configuration values. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_GetMiscOpts

AL_GetMiscOpts will return the current settings configured using **AL_SetMiscOpts**. For complete details about return values, please see the **AL_SetMiscOpts** routine for possible configuration values. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_GetColOpts

AL_GetColOpts will return the current settings configured using **AL_SetColOpts**. For complete details about return values, please see the **AL_SetColOpts** routine for possible configuration values. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_GetRowOpts

AL_GetRowOpts will return the current settings configured using **AL_SetRowOpts**. For complete details about return values, please see the **AL_SetRowOpts** routine for possible configuration values. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_SetHeaderIcon

AL_SetHeaderIcon provides the ability to procedurally place icons with headers (enhances icon support which was added in 7.8). Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_SetCellIcon

AL_SetCellIcon provides the ability to procedurally place icons within cells (enhances icon support which was added in 7.8). Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_SetCellValue

AL_SetCellValue provides the ability to procedurally set the contents of a given cell (use **AL_GetCellValue** to retrieve individual cell values). Used in conjunction with the new **AL_GetRow** command (see description below) you can provide additional feedback and control over cell contents. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

New Routine - AL_GetRow

AL_GetRow is similar to the **AL_GetColumn** routine, except it will return the current row number selected by the user. This routine may be executed in the ALP plug-in area event. Please refer to the routine definition in **AreaList Pro New / Modified Routines** for complete details.

Fixed Exit Callback Procedure

Fixed a number of issues which occur when using callback methods with 4D 2004. When perform data entry, the ALP callback methods were not correctly executed.

Redraw Issues

Fixed a few left over redraw issues which manifest when switching between output and input forms within same window, or when switching pages between forms which contain ALP areas.

Menu flashing

Fixed menu flashing issues when using ALP with 4D 2004.2 (see associated handling when using **AL_SetEventCallback** interface)

New / Modified AreaList Pro Routines

The following routines have been added or modified in AreaList Pro 8.0

AL_IsValidArea(areaRef:L)-> result:L

AL_IsValidArea will determine if the supplied area reference is valid.

areaReference - ALP Reference

-> *result* - Returns one of two values

0 - Invalid Area
1 - Valid Area

AL_SetDefaultStyle(selector:L;fontName:S;fontSize:I;fontStyle:I)

selector - The desired attribute you wish to set (header, list, footer)

1 (AL_Style_Header) - Sets header style
2 (AL_Style_List) - Sets list style
3 (AL_Style_Footer) - Sets footer style

fontName - Desired font name. You can pass a null string ("") to use default fontName

fontSize - Desired font size. You can pass a value of -1 to use default fontSise

fontStyle - Desired font style. You can pass a value of -1 to use default fontSize

Example:

The following example will set the default list style to Vernda, 10 point, plain and default header style to Verdana, 10 point, bold

AL_SetDefaultStyle(AL_Style_Header;"Verdana";10;Bold)

AL_SetDefaultStyle(AL_Style_List;"Verdana";10;Plain)

AL_SetDefaultFormat(selector:L;format:S)

selector - The desired data type you wish to set default filter

1 (AL Format Integer) - Sets integer column display
2 (AL Format Longint) - Sets longint column display
3 (AL Format Real) - Sets real column display
4 (AL Format Boolean) - Sets boolean column display
5 (AL Format Date) - Sets date column display
6 (AL Format Picture) - Sets picture column display

displayFilter - Desired display filter, default values

1 - ##,##0
2 - #,###,##0
3 - #,###,##0.00*
4 - True;False
5 - uses setting in control panel
6 - n/a

* AreaList Pro will now check system decimal dot usage in System Preferences and use its behavior and formats accordingly, eliminating need to potentially override

AL_SetAreaName(area:L;areaName:S)

-> *area* longint ALP Area Reference
-> *areaName* string ALP Area Name

AL_SetAreaName provides an interface for defining a name to a given ALP area. This can be helpful when using a generic event handler routine (see AL_SetEventCallback), providing a method for determining which area has passed along the desired event.

area - ALP Area Reference

areaName - Desired area name

AL_GetAreaName(area:L;areaName:S)

-> *area* longint ALP Area Reference
<- *areaName* string ALP Area Name

AL_GetAreaName will return the name defined by AL_SetAreaName. This value will also be available in event callback (parameter 8).

area - ALP Area Reference

areaName - Returns area name

AL_GetCellText(area:L;outText:T;inFlag:I)

-> *area* longint ALP Area Reference
<- *outText* text/string Returns current cell highlight text
-> *inFlag* longint Flag

AL_GetCellText will return the currently highlighted cell text which can be obtained during Edit Menu Callback

Note: This routine will only work correctly during Edit menu callback.

area - ALP Area Reference

outText - Returns current highlight text based on flags.

flags - Formatting flags

0 - get whole cell text
1 - get selected text only

AL_SetCellText(area:L;inText:T;inFlag:L)

-> area longint ALP Area Reference
-> inText text/string Text you wish to set
-> inFlag longint Flag

AL_SetCellText will set the currently highlighted cell text which can be obtained during Edit Menu Callback

Note: This routine will only work correctly during Edit menu callback.

area - ALP Area Reference

inText - Sets the current cell text

flags - Formatting flags

0 - replace whole cell text
1 - replace highlighted text only

AL_SetEditMenuCallback(area:L;callbackMethod:S) -> result:L

-> area longint ALP Area Reference
-> callbackMethod string Callback method

AL_SetEditMenuCallback provides an interface of overriding the default behavior when work with the 4D edit menu.

area - ALP Area Reference

callbackMethod - Desired callback method to use for given area.

Callback Parameters

The edit menu callback will return the area (parameter 1) and edit event (parameter 2). For an example callback, please refer to the example database **ALP Edit Menu Callback**.

area - ALP area reference

editEvent - Returns edit events in a longint object, using bitwise operators to extract subevents. When you wish act on one more more edit menu items, you return the value is a bitwise value.

Note: For more information on work with bitwise values, please refer to the 4th Dimension Command Reference

AL_Edit_Menu_Setup_Mask (65536) - Indicates that ALP is setting up edit menu, before menu is displayed

AL_Edit_Menu_Setup_Bit (16) - Bitwise position in event value

AL_Edit_Menu_Entry_Mask (32768) - Indicates event ALP cell editing is in progress

AL_Edit_Menu_Entry_Bit (15) - Bit position

AL_Edit_Menu_All_Items_Mask (127) - Indicates that ALP is using all possible edit menu items

AL_Edit_Menu_Select_All_Mask (64) - Reference to Edit menu select all menu item

AL_Edit_Menu_Select_All_Bit (6) - Bit position

AL_Edit_Menu_Clear_Mask (32) - Reference to Edit menu clear menu item

AL_Edit_Menu_Clear_Bit (5) - Bit position

AL_Edit_Menu_Paste_Mask (16) - Reference to Edit menu paste menu item

AL_Edit_Menu_Paste_Bit (4) - Bit position

AL_Edit_Menu_Copy_Mask (8) - Reference to Edit menu copy menu item

AL_Edit_Menu_Copy_Bit (3) - Bit position

AL_Edit_Menu_Cut_Mask (4) - Reference to Edit menu cut menu item

AL_Edit_Menu_Cut_Bit (2) - Bit position

AL_Edit_Menu_Redo_Mask (2) - Reference to Edit menu redo menu item

AL_Edit_Menu_Redo_Bit (1) - Bit position

AL_Edit_Menu_Undo_Mask (1) - Reference to Edit menu undo menu item

AL_Edit_Menu_Undo_Bit (0) - Bit position

-> *result* - The Edit menu callback expects a longint result, containing details of what action(s) should take place

When working with callback, you can return a series of values. If you want ALP to handle Edit menu selection, return a value of zero (0). If wish to customize the result, you return AL Edit Menu Handled Mask (tells ALP we handled menu).

AL_Edit_Menu_Handled_Mask (131072) - Tells ALP that callback method handled Edit menu operation

AL_Edit_Menu_Handled_Bit (17) - Bit position

Example:

For example, If you wish to tell ALP to enable only the Copy and Select All menu items (overriding default settings), you would return the following result.

```
$AL_Result:=( $AL_Result & AL_Edit_Menu_All_Items_Mask ) ?+ AL_Edit_Menu_Handled_Bit
$AL_Result:=$AL_Result ?+ AL_Edit_Menu_Copy_Bit `enable copy menu item
$AL_Result:=$AL_Result ?+ AL_Edit_Menu_Select_All_Bit `enable select all menu item

$0:=$AL_Result
```

Edit Menu Callback Framework

The callback method should include the following framework parameter declaration. Each parameter must be declared in every callback method and a result value must be returned. Failure to properly declare variables, or omit the result variable will produce a compiler error when using database in compiled mode.

```
`PM: AL_Edit_CB(area:L;event:L;undo:S) -> result:L
`LM: 5/25/07, v8.0
```

```

`$0: result
` 0 - ALP will handle event (default)
` >0 - callback handled event
`$1: ALP Area Reference
`$2: ALP Edit Event
`$3: Undo (Unused, but required by internal plugin API for callback)

```

```

C_LONGINT($0;$AL_Result)
C_LONGINT($1;$AL_Area)
C_LONGINT($2;$AL_Event)
C_TEXT($3;$AL_Undo)

```

```

$AL_Result:=0 `default result, ALP will handle event
$AL_Area:=$1
$AL_Event:=$2
$AL_Undo:=$3

```

Case of

...

End case

```
$0:=$AL_Result
```

AL_GetVersion -> version:S

```
-> version      string      ALP Version
```

AL_GetVersion will return the current ALP version

Example:

```
$vers:=AL_GetVersion `returns current ALP plug-in version
```

AL_GetRowOpts(area:L;multiLines:L;allowNoSelection:L;dragLine:L;acceptDrag:L;moveWithData:L;disableRowHighlight:L)

```

-> area          longint      ALP Area Reference
<- multiRow     longint      multiline attribute
<- allowNoSelect longint      allow no row selection attribute
<- dragLine     longint      drag line attribute
<- acceptDrag   longint      accept drive attribute
<- moveWithData longint      move with data attribute
<- disableRowHilite longint    disable row highlight attribute

```

AL_GetRowOpts will return the current settings configured using **AL_SetRowOpts**. For complete details about return values, please see the **AL_SetRowOpts** routine for possible configuration values.

area - ALP area reference

multiLines - returns multiLine attribute

allowNoSelection - returns allowNoSelection attribute

dragLine - returns dragLine attribute

acceptDrag - returns acceptDrag attribute

moveWithData - returns moveWithData attribute

disableRowHighlight - returns disableRowHighlight attribute

AL_GetColOpts(area:L;allowColumnResize:L;allowColumnLock:L;hideLastColumns:L;displayPixelWidth:L;dragColumn:L;acceptDrag:L)

```

-> area          longint      ALP Area Reference
<- allowResize   longint      allow column resize attribute
<- hideLastCols  longint      hide last columns attribute
<- dispPixWidth  longint      display pixel width attribute
<- dragColumn    longint      drag column attribute
<- acceptDrag    longint      accept drag attribute

```

AL_GetColOpts will return the current settings configured using **AL_GetColOpts**. For complete details about return values, please see the **AL_GetColOpts** routine for possible configuration values.

area - ALP area reference

allowColumnResize - returns allowColumnResize attribute

resizeInDuring - returns resizeInDuring attribute

allowColumnLock - returns allowColumnLock attribute

hideLastColumns - returns the number of last-hidden columns

displayPixelWidth - returns displayPixelWidth attribute

dragColumn - returns dragColumn attribute

acceptDrag - returns acceptDrag attribute

AL_SetHeaderIcon(area:L;column:L;iconAlignment:L;picRef:L;horPosition:L;vertPosition:L;offset:L;scaling:L)

-> area	longint	ALP Area Reference
-> column	longint	column attribute
-> iconAlignment	longint	icon alignment attribute
-> picRef	longint	picture reference
-> horPosition	longint	horizontal position attribute
-> vertPosition	longint	vertical position attribute
-> offset	longint	pixel offset attribute
-> scaling	longint	scaling attribute

AL_SetHeaderIcon provides the ability to procedurally place icons in column headers.

area - ALP area reference

column - desired header column number

iconAlignment - position of icon

0 - places icon on left of cell
1 - places icon on right of cell

picRef - 4D PICT object containing icon value (Due to limitation of drawing of icons in header, you must first load the desired icon into a 4D PICT object)

horPosition - one the following options

0 - default (top)
1 - align top left
2 - align center
3 - align bottom right

vertPosition - one the following options

0 - default (top)
1 - align top left
2 - align center
3 - align bottom right

Default position is left for left icon and right for right icon.

offset - offset of "icon guide" Vertical position is relative tp "icon guide". If the vertical alignment is alignCenter, the icon is centered between guide and corresponding (left or right_ side of cell.

scaling - one the following options

0 - truncated
1 - scaled

Cell content (text) is drawn to place that is left after drawing icon. If icon is bigger then available space, text is drawn over the icon.

Example

The following example will use the same icon as AL_SetCellIcon, but it will first load the icon into a 4D PICT object.

C_PICTURE(\$pict)
C_LONGINT(\$col)

\$col:=3 `place icon in 3rd column
\$iconAlign:=0 `draw on left

\$horPos:=0 `default
\$verPos:=2 `align right
\$offset:=5
\$scaling:=0

GET PICTURE FROM LIBRARY(1717;\$pict)
AL_SetHeaderIcon (eAL_Output;\$col;\$iconAlign;\$pict;\$horPos;\$verPos;\$offset;\$scaling)

\$iconAlign:=1 `draw on right
GET PICTURE FROM LIBRARY(1717;\$pict)
AL_SetHeaderIcon (eAL_Output;\$col; iconAlign;\$pict;\$horPos;\$verPos;\$offset;\$scaling)

AL_SetCellIcon (area:L; column:L; row:L; iconRef:L; iconAlignment:L; horPosition:L; vertPosition:L; offset:L; scalingOption:L)
this call superceeds icons codes in text

-> area	longint	ALP Area Reference
-> column	longint	column attribute
-> row	longint	row attribute
-> iconRef	longint	icon Referecne attribute
-> iconAlignment	longint	icon alignment attribute
-> horPosition	longint	horizontal position attribute
-> vertPosition	longint	vertical position attribute
-> offset	longint	pixel offset attribute
-> scaling	longint	scaling attribute

area - ALP area reference

column - desired column where you wish to draw icon

row - desired row number where you wish to draw icon

iconRef - icon reference as now, depending of the value it uses icns resource, PICT resource or picture library picture

To associate an icon to the item, pass one of the following numeric values:

- N, where N is the resource ID of Mac OS-based 'cicn' resource
- Use `PICT resource+N`, where N is the the resource ID of a Mac OS-based 'PICT' resource
- Use `PicRef+N`, where N is the reference number of a Picture from the Design environment Picture Library

Pass zero (0), if you do not want any graphic for the item.

leftRight - each cell can contain up to two icons - one on the left and one on the right. Pass 0 for left, 1 for right

horPosition - one the following options

- 0 - default (top)
- 1 - align top left
- 2 - align center
- 3 - align bottom right

vertPosition - one the following options

- 0 - default (top)
- 1 - align top left
- 2 - align center
- 3 - align bottom right

Default position is left for left icon and right for right icon.

offset - offset of "icon guide" Vertical position is relative tp "icon guide". If the vertical alignment is `alignCenter`, the icon is centered between guide and corresponding (left or right_ side of cell.

scaling - one the following options

- 0 - truncated
- 1 - scaled

Cell content (text) is drawn to place that is left after drawing icon. If icon is bigger then available space, text is drawn over the icon.

Example:

The following example will draw an icon in `r3c2`, using an item (resID 1717) from the Picture Library.

```
$col:=2
$row:=3
$iconRef:=1717+Use_PicRef
$iconPos:=1 `right
$horPos:=0 `default
$verPos:=2 `align right
$offset:=5
$scaling:=0
```

AL_SetCellIcon (eAL_Output;\$col;\$row;\$iconRef ;\$iconPos;\$horPos;\$verPos;\$offset;\$scaling)

AL_GetMiscOpts(area:L;hideHeaders:L;areaSelected:L;postKey:S;showFooters:L;userModernLook:L)

-> area	longint	ALP Area Reference
<- hideHeaders	longint	hide headers attribute
<- areaSelected	longint	area selected attribute
<- postKey	string	post key attribute
<- showFooters	longint	show footers attribute
<- userModern	longint	use modern look attribute

AL_GetMiscOpts will return the current settings configured using **AL_SetMiscOpts**. For complete details about return values, please see the `AL_SetMiscOpts` routine for possible configuration values.

area - ALP area reference

hideHeaders - returns `hideHeaders` attribute

areaSelected - returns `areaSelected` attribute

postKey - returns `postKey` attribute

showFooters - returns `showFooters` attribute

useModernLock - returns `useModernLock` attribute

AL_GetHeaderOptions(area:L;options:L;iconRef:L;callbackMethod:S)

-> area	longint	ALP Area Reference
<- options	longint	options attribute
<- icon	longint	icon references attribute
<- callback	string	header callback method

AL_GetHeaderOptions will return the attributes set by **AL_SetHeaderOptions**. For complete description of header options, please refer to `AL_SetHeaderOptions`.

area - ALP Area Reference

options - returns desired options for overriding sort icon

iconRef - returns desired icon reference (uses same interface as 4D `hList` interface).

callbackMethod - returns desired callback method which is executed when icon is clicked.

AL_SetHeaderOptions(area:L;options:L;iconRef:L;callbackMethod:S)

-> area longint ALP Area Reference
-> options longint options attribute
-> icon longint icon references
-> callback string header callback method

AL_SetHeaderOptions provides the ability to customize the interface over the scrollbars (sort area). You can customize the icon which is displayed using 'cicn' or 'pict' resource, or an item from the 4D Picture Library (see details below).

area - ALP area

options - desired options for overriding sort icon

0 - no options, use default interface
1 - display custom icon and execute callbackMethod on mouseDown
2 - display custom icon and execute callbackMethod on mouseUp

iconRef - desired icon reference (uses same interface as 4D hList interface).

Note: For optimal results, the icon size should be 13w x 12h

To associate an icon to the item, pass one of the following numeric values:

- N, where N is the resource ID of Mac OS-based 'cicn' resource
- Use PICT_resource+N, where N is the the resource ID of a Mac OS-based 'PICT' resource
- Use PicRef+N, where N is the reference number of a Picture from the Design environment Picture Library

Pass zero (0), if you do not want any graphic for the item.

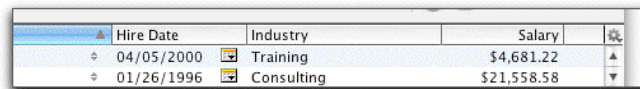
callbackMethod - desired callback method which is executed when icon is clicked.

- If you have passed an option value of 1, callback will be executed when user releases mouse button
- If you have passed an option value of 2, callback will be executed immediately when user click icon

Example:

The following example will create a custom icon, using a 4D Picture Library item

```
`configure area callback  
AL_SetHeaderOptions ($AL_AREA;2;TC_GetPictureRefNo ("AL_TBL_ConfigPop")+Use PicRef ;"AL_TBL_AreaConfig")
```



Hire Date	Industry	Salary
04/05/2000	Training	\$4,681.22
01/26/1996	Consulting	\$21,558.58

Figure 7 - Custom Header Icon and Action

AL_SetInterface(area:L;columnStyle:L;sortIcon:L;ellipsis;metaOptions:L;clickDelay:L;allowPartialRow:L)

-> area longint ALP Area Reference
-> columnStyle longint column style attribute
-> sortIcon longint sort icon icon reference
-> ellipsis longint use ellipsis reference
-> metaOption longint metaOption reference
-> clickDelay longint click delay to initiate data entry
-> allowPartialRow longint allow partial rows to be displayed

AL_SetInterface now includes an additional parameter, providing an interface for defining how ALP will utilize meta characters when popup menus are used for data entry.

metaOption - provides a global interface for disabling meta characters in ALP enterable popup controls

0 - meta characters enabled
1 - meta characters disabled

clickDelay - Provides the number of ticks (60 ticks = 1 second) which ALP will wait before automatically initiating data entry (see **AL_SetEntryOpts** and **AL_SetEnterable** form additional options).

allowPartialRow - Instructs ALP to display partial rows, thus causing ALP to draw in the exact areaRect defined in form.

0 - Don't display partial row, area will be resized vertically to only show full rows [default]
1 - Allow display of partial rows (area will not be resized)

AL_SetEnterable(area:L;columnNum:L;enterable:L;popupArray:X;menuPackRef:L;)

AL_SetEnterable now includes two new options for defining how enterability is used. You can use *enterableOpts* 4 or 5 to control meta characters for a given column. If you wish to define the meta character functions globally, use the routine **AL_SetInterface** (metaOption parameter).

enterableOpts - specifies the method of enterability for columnNum

Note: When using the *enterableOpts* 4 and 5, you can override the default setting configured by **AL_SetInterface** for a given area. Therefore, if you wish to configure ALP to globally disable meta characters for popup controls, you can do so using **AL_SetInterface**, then enable for a given column using **AL_SetEnterable**.

0 - non enterable
1 - enterable using typed characters only (default)
2 - enterable using popup menu only
3 - enterable using both typed characters and popup menu
4 - enterable using popup menu only (no meta characters)
5 - enterable using both typed characters and popup menu (no meta characters)

Example:

The following example will disable meta characters globally, for all ALP areas

AL_SetInterface(-1;-1;-1;-1) `0 as area parameter defines for all ALP areas

Then, you can enable meta characters for a given column in a specific area

AL_SetEnterable(eClients;2;2;atTest)

AL_GetCopyOpts(area:L;includeHiddenCols:L;fieldDelim:L;recDelim:L;fieldWrapper:L)

AL_GetCopyOpts will return the current values as defined by **AL_SetCopyOpts**.

area - ALP area reference

includeHiddenCols - returns the includeHiddenCols attribute

fieldDelim - returns the fieldDelim attribute

recDelim - returns the recDelim attribute

fieldWrapper - returns the fieldWrapper attribute

AL_GetFooters(area:L;footerList:X;option:L) -> result:L (New Routine)

-> area longint ALP Area Reference
-> footerList array text footer value list
-> options longint option reference

AL_GetFooters will return the footer information if you have enabled footers (see **AL_SetColOpts**). When calling **AL_GetFooters**, you have the option of including or omitting invisible column(s).

area - ALP Area Reference

footerList - A valid 4th Dimension array (TEXT or STRING) which will contain a list of all footers

options - When extracting footer information, you may optionally return only visible footers.

0 - No options, return all footer values
1 - Return only visible footer values

-> *result* - Returns a valid ALP Result Code

Example:

The following example will return all the footer values from visible columns only

AL_SetMiscOpts(eList;0;0;";";1;1)

AL_SetColOpts(eList;1;0;0;1) `hide last column

AL_SetFooters(eList;1;"Footer1")

AL_SetFooters(eList;2;"Footer2")

AL_SetFooters(eList;3;"Footer3")

ARRAY TEXT(atAL_FooterList;0)

\$ret=**AL_GetFooters**(eList;atAL_FooterList;1) `retrieve footer data, only visible columns

When the routine is complete, atAL_FooterList will contain two elements

atAL_FooterList{1} Footer1
atAL_FooterList{2} Footer2

AL_SetMiscRGBColor(area:L;selector:L;redPart:L;greenPart:L;bluePart:L)

-> area longint ALP Area Reference
-> selector longint routine selectors
-> red longint RGB - red part
-> green longint RGB - green part
-> blue longint RGB - blue part

AL_SetMiscRGBColor provides the ability to define miscellaneous color attributes using the associated RGB values. This routine is similar to **AL_SetMiscColors**.

area - ALP Area Reference

selector - Value 0 to 3

0 - The background color of the area in the header above the scroll bar. This area only exists if the header and the vertical scrollbar are displayed.

1 - The background color of the area in the footer below the vertical scrollbar. This area only exists if the footer and the vertical scrollbar are shown.

2 - The background color of the area to the left of the horizontal scrollbar. This area only exists if the horizontal scrollbar is shown and at least one column is locked.

3 - The background color of the area to the right of the horizontal scrollbar and the vertical scrollbar are shown.

RedColor - Desired 'red' color in RGB color pattern

GreenColor - Desired 'green' color in RGB color pattern

BlueColor - Desired 'blue' color in RGG color pattern

AL_SetEventCallback(area:L;callbackMethod:S;flag:L) -> result:L

-> area	longint	ALP Area Reference
-> callbackMethod	string	4D method
-> flags	longint	Flags
<- result	longint	Result

AL_SetEventCallback provides an enhanced method for dealing with all the events which could be triggered when working with ALP areas. While the existing event triggering system works (ALProEvt), the callback method provides enhanced developer control for more precise event handling.

-> *area* - ALP Area Reference

-> *callbackMethod* - A valid 4th Dimension method which will be called during ALP event execution.

-> *flags* - Handling flags

<0 - Reserved (will not update variables and will not force script execution)
 0 - do not execute area script, update 4D variables
 1 - execute area script, update 4D variables
 2 - do not execute area script, do not update variables
 3 - execute area script, no do not update 4D variables

Example:

The following will install an event callback

```
$ret:=AL_SetEventCallback (eALP;"AL_EventCallback")
```

CallbackMethod is 4D method with following declaration:

```
`PM: AL_EventCallback
`LM: 03/12/07

C_LONGINT($0) `status, return 0
C_LONGINT($1;$area) `ALP area
C_LONGINT($2;$alpEvent) `alp event
C_LONGINT($3;$alpEventMod) `event modifier - unused now, may be used later for passing additional info about the event
C_LONGINT($4;$col) `column - last clicked column
C_LONGINT($5;$row) `row - last clicked row
C_LONGINT($6;$modifiers) `modifiers
C_STRING(255;$7;$tip) `tip string
C_STRING(255;$8;$areaName) `plug-in area name (see AL_SetAreaName)
```

*** Control click events are now reported to event method immediately (it means when the mouse is still down.) ***

-> *flags* - Compatibility flag that defines how and when an area script is executed.

0 = compatible mode - area script is executed the same way as now
 1 = area script is executed except for events that require posting cmd-\ (single click and scroll events)
 2 = area script is not executed at all

-> *results* - Controls action of event execution.

0 = indicates event was handled by callback method
 1 = indicates event was not handled by callback method, normal ALP events will execute

Return value should be 0.

Note:

\$2 contains ALP event as passed by ALProEvt variable to area script. There are no new event added (so they can be in the future.) The handling of events is similar, so if user wants to have single and double clicke reported, ALP will still wait for double-click time to decide if it received single or double click.

Example:

The following example installs an event callback method which is executed during any ALP event.

Event callback is initialized with call

```
$err:=AL_SetEventCallback(area;"CallbackMethod";compatibilityFlag)
```

AL_SetCellValue(area:L;row:L;column:L;alphaNumericData:S;pictData:P)

-> area	longint	ALP Area Reference
-> row	longint	row
-> column	longint	column
-> alphaData	text	alphanumeric value
-> pictData	pictData	picture data

AL_SetCellValue provides the ability to update the contents of a given cell. You can set either alphanumeric or picture data.

Parameters:

-> *area* - ALP Area Reference

-> *rowNumber* - Selected row number.

-> *columnNumber* - Selected column number.

-> *alphaNumericData* - Alphanumeric (non-picture) data you wish to use as new value.

-> *pictData* - Picture data you wish to use as new value.

Example:

See AL_GetRow for an example of using AL_SetCellValue.

AL_GetRow(area:L) -> rowNumber:L

-> area longint ALP Area Reference
<- row longint Returns selected row number

AL_GetRow will return the selected row number for the supplied ALP area.

Parameters:

-> *area* - ALP Area Reference
<- *rowNumber* - Selected row number.

Example:

The following example will return the selected column and row numbers (cell), then use the **AL_GetCellValue** routine to return the associated data displayed in the selected cell.

Case of

```
:(Form event=On plugin_area)
$row:=AL_GetRow(eArea)
$col:=AL_GetColumn(eArea)
$ret:=AL_GetCellValue(eArea;$row;$col;$sData)
$sData:="new value"
AL_SetCellValue(eArea;$row;$col;$sData)
AL_UpdateArrays(eArea;-1)
```

End case

AL_GetCellOpts(area:L;cellSelection:L;moveWithData:L;optimization:L)

AL_GetCellOpts will return the current values set by **AL_SetCellOpts** (or default values if this routine has not been called).

area - ALP area reference
cellSelection - returns the cell selection attribute
moveWithData - returns the moveWithData attribute
optimization - returns the optimization attribute

AL_SetCellBorder(area:L;column:L;row:L;borderLeft:L;borderTop:L;borderRight:L;borderBottom:L;offset:L;width:L;redColor:L;greenColor:L;blueColor:L)

-> area longint ALP Area Reference
>- row longint row
>- column longint column
>- borderLeft longint Draw left border
>- borderTop longint Draw top border
>- borderRight longint Draw right border
>- borderBottom longint Draw bottom border
>- offset longint Offset from cell boundary in points
>- width longint Width of line
>- redColor longint RGB red color
>- greenColor longint RGB green color
>- blueColor longint RGB blue color

AL_SetCellBorder provide the ability to set the border style when users are editing cells (enterable entry required).

area - ALP area reference
row - row of cell where border will be applied
column - column of cell where border will be applied
borderLeft - draw left border
borderTop - draw top border
borderRight - draw right border
borderBottom - draw bottom border
offset - Offset from cell boundary in points
0 = if border should be drawn at cell boundary (default)
width - width of line. May use fractional value but only integer width can be drawn in content of the cell.
redColor - RGB red color used for border
greenColor - RGB green color used for border
blueColor - RGB blue color used for border

Example:

The following example will configure the cell border in **CellEntry** callback method.

Case of

:(Form event=On load\$)

\$row:=AL GetRow(eArea)

\$col:=AL GetColumn(eArea)

\$ret:=AL GetCellValue(eArea;\$row;\$col;\$sData)

\$sData:="new value"

AL SetCellValue(eArea;\$row;\$col;\$sData)

AL UpdateArrays(eArea;-1)

End case

Compatibility Information

AreaList Pro™ is fully compatible with 4D/4D Server 2003 or greater (including 4D 11.3/11.4). AreaList Pro™ is compatible with Macintosh and Windows clients.

Technical Support

Technical support for AreaList Pro will be provided via electronically via e-mail or our online support reporting system. You are encouraged to use the online web reporting form as it will be correctly routed to the appropriate support personnel.

Tel (714) 375-4252

Fax (714) 848-0382

<<http://www.asgsoft.com/support/>>

<<mailto:support@asgsoft.com>>