10 Drag and Drop

AreaList Pro enables rows, columns and cells to be dragged and dropped from and to AreaList Pro areas.

Overview

You can control which areas can be dragged from or to, what options are available (e.g. whether multiple rows can be dragged or not), and what happens after a drop. Row dragging can be initiated either by alt / option-clicking on an item (cell, row, or column) and dragging it or by simply dragging it, depending on how it has been configured.

When an item is clicked and dragged, the pointer will change.

If the drop is allowed, the pointer will have a plus symbol attached to it when it hovers over the "drop" area: ⊕

If the drop is not allowed by the destination object, you'll see a "no entry" sign instead: ⊘

Dragging

You can allow rows, columns and cells to be dragged from an AreaList Pro area and dropped to various destinations:

- a row, a column or a cell in the same AreaList Pro area
- a row, a column or a cell in another AreaList Pro area
- a day, an event or a banner in a CalendarSet area
- any 4D droppable object
- another application that accepts text

Dropping

You can also allow dropping onto rows, columns and cells in an AreaList Pro area from various sources:

- row(s), a column or cell(s) from the same AreaList Pro area
- row(s), a column or a cell(s) from another AreaList Pro area
- events and banners from a CalendarSet area
- any 4D draggable object
- text or other contents displayed in another application window
- a document in a MacOS Finder or Windows Explorer window

Item types

When dragging and dropping between AreaList Pro areas, or within the same area, the destination item will match the source item:

- row(s) will be dropped onto a row
- column will be dropped onto a column (use the header to select the source column that you want to drag)
- cell(s) will be dropped onto a cell

Controlling the Drag and Drop

You can control each AreaList Pro area's draggability and droppability:

- if the area can be dragged from
- if the area can be dropped to
- which item types (rows, columns and / or cells) can be dropped
- which AreaList Pro or CalendarSet areas can be the source or destination
- if external sources (non-AreaList Pro / CalendarSet objects) are allowed
- whether an attempted a drop on the area is accepted or rejected
- what happens after a drop

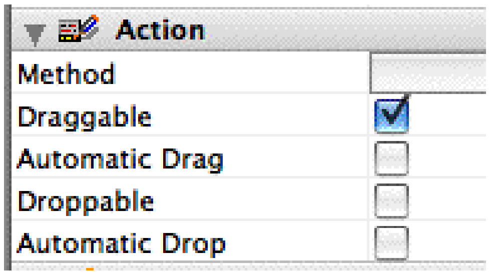
Configuring Drag and Drop

You must configure AreaList Pro to allow dragging out of and into an AreaList Pro area.

Setting the 4D Object Properties

The first thing you must do is select the Drag and Drop properties for the AreaList Pro areas as 4D objects.

1. Select the source object (the AreaList Pro area that you want to enable dragging **from**)
2. In the **Action** topic of the **Property List** dialog, select the **Draggable** checkbox:



Action

Method	
Draggable	✓
Automatic Drag	☐
Droppable	☐
Automatic Drop	☐

3. Select the destination object (the AreaList Pro area that want to drop **to**)
4. In the **Action** topic of the **Property List** dialog, select the **Droppable** checkbox.

An area can be both Draggable and Droppable, which will amongst others enable drag and drop within the area.

Note: in compatibility mode, the AreaList Pro area is draggable and droppable even if it is not set as draggable or droppable in the object properties.

Drag and drop Properties

AreaList Pro properties provide the control necessary to allow or disallow dragging within an area, between two or more areas or from non-plugin objects. You can allow dropping from and onto rows, columns and / or cells.

In order to facilitate dragging and dropping, you need to tell AreaList Pro which area(s) you want to allow the dragging between, and specify various options such as which objects can be dragged (rows, columns, and/or cells), which areas those objects can be dragged to and from, and whether certain keys - such as the Alt / Option key - will have any particular effect.

Access “codes” overview

To allow dragging out of AreaList Pro, you must pass an “access code” for each type of item (rows, columns and / or cells) that can be dragged from this area. You must specify at least one code to enable dragging, using ALP_Area_DragSrcXXXCodes properties (where XXX is Row, Col or Cell).

Any number of codes can be passed. Allowing many codes provides for more flexibility in enabling and disabling dragging between various areas.

In order to allow dropping into AreaList Pro, you must pass an “access code” for each type of item (rows, columns and / or cells) that can be the destination of a drop. You must specify at least one code to enable dropping, using ALP_Area_DragDstXXXCodes properties (where XXX is Row, Col or Cell). As with source code properties, any number of codes can be passed for flexibility reasons.

Property list

AreaList Pro provides a number of properties that you can use to set and find the required details. Use these properties with the ***ALP_SetArea…*** and ***ALP_GetArea…*** commands:

Property	Type	Description
ALP_Area_DragDataType	longint	Dragged data: 1 = row(s) 2 = column 3 = cell(s)
ALP_Area_DragDstArea	longint	Destination area
ALP_Area_DragDstCell	longint	Destination area cell
ALP_Area_DragDstCellCodes	text	Drag destination cell codes
ALP_Area_DragDstCol	longint	Destination area column
ALP_Area_DragDstColCodes	text	Drag destination column codes
ALP_Area_DragDstProcessID	longint	4D’s process ID of the destination area
ALP_Area_DragDstRow	longint	Destination area row
ALP_Area_DragDstRowCodes	text	Drag destination row codes
ALP_Area_DragProcessID	longint	4D’s process ID of the source area
ALP_Area_DragSrcArea	longint	The dragged AreaList Pro area
ALP_Area_DragSrcCell	longint	Source area cell
ALP_Area_DragSrcCellCodes	text	Drag source cell codes
ALP_Area_DragSrcCol	longint	The source area column
ALP_Area_DragSrcColCodes	text	Drag source column codes
ALP_Area_DragSrcRow	longint	Source area row
ALP_Area_DragSrcRowCodes	text	Drag source row codes

Alt/Option key

A default setting that you may want to change is the “drag with Alt key” option.

The default setting for this is that the user must hold down the Alt or Option key to effect a drag. You can turn this off by setting the ALP_Area_DragOptionKey property for the source area to False - for example:

AL_SetAreaLongProperty (ProductList;ALP_Area_DragOptionKey;0) //don’t need alt key to drag

What are access “codes”?

The access codes that are passed in ALP_Area_DragSrcXXXCodes and ALP_Area_DragDstXXXCodes (where XXX is Row, Col or Cell) are used to enable dragging between specific drag partners.

These drag partners can be the same AreaList Pro area, different AreaList Pro areas, CalendarSet plugin areas, text selections from 4D or other applications and external documents.

- Setting at least one source access code to an area will make it draggable (from the specified object type).
- Setting at least one destination access code to an area will make it droppable (on the specified object type).

Drag and drop between plugin areas

When dragging from and to AreaList Pro or CalendarSet areas (or within the same area) the source of the drag and the target of the drop are designated as drag and drop partners*.*

You need to tell AreaList Pro (and CalendarSet if used) what these partnerships are, and to do this you create **access codes**.

An access code is simply a text code that you create. Let’s say you have a layout that contains two AreaList Pro areas: **ProductList** and **SelectedItems**, and you want to enable items to be dragged from ProductList and dropped onto SelectedItems. You might decide on the access code “select”.

To enable row dragging you will need two lines of code:

AL_SetAreaTextProperty (ProductList;ALP_Area_DragSrcRowCodes;"select")

AL_SetAreaTextProperty (SelectedItems***;***ALP_Area_DragDstRowCodes***;"select")***

You can list any number of access codes.

For example, suppose you have four AreaList Pro areas on a form - AreaA, AreaB, AreaC and AreaD. You want to allow drag and drop from AreaA to AreaB or AreaC, and from AreaD to AreaC but not AreaB:

1. Create two access codes: “dropB” and “dropC”.

2. Set the access code properties for the four areas as follows:

AL_SetAreaTextProperty (AreaA;ALP_Area_DragSrcRowCodes;"dropB|dropC")

AL_SetAreaTextProperty (AreaD;ALP_Area_DragSrcRowCodes;"dropC")

AL_SetAreaTextProperty (AreaB;ALP_Area_DragDstRowCodes;"dropB")

AL_SetAreaTextProperty (AreaC;ALP_Area_DragDstRowCodes;"dropC")

Note that the items in the list of codes are separated by a pipe character: “dropB|dropC”

You can specify different codes for cells, rows, and columns.

Note: as opposed to CalendarSet commands where each access code is a parameter by itself: "dropB";"dropC", with AreaList Pro properties all access codes are passed in the same string (list of codes separated by ‘|’).

That’s all you need to do to enable basic drag and drop functionality between two areas with default settings.

When a drag and drop takes place, the drag sender’s plugin code communicates its access codes to the drop receiver’s plugin code. The drop receiver will compare the access codes of the sender to its own codes. If any of the codes match, the drop is allowed.

This mechanism allows a number of combinations between several drag and drop partners.

Note: access codes are strictly compared, taking into account case and diacritics.

Example (one area)

The following is an example of enabling the dragging and dropping of events within the same AreaList Pro area by setting a unique identifier that only enables dragging within this area.

```
  //enable drag rows to rows within this area
vSelfStr:="ALParea"+String(eList)  //only allows dragging and dropping within this area
AL_SetAreaTextProperty (eList;ALP_Area_DragSrcRowCodes;vSelfStr)
AL_SetAreaTextProperty (eList;ALP_Area_DragDstRowCodes;vSelfStr)
```

Example (two areas)

Suppose we have a form on which there are two AreaList Pro areas: **ProductList** contains a list of products, and **SelectedItems** contains a list of products that have been selected from the list. We want to allow users to add products to **SelectedItems** by dragging rows from **ProductList**.

We’ve set up the two areas as described above (under Set the access code properties). Three arrays have been initialised and added to **SelectedItems** (atProductPurch, aiQty, and arTotal).

When a product is dropped onto **SelectedItems** we need to add a row to each of the **SelectedItems** arrays and fill them with the appropriate data if the product hasn’t already been selected, or update the arrays if it has already been selected.

In the object method for **SelectedItems** we call a method called ***AddProductToOrder***:

```
Case of
    : (Form event=On Drop)
    AddProductToOrder (Self)
End case
```

```
  //AddProductToOrder project method
C_LONGINT($DropArea;$SourceRow;$ProductRow)
$DropArea:=$1->
$SourceRow:=AL_GetAreaLongProperty ($DropArea;ALP_Area_DragSrcRow)
GOTO SELECTED RECORD([product];$SourceRow)
$ProductRow:=Find in array(atProductPurch;[product]product_name)
If ($ProductRow<1)
    APPEND TO ARRAY(atProductPurch;[product]product_name)
    APPEND TO ARRAY(aiQty;1)
    APPEND TO ARRAY(arTotal;[product]retail_price)
Else
    aiQty{$ProductRow}:=aiQty{$ProductRow}+1
    arTotal{$ProductRow}:=aiQty{$ProductRow}*[product]retail_price)
End if
AL_SetAreaLongProperty ($DropArea;ALP_Area_CheckData;0)  //tell AreaList Pro we expanded the arrays
```

Drag and drop with external objects

External objects are defined in this context as non AreaList Pro (or CalendarSet) plugin areas:

- Dragging from and dropping to 4D fields, variables or any droppable / draggable object
- Dragging from and dropping to 4D text selections (like a text variable content)
- Dragging from and dropping to text selections in other applications (open windows from e.g. a text editor)
- Dragging from external files (no dropping to)

Since external objects obviously don't support access codes they will be potential recipients of any drag from a draggable AreaList Pro area, or source of any drop to a droppable AreaList Pro area.

Any source access code will make a AreaList Pro area draggable to external objects.

The "_external" special access code must be used as a destination access code in order to make an AreaList Pro area droppable from external objects:

- "_external" can only be used as a destination access code and is the only way to make an AreaList Pro area accept a drop from objects other that AreaList Pro or CalendarSet areas
- "_external" as a destination access code means "drop onto this AreaList Pro area from any external object"

We can modify the example above to allow dragging from an external object to area B:

AL_SetAreaTextProperty (AreaA;ALP_Area_DragSrcRowCodes;"dropB|dropC")
//drag to B, C and external objects

AL_SetAreaTextProperty (AreaD;ALP_Area_DragSrcRowCodes;"dropC") //drag to C and external objects

AL_SetAreaTextProperty (AreaB;ALP_Area_DragDstRowCodes;"dropB|_external")
//accept drop from A & external objects

AL_SetAreaTextProperty (AreaC;ALP_Area_DragDstRowCodes;"dropC") //accept drop from A & D

When dragging from an external object to an AreaList Pro area, the destination type will either be a row or a cell according to the area's current selection mode (ALP_Area_SelType).

Using the Event callback method

The processing of the drop event can be handled either in the event callback method or in the On Drop event on the AreaList Pro area method.

Generally it is best to handle the processing in the On Drop form event.

AL_GetAreaLongProperty with ALP_Area_AlpEvent is used to find out that a drop occured: in this case, the AL Row drop event, AL Column drop event, AL Cell drop event or AL Object drop event value tells us that something was dropped onto the area, and what was dropped (row(s), column, cell(s) or non-AreaList Pro object).

You can also fine-tune your control over the user's drag and drop from the event callback method set by the ALP_Area_CallbackMethOnEvent property (callback methods are explained in detail elsewhere):

AL_SetAreaTextProperty (ProductList;ALP_Area_CallbackMethOnEvent;"AlpEventCallback")

During a drag and drop, this method will be called under two circumstances:

- When a drop is attempted from an external object, while the pointer is still hovering over the AreaList Pro destination area: the $0 value returned by the callback will allow the subsequent drop action or not (as with setting access codes to control drag and drop between plugin areas): AL Allow drop event.
- After a drop has been performed, whatever the source was (plugin area or external object): AL Row drop event, AL Column drop event, AL Cell drop event or AL Object drop event.

Note: the callback method will **not** be called with AL Allow drop event when the source of the drag and drop is a plugin area, where access codes are used instead to specifically allow drag and drop between areas.

When the drop is completed, the form method / AreaList Pro area object (or callback method if previously set by ALP_Area_CallbackMethOnEvent) is executed. You can then determine what the last user action was using ALP_Area_AlpEvent (form / object method) or $2 (callback method).

Note: the event reported by ALP_Area_AlpEvent will never be AL Allow drop event for an external object source, but the callback will receive this event in $2. See “Allow drop” below.

The source area's last event is AL Row drag event, AL Column drag event or AL Cell drag event (drag events). The destination area's last event is AL Row drop event, AL Column drop event, AL Cell drop event or AL Object drop event.

The drag events are not reported if the drag ended as a drop into the same AreaList Pro area.

Note: AL Row drag event, AL Column drag event and AL Cell drag event drag event should no longer be used anyway. Use the AL Row drop event, AL Column drop event, AL Cell drop event and AL Object drop event callback events instead, or the form / object method On Drop 4D event using ***AL_GetAreaLongProperty*** with ALP_Area_AlpEvent.

Allow drop

The callback method is called with the AL Allow drop event event when an external object is dragged over the area. It is used to allow or reject the drop.

For example:

```
Case of
    : ($2=AL Allow drop event)
        $0:=1  // allow
End case
```

Note: never display a window in the callback while processing this AL Allow drop event, including **TRACE** or **ALERT**: 4D would freeze (ACI0087433).

This event callback method (actually a function in this case) receives six parameters, and must return a result:

- $1 is the destination AreaList Pro area reference.
- $2 is the event, in this case AL Allow drop event
- $0 is expected by AreaList Pro, with a special meaning for this event: 0 (disallow drop) or 1 (allow drop).

Note: if no callback is set the drop from an external object will **not** be accepted (even when "_external" is set as a destination access code for the area). Therefore the callback is the only way to allow a drop from an external object on a droppable area.

The pointer shape will depend upon this result:

- plus symbol if the drop is allowed:
- “no entry” sign if the drop is non allowed:

Note: once an allowed destination has been rolled over, the pointer will stay as “+” even when you subsequently move to a place where the drop is not allowed: the drop will nevertheless be allowed and the On Drop form event will be triggered. This is due to a limitation of 4D, which does not call the plugin again to ask if the drop is allowed unless you leave the area and re-enter it again.
In this case ALP_Area_DragSrcArea, ALP_Area_DragSrcRow, ALP_Area_DragSrcCol and ALP_Area_DragSrcCell will return zeros.

After the drop

When an item is dropped onto an AreaList Pro area, the following information is available to you:

- Notification that a drop occurred
- Which item was dragged and which type (row, column or cell)
- Where the item was dragged from (this area or another area, or another kind of object)
- The type of data that was the recipient of the drop (row, column or cell)

The processing of the drop event can either be handled in the callback with the AL Object drop event event or the form method / AreaList Pro area object method with the On Drop event, which is always executed in the context of the destination area / process.The call sequence is:

1. Callback method with AL Row drop event, AL Column drop event, AL Cell drop event or AL Object drop event in $2

2. Area object method with On Drop form event

3. Form method with On Drop form event

Note: when displaying AreaList Pro in an external window there is no form / object method to execute, therefore the callback is the only way to control drag and drop in this context.

The callback method receives six parameters as usual, the first two being:

- $1 the destination AreaList Pro area reference.
- $2 the event code, here AL Row drop event, AL Column drop event, AL Cell drop event or AL Object drop event

You can also determine which AreaList Pro object type was dropped using ALP_Area_DragDataType.

To determine which area was the source of the drag, use ALP_Area_DragSrcArea. This property returns the area reference (a long integer) and ALP_Area_DragProcessID is the process ID of the source area, whether it is the same AreaList Pro area or another AreaList Pro area, or a CalendarSet area (the area reference will be negative in this case).

To find out which row, column or cell was dropped, use either ALP_Area_DragSrcRow, ALP_Area_DragSrcCol or ALP_Area_DragSrcCell.

The above properties will return a single source row, column or cell. For example ALP_Area_DragSrcRow is the selected row (ALP_Area_SelRow) at the time the drag was initiated.

If you want to handle multiple rows (ALP_Area_DragRowMultiple+ALP_Area_SelMultiple) or cells as sources, use the Object Properties with the Objects command theme to retrieve the source area's selection:

```
ARRAY LONGINT(aRows;0)
$error:=AL_GetObjects (eList;ALP_Object_Selection;aRows)  //get the rows selected by user
```

When dragging to/from another area or a 4D object, that object can either reside in the same window or on another window, which may require use of 4D’s process communication commands to take action on the drop.

Note: when dragging and dropping to or from other objects, AreaList Pro is only providing a user interface to the drag, and notifying you, the developer, that the drop has occurred. You are responsible for manipulating any arrays or other data structures.

See also Reordering after dragging within one area and Dragging to a 4D object.

Receiving a drop from a non-AreaList Pro Object

As well as dragging between two AreaList Pro areas, you can also drag between non-AreaList Pro objects and AreaList Pro areas — for example, an event or a banner from CalendarSet, another 4D object, a text selection in any (drag and drop savvy) application window or a drop from an external document.

Receiving a drop from a CalendarSet area is identified by ALP_Area_DragSrcArea returning a negative value (opposite of the CalendarSet area reference) and its process in ALP_Area_DragProcessID.

If the source of the drag is an external object, the callback (if set for the area) will be triggered twice:

- when dragging over the area ($2=AL Allow drop event)
- after the drop ($2=AL Object drop event)

In both cases (and in the 4D object/form method after the drop) ALP_Area_DragSrcArea will be zero and ALP_Area_DragProcessID will be:

- the 4D originating object's process if the source is a 4D object
- -1 if the source is a document on disk or another application

Also in both callback calls the content of the dragged and dropped object is the pasteboard data (text, picture and/or other).

Use **GET PASTEBOARD DATA** to analyze the dragged data and allow the drop (AL Allow drop event) or process it (AL Object drop event).

Allowing the drop from external objects in the callback

When dragging from non-plugin objects, you can use an event callback method to “catch” the user action and allow it or not (event callback methods are set with the ALP_Area_CallbackMethOnEvent property). If no callback is set, the drop will be allowed.

In order for external object drag and drop to be disallowed once an area has been made droppable with the "_external" destination access code, the callback method must be set and handle the AL Allow drop event case.

See the External documents example below.

CalendarSet

The source CalendarSet area is referenced as a negative value in the ALP_Area_DragSrcArea property (opposite of the CalendarSet area reference).

Plugin area references use sequential numbering: 1 can be AreaList Pro and CalendarSet area references at the same time, therefore CalendarSet area #1 will be referenced from AreaList Pro's point of view as -1, to differentiate from AreaList Pro area #1.

```
C_LONGINT($srcArea;$srcCSArea)
If (Form event=On Drop)  //here we use the object method, no callback set
   $srcArea:=AL_GetAreaLongProperty (eList; ALP_Area_DragSrcArea)
   $srcCSArea:=-$srcArea  //CalendarSet's area reference (note the minus sign)
   If ($srcCSArea=CSappointments)  //is this CalendarSet area our appointment calendar?
     //Do something with the appointements
   End if
End if
```

Refer to the CalendarSet manual regarding accepting a drop in a CalendarSet area.

4D

You can use the following code in the On Drop event when accepting a drop from a 4D object:

```
DRAG AND DROP PROPERTIES($srcObject;$srcElement;$srcProcess)
```

Note: $srcObject is **Nil** if the source 4D object has Automatic Drag enabled.
$srcObject is also **Nil** if it comes from a different application (or 4D instance).

Then you can use the following code to check what has been dropped:

```
ARRAY TEXT($4Dsignatures;0)
ARRAY TEXT($nativeTypes;0)
ARRAY TEXT($formatNames;0)
GET PASTEBOARD DATA TYPE($4Dsignatures;$nativeTypes;$formatNames)
```

Note: the On Drop event code will work correctly after the drop when used in an area’s object method but in an event callback the form event is zero and drag & drop properties from 4D will not function. However the pasteboard can still be analyzed in the callback with both events AL Allow drop event and AL Object drop event.

External documents

After Allowing the drop from external objects in the callback, the AL Object drop event AreaList Pro event or the On Drop 4D event are used to open the document according to the pathname retrieved from the pasteboard, then process it.

Let's suppose we want to import some data into a series of arrays by dropping a text file onto an AreaList Pro area. We've saved a spreadsheet that contains information on some new Nuts products as a tab-delimited text file:

"Macadamia nuts, 50g"	MAC-001	Nuts	Snack-sized bag of nuts	2.5
"Macadamia nuts, 100g"	MAC-002	Nuts	Family-sized bag of nuts	4.5
"Pecans, 50g"	PEC-001	Nuts	Snack-sized bag of pecan nuts	3.5
"Macadamia nuts, 100g"	PEC-002	Nuts	Family-sized bag of pecan nuts	5.5
"Dry roasted Peanuts 50g"	PEA-001	Nuts	Snack-sized bag of salted, roasted peanuts	2.5

When this text file is dropped onto a list of products, we want to create a new row for each new product and populate the appropriate arrays with the product's details.

Setting up the Area

1. Create a new AreaList Pro area on your form
2. In the Property List, select the **Droppable** option under the **Action** topic, and the **On Drop** event.
3 Create a callback method:

```
//AlpEventCallback
C_LONGINT($1) //AreaList Pro object reference
C_LONGINT($2) //AreaList Pro event
C_LONGINT($3) //4D event
C_LONGINT($4) //last clicked column (or column under the pointer for mouse moved event)
C_LONGINT($5) //last clicked row (or row under the pointer for mouse moved event)
C_LONGINT($6) //modifiers
C_LONGINT($0)
Case of
  : ($2=AL Allow drop event)
    $0:=1 //allow
End case
```

4. Assign that callback method to the area:

```
Case of
  : (Form event=On Load)
    AL_SetAreaTextProperty (ProductList;ALP_Area_CallbackMethOnEvent;"AlpEventCallback")
End case
```

This code can go either on the AreaList Pro destination area object method or in the form method.

Handling the Drop

Add some code to the On Drop event section of the AreaList Pro destination area object method:

```
Case of
   : (Form event=On drop)
      DRAG AND DROP PROPERTIES($srcObject;$srcElement;$srcProcess)
      $dragSource:=AL_GetAreaLongProperty (Self->;ALP_Area_DragSrcArea)
      If ($dragSource=0)  //not an AreaList Pro area
         If (Nil($srcObject))  //external source
            GET PASTEBOARD DATA(“com.4d.private.file.url";$data)  //gets file pathname
            If (OK=1)
               PLATFORM PROPERTIES($platform)
                  If ($platform=Windows)
                     $LineDelimit:=Char(10)
                  Else
                     $LineDelimit:=Char(13)
                  End if
               $path:=Get file from pasteboard(1)  //first file
               $FileType:=Document type($path)
               If (($FileType="txt") | ($FileType="text"))
                  SET CHANNEL(10;$path)  //open the file
                  While (OK=1)  //file opened OK & more data to receive
                  RECEIVE PACKET($tdata;$LineDelimit)  //get one row
                     ARRAY TEXT(atVals;0)
                     explode ($tdata;9;->atVals)  //parse the text into the array (see below)
                     If (Size of array(atVals)=5)
                        APPEND TO ARRAY(atName;Replace string(atVals{1};Char(34);""))
                        APPEND TO ARRAY(atCode;atVals{2})
                        APPEND TO ARRAY(atType;atVals{3})
                        APPEND TO ARRAY(atDesc;Replace string(atVals{4};Char(34);""))
                        APPEND TO ARRAY(arWprice;Num(atVals{5}))
                     End if
                  End while
                  SET CHANNEL(11)  //close the file
                  SORT ARRAY(atName;atCode;atType;atDesc;arWprice)
                  AL_SetAreaLongProperty (Self->;ALP_Area_CheckData;0)
                   //tell AreaList Pro we expanded the arrays
               End if
            End if
         End if
      End if
End case
```

5. To test it, load the form and then drop the text file onto the area. The On Drop event will execute and the five new products will be added to the area.

Utility

The ***explode*** project method splits a delimited line of text into an array:

```
//explode
//explodes a text string into parts using designated separator character
//and returns them in an array
//parameters: $1 = the text string
//$2 = the separation character(ASCII value)
//$3 = pointer to the array to put the values in
//supports only TEXT arrays
//array must be declared and zero'd first
//example: explode (tText;9;->atVals))
C_TEXT($text;$char)
$text:=$1
$char:=Char($2)
$Elements:=0
While (Length($text)>0)
    $pos:=Position($char;$text)
    If ($pos>0)
        $value:=Substring($text;1;$pos-1)
        $text:=Substring($text;$pos+1)
    Else
        $value:=$text
        $text:=""
    End if
    APPEND TO ARRAY($3->;$value)
End while
```

Hints and Tips

Here is some feedback from our support regarding Drag & Drop in AreaList Pro, which you may find useful in addition to the above explanations.

Feel free to ask for more using the AreaList Pro / PrintList Pro forum.

Row dragging in cell selection mode

Row dragging isn't enabled when an AreaList Pro object is in cell selection mode.

To set the selection mode, use the ALP_Area_SelType property of ***AL_SetAreaLongProperty*** - for example:

AL_SetAreaLongProperty (area; ALP_Area_SelType;0) // selection mode = rows

Dragging a row to the bottom of the list

If you drag a row from another AreaList Pro area and release below the bottom existing row, ***AL_GetAreaLongProperty*** with ALP_Area_DragDstRow will return the last existing row. This behaviour is compatible with versions 8.x.

Dragging "onto" a row means just that: drag onto an existing row.

You can drop a row onto the empty area below the last row (you can also drop data into an empty AreaList Pro area – without any rows), but the last row is reported as the destination.

You can set the area to "insert" mode:

AL_SetAreaLongProperty ($area; ALP_Area_DragRowOnto; 0)

In this case, it will append the row to the list rather than inserting above the last existing row.

Drag Line property

ALP_Area_DragLine is used when the source (referenced area) does not have the ALP_Area_DragSrcRowCodes property set. This is also true for ALP_Drop_DragAcceptLine (with the destination area).

It is useless if you call ALP_Area_DragSrcRowCodes to set the matching codes between source and destination. Drag will only be allowed to a matching destination.

The option key values to ALP_Area_DragLine (1-3 = with option, 4-6 = without option) can be set using ALP_Area_DragOptionKey.

Using ALP_Area_DragSrcRowCodes makes it consistent with ALP_Area_DragSrcColCodes (which in turn makes ALP_Drop_DragAcceptColumn obsolete) and ALP_Area_DragSrcCellCodes.

Drag and drop and compatibility mode

If drag within the area does not work unless the compatibility is turned on, check the AreaList Pro object properties on the 4D form - it has to be draggable / droppable.

In compatibility mode, the area is draggable / droppable even if it is not marked as such on the 4D form (previous AreaList Pro versions ignored this setting).

Be aware that Drop has to be handled On Drop, not in the drag context as opposed to previous versions (the drop can end anywhere, not necessarily in an AreaList Pro area).

Reordering after dragging within one area

Rows

AreaList Pro will automatically reorder the rows (i.e. array elements) if all the following conditions are met.

- drag and drop occurs in the same area
- the area is in arrays mode
- ALP_Area_SelType is 0 (row selection)
- ALP_Area_DragRowOnto is 0 (the feedback to the user is "insert" - highlight between rows) - when using the old v8.x API, you have to use 0 as the fifth parameter in your call to ***AL_SetDrgOpts***
- ALP_Area_DragRowMultiple or ALP_Area_SelMultiple is 0

Otherwise your 4D code must process the drag as in the evtDragWithin method from Example 10.

Column

Only one colum at a time can be dropped within the area or to another destination.

If dropped within the area the dropped colum is moved (inserted) to the target column's position and the columns are automatically reordered.

If the area is not in compatibility mode the grid order is updated (ALP_Object_Grid).

Note: the above behavior does not apply to Grids where rows include several lines.

Selection mode effects

Keep in mind that the selection mode (as defined by ALP_Area_SelType) may limit the ability to drag and drop depending on the object types (specified by their access codes).

- specifying column source access codes makes the area draggable: a column can be dragged
- specifying column destination access codes makes the area droppable: a column can be dropped
- specifying row source access codes in cell selection mode does not make the area draggable
- specifying row source access codes in row selection mode makes the area draggable: row(s) can be dragged
- specifying row destination access codes makes the area droppable: row(s) can be dropped; in cell selection mode, only from a different area (which must be in row selection mode)
- specifying cell source access codes does not make the area draggable when you are in rows mode
- specifying cell source access codes in cell selection mode makes the area draggable: cells(s) can be dragged
- specifying cell destination access codes makes the area droppable: cells(s) can be dropped; if in row selection mode, only from a different area (which must be in cell selection mode)

Dragging to a 4D object

Suppose you want to drag from AreaList Pro to another 4D object (non-AreaList Pro) and you need to collect information about the dragged row(s).

Since AreaList Pro simply starts a drag (when allowed), you must handle On Drag Over & On Drop in the 4D object/form method.

You can retrieve the source variable name from **DRAG AND DROP PROPERTIES** (to know that drag is from an AreaList Pro area) or you can directly access "net.e-node.alp.object" (which is a longint in native byte order) to get the AreaList Pro area reference.

Then you can get the dragged row number using ALP_Area_DragSrcRow (or selection if multiple-row drag is allowed).

For that to work, the 4D object must be droppable, must have enabled On Drop event and must not handle the drop automatically (“Automatic Drop” must not be checked).

```
C_LONGINT ($srcALP)
C_BLOB ($blob)
GET PASTEBOARD DATA ("net.e-node.alp.object";$blob)
If (OK=1)
   $srcALP:=BLOB to longint($blob;Native byte ordering)
   $row:=AL_GetAreaLongProperty ($srcALP;ALP_Area_DragSrcRow)
   // do something
End if
```

Disabling Drag and/or Drop with Read-only mode

It it possible to completely prevent dragging from or dropping to an area using the Read-only mode (ALP_Area_ReadOnly property). The respective bits in this property, if set, will supersede any relevant settings described in this chapter.

See Read-only mode.

See also Drag and Drop from the Upgrading from Previous versions of AreaList Pro section.