

Libraries_Manual

COLLABORATORS

	<i>TITLE :</i> Libraries_Manual		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 18, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Libraries_Manual	1
1.1	Amiga® RKM Libraries: 16 ASL Library	1
1.2	16 ASL Library / About Requesters	1
1.3	16 ASL Library / Creating a File Requester	2
1.4	16 / Creating a File Requester / Specifying Options with TagItems	3
1.5	16 // File Pattern Matching and Multiple Selects	4
1.6	16 / Creating a File Requester / ASL Requesters and Custom Screens	5
1.7	16 / Creating a File Requester / The Save Requester	5
1.8	16 / Creating a File Requester / The Directory Requester	6
1.9	16 ASL Library / Creating a Font Requester	6
1.10	16 / Creating a Font Requester / Specifying Options with TagItems	7
1.11	16 ASL Library / Calling Custom Functions from a Requester	8
1.12	16 / Calling Functions / Parameters Passed to Custom Hook Functions	9
1.13	16 ASL Library / Function Reference	10

Chapter 1

Libraries_Manual

1.1 Amiga® RKM Libraries: 16 ASL Library

This chapter describes the asl.library. The sole purpose of this library is to provide standard file and font requesters for application programs.

It is easier to understand the asl.library if you are familiar with some basic concepts of the Amiga operating system, especially TagItem arrays (described in the "Utility Library" chapter), Intuition screens and windows, graphics library font structures, and AmigaDOS pattern matching.

About Requesters	Calling Custom Functions from a Requester
Creating a File Requester	Function Reference
Creating a Font Requester	

1.2 16 ASL Library / About Requesters

Requesters are temporary sub-windows used for confirming actions or selecting options. The most common type of requester is a file requester which is used to pick a file name for a load or save operation.

Under 1.3 (V34) and earlier versions of the Amiga operating system there was limited support for requesters. Intuition provides simple requesters which can be used to request responses such as OK or Cancel from the user. More elaborate Intuition requesters can be created by adding additional features such as string gadgets, however the result of this is that each application writer develops their own style of requester. Hence, the asl.library has been added to Release 2 of the Amiga operating system to make requesters more consistent. With asl.library, requesters are also much easier to create and take less memory.

The ASL Library Requires Release 2.

The asl.library requires Release 2 of the Amiga operating system, so only applications running under Release 2 and later versions of the Amiga OS can call its functions.

Requesters are very flexible and can be used for many different purposes.

The Release 2 asl.library supports the two most common type of requesters:

- * File requesters for choosing a file name in a load or save operation
- * Font requesters for choosing a font in a text operation

1.3 16 ASL Library / Creating a File Requester

Opening an ASL requester requires the use of three functions:

```
APTR request = AllocAslRequest( unsigned long type,
                                struct TagItem *tagList );
BOOL success = AslRequest( APTR request, struct TagItem *tagList );
void          FreeAslRequest( APTR request );
```

The first function you should call is `AllocAslRequest()`. This allocates the main data structure you will use, either a `FileRequester` structure or a `FontRequester` structure. You specify the type of requester you want for `AllocAslRequest()` by setting the type argument. This can be one of two values defined in `<libraries/asl.h>`: either `ASL_FileRequest`, to ask for a `FileRequester` structure, or `ASL_FontRequest`, to ask for a `FontRequester` structure.

Here's a listing of the `FileRequester` structure. (The `FontRequester` structure is discussed in more detail later in this chapter.)

```
struct FileRequester {          /* (from <libraries/asl.h>) */
    APTR    rf_Reserved1;
    BYTE    *rf_File;           /* Filename pointer */
    BYTE    *rf_Dir;            /* Directory name pointer */
    CPTR    rf_Reserved2;
    UBYTE    rf_Reserved3;
    UBYTE    rf_Reserved4;
    APTR    rf_Reserved5;
    WORD    rf_LeftEdge,rf_TopEdge; /* Preferred window pos */
    WORD    rf_Width,rf_Height;    /* Preferred window size */
    WORD    rf_Reserved6;
    LONG    rf_NumArgs;           /* A-la WB Args, for multiselects */
    struct WBArg *rf_ArgList;
    APTR    rf_UserData;         /* Applihandle (you may write!!) */
    APTR    rf_Reserved7;
    APTR    rf_Reserved8;
    BYTE    *rf_Pat;             /* Pattern match pointer */
};                               /* note - more reserved fields follow */
```

Whichever requester type you use, you must allocate the requester structure with the `AllocAslRequest()` function call. Do not create the data structure yourself. The values in this structure are for read access only. Any changes to them must be performed only through `asl.library` function calls.

Once you have set up a requester structure with `AllocAslRequest()`, call `AslRequest()` to make the requester appear on screen. `AslRequest()` takes the requester data structure as an argument using it as a specification for the requester that it creates on screen.

Figure 16-1: The ASL File Requester

AslRequest() is always synchronous to the calling program. That is, control does not return to your program until the user makes a selection or cancels. AslRequest() returns TRUE, if the user selects a file (or a font). In that case the file (or font) name that the user selected is returned in the requester data structure. AslRequest() returns FALSE if the user cancels the requester or the requester failed for some reason.

When you have finished with a requester use the FreeAslRequest() function to deallocate the requester data structure.

Specifying Requester Options with TagItems
 Simple ASL File Requester Example
 File Pattern Matching and Multiple Selects
 ASL Requesters and Custom Screens
 The Save Requester
 The Directory Requester

1.4 16 / Creating a File Requester / Specifying Options with TagItems

Both AllocAslRequest() and AslRequest() accept a TagItem array or tag list as an argument. The tag list is used to initialize or alter the values in the requester data structure.

A single TagItem consists of a tag name and an associated tag value. Tag items that apply to the asl.library are defined in <libraries/asl.h>. The basic tag items (used in the first example listed below) are:

Requester Tag Name	Used For
-----	-----
ASL_Hail	String to place in the title bar of the requester window
ASL_Width	Requester window width
ASL_Height	Requester window height
ASL_LeftEdge	Requester window y origin
ASL_TopEdge	Requester window x origin
ASL_OKText	String to place in OK gadget of requester
ASL_CancelText	String to place in Cancel gadget of requester
ASL_File	Default file name (for file requesters only)
ASL_Dir	Default directory name (for file requesters only)

Note that you are currently limited to about six characters for the replacement text if you use either the ASL_OKText or ASL_CancelText tags to change the text that appears in the OK and Cancel gadgets.

The contents of an ASL requester data structure are preserved across calls to AslRequest(). So, until the requester is freed, tag settings and user selections will remain in the data structure unless they are altered by tags in subsequent calls to AslRequest(). This is very useful because it allows the requester to remember and redisplay the user's previous selections. However, this also means that the programmer must assure that

any addresses passed in ASL tags remain valid, or are refreshed on each call to `AslRequest()`.

Generally, options that you wish to specify only once, such as the initial position and size, should be specified as tags when you allocate the requester. Options that you wish to control for each use of the requester should be passed as tags each time the requester is opened with `AslRequest()`.

1.5 16 // File Pattern Matching and Multiple Selects

A file requester can filter out certain file and directory entries using the "wildcard" feature of AmigaDOS. To activate the wildcard feature for a file requester, you use the `ASL_FuncFlags` tag. Each bit in the `ASL_FuncFlags` tag item controls a special option of the requester depending on its type (file or font). See `<libraries/asl.h>` for a complete listing of the options that the `ASL_FuncFlags` tag controls.

File Requester Flag	Used For
-----	-----
<code>FILE_PATGAD</code>	Enables the file name pattern matching gadget
<code>FILE_MULTISELECT</code>	Enables multiple selection of files
<code>FILE_NEWIDCMP</code>	Use separate IDCMP for requester sharing a custom screen (see below)
<code>FILE_SAVE</code>	Makes the file requester a save requester (see below)

If the `FILE_PATGAD` bit of the `ASL_FuncFlags` tag is set, the file requester will appear with a "Pattern" gadget in addition to the usual file name and directory name gadgets. The user can type an AmigaDOS wildcard pattern into this gadget and the pattern will be used to limit the file names that appear in the requester. An application can also supply a default pattern using the `ASL_Pattern` tag item. A hidden unchangeable pattern can be created by supplying an `ASL_Pattern` without a `FILE_PATGAD` gadget. Such invisible patterns should not be used if there is any reason that the user may need to access a file which does not match the pattern.

Another feature of the ASL file requester is multiple selection. When multiple selection is enabled, the user can choose more than one file name in a single directory by selecting names in the requester's scrolling list gadget with the mouse. This option, like pattern matching, is set up with the `ASL_FuncFlags` tag.

If the `FILE_MULTISELECT` bit of the `ASL_FuncFlags` tag is set, the file requester will allow multiple selection. When the user selects several file names through the multiple selection feature, the FileRequester's `rf_NumArgs` field contains the number of files selected and the `rf_ArgList` field contains a pointer to an array of `WBArg` structures (defined in `<workbench/startup.h>`). There is a `WBArg` structure containing a file name for each file the user selected.

The following example illustrates a file requester with both a pattern matching gadget and multiple selection enabled.

```
filepat.c
```

The previous example demonstrates two alternate functions for creating and using ASL requesters:

```
APTR AllocAslRequestTags( unsigned long type, Tag Tag1, ... );  
BOOL AslRequestTags( APTR request, Tag Tag1, ... );
```

`AllocAslRequestTags()` can be used instead of `AllocAslRequest()` to allocate and set up the file requester. This is an `amiga.lib` function that will accept `TagItems` directly in its parameter list, rather than a pointer to an array of `TagItems`.

Similarly, `AslRequestTags()` will accept `TagItems` directly instead of requiring a pointer to an array of `TagItems` as `AslRequest()` does.

1.6 16 / Creating a File Requester / ASL Requesters and Custom Screens

An application that uses a custom screen normally wants its requesters to open on its screen. Using the `ASL_Window` tag, a program can associate a requester with a specific window so that the requester appears on the same screen as the window. The `ASL_Window` tag is followed by a pointer to a window structure. `ASL_Window` works with both file and font requesters. The example above shows how the `ASL_Window` tag is used with a file requester.

Normally, a requester associated with a window (using `ASL_Window`) shares that window's IDCMP port for its communication. An application may not want to share an IDCMP port with the requester. Using the `ASL_FuncFlags` tag, a program can ask for a requester that creates its own IDCMP port. There are two flags that accomplish this. The first, `FILE_NEWIDCMP`, is used on file requesters. The other, `FONF_NEWIDCMP`, is used on font requesters.

1.7 16 / Creating a File Requester / The Save Requester

The save requester is a special type of file requester used for save operations. It differs from the regular ASL file requester in several ways. First, the color of the text making up the file names and the background color are interchanged. This makes it more apparent to the user that they are looking at a save requester (instead of the usual load requester).

Another difference, is that a save requester does not allow the user to select an existing file name by double-clicking on an entry in the scrolling list gadget. This helps prevent the user from accidentally overwriting the wrong file.

Save requesters can also create directories. If the user types a directory name into the save requester and the directory doesn't exist, the save requester will create that directory (after getting the user's permission via another requester).

To create a save requester, set the `FILF_SAVE` flag of the `ASL_FuncFlags` tag. Remember that ASL tags and flag values are preserved across calls to `AslRequest()`, so if you use a save requester, you must clear the `FILF_SAVE` bit and reset your `ASL_FuncFlags` when you want a load requester. Note that it does not make sense to have multiselection in a save requester, so the `FILF_SAVE` flag overrides the `FILF_MULTISELECT` flag.

1.8 16 / Creating a File Requester / The Directory Requester

Sometimes a program may only require a directory name from the user. There is another variation on `asl.library`'s file requester that allows this. The `ASL_ExtFlags1` tag contains a flag bit to toggle this option. If the `FILF_NOFILES` flag of `ASL_ExtFlags1` is set, the requester will appear without a string gadget for file names and will display only directory names in the scrolling list gadget. When `AslRequest()` (or `AslRequestTags()`) returns successfully, the `rf_Dir` field of the `FileRequester` structure contains the name of the directory the user selected.

Another flag defined for `ASL_ExtFlags1` is `FILF_MATCHDIRS`. If file pattern matching is on (see the `FILF_PATGAD` flag for `ASL_FuncFlags`, setting `FILF_MATCHDIRS` tells the file requester to pattern match directory names as well as file names. Of course, if both of these `ASL_ExtFlags1` flags are set, the requester will only pattern match directory names.

1.9 16 ASL Library / Creating a Font Requester

The ASL library also contains a font requester. Using the font requester is very similar to using the file requester. First, allocate a requester structure with `AllocAslRequest()` or `AllocAslRequestTags()`. The type should be set to `ASL_FontRequest` in order to get a `FontRequester` structure:

```
struct FontRequester {
    APTR    fo_Reserved1[2];
    struct TextAttr fo_Attr;           /* Returned TextAttr          */
    UBYTE    fo_FrontPen;             /* Returned pens, if selected */
    UBYTE    fo_BackPen;
    UBYTE    fo_DrawMode;
    APTR    fo_UserData;
    /* missing from asl.h but present in this structure */
    SHORT    fo_LeftEdge, fo_TopEdge, fo_Width, fo_Height;
};
```

Once the requester is set up, call `AslRequest()` or `AslRequestTags()` to make the requester appear on screen. These functions return `TRUE` if the user makes a selection. In that case, the font selected is returned as a `TextAttr` structure in the `fo_Attr` field of the `FontRequester` structure. (The `TextAttr` structure is defined in `<graphics/text.h>`. See the Amiga ROM Kernel Manual: Includes and Autodocs for a complete listing.) If the user cancels the font requester `FALSE` is returned.

Figure 16-2: The ASL Font Requester

When the requester is no longer needed, call `FreeAslRequest()` to deallocate the requester data structure.

Specifying Font Requester Options with TagItems
Example Font Requester

1.10 16 / Creating a Font Requester / Specifying Options with TagItems

As with a file requester, the font requester is specified with a TagItem list. There are several tags that are specific to the font requester:

Font Requester Tag Name	Used For
ASL_FontName	Default font (<code>fo_Attr.ta_Name</code>)
ASL_FontHeight	Default font size (<code>fo_Attr.ta_YSize</code>)
ASL_FontStyles	Default font style (<code>fo_Attr.ta_Style</code>)
ASL_FontFlags	Default font flags (<code>fo_Attr.ta_Flags</code>)
ASL_FrontPen	Default font color (<code>fo_FrontPen</code>)
ASL_BackPen	Default font background color (<code>fo_BackPen</code>)
ASL_ModeList	Alternate strings for the drawing mode gadget (see below)
ASL_MinHeight	Minimum font height the requester will display
ASL_MaxHeight	Maximum font height the requester will display

Note that the last two tags only limit the range of font sizes that the font requester displays, the user is free to type in any value.

Font requesters have additional special options that are controlled through the `ASL_FuncFlags` tag. This tag works the same way as it does with file requesters but with different options available. Recall that the data for this tag is divided into bit fields, each of which controls a requester option. The flags used with the `ASL_FuncFlags` tag in a font requester are defined in `<libraries/asl.h>`:

Font Requester Flags	Used For
FONF_FRONTCOLOR	Enables font color selection gadgets
FONF_BACKCOLOR	Enables font background color selection gadget
FONF_STYLES	Enables font style selection gadget
FONF_FIXEDWIDTH	Limits display to fixed width fonts only
FONF_DRAWMODE	Enables font draw mode gadget

A simple font requester (one without any of the above `FONF_` flags set) only lets the user choose a font and a Y size. Setting the flags above adds options to the font requester. `FONF_FRONTCOLOR` and `FONF_BACKCOLOR` add color selection gadgets to the requester, one for choosing a font's foreground color (labeled "Text") and the other for choosing the background color (labeled "Field"). The font requester records the user's setting in the `FontRequester`'s `fo_FrontPen` and `fo_BackPen` fields.

FONF_STYLES sets up several gadgets to choose the style of the font (bold, italics, underline). The font requester saves these settings in the fo_Attr.ta_Style bit field according to the style flags defined in <graphics/text.h>. FONF_FIXEDWIDTH limits the font name display to fixed width (non-proportional) fonts (note that this does not prevent the user from typing in a proportional font name).

FONF_DRAWMODE adds a cycle gadget to the font requester so the user can choose the draw mode. The draw mode is saved in the requester's fo_DrawMode field. The number stored there corresponds to the draw mode's position in the gadget's cycle.

The draw mode cycle gadget initially is labeled "Mode" and has three elements in its cycle: "JAM1", "JAM2", and "Complement". These yield a result of 0, 1, and 2, respectively. It is possible to change the names and number of draw modes with the ASL_ModeList tag. This tag accepts a pointer to an array of strings. The first string replaces "Mode" as the label for the draw mode cycle gadget. The strings that follow replace the elements of the cycle gadget. The last entry in the array has to be NULL to tell the requester where the list of entries ends.

1.11 16 ASL Library / Calling Custom Functions from a Requester

The ASL_HookFunc tag passes an ASL requester a pointer to a custom function. The requester can use this function for two purposes. The first is to determine if the requester should display a particular file or font name. The other purpose is to process messages that the requester receives at its IDCMP port that are not meant for the requester. Hook functions are set up through flag values used with the ASL_FuncFlags tag:

Hook Function Flag	Used For
FILE_DOWILDFUNC	Call user hook function on each name in a file requester
FONF_DOWILDFUNC	Call user hook function on each name in a font requester
FILE_DOMSGFUNC	Call user hook function for IDCMP messages not used by a file requester
FONF_DOMSGFUNC	Call user hook function for IDCMP messages not used by a font requester

The FILE_DOWILDFUNC and FONF_DOWILDFUNC flags cause a requester to call the function you specify with the ASL_HookFunc tag for every file or font entry. The requester displays the file or font name only if your hook function tells it to. For a file requester, if your hook function returns a zero, the file requester will display the file name. For a font requester, if your hook function returns anything but zero, the font requester will display the font name and size.

The FILE_DOMSGFUNC and FONF_DOMSGFUNC flags cause a requester to call your hook function whenever it receives an IntuiMessage that it cannot use at the IDCMP port that it shares with your window. (See the section on "ASL Requesters and Custom Screens" earlier in this chapter for more

information about sharing IDCMP ports.) If the requester receives any messages that are not meant for the requester it will call your hook function (specified with the ASL_HookFunc tag). Your hook function is responsible for returning a pointer to the IntuiMessage. The requester will take care of replying to the message.

Parameters Passed to Custom Hook Functions

Example ASL Requester with Custom Hook Function

1.12 16 / Calling Functions / Parameters Passed to Custom Hook Functions

A requester always passes three parameters to your custom hook function:

```
ULONG MyHookFunc(ULONG type, CPTR object, CPTR AslRequester)
```

If MyHookFunc() is called from a file requester doing _DOWILDFUNC, the three parameters are:

```
type = FILE_DOWILDFUNC
```

```
object = pointer to an AnchorPath structure (from <dos/dosasl.h>)
```

```
AslRequester = pointer to the FileRequester that called the hook  
function (Return a zero to display this file)
```

The AnchorPath structure is a dos.library structure used in pattern matching. Refer to the AmigaDOS Manual, 3rd Edition by Bantam Books for more information.

If MyHookFunc() is called from a font requester doing _DOWILDFUNC, the three parameters are:

```
type = FONF_DOWILDFUNC
```

```
object = pointer to a TextAttr structure (from <graphics/text.h>)
```

```
AslRequester = pointer to the FontRequester that called the hook  
function (Return non-zero to display this particular  
font size)
```

If MyHookFunc() is called from a file or font requester doing _DOMSGFUNC, the three parameters are:

```
type = FILE_DOMSGFUNC (file requester) or FONF_DOMSGFUNC (font  
requester)
```

```
object = pointer to the IntuiMessage for the function to process
```

```
AslRequester = pointer to the FileRequester or FontRequester that  
called the hook function (Return a pointer to the
```

IntuiMessage)

Notice that it is possible for a requester to use both `_DOWILDFUNC` and `_DOMSGFUNC` at the same time. Your hook function has to differentiate between the two cases by testing the type passed to it. It is not possible for a font and file requester to share a hook function for a `_DOWILDFUNC`, because `FILF_DOWILDFUNC` is defined to be the same value as `FONF_DOWILDFUNC`, so the hook function cannot tell if the object (from the prototype above) is a pointer to an `AnchorPath` structure or a pointer to a `TextAttr` structure. It is possible for font and file requesters to share one hook function for `_DOMSGFUNC` (even though `FILF_DOMSGFUNC` and `FONF_DOMSGFUNC` are equal) because, in this case, font and file requesters both call your hook function in the same manner.

1.13 16 ASL Library / Function Reference

The following are brief descriptions of the ASL library functions. See the *Amiga ROM Kernel Reference Manual: Includes and Autodocs* for details on each function call. All of these functions require Release 2 or a later version of the operating system.

Table 16-1: Functions for ASL Requesters

Function	Description
<code>AllocAslRequest()</code>	Allocates an ASL font or file requester from a <code>TagItem</code> array
<code>AllocAslRequestTags()</code>	Same as <code>AllocAslRequest()</code> but accepts tags directly
<code>AslRequest()</code>	Displays an ASL requester with options set up in a <code>TagItem</code> array
<code>AslRequestTags()</code>	Same as <code>AslRequest()</code> but accepts tags directly
<code>FreeAslRequest()</code>	Deallocates an ASL requester created with <code>AllocAslRequest()</code>