Libraries

COLLABORATORS

	TITLE : Libraries	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Libraries

1.1 Amiga® RKM Libraries: 7 Intuition Requesters and Alerts

This chapter explains how to create requesters, the information exchange boxes that both the system and applications can use for confirming actions, getting command options and similar operations. These boxes are called requesters because they generally request information from the user.

Alerts provide a function similar to requesters but are reserved for emergency messages. Alerts are discussed later in this chapter.

1.2 7 Intuition Requesters and Alerts / Types Of Requesters

There are at least three kinds of display objects in Amiga terminology called requesters: true requesters, system requesters and ASL requesters.

True requesters are general purpose display areas that can be thought of as temporary sub-windows. They display information to the user and allow the user to make a selection. True requesters always open within an existing window and are constrained to the boundaries of that window (often referred to as the parent window). If a requester extends beyond the edge of its parent window, either its position is adjusted or its graphics are clipped. True requesters always block input to their parent window as long as they are present.

System requesters are typically used for warnings or to confirm an action the user has just initiated. System requesters differ from true requesters in that they cannot block input to the parent window. In fact, system requesters do not open in a parent window at all, but instead open their own separate window in the screen. Since these requesters are so different from true requesters, they will be discussed separately later in the chapter. See the sections on "Easy Requesters" and "System Requests" for more information.

The third type of requester, the ASL requester, is a special purpose requester available only in Release 2 and later versions of the OS. ASL requesters provide an easy, standard way to get a filename from the user for load and save operations. They can also be used to get a font selection from the user. Since selecting a file or font name is one of the most common uses for a requester, it has been incorporated into Release 2 as a standard feature. For the details about ASL file and font requesters, see Chapter 16, "ASL Library".

1.3 7 Intuition Requesters and Alerts / True Requesters

The primary function of a requester is to display information to the user from which the user is to make a selection. Conceptually, requesters are similar to menus since both menus and requesters offer options to the user. Requesters, however, go beyond menus because they can have customized imagery, can be placed anywhere in a window, can be activated by the application and may have any type of gadget attached.

For instance, to select a color for a given operation using a menu could be awkward, especially in an application that supports a large number of colors. In that case a requester could be used instead (see figure).

Figure 7-1: Requester Deluxe

The ability of a true requester to block input to its parent window is important in understanding how requesters are used. When input is blocked by a true requester (also known as a modal requester), the user must take some action before the program will proceed further, such as making a selection, correcting an error condition, or acknowledging a warning. These are situations where a true (modal) requester is appropriate, however, keep in mind that your application should try to be as user-responsive as possible. Putting up a requester merely because you are in a phase of the program where it would be difficult to deal with user input is bad style. Modal requesters should be used only when the program requires user interaction before proceeding.

True requesters can be created in a window in two different ways.

* An application can display a requester at any time by calling the Request() function.

* The application can declare a requester as the window's double menu requester, which the user can bring up with a double-click of the menu button (this method is rarely used).

Creating Application Requesters
Requester I/O
Rendering Requesters
Requester Refresh Type
Requester Display Position
Gadgets in Requesters
Using a Requester to Block Window Input
Double Menu Requesters
IDCMP Requester Features

1.4 7 / True Requesters / Creating Application Requesters

To create a requester, the application first allocates memory for or declares an instance of the Requester structure as defined in <intuition/intuition.h>. Once the Requester structure is set up, it is initialized with the InitRequester() function.

```
    void InitRequester( struct Requester *requester)
```

This function simply clears the Requester structure. The application should do further initialization depending on its needs. See the section on the "Requester Structure" below for an explanation of all the Requester fields and how to set them.

A true (modal) requester is attached to its parent window and displayed with the Request() function.

```
    BOOL Request(struct Requester *requester, struct Window *window)
```

This function returns TRUE if the requester opens successfully or FALSE if the requester cannot be opened. If the requester opens successfully, menu and gadget input in the parent window is blocked as long as the requester is displayed. The application should process input events from the requester, which are sent to the parent window's Window.UserPort, until the requester is satisfied.

To remove a requester from its parent window and update the display, use EndRequest().

```
    void EndRequest( struct Requester *requester, struct Window *window );
```

This removes only the one requester specified. It is possible to set up a requester with a special gadget that, if selected, will automatically close the requester. In that case, EndRequest() need not be called. If the program needs to cancel the request early, or cancel it only after some specific manipulation of the gadgets, EndRequest() should be used.

The application should always provide a safe way for the user to back out of a requester without taking any action that affects the user's work. Providing an escape hatch is important, for instance, a requester with the message "Overwrite File?" should allow the user to cancel the operation without losing the old data.

1.5 7 / True Requesters / Requester I/O

So long as a requester is active in a window, the only gadgets that can be used are those that are in the requester, plus all of the window's system gadgets except for the close gadget (i.e., the drag bar, size gadget, depth gadget, and zoom gadget). A requester also makes the menus of the parent window inaccessible. Additionally, mouse button and keyboard events will be blocked (unless the requester's NOISYREQ flag is set; see "Requester Structure" below). Mouse movement events, if enabled in the parent window (with WFLG_REPORTMOUSE), are not blocked.

Requesters do not have their own IDCMP message ports. Instead, events for a requester are sent to the IDCMP port of the requester's parent window (Window.UserPort). Since the window's menus and application gadgets are inaccessible, no IDCMP events will be sent for them.

Even though the window containing the requester is blocked for input, the user can work in another application or even in a different window of the same application without satisfying the requester. Only input to the parent window is blocked by a requester.

Output is not blocked by a requester so nothing prevents the application from writing to the window. Be aware, however, that the requester obscures part of the display and cannot be moved within the window so this may limit the usefulness of any output you send to the parent window. There are several ways to monitor the comings and goings of requesters that allow the program to know if requesters are currently displayed in a given window. See "IDCMP Requester Features" below.

The information displayed in a requester is placed in its own layer, so it does not overwrite the information in the window. Output to the window while the requester is displayed will not change the requester's display, it will go into the window's layer. The requester's layer is clipped to the window's boundaries, so the data in the requester is only visible if the window is large enough to allow for the complete display of that data.

The requester will remain in the window and input will remain blocked until the user satisfies the request or the application removes the requester. Applications can set up some or all of the gadgets in the requester to automatically terminate the requester when the gadget is selected. This allows the requester to be removed from the window by user action. The application may also remove requesters from the window based on some event internal to the program.

Multiple requesters may be nested in a single window. Such requesters must be satisfied in the reverse order in which they were posted; the last requester to be displayed must be satisfied first. Input will not be restored to a previous requester until all later requesters are satisfied.

Note that the application may not bring up a limitless number of requesters in a window. Each requester creates a new layer for rendering in its window and the system currently has a limit of ten layers per window. Normal windows use one layer for the window rendering, GimmeZeroZero windows use a second layer for the border rendering. This leaves a maximum of eight or nine simultaneous requesters open in a window at any given time.

If the requester is being brought up only to display an error message, the application may want to use a less intrusive method of bringing the error to the user's attention than a requester. Requesters interrupt the flow of the user's work, and force them to respond before continuing.

As an alternative to bringing up an error requester, the application could flash the screen instead with Intuition's DisplayBeep() function. This allows the application to notify the user of an error that is not serious enough to warrant a requester and to which the user does not really need to respond. For more information, see the description of DisplayBeep() in

the "Intuition Screens" chapter.

1.6 7 / True Requesters / Rendering Requesters

The application may choose to use Intuition's rendering facilities to display the requester, or it may define its own custom bitmap. The Requester structure is initialized differently according to the rendering method chosen.

To use Intuition's rendering facilities, you supply a list of one or more display objects with the Requester structure and submit the Requester to Intuition, allowing it to draw the objects. These objects can include independent lists of Borders, IntuiText, Images and Gadgets. Note that the abilty to provide a list of Image structures is new in V36, and the USEREQIMAGE flag must be set for them to be rendered. For more about Intuition rendering see the chapter on "Intuition Images, Line Drawing and Text".

The gadgets in a requester also have their own borders, images and text to add to the display imagery. Intuition will allocate the buffers, construct a bitmap that lasts for the duration of the display, and render the requester into the window. This rendering is all done over a solid color, filled background specified by the BackFill pen in the Requester structure. The backfill may be disabled by setting the NOREQBACKFILL flag (this also a new feature of V36).

On the other hand, a custom requester may be designed with pre-defined, bitmap imagery for the entire object. The image bitmap is submitted to Intuition through the ImageBMap field of the Requester structure. The bitmap should be designed to reduce user confusion; gadgets should line up with their images, and the designer should attempt to use glyphs (symbols) familiar to the user.

To provide imagery for the requester, applications should always try to use data structures attached to the Requester structure as described above. Although, rendering directly into the requester layer's RastPort is tolerated, it must be done with great care.

First, a requester is allowed to have gadgets that automatically close the requester when they are selected (GACT_ENDGADGET). If such a gadget is selected, the requester, its layer, and its layer's RastPort will be deleted asynchronously to your application. If your application is trying to render directly into the requester at that time, the results are unpredictable. Therefore, do not put GACT_ENDGADGET gadgets into a requester if you plan on rendering directly into its RastPort.

Second, recall that requesters are clipped to the inside of the window (not including the borders). If the window can be sized smaller such that the requester would be entirely clipped, the requester's layer may be deleted by Intuition. If your window's minimum size and the requester size and position are such that the requester can be completely clipped, then reading Requester.ReqLayer is unsafe without additional protection. It would be correct to LockLayerInfo() the screen's Layer_Info, then check for the existence of the requester's ReqLayer, then render, then unlock.

For reasons such as these, direct rendering is discouraged.

1.7 7 / True Requesters / Requester Refresh Type

A requester appears in a Layer. By default, the requester layer is of type LAYERSMART, or, in window terminology, WFLG_SMART_REFRESH; so rendering is preserved in the requester when the window is moved or revealed.

Requesters may also be simple refresh. This is the recommended type. If possible, make the requester a simple refresh layer requester by specifying the SIMPLEREQ flag.

For all refresh types, Intuition will keep the gadget, border, image and bitmap imagery properly refreshed.

1.8 7 / True Requesters / Requester Display Position

The location of true requesters may be specified in one of three ways. The requester may either be a constant location, which is an offset from the top left corner of the window; a location relative to the current location of the pointer; or a location relative to the center of the window.

To display the requester as an offset from the upper left corner of the window, initialize the TopEdge and LeftEdge variables and clear the POINTREL flag. This will create a requester with a fixed position relative to the upper left corner for both normal requesters and double menu requesters.

Displaying the requester relative to the pointer can get the user's attention immediately and closely associates the requester with whatever the user was doing just before the requester was displayed in the window. However, only double menu requesters may be positioned relative to the pointer position. See below for more information on double menu requesters.

Requesters that are not double menu requesters may be positioned relative to the center of the window on systems running Release 2 or a later version of the OS. This is done by setting the POINTREL flag and filling in the relative top and left of the gadget. Setting RelTop and RelLeft to zero will center the requester in the window. Positive values of RelTop and RelLeft will move the requester down and to the right, negative values will move it up and to the left.

1.9 7 / True Requesters / Gadgets in Requesters

Each requester gadget must have the GTYP_REQGADGET flag set in the GadgetType field of its Gadget structure. This informs Intuition that this gadget is to be rendered in a requester rather than a window.

Requesters can have gadgets in them that automatically satisfy the request and end the requester. When one of these gadgets is selected, Intuition will remove the requester from the window. This is equivalent to the application calling EndRequest(), and, if the request is terminated by selection of such a gadget, the application should not call EndRequest() for that requester.

Set the GACT_ENDGADGET flag in the Activation field of the Gadget structure to create a gadget that automatically terminates the requester. Every time one of the requester's gadgets is selected, Intuition examines the GACT_ENDGADGET flag. If GACT_ENDGADGET is set, the requester is removed from the display and unlinked from the window's active requester list.

Requesters rendered via Intuition and those that use a custom bitmap differ in how their gadgets are rendered. For requesters rendered via Intuition, the application supplies a regular gadget list just as it would for application gadgets in a window.

In custom bitmap requesters, however, any gadget imagery is part of the bitmap supplied for the requester. Therefore the list of gadgets supplied for custom bitmap requesters should not provide gadget imagery but rather it should define only the select boxes, highlighting, and gadget types for the gadgets.

The Gadget structures used with a custom bitmap requester should have their GadgetRender, SelectRender and GadgetText fields set to NULL as these will be ignored. Other gadget information--select box dimensions, highlighting, and gadget type--is still relevant. The select box information is especially important since the select box must have a well defined correspondence with the custom bitmap imagery supplied. The basic idea is to make sure that the user understands the requester imagery and gadgets.

1.10 7 / True Requesters / Using a Requester to Block Window Input

There may be times when an application needs to block user input without a visible requester. In some cases, the application needs to be busy for a while. Other times, an application wants the blocking properties of a requester, but prefers to use a window instead of a true requester. In this case, the application can create a requester with no imagery, attaching it to the parent window to block input. A new window may then be opened to act as the requester.

Some of the advantages of using a window as a requester instead of a real requester include:

* A window can be resized, and moves independently of the parent window.

* It is legal to render directly into a window.

* The window can have its own menus since only the parent window's menus are disabled (this is only occasionally useful).

```
  * Certain code or a library you are using may not work in requesters
    (GadTools library is an example of this).

Of course, using a true requester instead of a window has the advantage
that the requester automatically moves and depth-arranges along with the
parent window.

 A Requester Example
```

1.11 7 / True Requesters / Double Menu Requesters

A double menu requester is exactly like other requesters with one exception: it is displayed only when the user double clicks the mouse menu button. Double menu requesters block input in exactly the same manner as other true requesters. A double menu requester is attached to a window by calling SetDMRequest().

```
    BOOL SetDMRequest( struct Window *window,
                       struct Requester *requester );
```

This call does not display the requester, it simply prepares it for display. The requester will be brought up when the user double clicks the mouse menu button. The parent window will receive IDCMP_REQSET and IDCMP_REQCLEAR messages when the requester is added and removed.

To prevent the user from bringing up a double menu requester, unlink it from the window by calling ClearDMRequest(). If a double menu request is set for a window, ClearDMRequest() should be called to remove the requester before that window is closed.

```
    BOOL ClearDMRequest( struct Window *window );
```

This function unlinks the requester from the window and disables the ability of the user to bring it up. ClearDMRequest() will fail if the double menu request is currently being displayed.

Double menu requesters can be positioned relative to the current mouse pointer position. For a mouse relative requester, specify POINTREL in the Flags field and initialize the RelLeft and RelTop variables. RelLeft and RelTop describe the offset of the upper, left corner of the requester from the pointer position at the time the requester is displayed. These values can be either negative or positive.

The values of RelLeft and RelTop are only advisory; the actual position will be restricted such that the requester is entirely contained within the borders of its parent window, if possible. The actual top and left positions are stored in the TopEdge and LeftEdge variables.

Positioning relative to the mouse pointer is possible only with double menu requesters. Setting POINTREL in a requester which is not a double menu requester will position the requester relative to the center of the window.

1.12 7 / True Requesters / IDCMP Requester Features

Intuition can notify your application about user activity in the requester by sending a message to the parent window's IDCMP port (Window.UserPort). When using the IDCMP for input, the following IDCMP flags control how requester input events will be handled.

IDCMP_REQSET
    With this flag set, the program will receive a message whenever a requester opens in its window. The application will receive one IDCMP_REQSET event for each requester opened in the window.

IDCMP_REQCLEAR
    With this flag set, the program will receive a message whenever a requester is cleared from its window. The application will receive one IDCMP_REQCLEAR event for each requester closed in the window. By counting the number of IDCMP_REQSET and IDCMP_REQCLEAR events, the application may determine how many requesters are currently open in a window.

IDCMP_REQVERIFY
    With this flag set, the application can ensure that it is ready to allow a requester to appear in the window before the requester is displayed.

    When the program receives an IDCMP_REQVERIFY message, it must reply to that message before the requester is added to the window. If multiple requesters are opened in the window at the same time, only the first one will cause an IDCMP_REQVERIFY event. It is assumed that once a requester is in a window others may be added without the program's consent. After the requester count drops to zero and there are no open requesters in the window, the next requester to open will cause another IDCMP_REQVERIFY event.

    IDCMP_REQVERIFY is ignored by the Request() function. Since Request() is controlled by the application, it is assumed that the program is prepared to handle the request when calling this function. Since the system does not render true requesters into an application's window (EasyRequest() and AutoRequest() come up in their own window, not in the application's window), IDCMP_REQVERIFY will only control the timing of double menu requesters.

These flags are set when the parent window is first opened by using either the WA_IDCMP tag or NewWindow.IDCMPFlags. They can also be set after the parent window is open by using the ModifyIDCMP() call. See the chapter entitled "Intuition Input and Output Methods," for further information about these IDCMP flags. See the "Intuition Windows" chapter for details on setting IDCMP flags when a window is opened.

1.13 7 Intuition Requesters and Alerts / Requester Structure

Unused fields in the Requester structure should be initialized to NULL or zero before using the structure. For global data that is pre-initialized, be sure to set all unused fields to zero. For dynamically allocated

structures, allocate the storage with the MEMF_CLEAR flag, or call the InitRequester() function to clear the structure.

Requesters are Initialized According to Their Type.

See "Rendering Requesters" and "Gadgets in Requesters" above for information about how the initialization of the structure differs according to how the requester is rendered.

The specification for a Requester structure, defined in <intuition/intuition.h>, is as follows.

```
struct Requester
    {
    struct Requester *OlderRequest;
    WORD LeftEdge, TopEdge;
    WORD Width, Height;
    WORD RelLeft, RelTop;
    struct Gadget *ReqGadget;
    struct Border *ReqBorder;
    struct IntuiText *ReqText;
    UWORD Flags;
    UBYTE BackFill;
    struct Layer *ReqLayer;
    UBYTE ReqPad1[32];
    struct BitMap *ImageBMap;
    struct Window *RWindow;
    struct Image  *ReqImage;
    UBYTE ReqPad2[32];
    };
```

Here are the meanings of the fields in the Requester structure:

OlderRequest
: For system use, initialize to NULL.

LeftEdge, TopEdge
: The location of the requester relative to the upper left corner of the window. These values must be set if the POINTREL flag is not set. Use RelLeft and RelTop for POINTREL requesters.

Width, Height
: These fields describe the size of the entire requester rectangle, containing all the text and gadgets.

RelLeft, RelTop
: These values are only used if the POINTREL flag in the requester's Flags field is set.

  If the requester is a double menu requester and POINTREL is set then these values contain the relative offset of the requester's upper left corner from the current pointer position.

  If the requester is not a double menu requester and POINTREL is set, then these values contain the relative offset of the requester's center from the center of the window that the requester is to be displayed in. For example, using POINTREL with a requester which is

not a double menu requester with RelLeft and RelTop of zero will center the requester in the window. The requester is centered within the inner part of the window, that is, within the inside edge of the window's borders.

If the requester is POINTREL and part of the containing box will appear out of the window, Intuition will adjust the requester so that the upper left corner is visible and as much of the remaining box as possible is visible. The adjustment attempts to maintain the requester within the window's borders, not within the window's bounding box.

ReqGadget
This field is a pointer to the first in a linked list of Gadget structures. GTYP_REQGADGET must be specified in the GadgetTypes field of all Gadget structures that are used in a requester. Take care not to specify gadgets that extend beyond the Requester rectangle specified by the Width and Height fields, as Intuition does no boundary checking.

For requesters with custom imagery, where PREDRAWN is set, ReqGadget points to a valid list of gadgets, which are real gadgets in every way except that the gadget text and imagery information are ignored (and can be NULL). The select box, highlighting, and gadget type data are still used. Try to maintain a close correspondence between the gadgets' select boxes and the supplied imagery.

String Gadgets and Pre-drawn Requesters.

Intuition will not render string gadget text in a predrawn requester. The application must use other rendering means than the predrawn bitmap if it wishes to use string gadgets with a requester.

ReqBorder
This field is a pointer to an optional linked list of Border structures for drawing lines around and within the requester. The lines specified in the border may go anywhere in the requester; they are not confined to the perimeter of the requester.

For requesters with custom imagery, where PREDRAWN is set, this variable is ignored and may be set to NULL.

ReqText
This field is a pointer to an optional linked list of IntuiText structures containing text for the requester. This is for general text in the requester.

For requesters with custom imagery, where PREDRAWN is set, this variable is ignored and can be set to NULL.

Flags
The following flags may be specified for the Requester:

POINTREL
Specify POINTREL to indicate that the requester is to appear in a relative rather than a fixed position.

For double menu requesters, the position is relative to the pointer. Otherwise, the position of POINTREL requesters is relative to the center of the window.

See the discussion of RelLeft and RelTop, above.

PREDRAWN
Specify PREDRAWN if a custom BitMap structure is supplied for the requester and ImageBMap points to the structure.

NOISYREQ
Normally, when a requester is active, any gadget, menu, mouse and keyboard events within the parent window are blocked. Specify the NOISYREQ requester flag to allow keyboard and mouse button IDCMP events to be posted, even though the requester is active in the parent window.

If the NOISYREQ requester flag is set, the application will receive IDCMP_RAWKEY, IDCMP_VANILLAKEY and IDCMP_MOUSEBUTTONS events. Note that with NOISYREQ set, IDCMP_MOUSEBUTTON events will also be sent when the user clicks on any of the blocked gadgets in the parent window.

Although the reporting of mouse button events depends on NOISYREQ, mouse movement events do not. IDCMP_MOUSEMOVE events are reported if the window flag WFLG_REPORTMOUSE is set in the parent window, or if one of the requester gadgets is down and has the GACT_FOLLOWMOUSE flag set. This is true even if the requester is a double menu requester.

USEREQIMAGE
Render the linked list of images pointed to by ReqImage after rendering the BackFill color but before gadgets and text.

NOREQBACKFILL
Do not backfill the requester with the BackFill pen.

In addition, Intuition uses these flags in the Requester:

REQOFFWINDOW
Set by Intuition if the requester is currently active but is positioned off window.

REQACTIVE
This flag is set or cleared by Intuition as the requester is posted and removed. The active requester is indicated by the value of Window.FirstRequest.

SYSREQUEST
This flag is set by Intuition if this is a system generated requester. Since the system will never create a true requester in an application window, the application should not be concerned with this flag.

BackFill
BackFill is the pen number to be used to fill the rectangle of the requester before any drawing takes place. For requesters with custom

imagery, where PREDRAWN is set, or for requesters with NOREQBACKFILL set, this variable is ignored.

ReqLayer
While the requester is active, this contains the address of the Layer structure used in rendering the requester.

ImageBMap
A pointer to the custom bitmap for this requester. If this requester is not PREDRAWN, Intuition ignores this variable.

When a custom bitmap is supplied, the PREDRAWN flag in the requester's Flags field must be set.

RWindow
Reserved for system use.

ReqImage
A pointer to a list of Image structures used to create imagery within the requester. Intuition ignores this field if the flag USEREQIMAGE is not set. This imagery is automatically redrawn by Intuition each time the requester needs refreshing. The images are drawn after filling with the BackFill pen, but before the gadgets and text.

ReqPad1, ReqPad2
Reserved for system use.

1.14 7 Intuition Requesters and Alerts / Easy Requesters

EasyRequest() provides a simple way to make a requester that allows the user to select one of a limited number of choices. (A similar function, AutoRequest(), is also available but is not as flexible or as powerful. See the Amiga ROM Kernel Reference Manual: Includes and Autodocs for more information.)

Figure 7-2: A Simple Requester Made with EasyRequest()

The program supplies the text for the body of the requester, text for each of the possible options, an optional title for the window, and other arguments. The body text can consist of one or more lines with lines separated by the linefeed character.

Each option for an easy requester is displayed as a simple button gadget positioned beneath the body text you specify. The layout of the requester, its text and buttons, is done automatically and is font sensitive. The screen font (Screen.Font) is used for all text in the requester.

Typically, easy requesters have one selection indicating a positive action and one selection indicating a negative action. The text used for the positive action might be "OK", "Yes," "True," "Retry," or similar responses. Likewise, the text used for the negative action might be "No," "False," "Cancel," and so on. The negative choice should always be the rightmost or final choice and will return a zero if selected.

When EasyRequest() is called, Intuition will build the requester, display it, and wait for user response.

```
LONG EasyRequest( struct Window *window,
                  struct EasyStruct *easyStruct,
                  ULONG *idcmpPtr, APTR arg1, ... );

LONG EasyRequestArgs( struct Window *window,
                      struct EasyStruct *easyStruct,
                      ULONG *idcmpPtr, APTR args );
```

The window argument is a pointer to the reference window. The requester will be displayed on the same screen that the reference window is on and also takes its title from the reference window, if not otherwise specified. This argument can be NULL, which means the requester is to appear on the Workbench screen, or the default public screen, if defined.

The easyStruct argument is a pointer to an EasyStruct structure which defines the setup and the text of this easy requester (described below).

The idcmpPtr argument is a pointer to a ULONG containing the IDCMP flags for the event that you want to terminate this requester. If such an event occurs the requester is terminated (with a result of -1) and the ULONG that idcmpPtr points to will contain the actual class of the event message. This feature allows external events to satisfy the request, such as the user inserting a disk in the disk drive. This argument can be set to NULL for no automatic termination.

The gadget and body text for an easy requester is specified in an EasyStruct structure (see below). Body text can be specified using a printf()-style format string that also accepts variables as part of the text. If variables are specified in the requester text, their value is taken from the args (or arg1,...) parameters shown in the prototypes above. EasyRequestArgs() takes a pointer to an array of pointers to arguments, while EasyRequest() has a varargs interface and takes individual arguments as part of the function call. The types of these arguments are specified in the format strings of the EasyStruct structure. Arguments for es_GadgetFormat follow arguments for es_TextFormat.

The EasyRequest() functions return an integer from 0 to n - 1, where n is the number of choices specified for the requester. The numbering from left-to-right is: 1, 2, ..., n - 1, 0. This is for compatibility with AutoRequest() which returns FALSE for the rightmost gadget.

The function will return -1 if it receives an IDCMP event that matches one of the termination events specified in the idcmpPtr argument.

Turn Off the Verify Messages.

Use ModifyIDCMP() to turn off all verify messages (such as MENUVERIFY) before calling EasyRequest() or AutoRequest(). Neglecting to do so can cause situations where Intuition is waiting for the return of a message that the application program cannot receive because its input is shut off while the requester is up. If Intuition finds itself in a deadlock state, the verify function will timeout and will be automatically replied.

1.15 7 / Easy Requesters / The EasyStruct Structure

The text and setup of an easy requester is specified in an EasyStruct structure, defined in <intuition/intuition.h>.

```
    struct EasyStruct
        {
        ULONG       es_StructSize;
        ULONG       es_Flags;
        UBYTE       *es_Title;
        UBYTE       *es_TextFormat;
        UBYTE       *es_GadgetFormat;
        };
```

es_StructSize
    Contains the size of the EasyStruct structure, sizeof(struct EasyStruct).

es_Flags
    Set to zero.

es_Title
    Title of easy requester window. If this is NULL, the title will be taken to be the same as the title of the reference window, if one is specified in the EasyRequest() call, else the title will be "System Request".

es_TextFormat
    Format string for the text in the requester body, with printf()-style variable substitution as described in the Exec library function RawDoFmt(). Multiple lines are separated by the linefeed character (hex 0x0a or '\n' in C). Formatting '%' functions are supported exactly as in RawDoFmt(). The variables that get substituted in the format string come from the last argument passed to EasyRequest() (see prototype above).

es_GadgetFormat
    Format string for gadgets, where the text for separate gadgets is separated by '|' (vertical bar). As with the body text, printf()-style formatting with variable substitution is supported, but multi-line text in the gadgets is not supported. At least one gadget must be specified.

Requesters generated with EasyRequest() and BuildEasyRequest() (including system requesters, which use SysReqHandler() to handle input) can be satisfied by the user via the keyboard. The key strokes left Amiga V and left Amiga B correspond to selecting the requester's leftmost or rightmost gadgets with the mouse, respectively.

An easy request must have at least one choice. Multiple choices are specified through the "|" (vertical bar) separator character in the es_GadgetFormat string. The buttons are displayed evenly spaced, from left-to-right in the order in which they appear in the string.

The requesters generated by EasyRequest() appear in the visible portion of the specified screen. They do not cause the screen to scroll. Under the current implementation, the window for an easy requester will appear in the upper left corner of the display clip for the specified screen.

When a request is posted using EasyRequest() or BuildEasyRequest(), it will move the screen it appears on to the front, if that screen is not already the frontmost. This brings the request to the attention of the user. The request also comes up as the active window and could potentially steal the input focus from the current window.

When the request is satisfied the screen will be moved to back if the request caused the screen to move to the front when it was displayed. Note that the final screen position may not be the same as the original screen position.

Example Using EasyRequest()

1.16 7 / Easy Requesters / Low Level Access to Easy Requesters

The EasyRequest() function calls a lower level Intuition function named BuildEasyRequest() to construct the requester. An application can call BuildEasyRequest() directly if it needs to use an easy requester but requires custom handling of the events sent to the requester. Handling of the events should be done using the SysReqHandler() function as described below.

The BuildEasyRequest() functions take the same arguments as EasyRequest():

```
    struct Window *BuildEasyRequestArgs( struct Window *window,
                                         struct EasyStruct *easyStruct,
                                         unsigned long idcmp, APTR args );

    struct Window *BuildEasyRequest( struct Window *window,
                                     struct EasyStruct *easyStruct,
                                     unsigned long idcmp, APTR arg1, ... );
```

To process input event information directly while an easy requester is displayed, first call BuildEasyRequest() then call SysReqHandler() periodically to process user input.

```
    LONG SysReqHandler( struct Window *window, ULONG *idcmpPtr,
                        long waitInput );
```

This will provide standard handling of events but allow the application to control the timing of checking the events. This handling includes checks for left Amiga keys.

The FreeSysRequest() function must be called after an application has finished with a requester (if it was created with BuildEasyRequest() call).

```
    void FreeSysRequest( struct Window *window );
```

This function ends the requester and frees any resources allocated with

the BuildEasyRequest() call.

1.17 7 Intuition Requesters and Alerts / System Requesters

System requesters, such as DOS requests to "Insert volume foo in any drive," are created by the system using EasyRequest(). Unless otherwise specified, these requests appear on the default public screen.

System requests may appear at any time the system requires a resource that is not available. The user may be in the middle of an action, the program may be in any state.

Use the function ModifyIDCMP() to turn off all verify messages before calling any function that might generate a system requester. Neglecting to do so can cause situations where Intuition is waiting for the return of a message which the application program is unable to receive because its input is shut off while the requester is up. If Intuition finds itself in a deadlock state, the verify function will timeout and be automatically replied.

Redirecting System Requesters

1.18 7 / System Requesters / Redirecting System Requesters

A process can force the system requests which are caused by its actions to appear on a custom screen by changing the pr_WindowPtr field of its Process structure. This field may be set to three values: zero, negative one or a valid pointer to the Window structure of an open window. If pr_WindowPtr is set to zero, the request will appear on the default public screen. If pr_WindowPtr is set to negative one, the system request will never appear and the return code will be as if the user had selected the rightmost button (negative response). If pr_WindowPtr is set to a valid window pointer, then the request will appear on the same screen as the window.

The original value of pr_WindowPtr should be cached and restored before the window is closed.

1.19 7 Intuition Requesters and Alerts / Alerts

Alerts are for emergency messages. They can be displayed even when the system is in a very fragile state, such as when the system is low on memory or when some of the system lists are corrupt.

Alerts can be displayed by either the system or an application. They are reserved for urgent messages and dire warnings in situations that require the user to take some immediate action. Alerts should only be used where no other display type is possible. For instance, when the system has crashed or is about to crash, an alert could be used to inform the user of the cause.

The sudden display of an alert is a jarring experience for the user. The system stops dead while the alert is displayed and waits for the user input. For this reason, alerts should only be used when there is no recourse. If possible, use requesters or windows to display warning messages in place of alerts.

System alerts are managed entirely by Intuition. The program does not have to take any action to invoke or process these alerts. Alerts do not have access to the display database or other information required to open in specialized display modes. For this reason, alerts must appear in a display mode available on all machines, namely high resolution, non-interlaced. Alerts do not use overscan, so the display is limited to 640 by 200 on an NTSC machine, and 640 by 256 on a PAL machine.

The alert appears at the top of the video display. They are displayed the full 640 pixels wide and as tall as needed, up to the limits described above. Alerts are always displayed on a black background. The text of the alert is displayed within a rectangular border. Both the text and the border are displayed in a single color which is determined by the type of the alert.

The user responds to an alert by pressing one of the mouse buttons. The left mouse button signifies a positive response such as "Retry" or "OK". The right mouse button signifies a negative response such as "Cancel" or "Abort".

    Alerts Save Up User Input.

    The events produced by the user during an alert are not consumed by
    the alert. These events are passed through to the program when the
    alert returns. There could be a great deal of input queued and
    waiting for processing by the application.

Types of Alerts Creating Alerts Display Alert Example

1.20 7 / Alerts / Types of Alerts

There are two levels of severity for alerts:

RECOVERY_ALERT
    Recovery alerts are used in situations where the caller believes that the system can resume operations after handling the error. The alert is used as a warning, and is displayed in amber.

    A recoverable alert displays the text of the alert and flashes the border while waiting for the user to respond. It returns TRUE if the user presses the left mouse button in response to the alert, otherwise FALSE is returned.

DEADEND_ALERT
    Deadend alerts are used in situations where the caller believes that no recovery from the error is possible, and further operation of the system is impossible. This alert is used to inform the user of a fatal problem and is displayed in red. Deadend alerts are the same

as recoverable alerts in every way except color.

1.21 7 / Alerts / Creating Alerts

The function DisplayAlert() creates and displays an alert message. The message will almost always be displayed regardless of the state of the machine (with the exception of catastrophic hardware failures). If the user presses one of the mouse buttons, the display is restored to its original state, if possible. If a recoverable alert cannot be displayed (because memory is low), DisplayAlert() will return FALSE, as if the user had selected cancel. DisplayAlert() is also used by the system to display the Amiga system alert messages.

```
BOOL DisplayAlert( unsigned long alertNumber, UBYTE *string,
                   unsigned long height );
```

The alertNumber argument is a LONG value, specifying whether this is a RECOVERY_ALERT or a DEADEND_ALERT (see the <intuition/intuition.h> include file).

The string argument points to a string that is made up of one or more substrings. Each substring contains the following:

* The first component is a 16 bit x-coordinate and an 8 bit y-coordinate describing where to position the substring within the alert display. The units are in pixels. The y-coordinate describes the location of the text baseline.

* The second component is the text itself. The substring must be NULL terminated (it ends with a zero byte).

* The last component is the continuation byte. If this byte is zero, this is the last substring in the message. If this byte is non-zero, there is another substring in this alert message.

The complete string must be terminated by two NULL characters; one as the end of the last substring, and one as a NULL continuation byte, indicating that this was the last substring. The height argument is the number of display lines required for the alert.

1.22 7 Intuition Requesters and Alerts / Function Reference

The following are brief descriptions of the Intuition functions that relate to the use of Intuition requesters and alerts. See the Amiga ROM Kernel Reference Manual: Includes and Autodocs for details on each function call.

Table 7-1: Functions for Intuition Requesters and Alerts

Function	Description

Request()	Open a requester in an open window.
EndRequest()	Close an open requester in a window.
InitRequester()	Clear a requester structure before use.
EasyRequestArgs()	Open a system requester.
EasyRequest()	Alternate calling sequence for EasyRequestArgs().
BuildEasyRequestArgs()	Low level function to open an EasyRequester.
BuildEasyRequest()	Alternate calling sequence for BuildEasyRequestArgs().
SysReqHandler()	Event handler function for EasyRequestArgs().
AutoRequest()	Open a pre-V36 system requester.
BuildSysRequest()	Low level function to open an AutoRequest().
FreeSysRequest()	Low level function to close an AutoRequest().
SetDMRequest()	Set a double menu requester for an open window.
ClearDMRequest()	Clear a double menu requester from an open window.
DisplayAlert()	Open an alert on the screen.