"SIZE' Resource

Every application executing under System 7.0+, as well as every application executing under **MultiFinder**, should contain a 'SIZE' resource. One of the principal functions of the 'SIZE' resource is to inform the Operating System about the memory size requirements for the application (hence the name 'SIZE') so that the Operating System can set up an appropriately sized partition for the application. The 'SIZE' resource is also used to indicate certain scheduling options to the Operating System, such as whether the application can run in the background, whether it can accept suspend and resume events, and so forth. The 'SIZE' resource in System 7.0+ contains additional information indicating whether the application is 32-bit clean, whether the application wishes to receive notification of the termination of any applications it has launched, and whether the application wishes to receive high-level events.

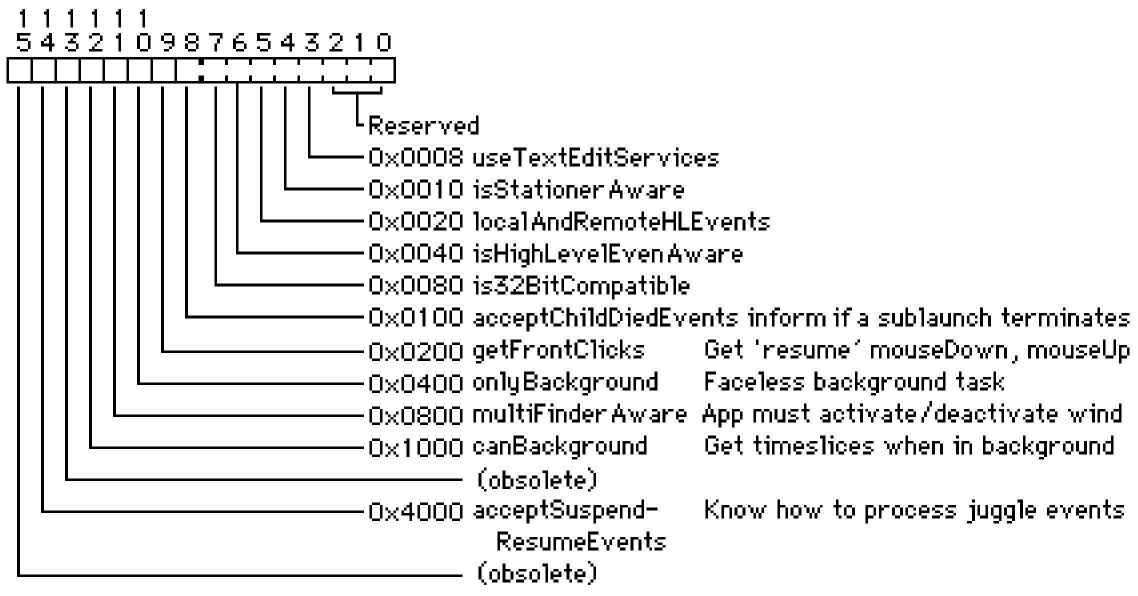
This section explains the structure of a 'SIZE' resource and the meaning of each of its fields. It also shows how to specify the Rez input for a 'SIZE' resource. You are responsible for creating the information in this resource.

A 'SIZE' resource consists of a 16-bit flags field, followed by two 32-bit size fields. The flags field specifies operating characteristics of the application, and the size fields indicate the minimum and preferred partition sizes for the application. The **minimum partition size** is the actual limit below which your application will not run. The **preferred partition size** is the memory size at which your application can run most effectively and which the Operating System attempts to secure upon launch of the application. If that amount of memory is unavailable, the application is placed into the largest contiguous block available, provided that it is larger than the specified minimum size.

Note: If the amount of available memory is between the minimum and the preferred sizes, the **Finder** displays a dialog box asking if the user wants to run the application using the amount of memory available. If your application does not have a 'SIZE' resource, it is assigned a default partition size of 512 KB.

When you define a 'SIZE' resource, you should give it a resourceID of -1. A user can modify the preferred size in the Finder's information window for your application. If the user does alter the partition size, the Operating System creates a new 'SIZE' resource having a resourceID of zero (0). At application launch time, the **Launch** function looks for a 'SIZE' resource with ID 0; if this resource is not found, it uses your original 'SIZE' resource with ID -1. This new 'SIZE' resource is also created when the user modifies any of the other settings in the resource.

The bits of the 'SIZE' resource are formatted as follows:



1 1 1 1 1 1
5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
Reserved
0x0008 useTextEditServices
0x0010 isStationerAware
0x0020 localAndRemoteHLEvents
0x0040 isHighLevelEvenAware
0x0080 is32BitCompatible
0x0100 acceptChildDiedEvents inform if a sublaunch terminates
0x0200 getFrontClicks Get 'resume' mouseDown, mouseUp
0x0400 onlyBackground Faceless background task
0x0800 multiFinderAware App must activate/deactivate wind
0x1000 canBackground Get timeslices when in background
(obsolete)
0x4000 acceptSuspend-ResumeEvents Know how to process juggle events
(obsolete)

The following Rez template shows the structure of the 'SIZE' resource.

A template for a 'SIZE' resource

```
type 'SIZE' {

    Boolean     reserved;
                //reserved

    Boolean     ignoreSuspendResumeEvents;
                //ignores suspend-resume events

    Boolean     acceptSuspendResumeEvents;
                //accepts suspend-resume events

    Boolean     reserved;
                //reserved

    Boolean     cannotBackground;
                //does no background processing

    Boolean     canBackground;
                //can use background null events

    Boolean     needsActivateOnFGSwitch;
                //needs activate event

    Boolean     doesActivateOnFGSwitch;
                //needs no activate event

    Boolean     backgroundAndForeground;
                //app has a user interface

    Boolean     onlyBackground;
```

	//app has no user interface
Boolean	dontGetFrontClicks; //no mouse events on resume
Boolean	getFrontClicks; //get mouse events on resume
Boolean	ignoreAppDiedEvents; //applications use this
Boolean	acceptAppDiedEvents; //app launchers use this
Boolean	not32BitCompatible; //works with 24-bit addr
Boolean	is32BitCompatible; //works with 24- or 32-bit addr
Boolean	notHighLevelEventAware; //can't use high-level events
Boolean	isHighLevelEventAware; //can use high-level events
Boolean	onlyLocalHLEvents; //only local high-level events
Boolean	localAndRemoteHLEvents; //also remote high-level events
Boolean	notStationeryAware; //can't use stationery documents
Boolean	isStationeryAware; //can use stationery documents
Boolean	dontUseTextEditServices; //can't use inline services
Boolean	useTextEditServices; //can use inline services
Boolean	reserved; //reserved
Boolean	reserved; //reserved
Boolean	reserved; //reserved
long	//memory sizes in bytes
long	//preferred memory size

```
    long        //minimum memory size
 };
```

The nonreserved bits in the flags field have the following meanings.

Flag descriptions

acceptSuspendResumeEvents	When set, indicates that your application can process suspend and resume events (which the Operating System sends to your application before sending it into the background or when bringing it into the foreground). In this way, your application knows when to process the global scrap.
canBackground	When set, indicates that your application wants to receive null **event processing** time while in the background. If your application has nothing to do in the background, you should not set this flag.
doesActivateOnFGSwitch	When set, indicates that your application takes responsibility for activating and deactivating any windows in response to a suspend or resume event. If the acceptSuspendResumeEvents flag is set, if the doesActivateOnFGSwitch flag is not set, and if the application is suspended, then the application receives an activate event. However, if you set the doesActivateOnFGSwitch flag, then your application won't receive activate events, and you must take care of activation and deactivation when it receives the corresponding suspend or resume event. This means that if the application's window is frontmost, the suspend event should be treated as though a deactivate event were received as well (assuming that both the doesActivateOnFGSwitch and acceptSuspendResumeEvents flags are set). For example, scroll bars should be deactivated, blinking insertion points should be hidden, and selected text should be deselected if your application moves to the back-ground. If you do not set this flag, then a window must be created to force the activate and deactivate events to occur.
onlyBackground	When set, indicates that your application runs only in the background. Usually this is because it does not have a user interface and cannot run in the foreground.
getFrontClicks	When set, indicates that your application is to receive the mouseDown and mouseUp events that are used to bring your application into the foreground when the user clicks in your application's frontmost window. Typically, the user simply wants to bring your application into the foreground, so it is usually not desirable to receive the mouse events (which would probably move the insertion point or start drawing immediately, depending on the application). The

	Finder is one application, however, that has the getFrontClicks flag set.
acceptAppDiedEvents	When set, indicates that your application is to be notified that an application launched by this application has terminated or crashed. See the **Process Manager**description for more information about launching applications and receiving application-died events.
is32BitCompatible	When set, indicates that your application can be run with the 32-bit **Memory Manager**. You should not set this flag unless you have thoroughly tested your application on a 32-bit system (such as a Macintosh IIci running System 7.0+ in 32-bit mode, or under A/UX).

The following flags have meaning only under System 7 or later.

isHighLevelEventAware	When set, indicates that your application can send and receive high-level events. If this flag is not set, the **Event Manager** does not give your application high-level events when you call **WaitNextEvent**. There is no way to mask out types of high-level events; if his flag is set, you will receive all types of high-level events sent to your application.
localAndRemoteHLEvents	When set, indicates that your application is to be visible to applications running on other computers on a network (in addition to applications running on the local machine). If this flag is not set, your application does not receive high-level events across a network.
isStationeryAware	When set, indicates that your application can recognize stationery documents. If this flag is not set and the user opens a stationery document, the Finder duplicates the document and prompts the user for a name for the duplicate document.
useTextEditServices	When set, indicates that your application can use the inline text services provided by **TextEdit**. See the **TextEdit** description for information about the inline input capabilities of **TextEdit**.

Note: If you set the acceptSuspendResumeEvents flag, you should also set the doesActivateOnFGSwitch flag.

The modifiers field in the EventRecord now contains additional information about a mouseDown event. In System 7.0+, the activeFlag modifier flag in the modifiers field of a mouseDown event record is set to indicate that the mouseDown event caused a foreground switch. Your application can use this flag to determine whether to process the mouseDown event (probably depending on whether the clicked item was visible before the foreground switch). This modifier is set for all mouseDown events that cause a foreground switch, regardless of whether your application's getFrontClicks flag is set or whether the mouse click was in your application's front window. In system versions prior to 7.0, this flag is never set for mouseDown events, and your application cannot tell if the mouse click caused a foreground switch. As a result, your application should always process a mouseDown event if its getFrontClicks flag is set.

The following program shows the input for a sample 'SIZE' resource.

The Rez input for a sample 'SIZE' resource

```
resource 'SIZE' (-1)
{
    reserved,  //reserved
    acceptSuspendResumeEvents, //accepts suspend-resume events
    reserved,  //reserved
    canBackground,    //can use background null events
    doesActivateOnFGSwitch,       //needs no activate event
    backgroundAndForeground,    //app has a user interface
    dontGetFrontClicks,    //no mouse events on resume
    ignoreAppDiedEvents,      //applications use this
    is32BitCompatible,        //works with 24- or 32-bit addr
    isHighLevelEventAware,  //can use high-level events
    localAndRemoteHLEvents, //also remote high-level events
    isStationeryAware,    //can use stationery documents
    dontUseTextEditServices, //can't use inline input services
    reserved,  //reserved
    reserved,  //reserved
    reserved,  //reserved
    kPrefSize * 1024,    //preferred memory size
    kMinSize * 1024 //minimum memory size
};
```

This resource specification indicates, among other things, that the application is 32-bit clean, can handle stationery documents, and accepts both local and network high-level events. You are responsible for defining the constants kPrefSize and kMinSize; for example, if you set kPrefSize to 50, the preferred partition size will be 50 KB.