Exhibit 42

android developers

public static class
WindowManager.LayoutParams

extends ViewGroup.LayoutParams
implements Parcelable

java.lang.Object
 ↳android.view.ViewGroup.LayoutParams
  ↳android.view.WindowManager.LayoutParams

Summary

Inherited XML Attributes	[Expand]
▶From class android.view.ViewGroup.LayoutParams	

Constants

int	ALPHA_CHANGED	
int	ANIMATION_CHANGED	
float	BRIGHTNESS_OVERRIDE_FULL	Value for screenBrightness and buttonBrightness indicating that the screen or button backlight brightness should be set to the hightest value when this window is in front.
float	BRIGHTNESS_OVERRIDE_NONE	Default value for screenBrightness and buttonBrightness indicating that the brightness value is not overridden for this window and normal brightness policy should be used.
float	BRIGHTNESS_OVERRIDE_OFF	Value for screenBrightness and buttonBrightness indicating that the screen or button backlight brightness should be set to the lowest value when this window is in front.
int	DIM_AMOUNT_CHANGED	
int	FIRST_APPLICATION_WINDOW	Start of window types that represent normal application windows.
int	FIRST_SUB_WINDOW	Start of types of sub-windows.
int	FIRST_SYSTEM_WINDOW	Start of system-specific window types.
int	FLAGS_CHANGED	
int	FLAG_ALLOW_LOCK_WHILE_SCREEN_ON	Window flag: as long as this window is visible to the user, allow the lock screen to activate while the screen is on.
int	FLAG_ALT_FOCUSABLE_IM	Window flag: invert the state of FLAG_NOT_FOCUSABLE with respect to how this window interacts with the current method.
int	FLAG_BLUR_BEHIND	*This constant is deprecated. Blurring is no longer supported.*
int	FLAG_DIM_BEHIND	Window flag: everything behind this window will be dimmed.
int	FLAG_DISMISS_KEYGUARD	Window flag: when set the window will cause the keyguard to be dismissed, only if it is not a secure lock keyguard.

int	FLAG_DITHER	Window flag: turn on dithering when compositing this window to the screen.
int	FLAG_FORCE_NOT_FULLSCREEN	Window flag: Override {@link #FLAG_FULLSCREEN and force the screen decorations (such as status bar) to be shown.
int	FLAG_FULLSCREEN	Window flag: Hide all screen decorations (e.g.
int	FLAG_HARDWARE_ACCELERATED	Indicates whether this window should be hardware accelerated.
int	FLAG_IGNORE_CHEEK_PRESSES	Window flag: intended for windows that will often be used when the user is holding the screen against their face, it will aggressively filter the event stream to prevent unintended presses in this situation that may not be desired for a particular window, when such an event stream is detected, the application will receive a CANCEL motion event to indicate this so applications can handle this accordingly by taking no action on the event until the finger is released.
int	FLAG_KEEP_SCREEN_ON	Window flag: as long as this window is visible to the user, keep the device's screen turned on and bright.
int	FLAG_LAYOUT_INSET_DECOR	Window flag: a special option only for use in combination with `FLAG_LAYOUT_IN_SCREEN`.
int	FLAG_LAYOUT_IN_SCREEN	Window flag: place the window within the entire screen, ignoring decorations around the border (a.k.a.
int	FLAG_LAYOUT_NO_LIMITS	Window flag: allow window to extend outside of the screen.
int	FLAG_NOT_FOCUSABLE	Window flag: this window won't ever get key input focus, so the user can not send key or other button events to it.
int	FLAG_NOT_TOUCHABLE	Window flag: this window can never receive touch events.
int	FLAG_NOT_TOUCH_MODAL	Window flag: Even when this window is focusable (its {@link #FLAG_NOT_FOCUSABLE is not set), allow any pointer events outside of the window to be sent to the windows behind it.
int	FLAG_SCALED	Window flag: a special mode where the layout parameters are used to perform scaling of the surface when it is composited to the screen.
int	FLAG_SECURE	Window flag: don't allow screen shots while this window is displayed.
int	FLAG_SHOW_WALLPAPER	Window flag: ask that the system wallpaper be shown behind your window.
int	FLAG_SHOW_WHEN_LOCKED	Window flag: special flag to let windows be shown when the screen is locked.
int	FLAG_SPLIT_TOUCH	Window flag: when set the window will accept for touch events outside of its bounds to be sent to other windows that also support split touch.
int	FLAG_TOUCHABLE_WHEN_WAKING	Window flag: When set, if the device is asleep when the touch screen is pressed, you will receive this first touch event.
int	FLAG_TURN_SCREEN_ON	Window flag: when set as a window is being added or made visible, once the window has been shown then the system will poke the power manager's user activity (as if the user had woken up the device) to turn the screen on.
int	FLAG_WATCH_OUTSIDE_TOUCH	Window flag: if you have set `FLAG_NOT_TOUCH_MODAL`, you can set this flag to receive a single special MotionEvent with the action `MotionEvent.ACTION_OUTSIDE` for touches that occur outside of your window.

int	FORMAT_CHANGED	
int	LAST_APPLICATION_WINDOW	End of types of application windows.
int	LAST_SUB_WINDOW	End of types of sub-windows.
int	LAST_SYSTEM_WINDOW	End of types of system windows.
int	LAYOUT_CHANGED	
int	MEMORY_TYPE_CHANGED	
int	MEMORY_TYPE_GPU	*This constant is deprecated. this is ignored, this value is set automatically when needed.*
int	MEMORY_TYPE_HARDWARE	*This constant is deprecated. this is ignored, this value is set automatically when needed.*
int	MEMORY_TYPE_NORMAL	*This constant is deprecated. this is ignored, this value is set automatically when needed.*
int	MEMORY_TYPE_PUSH_BUFFERS	*This constant is deprecated. this is ignored, this value is set automatically when needed.*
int	SCREEN_BRIGHTNESS_CHANGED	
int	SCREEN_ORIENTATION_CHANGED	
int	SOFT_INPUT_ADJUST_NOTHING	Adjustment option for `softInputMode`: set to have a window not adjust for a shown input method.
int	SOFT_INPUT_ADJUST_PAN	Adjustment option for `softInputMode`: set to have a window pan when an input method is shown, so it doesn't need to deal with resizing but just panned by the framework to ensure the current input focus is visible.
int	SOFT_INPUT_ADJUST_RESIZE	Adjustment option for `softInputMode`: set to allow the window to be resized when an input method is shown, so that its contents are not covered by the input method.
int	SOFT_INPUT_ADJUST_UNSPECIFIED	Adjustment option for `softInputMode`: nothing specified.
int	SOFT_INPUT_IS_FORWARD_NAVIGATION	Bit for `softInputMode`: set when the user has navigated forward to the window.
int	SOFT_INPUT_MASK_ADJUST	Mask for `softInputMode` of the bits that determine the way that the window should be adjusted to accommodate the soft input window.
int	SOFT_INPUT_MASK_STATE	Mask for `softInputMode` of the bits that determine the desired visibility state of the soft input area for this window.
int	SOFT_INPUT_MODE_CHANGED	
int	SOFT_INPUT_STATE_ALWAYS_HIDDEN	Visibility state for `softInputMode`: please always hide any soft input area when this window receives focus.
int	SOFT_INPUT_STATE_ALWAYS_VISIBLE	Visibility state for `softInputMode`: please always make the soft input area visible when this window receives input focus.
int	SOFT_INPUT_STATE_HIDDEN	Visibility state for `softInputMode`: please hide any soft input area when normally appropriate (when the user is navigating forward to your window).
int	SOFT_INPUT_STATE_UNCHANGED	Visibility state for `softInputMode`: please don't change the state of the soft input area.
int	SOFT_INPUT_STATE_UNSPECIFIED	Visibility state for `softInputMode`: no state has been specified.
int	SOFT_INPUT_STATE_VISIBLE	Visibility state for `softInputMode`: please show the soft input

		area when normally appropriate (when the user is navigating forward to your window).
int	TITLE_CHANGED	
int	TYPE_APPLICATION	Window type: a normal application window.
int	TYPE_APPLICATION_ATTACHED_DIALOG	Window type: like `TYPE_APPLICATION_PANEL`, but layout of the window happens as that of a top-level window, *not* as a child of its container.
int	TYPE_APPLICATION_MEDIA	Window type: window for showing media (e.g.
int	TYPE_APPLICATION_PANEL	Window type: a panel on top of an application window.
int	TYPE_APPLICATION_STARTING	Window type: special application window that is displayed while the application is starting.
int	TYPE_APPLICATION_SUB_PANEL	Window type: a sub-panel on top of an application window.
int	TYPE_BASE_APPLICATION	Window type: an application window that serves as the "base" window of the overall application; all other application windows will appear on top of it.
int	TYPE_CHANGED	
int	TYPE_INPUT_METHOD	Window type: internal input methods windows, which appear above the normal UI.
int	TYPE_INPUT_METHOD_DIALOG	Window type: internal input methods dialog windows, which appear above the current input method window.
int	TYPE_KEYGUARD	Window type: keyguard window.
int	TYPE_KEYGUARD_DIALOG	Window type: dialogs that the keyguard shows
int	TYPE_PHONE	Window type: phone.
int	TYPE_PRIORITY_PHONE	Window type: priority phone UI, which needs to be displayed even if the keyguard is active.
int	TYPE_SEARCH_BAR	Window type: the search bar.
int	TYPE_STATUS_BAR	Window type: the status bar.
int	TYPE_STATUS_BAR_PANEL	Window type: panel that slides out from over the status bar
int	TYPE_SYSTEM_ALERT	Window type: system window, such as low power alert.
int	TYPE_SYSTEM_DIALOG	Window type: panel that slides out from the status bar
int	TYPE_SYSTEM_ERROR	Window type: internal system error windows, appear on top of everything they can.
int	TYPE_SYSTEM_OVERLAY	Window type: system overlay windows, which need to be displayed on top of everything else.
int	TYPE_TOAST	Window type: transient notifications.
int	TYPE_WALLPAPER	Window type: wallpaper window, placed behind any window that wants to sit on top of the wallpaper.

Inherited Constants [Expand]

▸From class android.view.ViewGroup.LayoutParams

▸From interface android.os.Parcelable

Fields

public static final Creator<WindowManager.LayoutParams>	CREATOR	
public float	alpha	An alpha value to apply to this entire window.
public float	buttonBrightness	This can be used to override the standard behavior of the button and keyboard backlights.
public float	dimAmount	When `FLAG_DIM_BEHIND` is set, this is the amount of dimming to apply.
public int	flags	Various behavioral options/flags.
public int	format	The desired bitmap format.
public int	gravity	Placement of window within the screen as per `Gravity`.
public float	horizontalMargin	The horizontal margin, as a percentage of the container's width, between the container and the widget.
public float	horizontalWeight	Indicates how much of the extra space will be allocated horizontally to the view associated with these LayoutParams.
public int	memoryType	*This field is deprecated. this is ignored*
public String	packageName	Name of the package owning this window.
public float	screenBrightness	This can be used to override the user's preferred brightness of the screen.
public int	screenOrientation	Specific orientation value for a window.
public int	softInputMode	Desired operating mode for any soft input area.
public int	systemUiVisibility	Control the visibility of the status bar.
public IBinder	token	Identifier for this window.
public int	type	The general type of window.
public float	verticalMargin	The vertical margin, as a percentage of the container's height, between the container and the widget.
public float	verticalWeight	Indicates how much of the extra space will be allocated vertically to the view associated with these LayoutParams.
public int	windowAnimations	A style resource defining the animations to use for this window.
public int	x	X position for this window.
public int	y	Y position for this window.

Inherited Fields	[Expand]
▶From class android.view.ViewGroup.LayoutParams	

Public Constructors
WindowManager.LayoutParams ()
WindowManager.LayoutParams (int _type)
WindowManager.LayoutParams (int _type, int _flags)
WindowManager.LayoutParams (int _type, int _flags, int _format)
WindowManager.LayoutParams (int w, int h, int _type, int _flags, int _format)
WindowManager.LayoutParams (int w, int h, int xpos, int ypos, int _type, int _flags, int _format)
WindowManager.LayoutParams (Parcel in)

Public Methods	
final int	copyFrom (WindowManager.LayoutParams o)
String	debug (String output) Returns a String representation of this set of layout parameters.
int	describeContents () Describe the kinds of special objects contained in this Parcelable's marshalled representation.
final CharSequence	getTitle ()
static boolean	mayUseInputMethod (int flags) Given a particular set of window manager flags, determine whether such a window may be a target for an input method when it has focus.
final void	setTitle (CharSequence title)
String	toString () Returns a string containing a concise, human-readable description of this object.
void	writeToParcel (Parcel out, int parcelableFlags) Flatten this object in to a Parcel.

Inherited Methods	[Expand]
▶ From class android.view.ViewGroup.LayoutParams	
▶ From class java.lang.Object	
▶ From interface android.os.Parcelable	

Constants

public static final int **ALPHA_CHANGED** Since: API Level 1

Constant Value: 128 (0x00000080)

public static final int **ANIMATION_CHANGED** Since: API Level 1

Constant Value: 16 (0x00000010)

public static final float **BRIGHTNESS_OVERRIDE_FULL** Since: API Level 8

Value for `screenBrightness` and `buttonBrightness` indicating that the screen or button backlight brightness should be set to the hightest value when this window is in front.

Constant Value: 1.0

public static final float **BRIGHTNESS_OVERRIDE_NONE** Since: API Level 8

Default value for `screenBrightness` and `buttonBrightness` indicating that the brightness value is not overridden for this window and normal brightness policy should be used.

Constant Value: -1.0

public static final float **BRIGHTNESS_OVERRIDE_OFF** Since: API Level 8

Value for `screenBrightness` and `buttonBrightness` indicating that the screen or button backlight brightness should be set to the lowest value when this window is in front.

Constant Value: 0.0

public static final int **DIM_AMOUNT_CHANGED** Since: API Level 1

Constant Value: 32 (0x00000020)

public static final int **FIRST_APPLICATION_WINDOW** Since: API Level 1

Start of window types that represent normal application windows.

Constant Value: 1 (0x00000001)

public static final int **FIRST_SUB_WINDOW** Since: API Level 1

Start of types of sub-windows. The `token` of these windows must be set to the window they are attached to. These types of windows are kept next to their attached window in Z-order, and their coordinate space is relative to their attached window.

Constant Value: 1000 (0x000003e8)

public static final int **FIRST_SYSTEM_WINDOW** Since: API Level 1

Start of system-specific window types. These are not normally created by applications.

Constant Value: 2000 (0x000007d0)

public static final int **FLAGS_CHANGED** Since: API Level 1

Constant Value: 4 (0x00000004)

public static final int **FLAG_ALLOW_LOCK_WHILE_SCREEN_ON** Since: API Level 8

Window flag: as long as this window is visible to the user, allow the lock screen to activate while the screen is on. This can be used independently, or in combination with `FLAG_KEEP_SCREEN_ON` and/or `FLAG_SHOW_WHEN_LOCKED`

Constant Value: 1 (0x00000001)

public static final int **FLAG_ALT_FOCUSABLE_IM** Since: API Level 3

Window flag: invert the state of `FLAG_NOT_FOCUSABLE` with respect to how this window interacts with the current method. That is, if FLAG_NOT_FOCUSABLE is set and this flag is set, then the window will behave as if it needs to interact with the input method and thus be placed behind/away from it; if FLAG_NOT_FOCUSABLE is not set and this flag is set, then the window will behave as if it doesn't need to interact with the input method and can be placed to use more space and cover the input method.

Constant Value: 131072 (0x00020000)

public static final int **FLAG_BLUR_BEHIND**

Since: API Level 1

This constant is deprecated.
Blurring is no longer supported.

Window flag: blur everything behind this window.

Constant Value: 4 (0x00000004)

public static final int **FLAG_DIM_BEHIND**

Since: API Level 1

Window flag: everything behind this window will be dimmed. Use `dimAmount` to control the amount of dim.

Constant Value: 2 (0x00000002)

public static final int **FLAG_DISMISS_KEYGUARD**

Since: API Level 5

Window flag: when set the window will cause the keyguard to be dismissed, only if it is not a secure lock keyguard. Because such a keyguard is not needed for security, it will never re-appear if the user navigates to another window (in contrast to `FLAG_SHOW_WHEN_LOCKED`, which will only temporarily hide both secure and non-secure keyguards but ensure they reappear when the user moves to another UI that doesn't hide them). If the keyguard is currently active and is secure (requires an unlock pattern) than the user will still need to confirm it before seeing this window, unless `FLAG_SHOW_WHEN_LOCKED` has also been set.

Constant Value: 4194304 (0x00400000)

public static final int **FLAG_DITHER**

Since: API Level 1

Window flag: turn on dithering when compositing this window to the screen.

Constant Value: 4096 (0x00001000)

public static final int **FLAG_FORCE_NOT_FULLSCREEN**

Since: API Level 1

Window flag: Override {@link #FLAG_FULLSCREEN and force the screen decorations (such as status bar) to be shown.

Constant Value: 2048 (0x00000800)

public static final int **FLAG_FULLSCREEN**

Since: API Level 1

Window flag: Hide all screen decorations (e.g. status bar) while this window is displayed. This allows the window to use the entire display space for itself -- the status bar will be hidden when an app window with this flag set is on the top layer.

Constant Value: 1024 (0x00000400)

public static final int **FLAG_HARDWARE_ACCELERATED**

Since: API Level 11

Indicates whether this window should be hardware accelerated. Requesting hardware acceleration does not guarantee it will happen.

This flag can be controlled programmatically *only* to enable hardware acceleration. To enable hardware acceleration for a given window programmatically, do the following:

```
Window w = activity.getWindow(); // in Activity's onCreate() for instance
w.setFlags(WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED,
        WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED);
```

It is important to remember that this flag **must** be set before setting the content view of your activity or dialog.

This flag cannot be used to disable hardware acceleration after it was enabled in your manifest using `hardwareAccelerated`. If you need to selectively and programmatically disable hardware acceleration (for automated testing for instance), make sure it is turned off in your manifest and enable it on your activity or dialog when you need it instead, using the method described above.

This flag is automatically set by the system if the `android:hardwareAccelerated` XML attribute is set to true on an activity or on the application.

Constant Value: 16777216 (0x01000000)

public static final int **FLAG_IGNORE_CHEEK_PRESSES** Since: API Level 1

Window flag: intended for windows that will often be used when the user is holding the screen against their face, it will aggressively filter the event stream to prevent unintended presses in this situation that may not be desired for a particular window, when such an event stream is detected, the application will receive a CANCEL motion event to indicate this so applications can handle this accordingly by taking no action on the event until the finger is released.

Constant Value: 32768 (0x00008000)

public static final int **FLAG_KEEP_SCREEN_ON** Since: API Level 1

Window flag: as long as this window is visible to the user, keep the device's screen turned on and bright.

Constant Value: 128 (0x00000080)

public static final int **FLAG_LAYOUT_INSET_DECOR** Since: API Level 1

Window flag: a special option only for use in combination with `FLAG_LAYOUT_IN_SCREEN`. When requesting layout in the screen your window may appear on top of or behind screen decorations such as the status bar. By also including this flag, the window manager will report the inset rectangle needed to ensure your content is not covered by screen decorations. This flag is normally set for you by Window as described in `setFlags(int, int)`.

Constant Value: 65536 (0x00010000)

public static final int **FLAG_LAYOUT_IN_SCREEN** Since: API Level 1

Window flag: place the window within the entire screen, ignoring decorations around the border (a.k.a. the status bar). The window must correctly position its contents to take the screen decoration into account. This flag is normally set for you by Window as described in `setFlags(int, int)`.

Constant Value: 256 (0x00000100)

public static final int **FLAG_LAYOUT_NO_LIMITS** Since: API Level 1

Window flag: allow window to extend outside of the screen.

Constant Value: 512 (0x00000200)

public static final int **FLAG_NOT_FOCUSABLE** Since: API Level 1

Window flag: this window won't ever get key input focus, so the user can not send key or other button events to it. Those will instead go to whatever focusable window is behind it. This flag will also enable `FLAG_NOT_TOUCH_MODAL` whether or not that is explicitly set.

Setting this flag also implies that the window will not need to interact with a soft input method, so it will be Z-ordered and positioned independently of any active input method (typically this means it gets Z-ordered on top of the input method, so it can use the full screen for its content and cover the input method if needed. You can use `FLAG_ALT_FOCUSABLE_IM` to modify this behavior.

Constant Value: 8 (0x00000008)

public static final int **FLAG_NOT_TOUCHABLE**

Window flag: this window can never receive touch events.

Constant Value: 16 (0x00000010)

public static final int **FLAG_NOT_TOUCH_MODAL** Since: API Level 1

Window flag: Even when this window is focusable (its {@link #FLAG_NOT_FOCUSABLE is not set), allow any pointer events outside of the window to be sent to the windows behind it. Otherwise it will consume all pointer events itself, regardless of whether they are inside of the window.

Constant Value: 32 (0x00000020)

public static final int **FLAG_SCALED** Since: API Level 1

Window flag: a special mode where the layout parameters are used to perform scaling of the surface when it is composited to the screen.

Constant Value: 16384 (0x00004000)

public static final int **FLAG_SECURE** Since: API Level 1

Window flag: don't allow screen shots while this window is displayed. Maps to Surface.SECURE.

Constant Value: 8192 (0x00002000)

public static final int **FLAG_SHOW_WALLPAPER** Since: API Level 5

Window flag: ask that the system wallpaper be shown behind your window. The window surface must be translucent to be able to actually see the wallpaper behind it; this flag just ensures that the wallpaper surface will be there if this window actually has translucent regions.

Constant Value: 1048576 (0x00100000)

public static final int **FLAG_SHOW_WHEN_LOCKED** Since: API Level 5

Window flag: special flag to let windows be shown when the screen is locked. This will let application windows take precedence over key guard or any other lock screens. Can be used with `FLAG_KEEP_SCREEN_ON` to turn screen on and display windows directly before showing the key guard window. Can be used with `FLAG_DISMISS_KEYGUARD` to automatically fully dismisss non-secure keyguards. This flag only applies to the top-most full-screen window.

Constant Value: 524288 (0x00080000)

public static final int **FLAG_SPLIT_TOUCH** Since: API Level 11

Window flag: when set the window will accept for touch events outside of its bounds to be sent to other windows that also support split touch. When this flag is not set, the first pointer that goes down determines the window to which all subsequent touches go until all pointers go up. When this flag is set, each pointer (not necessarily the first) that goes down determines the window to which all subsequent touches of that pointer will go until that pointer goes up thereby enabling touches with multiple pointers to be split across multiple windows.

Constant Value: 8388608 (0x00800000)

public static final int **FLAG_TOUCHABLE_WHEN_WAKING** Since: API Level 1

Window flag: When set, if the device is asleep when the touch screen is pressed, you will receive this first touch event. Usually the first touch event is consumed by the system since the user can not see what they are pressing on.

Constant Value: 64 (0x00000040)

public static final int **FLAG_TURN_SCREEN_ON** Since: API Level 5

Window flag: when set as a window is being added or made visible, once the window has been shown then the system will poke the power manager's user activity (as if the user had woken up the device) to turn the screen on.

Constant Value: 2097152 (0x00200000)

public static final int **FLAG_WATCH_OUTSIDE_TOUCH** Since: API Level 3

Window flag: if you have set `FLAG_NOT_TOUCH_MODAL`, you can set this flag to receive a single special MotionEvent with the action `MotionEvent.ACTION_OUTSIDE` for touches that occur outside of your window. Note that you will not receive the full down/move/up gesture, only the location of the first down as an ACTION_OUTSIDE.

Constant Value: 262144 (0x00040000)

public static final int **FORMAT_CHANGED** Since: API Level 1

Constant Value: 8 (0x00000008)

public static final int **LAST_APPLICATION_WINDOW** Since: API Level 1

End of types of application windows.

Constant Value: 99 (0x00000063)

public static final int **LAST_SUB_WINDOW** Since: API Level 1

End of types of sub-windows.

Constant Value: 1999 (0x000007cf)

public static final int **LAST_SYSTEM_WINDOW** Since: API Level 1

End of types of system windows.

Constant Value: 2999 (0x00000bb7)

public static final int **LAYOUT_CHANGED** Since: API Level 1

Constant Value: 1 (0x00000001)

public static final int **MEMORY_TYPE_CHANGED** Since: API Level 1

Constant Value: 256 (0x00000100)

public static final int **MEMORY_TYPE_GPU** Since: API Level 1

This constant is deprecated.
this is ignored, this value is set automatically when needed.

Constant Value: 2 (0x00000002)

public static final int **MEMORY_TYPE_HARDWARE** Since: API Level 1

This constant is deprecated.
this is ignored, this value is set automatically when needed.

Constant Value: 1 (0x00000001)

public static final int **MEMORY_TYPE_NORMAL** Since: API Level 1

This constant is deprecated.
this is ignored, this value is set automatically when needed.

Constant Value: 0 (0x00000000)

public static final int **MEMORY_TYPE_PUSH_BUFFERS**

This constant is deprecated.
this is ignored, this value is set automatically when needed.

Constant Value: 3 (0x00000003)

public static final int **SCREEN_BRIGHTNESS_CHANGED** Since: API Level 3

Constant Value: 2048 (0x00000800)

public static final int **SCREEN_ORIENTATION_CHANGED** Since: API Level 3

Constant Value: 1024 (0x00000400)

public static final int **SOFT_INPUT_ADJUST_NOTHING** Since: API Level 11

Adjustment option for `softInputMode`: set to have a window not adjust for a shown input method. The window will not be resized, and it will not be panned to make its focus visible.

Constant Value: 48 (0x00000030)

public static final int **SOFT_INPUT_ADJUST_PAN** Since: API Level 3

Adjustment option for `softInputMode`: set to have a window pan when an input method is shown, so it doesn't need to deal with resizing but just panned by the framework to ensure the current input focus is visible. This can *not* be combined with `SOFT_INPUT_ADJUST_RESIZE`; if neither of these are set, then the system will try to pick one or the other depending on the contents of the window.

Constant Value: 32 (0x00000020)

public static final int **SOFT_INPUT_ADJUST_RESIZE** Since: API Level 3

Adjustment option for `softInputMode`: set to allow the window to be resized when an input method is shown, so that its contents are not covered by the input method. This can *not* be combined with `SOFT_INPUT_ADJUST_PAN`; if neither of these are set, then the system will try to pick one or the other depending on the contents of the window.

Constant Value: 16 (0x00000010)

public static final int **SOFT_INPUT_ADJUST_UNSPECIFIED** Since: API Level 3

Adjustment option for `softInputMode`: nothing specified. The system will try to pick one or the other depending on the contents of the window.

Constant Value: 0 (0x00000000)

public static final int **SOFT_INPUT_IS_FORWARD_NAVIGATION** Since: API Level 3

Bit for `softInputMode`: set when the user has navigated forward to the window. This is normally set automatically for you by the system, though you may want to set it in certain cases when you are displaying a window yourself. This flag will always be cleared automatically after the window is displayed.

Constant Value: 256 (0x00000100)

public static final int **SOFT_INPUT_MASK_ADJUST** Since: API Level 3

Mask for `softInputMode` of the bits that determine the way that the window should be adjusted to accommodate the soft input window.

Constant Value: 240 (0x000000f0)

public static final int **SOFT_INPUT_MASK_STATE** Since: API Level 3

Mask for `softInputMode` of the bits that determine the desired visibility state of the soft input area for this window.

Constant Value: 15 (0x0000000f)

public static final int **SOFT_INPUT_MODE_CHANGED** Since: API Level 3

Constant Value: 512 (0x00000200)

public static final int **SOFT_INPUT_STATE_ALWAYS_HIDDEN** Since: API Level 3

Visibility state for `softInputMode`: please always hide any soft input area when this window receives focus.

Constant Value: 3 (0x00000003)

public static final int **SOFT_INPUT_STATE_ALWAYS_VISIBLE** Since: API Level 3

Visibility state for `softInputMode`: please always make the soft input area visible when this window receives input focus.

Constant Value: 5 (0x00000005)

public static final int **SOFT_INPUT_STATE_HIDDEN** Since: API Level 3

Visibility state for `softInputMode`: please hide any soft input area when normally appropriate (when the user is navigating forward to your window).

Constant Value: 2 (0x00000002)

public static final int **SOFT_INPUT_STATE_UNCHANGED** Since: API Level 3

Visibility state for `softInputMode`: please don't change the state of the soft input area.

Constant Value: 1 (0x00000001)

public static final int **SOFT_INPUT_STATE_UNSPECIFIED** Since: API Level 3

Visibility state for `softInputMode`: no state has been specified.

Constant Value: 0 (0x00000000)

public static final int **SOFT_INPUT_STATE_VISIBLE** Since: API Level 3

Visibility state for `softInputMode`: please show the soft input area when normally appropriate (when the user is navigating forward to your window).

Constant Value: 4 (0x00000004)

public static final int **TITLE_CHANGED** Since: API Level 1

Constant Value: 64 (0x00000040)

public static final int **TYPE_APPLICATION** Since: API Level 1

Window type: a normal application window. The `token` must be an Activity token identifying who the window belongs to.

Constant Value: 2 (0x00000002)

public static final int **TYPE_APPLICATION_ATTACHED_DIALOG** Since: API Level 3

Window type: like `TYPE_APPLICATION_PANEL`, but layout of the window happens as that of a top-level window, *not* as a child of its container.

Constant Value: 1003 (0x000003eb)

public static final int **TYPE_APPLICATION_MEDIA** Since: API Level 1

Window type: window for showing media (e.g. video). These windows are displayed behind their attached window.

Constant Value: 1001 (0x000003e9)

public static final int **TYPE_APPLICATION_PANEL** Since: API Level 1

Window type: a panel on top of an application window. These windows appear on top of their attached window.

Constant Value: 1000 (0x000003e8)

public static final int **TYPE_APPLICATION_STARTING** Since: API Level 1

Window type: special application window that is displayed while the application is starting. Not for use by applications themselves; this is used by the system to display something until the application can show its own windows.

Constant Value: 3 (0x00000003)

public static final int **TYPE_APPLICATION_SUB_PANEL** Since: API Level 1

Window type: a sub-panel on top of an application window. These windows are displayed on top their attached window and any `TYPE_APPLICATION_PANEL` panels.

Constant Value: 1002 (0x000003ea)

public static final int **TYPE_BASE_APPLICATION** Since: API Level 1

Window type: an application window that serves as the "base" window of the overall application; all other application windows will appear on top of it.

Constant Value: 1 (0x00000001)

public static final int **TYPE_CHANGED** Since: API Level 1

Constant Value: 2 (0x00000002)

public static final int **TYPE_INPUT_METHOD** Since: API Level 3

Window type: internal input methods windows, which appear above the normal UI. Application windows may be resized or panned to keep the input focus visible while this window is displayed.

Constant Value: 2011 (0x000007db)

public static final int **TYPE_INPUT_METHOD_DIALOG** Since: API Level 3

Window type: internal input methods dialog windows, which appear above the current input method window.

Constant Value: 2012 (0x000007dc)

public static final int **TYPE_KEYGUARD** Since: API Level 1

Window type: keyguard window.

Constant Value: 2004 (0x000007d4)

public static final int **TYPE_KEYGUARD_DIALOG** Since: API Level 1

Window type: dialogs that the keyguard shows

Constant Value: 2009 (0x000007d9)

public static final int **TYPE_PHONE** Since: API Level 1

Window type: phone. These are non-application windows providing user interaction with the phone (in particular incoming calls). These windows are normally placed above all applications, but behind the status bar.

Constant Value: 2002 (0x000007d2)

public static final int **TYPE_PRIORITY_PHONE** Since: API Level 1

Window type: priority phone UI, which needs to be displayed even if the keyguard is active. These windows must not take input focus, or they will interfere with the keyguard.

Constant Value: 2007 (0x000007d7)

public static final int **TYPE_SEARCH_BAR** Since: API Level 1

Window type: the search bar. There can be only one search bar window; it is placed at the top of the screen.

Constant Value: 2001 (0x000007d1)

public static final int **TYPE_STATUS_BAR** Since: API Level 1

Window type: the status bar. There can be only one status bar window; it is placed at the top of the screen, and all other windows are shifted down so they are below it.

Constant Value: 2000 (0x000007d0)

public static final int **TYPE_STATUS_BAR_PANEL** Since: API Level 1

Window type: panel that slides out from over the status bar

Constant Value: 2014 (0x000007de)

public static final int **TYPE_SYSTEM_ALERT** Since: API Level 1

Window type: system window, such as low power alert. These windows are always on top of application windows.

Constant Value: 2003 (0x000007d3)

public static final int **TYPE_SYSTEM_DIALOG** Since: API Level 1

Window type: panel that slides out from the status bar

Constant Value: 2008 (0x000007d8)

public static final int **TYPE_SYSTEM_ERROR** Since: API Level 1

Window type: internal system error windows, appear on top of everything they can.

Constant Value: 2010 (0x000007da)

public static final int **TYPE_SYSTEM_OVERLAY** Since: API Level 1

Window type: system overlay windows, which need to be displayed on top of everything else. These windows must not take input focus, or they will interfere with the keyguard.

Constant Value: 2006 (0x000007d6)

public static final int **TYPE_TOAST** Since: API Level 1

Window type: transient notifications.

Constant Value: 2005 (0x000007d5)

public static final int **TYPE_WALLPAPER** Since: API Level 5

Window type: wallpaper window, placed behind any window that wants to sit on top of the wallpaper.

Constant Value: 2013 (0x000007dd)

Fields

public static final Creator<WindowManager.LayoutParams> **CREATOR** Since: API Level 1

public float **alpha** Since: API Level 1

An alpha value to apply to this entire window. An alpha of 1.0 means fully opaque and 0.0 means fully transparent

public float **buttonBrightness** Since: API Level 8

This can be used to override the standard behavior of the button and keyboard backlights. A value of less than 0, the default, means to use the standard backlight behavior. 0 to 1 adjusts the brightness from dark to full bright.

public float **dimAmount** Since: API Level 1

When `FLAG_DIM_BEHIND` is set, this is the amount of dimming to apply. Range is from 1.0 for completely opaque to 0.0 for no dim.

public int **flags** Since: API Level 1

Various behavioral options/flags. Default is none.

See Also

`FLAG_ALLOW_LOCK_WHILE_SCREEN_ON`
`FLAG_DIM_BEHIND`
`FLAG_NOT_FOCUSABLE`
`FLAG_NOT_TOUCHABLE`
`FLAG_NOT_TOUCH_MODAL`
`FLAG_TOUCHABLE_WHEN_WAKING`
`FLAG_KEEP_SCREEN_ON`
`FLAG_LAYOUT_IN_SCREEN`
`FLAG_LAYOUT_NO_LIMITS`
`FLAG_FULLSCREEN`
`FLAG_FORCE_NOT_FULLSCREEN`
`FLAG_DITHER`
`FLAG_SECURE`
`FLAG_SCALED`
`FLAG_IGNORE_CHEEK_PRESSES`
`FLAG_LAYOUT_INSET_DECOR`
`FLAG_ALT_FOCUSABLE_IM`
`FLAG_WATCH_OUTSIDE_TOUCH`
`FLAG_SHOW_WHEN_LOCKED`
`FLAG_SHOW_WALLPAPER`
`FLAG_TURN_SCREEN_ON`
`FLAG_DISMISS_KEYGUARD`
`FLAG_SPLIT_TOUCH`
`FLAG_HARDWARE_ACCELERATED`

public int **format** Since: API Level 1

The desired bitmap format. May be one of the constants in `PixelFormat`. Default is OPAQUE.

public int **gravity**

Since: API Level 1

Placement of window within the screen as per `Gravity`. Both `Gravity.apply` and `Gravity.applyDisplay` are used during window layout, with this value given as the desired gravity. For example you can specify `Gravity.DISPLAY_CLIP_HORIZONTAL` and `Gravity.DISPLAY_CLIP_VERTICAL` here to control the behavior of `Gravity.applyDisplay`.

See Also

`Gravity`

public float **horizontalMargin**

Since: API Level 1

The horizontal margin, as a percentage of the container's width, between the container and the widget. See `Gravity.apply` for how this is used. This field is added with `x` to supply the *xAdj* parameter.

public float **horizontalWeight**

Since: API Level 1

Indicates how much of the extra space will be allocated horizontally to the view associated with these LayoutParams. Specify 0 if the view should not be stretched. Otherwise the extra pixels will be pro-rated among all views whose weight is greater than 0.

public int **memoryType**

Since: API Level 1

This field is deprecated.
this is ignored

public String **packageName**

Since: API Level 1

Name of the package owning this window.

public float **screenBrightness**

Since: API Level 3

This can be used to override the user's preferred brightness of the screen. A value of less than 0, the default, means to use the preferred screen brightness. 0 to 1 adjusts the brightness from dark to full bright.

public int **screenOrientation**

Since: API Level 3

Specific orientation value for a window. May be any of the same values allowed for `screenOrientation`. If not set, a default value of `SCREEN_ORIENTATION_UNSPECIFIED` will be used.

public int **softInputMode**

Since: API Level 3

Desired operating mode for any soft input area. May be any combination of:

- One of the visibility states `SOFT_INPUT_STATE_UNSPECIFIED`, `SOFT_INPUT_STATE_UNCHANGED`, `SOFT_INPUT_STATE_HIDDEN`, `SOFT_INPUT_STATE_ALWAYS_VISIBLE`, or `SOFT_INPUT_STATE_VISIBLE`.
- One of the adjustment options `SOFT_INPUT_ADJUST_UNSPECIFIED`, `SOFT_INPUT_ADJUST_RESIZE`, or `SOFT_INPUT_ADJUST_PAN`.

public int **systemUiVisibility**

Since: API Level 11

Control the visibility of the status bar.

See Also

`STATUS_BAR_VISIBLE`
`STATUS_BAR_HIDDEN`

public IBinder **token** Since: API Level 1

Identifier for this window. This will usually be filled in for you.

public int **type** Since: API Level 1

The general type of window. There are three main classes of window types:

- **Application windows** (ranging from `FIRST_APPLICATION_WINDOW` to `LAST_APPLICATION_WINDOW`) are normal top-level application windows. For these types of windows, the `token` must be set to the token of the activity they are a part of (this will normally be done for you if `token` is null).
- **Sub-windows** (ranging from `FIRST_SUB_WINDOW` to `LAST_SUB_WINDOW`) are associated with another top-level window. For these types of windows, the `token` must be the token of the window it is attached to.
- **System windows** (ranging from `FIRST_SYSTEM_WINDOW` to `LAST_SYSTEM_WINDOW`) are special types of windows for use by the system for specific purposes. They should not normally be used by applications, and a special permission is required to use them.

See Also

`TYPE_BASE_APPLICATION`
`TYPE_APPLICATION`
`TYPE_APPLICATION_STARTING`
`TYPE_APPLICATION_PANEL`
`TYPE_APPLICATION_MEDIA`
`TYPE_APPLICATION_SUB_PANEL`
`TYPE_APPLICATION_ATTACHED_DIALOG`
`TYPE_STATUS_BAR`
`TYPE_SEARCH_BAR`
`TYPE_PHONE`
`TYPE_SYSTEM_ALERT`
`TYPE_KEYGUARD`
`TYPE_TOAST`
`TYPE_SYSTEM_OVERLAY`
`TYPE_PRIORITY_PHONE`
`TYPE_STATUS_BAR_PANEL`
`TYPE_SYSTEM_DIALOG`
`TYPE_KEYGUARD_DIALOG`
`TYPE_SYSTEM_ERROR`
`TYPE_INPUT_METHOD`
`TYPE_INPUT_METHOD_DIALOG`

public float **verticalMargin** Since: API Level 1

The vertical margin, as a percentage of the container's height, between the container and the widget. See `Gravity.apply` for how this is used. This field is added with `y` to supply the *yAdj* parameter.

public float **verticalWeight** Since: API Level 1

Indicates how much of the extra space will be allocated vertically to the view associated with these LayoutParams. Specify 0 if the view should not be stretched. Otherwise the extra pixels will be pro-rated among all views whose weight is greater than 0.

public int **windowAnimations** Since: API Level 1

A style resource defining the animations to use for this window. This must be a system resource; it can not be an application resource because the window manager does not have access to applications.

public int **x** Since: API Level 1

X position for this window. With the default gravity it is ignored. When using `LEFT` or `START` or `RIGHT` or `END` it provides an offset from the given edge.

public int **y** Since: API Level 1

Y position for this window. With the default gravity it is ignored. When using `TOP` or `BOTTOM` it provides an offset from the given edge.

Public Constructors

public **WindowManager.LayoutParams** () Since: API Level 1

public **WindowManager.LayoutParams** (int _type) Since: API Level 1

public **WindowManager.LayoutParams** (int _type, int _flags) Since: API Level 1

public **WindowManager.LayoutParams** (int _type, int _flags, int _format) Since: API Level 1

public **WindowManager.LayoutParams** (int w, int h, int _type, int _flags, int _format) Since: API Level 1

public **WindowManager.LayoutParams** (int w, int h, int xpos, int ypos, int _type, int _flags, int _format) Since: API Level 1

public **WindowManager.LayoutParams** (Parcel in) Since: API Level 1

Public Methods

public final int **copyFrom** (WindowManager.LayoutParams o) Since: API Level 1

public String **debug** (String output) Since: API Level 1

Returns a String representation of this set of layout parameters.

Parameters

output	the String to prepend to the internal representation

Returns

a String with the following format: output + "ViewGroup.LayoutParams={ width=WIDTH, height=HEIGHT }"

public int **describeContents** () Since: API Level 1

Describe the kinds of special objects contained in this Parcelable's marshalled representation.

Returns

a bitmask indicating the set of special object types marshalled by the Parcelable.

public final CharSequence **getTitle** () Since: API Level 1 public static boolean **mayUseInputMethod** (int flags) Since: API Level 3

Given a particular set of window manager flags, determine whether such a window may be a target for an input method when it has focus. In particular, this checks the `FLAG_NOT_FOCUSABLE` and `FLAG_ALT_FOCUSABLE_IM` flags and returns true if the combination of the two corresponds to a window that needs to be behind the input method so that the user can type into it.

Parameters

flags	The current window manager flags.

Returns

Returns true if such a window should be behind/interact with an input method, false if not.

public final void **setTitle** (CharSequence title) Since: API Level 1

public String **toString** () Since: API Level 1

Returns a string containing a concise, human-readable description of this object. Subclasses are encouraged to override this method and provide an implementation that takes into account the object's type and data. The default implementation is equivalent to the following expression:

```
getClass().getName() + '@' + Integer.toHexString(hashCode())
```

See Writing a useful `toString` method if you intend implementing your own `toString` method.

Returns

a printable representation of this object.

public void **writeToParcel** (Parcel out, int parcelableFlags) Since: API Level 1

Flatten this object in to a Parcel.

Parameters

out	The Parcel in which the object should be written.
parcelableFlags	Additional flags about how the object should be written. May be 0 or `PARCELABLE_WRITE_RETURN_VALUE`.