EGL 1.4 API Quick Reference Card - Page 1

EGL™ is an interface between Khronos rendering APIs such as OpenGL ES or OpenVG and the underlying native platform window system. It handles graphics context management, surface/buffer binding, and rendering synchronization and enables high-performance, accelerated, mixed-mode 2D and 3D rendering using other Khronos APIs. An EGL implementation may not support all possible client APIs (OpenGL, OpenGL ES, and OpenVG) defined by the specification. Functions requiring an unsupported client API will generate errors when called.

- **[n.n.n]** refers to sections in the EGL 1.4 specification, available at www.khronos.org/egl

Types [2.1.1]

unsigned int	EGLBoolean
unsigned int	EGLenum
void	*EGLConfig
void	*EGLContext
void	*EGLDisplay
void	*EGLSurface
void	*EGLClientBuffer

The following types differ based on platform.

Windows platform:

HDC	EGLNativeDisplayType
HBITMAP	EGLNativePixmapType
HWND	EGLNativeWindowType

Linux/X11 platform:

Display	*EGLNativeDisplayType
Pixmap	EGLNativePixmapType
Window	EGLNativeWindowType

Android platform:

ANativeWindow*	EGLNativeWindowType

Attribute Lists

The *attrib_list* parameter used in various EGL functions may be either NULL, or a pointer to an array of <name,value> pairs terminated by EGL_NONE. The allowed <name> depends on the EGL function, and the <value>s depend on the <name>.

If an attribute is not specified in *attrib_list*, then the default value is used. If EGL_DONT_CARE is specified as an attribute value, then the attribute will not be checked. EGL_DONT_CARE may be specified for all attributes except EGL_LEVEL.

Attribute names for EGLConfig (see legend below table):

Attribute Name	Type	Sorting	Selection	Description and default value
EGL_BUFFER_SIZE	integer	(4) (-)	AtLeast	The total color component bits in the color buffer. Default is **0**.
EGL_RED_SIZE	integer	(3) (+)	AtLeast	EGL_DONT_CARE or the number of bits of Red in the color buffer. **0** means no Red is in the color buffer.
EGL_GREEN_SIZE	integer	(3) (+)	AtLeast	EGL_DONT_CARE or the number of bits of Green in the color buffer. **0** means no Green is in the color buffer.
EGL_BLUE_SIZE	integer	(3) (+)	AtLeast	EGL_DONT_CARE or the number of bits of Blue in the color buffer. **0** means no Blue is in the color buffer.
EGL_LUMINANCE_SIZE	integer	(3) (+)	AtLeast	The number of bits of Luminance in the color buffer. **0** means no Luminance in the color buffer.
EGL_ALPHA_SIZE	integer	(3) (+)	AtLeast	The number of bits of Alpha in the color buffer. **0** means no Alpha in the color buffer.
EGL_ALPHA_MASK_SIZE	integer	(9) (-)	AtLeast	The number of bits of Alpha Mask in the color buffer. **0** means no Alpha Mask in the color buffer.
EGL_BIND_TO_TEXTURE_RGB	boolean	(o)	Exact	**EGL_DONT_CARE**, EGL_TRUE, or EGL_FALSE. EGL_TRUE if bindable to RGB textures.
EGL_BIND_TO_TEXTURE_RGBA	boolean	(o)	Exact	**EGL_DONT_CARE**, EGL_TRUE, or EGL_FALSE. EGL_TRUE if bindable to RGBA textures.
EGL_COLOR_BUFFER_TYPE	enum	(2) (o)	Exact	EGL_DONT_CARE, **EGL_RGB_BUFFER**, or EGL_LUMINANCE_BUFFER to represent the color buffer type.
EGL_CONFIG_CAVEAT	enum	(1) (+)	Exact	**EGL_DONT_CARE** or one of the following values indicating caveats for the configuration: EGL_NONE; EGL_SLOW_CONFIG - rendering to a surface may run at reduced performance; EGL_NON_CONFORMANT_CONFIG - rendering to a surface will not pass the required OpenGL ES conformance tests
EGL_CONFIG_ID	integer	(11) (-)	Exact	**EGL_DONT_CARE** indicating unique EGLConfig identifier.
EGL_CONFORMANT	bitmask	(o)	Mask	Indicates whether contexts created are conformant. May be **0** or one or more of the following values: EGL_OPENGL_BIT (OpenGL 1.x or 2.x), EGL_OPENGL_ES_BIT (OpenGL ES 1.x), EGL_OPENGL_ES2_BIT (OpenGL ES 2.x), EGL_OPENVG_BIT (OpenVG 1.x)
EGL_DEPTH_SIZE	integer	(7) (-)	AtLeast	A positive integer indicating the number of bits of Z in the depth buffer. Default is **0**.
EGL_LEVEL	integer	(o)	Exact	The frame buffer level. Default is **0**.
EGL_MATCH_NATIVE_PIXMAP	integer	(o)	Special	**EGL_NONE** or handle of a valid native pixmap
EGL_MAX_PBUFFER_WIDTH	integer	(o)		The maximum width of pbuffer. Default is **0**.
EGL_MAX_PBUFFER_HEIGHT	integer	(o)		The maximum height of pbuffer. Default is **0**.
EGL_MAX_PBUFFER_PIXELS	integer	(o)		The maximum size of pbuffer. Default is **0**.
EGL_MAX_SWAP_INTERVAL	integer	(o)	Exact	**EGL_DONT_CARE** or the maximum value that can be passed to **eglSwapInterval**; indicating the maximum number of swap intervals that will elapse before a buffer swap takes place after calling **eglSwapBuffers**.
EGL_MIN_SWAP_INTERVAL	integer	(o)	Exact	**EGL_DONT_CARE** or the minimum value that can be passed to **eglSwapInterval**; indicating the minimum number of swap intervals that will elapse before a buffer swap takes place after calling **eglSwapBuffers**.
EGL_NATIVE_RENDERABLE	boolean	(o)	Exact	**EGL_DONT_CARE**, EGL_TRUE, or EGL_FALSE. EGL_TRUE if native rendering APIs can render to surface
EGL_NATIVE_VISUAL_ID	integer	(o)		Default is **0**.
EGL_NATIVE_VISUAL_TYPE	integer	(10) (+)	Exact	**EGL_DONT_CARE**, EGL_NONE, or the native visual type of the associated visual.
EGL_RENDERABLE_TYPE	bitmask	(o)	Mask	Indicates which client APIs are supported. May be one or more of the following values: EGL_OPENGL_BIT (OpenGL 1.x or 2.x), **EGL_OPENGL_ES_BIT** (OpenGL ES 1.x), EGL_OPENGL_ES2_BIT (OpenGL ES 2.x), EGL_OPENVG_BIT (OpenVG 1.x)
EGL_SAMPLE_BUFFERS	integer	(5) (-)	AtLeast	The number of multisample buffers. Default is **0**.
EGL_SAMPLES	integer	(6) (-)	AtLeast	The number of samples per pixel. Default is **0**.
EGL_STENCIL_SIZE	integer	(8) (-)	AtLeast	The number of bits of Stencil in the stencil buffer. **0** means no Stencil in the stencil buffer.
EGL_SURFACE_TYPE	bitmask	(o)	Mask	The types of EGL surfaces are supported. May be one or more of **EGL_WINDOW_BIT**, EGL_PIXMAP_BIT, EGL_PBUFFER_BIT, MULTISAMPLE_RESOLVE_BOX_BIT, EGL_VG_ALPHA_FORMAT_PRE_BIT, EGL_SWAP_BEHAVIOR_PRESERVED_BIT, EGL_VG_COLORSPACE_LINEAR_BIT.
EGL_TRANSPARENT_TYPE	enum	(o)	Exact	**EGL_NONE** means windows created with the EGLConfig will not have any transparent pixels; EGL_TRANSPARENT_RGB means the EGLConfig supports transparency.
EGL_TRANSPARENT_RED_VALUE	integer	(o)	Exact	**EGL_DONT_CARE** or an integer in the range 0..2^EGL_RED_SIZE - 1 to indicate the transparent red value.
EGL_TRANSPARENT_GREEN_VALUE	integer	(o)	Exact	**EGL_DONT_CARE** or an integer in the range 0..2^EGL_GREEN_SIZE - 1 to indicate the transparent green value.
EGL_TRANSPARENT_BLUE_VALUE	integer	(o)	Exact	**EGL_DONT_CARE** or an integer in the range 0..2^EGL_BLUE_SIZE - 1 to indicate the transparent blue value.

Errors [3.1]

Obtain information about the success or failure of the most recent EGL function called in the current thread with **eglGetError**.

EGLint **eglGetError**(void)

Error codes returned by eglGetError:

EGL_SUCCESS **0x3000**
Function succeeded.

EGL_NOT_INITIALIZED **0x3001**
EGL is not or could not be initialized, for the specified display.

EGL_BAD_ACCESS **0x3002**
EGL cannot access a requested resource (for example, a context is bound in another thread).

EGL_BAD_ALLOC **0x3003**
EGL failed to allocate resources for the requested operation.

EGL_BAD_ATTRIBUTE **0x3004**
An unrecognized attribute or attribute value was passed in an attribute list.

EGL_BAD_CONFIG **0x3005**
An **EGLConfig** argument does not name a valid EGLConfig.

EGL_BAD_CONTEXT **0x3006**
An **EGLContext** argument does not name a valid EGLContext.

EGL_BAD_CURRENT_SURFACE **0x3007**
The current surface of the calling thread is a window, pbuffer, or pixmap that is no longer valid.

EGL_BAD_DISPLAY **0x3008**
An EGLDisplay argument does not name a valid EGLDisplay.

EGL_BAD_MATCH **0x3009**
Arguments are inconsistent; for example, an otherwise valid context requires buffers (e.g. depth or stencil) not allocated by an otherwise valid surface.

EGL_BAD_NATIVE_PIXMAP **0x300A**
An **EGLNativePixmapType** argument does not refer to a valid native pixmap.

EGL_BAD_NATIVE_WINDOW **0x300B**
An **EGLNativeWindowType** argument does not refer to a valid native window.

EGL_BAD_PARAMETER **0x300C**
One or more argument values are invalid.

EGL_BAD_SURFACE **0x300D**
An **EGLSurface** argument does not name a valid surface (window, pbuffer, or pixmap) configured for rendering.

EGL_CONTEXT_LOST **0x300E**
A power management event has occurred. The application must destroy all contexts and reinitialise client API state and objects to continue rendering.

Legend for Attribute Lists table:

Defaults
Default values **shown in red**

Sorting
(n) Sort priority
(+) Sort order of Special
(-) Sort order of Smaller
(o) Sort order of None

Selection
Attributes are matched in an attribute-specific manner, as shown in the "Selection" column above:

- **AtLeast**: Only EGLConfigs with an attribute value that meets or exceeds the specified value are selected.
- **Exact**: Only EGLConfigs whose attribute value equals the specified value are matched.
- **Mask**: Only EGLConfigs for which the bits set in the attribute value include all the bits that are set in the specified value are selected (additional bits might be set in the attribute value).
- **Special**: As described for the specific attribute.

EGL 1.4 API Quick Reference Card - Page 2

Configuration Management [3.4]

The configurations available depend on the implementation. Applications must ask for configurations supporting all the capabilities required.

EGLBoolean **eglGetConfigs**(EGLDisplay *dpy*, EGLConfig **configs*, EGLint *config_size*, EGLint **num_config*)
*dpy, *configs, config_size, *num_config:* same as **eglChooseConfig**

EGLBoolean **eglGetConfigAttrib**(EGLDisplay *dpy*, EGLConfig *config*, EGLint *attribute*, EGLint **value*)
attribute: any from table in "Attribute Lists" except EGL_MATCH_NATIVE_PIXMAP

EGLBoolean **eglChooseConfig**(EGLDisplay *dpy*, const EGLint **attrib_list*, EGLConfig **configs*, EGLint *config_size*, EGLint **num_config*)
dpy: a valid EGLDisplay
attrib_list: NULL, or a list of zero or more attributes terminated with EGL_NONE. Contains elements from table in "Attribute Lists" except for EGL_MAX_PBUFFER_WIDTH, EGL_MAX_PBUFFER_HEIGHT, or EGL_MAX_PBUFFER_PIXELS.
config_size: number of elements in **configs*
configs: NULL, or a pointer to an array in which up to *config_size* EGLConfig handles are returned.May be any from table in "Attribute Lists" except EGL_MATCH_NATIVE_PIXMAP
**num_config:* number of configurations is returned in *num_config*, and elements 0 through *num_config*–1 of *configs* are filled in with the valid EGLConfigs.

Rendering Surfaces [3.5]

Use these functions to create, destroy, and work with rendering surfaces, including on-screen native platform window, off-screen pbuffer, and those with color buffers stored in native pixmaps.

EGLSurface **eglCreateWindowSurface**(EGLDisplay *dpy*, EGLConfig *config*, EGLNativeWindowType *win*, const EGLint **attrib_list*)
attrib_list: NULL, or zero or more of the following, terminated with EGL_NONE: EGL_RENDER_BUFFER, EGL_VG_COLORSPACE, EGL_VG_ALPHA_FORMAT

EGLSurface **eglCreatePbufferSurface**(EGLDisplay *dpy*, EGLConfig *config*, const EGLint **attrib_list*)
attrib_list: NULL, or zero or more of the following, terminated with EGL_NONE:
EGL_WIDTH, EGL_HEIGHT, EGL_LARGEST_PBUFFER, EGL_TEXTURE_FORMAT, EGL_TEXTURE_TARGET, EGL_MIPMAP_TEXTURE, EGL_VG_COLORSPACE, EGL_VG_ALPHA_FORMAT

EGLSurface **eglCreatePbufferFromClientBuffer**(EGLDisplay *dpy*, EGLenum *buftype*, EGLClientBuffer *buffer*, EGLConfig *config*, const EGLint **attrib_list*)
buftype: EGL_OPENVG_IMAGE
attrib_list: NULL, or zero or more of the following, terminated with EGL_NONE: EGL_TEXTURE_FORMAT, EGL_TEXTURE_TARGET, EGL_MIPMAP_TEXTURE

EGLBoolean **eglDestroySurface**(EGLDisplay *dpy*, EGLSurface *surface*)

EGLSurface **eglCreatePixmapSurface**(EGLDisplay *dpy*, EGLConfig *config*, EGLNativePixmapType *pixmap*, const EGLint **attrib_list*)
attrib_list: NULL, or zero or more of the following, terminated with EGL_NONE: EGL_VG_COLORSPACE, EGL_VG_ALPHA_FORMAT

EGLBoolean **eglSurfaceAttrib**(EGLDisplay *dpy*, EGLSurface *surface*, EGLint *attribute*, EGLint *value*)
attribute: EGL_MIPMAP_LEVEL, EGL_MULTISAMPLE_RESOLVE, EGL_SWAP_BEHAVIOR
value: EGL_BUFFER_PRESERVED, EGL_BUFFER_DESTROYED, EGL_MULTISAMPLE_RESOLVE_DEFAULT, EGL_MULTISAMPLE_RESOLVE_BOX, or mipmap level

EGLBoolean **eglQuerySurface**(EGLDisplay *dpy*, EGLSurface *surface*, EGLint *attribute*, EGLint **value*)
attribute: EGL_VG_ALPHA_FORMAT, EGL_CONFIG_ID, EGL_HORIZONTAL_RESOLUTION, EGL_MIPMAP_TEXTURE, EGL_MULTISAMPLE_RESOLVE, EGL_RENDER_BUFFER, EGL_TEXTURE_FORMAT, EGL_VERTICAL_RESOLUTION, EGL_VG_COLORSPACE, EGL_HEIGHT, EGL_LARGEST_PBUFFER, EGL_MIPMAP_LEVEL, EGL_PIXEL_ASPECT_RATIO, EGL_SWAP_BEHAVIOR, EGL_TEXTURE_TARGET, EGL_WIDTH
value: Returned value of *attribute*

Rendering Contexts [3.7]

Both the OpenGL ES and OpenVG client APIs rely on an implicit context used by all entry points, rather than passing an explicit context parameter. The implicit context for each API is set with EGL calls.

EGLBoolean **eglBindAPI**(EGLenum *api*)
api: EGL_OPENGL_API, EGL_OPENGL_ES_API, or EGL_OPENVG_API

EGLenum **eglQueryAPI**(void)
Note: This function returns one of the valid api parameters to **eglBindAPI**, or EGL_NONE.

EGLContext **eglCreateContext**(EGLDisplay *dpy*, EGLConfig *config*, EGLContext *share_context*, const EGLint **attrib_list*)
share_context: A context, or EGL_NO_CONTEXT.
attrib_list: EGL_CONTEXT_CLIENT_VERSION

EGLBoolean **eglDestroyContext**(EGLDisplay *dpy*, EGLContext *ctx*)

EGLBoolean **eglMakeCurrent**(EGLDisplay *dpy*, EGLSurface *draw*, EGLSurface *read*, EGLContext *ctx*)
draw, read: a surface or EGL_NO_SURFACE. The same EGLSurface may be specified for both draw and read; and for an OpenVG context, the same EGLSurface must be specified for both draw and read.
ctx: A context, or EGL_NO_CONTEXT

EGLContext **eglGetCurrentContext**(void)

EGLSurface **eglGetCurrentSurface**(EGLint *readdraw*)
readdraw: EGL_READ, EGL_DRAW

EGLDisplay **eglGetCurrentDisplay**(void)

EGLBoolean **eglQueryContext**(EGLDisplay *dpy*, EGLContext *ctx*, EGLint *attribute*, EGLint **value*)
attribute: EGL_CONFIG_ID, EGL_CONTEXT_CLIENT_TYPE, EGL_CONTEXT_CLIENT_VERSION, EGL_RENDER_BUFFER

Extending EGL

See http://www.khronos.org/registry/egl/ for information about extending EGL, as well as specifications of Khronos- and vendor-approved EGL extensions.

Extension: EGL_KHR_IMAGE_BASE

This extension defines EGLImage, a new EGL resource type suitable for sharing 2D arrays of image data between client APIs.

EGLImageKHR **eglCreateImageKHR**(EGLDisplay *dpy*, EGLContext *ctx*, EGLenum *target*, EGLClientBuffer *buffer*, const EGLint **attrib_list*)
target: the type of resource being used as the EGLImage source
buffer: buffer
attrib_list: NULL, EGL_NONE, or EGL_IMAGE_PRESERVED_KHR. (If EGL_IMAGE_PRESERVED_KHR is EGL_TRUE, then all pixel data values associated with *buffer* are preserved.)
Note: In the event of an error, **eglCreateImageKHR** returns EGL_NO_IMAGE_KHR

EGLBoolean **eglDestroyImageKHR**(EGLDisplay *dpy*, EGLImageKHR *image*)

Extension: EGL_SYNC_FENCE_KHR

This extension adds the concept of "sync objects" into EGL, specifically a fence sync object, which is comparable to the OpenGL fence sync object. Fence sync objects have corresponding fence commands, which are inserted into a client API command stream at the time the fence sync is created.

EGLSyncKHR **eglCreateSyncKHR**(EGLDisplay *dpy*, EGLenum *type*, const EGLint **attrib_list*)
type: EGL_SYNC_FENCE_KHR
attrib_list: NULL, or a pointer to a list of zero or more of the following attributes terminated with EGL_NONE: EGL_SYNC_TYPE_KHR, EGL_SYNC_FENCE_KHR, EGL_SYNC_STATUS_KHR, EGL_UNSIGNALED_KHR, EGL_SYNC_CONDITION_KHR, EGL_SYNC_PRIOR_COMMANDS_COMPLETE_KHR
Note: In the event of an error, **eglCreateSyncKHR** returns EGL_TIMEOUT_EXPIRED_KHR, EGL_CONDITION_SATISFIED_KHR, EGL_NO_SYNC_KHR

EGLint **eglClientWaitSyncKHR**(EGLDisplay *dpy*, EGLSyncKHR *sync*, EGLint *flags*, EGLTimeKHR *timeout*)
flags: a bitmask with EGL_SYNC_FLUSH_COMMANDS_BIT_KHR
timeout: An integer (number of nanoseconds), or 0 (tests current status of sync), or EGL_FOREVER_KHR (does not time out)

EGLBoolean **eglDestroySyncKHR**(EGLDisplay *dpy*, EGLSyncKHR *sync*)

EGLBoolean **eglGetSyncAttribKHR**(EGLDisplay *dpy*, EGLSyncKHR *sync*, EGLint *attribute*, EGLint **value*);
attribute: EGL_SYNC_TYPE_KHR, EGL_SYNC_STATUS_KHR, EGL_SYNC_CONDITION_KHR
**value:* Returns EGL_SIGNALED_KHR, EGL_UNSIGNALED_KHR, EGL_SYNC_PRIOR_COMMANDS_COMPLETE_KHR

Initialization & Terminating [3.2, 3.3, 3.11]

EGLDisplay **eglGetDisplay**(EGLNativeDisplayType *display_id*)
display id: implementation-dependent, or EGL_DEFAULT_DISPLAY

EGLBoolean **eglInitialize**(EGLDisplay *dpy*, EGLint **major*, EGLint **minor*)
major, minor: If not NULL, returns the major and minor version numbers of the EGL implementation.

EGLBoolean **eglTerminate**(EGLDisplay *dpy*)
Note: Termination marks for deletion all EGL-specific resources associated with the specified display, such as contexts and surfaces. **eglMakeCurrent** [3.7] and **eglReleaseThread** [3.11] should be called from all threads to complete deletion.

const char ***eglQueryString**(EGLDisplay *dpy*, EGLint *name*)
name: EGL_CLIENT_APIS, EGL_EXTENSIONS, EGL_VENDOR, EGL_VERSION
Note: The EGL_CLIENT_APIS string returned contains a space-separated list of API names with at least one of 'OpenGL', OpenGL_ES, or OpenVG. The format of EGL_VERSION is: <major_version.minor_version><space><vendor_specific_info>

EGLBoolean **eglReleaseThread**(void)
Note: EGL and its client APIs must be threadsafe. Resources explicitly allocated by calls to EGL, such as contexts, surfaces, and configuration lists are not affected by **eglReleaseThread**.

Synchronization Primitives [3.8]

Use these functions to prevent native rendering API functions from executing until any outstanding client API rendering affecting the same surface is complete. eglWaitGL is available for backwards compatibility.

EGLBoolean **eglWaitClient**(void)

EGLBoolean **eglWaitGL**(void)

EGLBoolean **eglWaitNative**(EGLint *engine*)
engine: EGL_CORE_NATIVE_ENGINE or implementation-defined

Posting the Color Buffer [3.9]

Use these functions to make the contents of the color buffer visible in a native window (**eglSwapBuffers**), or to copy the contents to a native pixmap (**eglCopyBuffers**).

EGLBoolean **eglSwapBuffers**(EGLDisplay *dpy*, EGLSurface *surface*)

EGLBoolean **eglCopyBuffers**(EGLDisplay *dpy*, EGLSurface *surface*, EGLNativePixmapType *target*)

EGLBoolean **eglSwapInterval**(EGLDisplay *dpy*, EGLint *interval*)
interval: Number of video frames displayed before a swap occurs, in range [EGL_MIN_SWAP_INTERVAL..EGL_MAX_SWAP_INTERVAL]. 0 means swaps are not synchronized to video. Default is 1.

Render to Textures [3.6]

Use these functions to render to an OpenGL ES texture using a pbuffer surface configured for this operation. Render to texture is not supported for OpenGL contexts, and OpenGL ES implementations are not required to support it.

EGLBoolean **eglBindTexImage**(EGLDisplay *dpy*, EGLSurface *surface*, EGLint *buffer*)
buffer: EGL_BACK_BUFFER

EGLBoolean **eglReleaseTexImage**(EGLDisplay *dpy*, EGLSurface *surface*, EGLint *buffer*)

Obtain Extension Function Pointers [3.10]

Use this function to query at runtime the address of extension functions.

void (***eglGetProcAddress**(const char **procname*))(void)
**procnme:* Valid name of extension function, terminated by NULL.