Software Test Specification

EGL 1.4

Filename	:	Software Test Specification.EGL 1.4.doc
Version	:	1.0.156 External Issue
Issue Date	:	06 Apr 2010
Author	:	PCG

Contents

1. Introduction

This document specifies the tests used during SQA to validate the Khronos Native Platform Graphics Interface (EGL Version 1.4) implementation on Imagination Technologies Hardware IP Cores.

1.1. Related Documentation

Document	Description
Khronos Native Platform Graphics Interface Specification (Version 1.4)	The EGL 1.4 specification

1.2. Document Scope

This specification describes the test to be performed to verify the implementation of EGL 1.4.

To test EGL, the OpenGL ES 1.1, OpenGL ES 2.0, OpenVG 1.0.1, OpenVG1.1 and OpenGL2.0 rendering APIs will be used.

1.3. Assumptions

This document is written with the following assumptions:

The reader is familiar with EGL in general, EGL 1.4 in particular, OpenGL ES 1.1, OpenGL ES 2.0, OpenVG 1.0.1, OpenVG1.1 and OpenGL2.0.

2. Test Scope

2.1.1. Features to be tested

This section defines the areas of functionality and features that EGL has been split into for testing purposes.

Feature	Description	Risk
Basic Management	The management of the EGL state machine including displays, extension handling and the synchronisation of the rendering API with the native window system	
Configuration Management	Functions used to query and select API configurations	
Context Management	Functions used to create, query, select and delete rendering contexts	
Surface Management	Functions used to create, query, select and delete rendering surfaces	
Buffer Management	Functions used to post the colour buffer	
Texture Management	Functions used to render to an OpenGL ES texture using a pbuffer	
Error Management	Information about the success or failure of the most recent EGL function called in a specific thread	
Thread Management	The use of multiple threads and rendering APIs	
Mixed Configuration Management	Support for mixed rendering configurations allowing OpenVG and OpenGLES to render to the same surface	

3. Test Catalogue

3.1. Functional Tests

ID	Application	Description	Verif.	Source
EGL_FN01	egltestkit	Tests for EGL 1.3 (only EGL1.3APIS) All new extensions and APIs Introduced in EGL1.4 have covered in tests listed in EGL_FN02 – EGL_FN07.	A	IMG
EGL_FN02	MixedAPITest1	Mixed APIs (OpenVG & OpenGLES1)	M	IMG
EGL_FN03	MixedAPITest2	Mixed APIs (OpenVG & OpenGLES2)	M	IMG
EGL_FN04	egl1_4_ogl2	Test of extensions specific to EGL 1.4, using OpenGL 2 as the target client API	A	IMG
EGL_FN07	egl1_4_ovg	Test of extensions specific to EGL 1.4, using OpenVG as the target client API	A	IMG
EGL_FN10	Eglinfo	Test to output supported egl configs-unittest.	M	IMG
EGL_FN11	Xmultiegltest	Unit test for Multi threaded egl test	M	IMG
EGL_FN12	eglchecksymbols	Script testing the list of exported symbols in the EGL library to check if unnecessary symbols are exported.	A	IMG

4. Test Cases

4.1. EGL_FN01 – EGL Testkit

Test Groups

Test Group ID	Test Group Name
100	Basic Management
200	Configuration Management
300	Context Management
400	Surface Management
500	Buffer Management
600	Texture Management
700	Error Management
800	Thread Management
900	Extensions Management
1000	Additional tests

Test Cases

Test Case ID	Function/API Under Test	Description	Expected Result
100	eglQueryString	Query the function with the *name* `EGL_CLIENT_APIS`	A pointer to a static, zero terminated string describing the supported APIs, `OpenGL_ES` and `OpenVG` should, be returned
101	eglQueryString	Query the function with the *name* `EGL_EXTENSIONS`	A pointer to a static, zero terminated string describing the supported extensions should be returned
102	eglQueryString	Query the function with the *name* `EGL_VENDOR`	A pointer to a static, zero terminated string should be returned
103	eglQueryString	Query the function with the *name* `EGL_VERSION`	A pointer to a static, zero terminated string describing the version information in the format `<1.3><space><vendor_specific_info>` should be returned
110	eglGetDisplay	Pass an implementation-specific *display_id* to the function, such as an `X Display` under X windows or a Windows Device Context under Microsoft Windows	A display should be returned
111	eglGetDisplay	Pass `EGL_DEFAULT_DISPLAY` as a *display_id* to the function	A *default display* should be returned
112	eglInitialize	Once a display has been obtained, initialise the display	`EGL_TRUE` should be returned, and *major* and *minor* should be updated with the major and minor values of the EGL implementation

Test Case ID	Function/API Under Test	Description	Expected Result
113	eglTerminate	Pass an `EGLDisplay` *dpy* to the function	All EGL-specific resources associated with the specified display should be marked for deletion
120	eglWaitClient	Perform a number of API rendering calls for a currently bound context and then call the function Call a number of EGL functions which will affect the surface associated with the bound context	All API rendering calls for the currently bound context should be executed before the native rendering calls `EGL_TRUE` should be returned on success
121	eglWaitGL	Perform a number of API rendering calls for a currently bound context and then call the function Call a number of EGL functions which will affect the surface associated with the bound context	All API rendering calls for the currently bound context should be executed before the native rendering calls `EGL_TRUE` should be returned on success This should be equivalent to `EGLenum api` = **eglQueryAPI**(); **eglBindAPI**(`EGL_OPENGL_ES_API`); **eglWaitClient**(); **eglBindAPI**(`api`);
122	eglWaitNative	Perform a number of API and native rendering calls for a currently bound context and surface, for a specified marking engine *engine*	Native rendering calls made with *engine* and which affect the surface of the current context should be executed before client API rendering calls
130	eglGetProcAddress	Call the function with a `NULL`-terminated string *procName* specifying an extension function Cast the pointer returned to a function pointer type that matches the extension function's definition	The address of the extension function, if supported, should be returned
140	eglReleaseThread	Create a number of contexts and surfaces and then call the function	Any currently bound contexts in the current thread should be released The rendering API should be reset to its value at thread initialisation Any implementation-dependent per-thread state maintained by EGL should be marked for deletion
200	eglGetConfigs	Call the function for each available display after the display has been initialised	All `EGLConfig`s available on the specified display should be returned in the buffer containing *num_config config_size* elements pointed to by *configs*

Test Case ID	Function/API Under Test	Description	Expected Result
210	eglChooseConfig	Call the function for each available display after the display has been initialised Specify a list of attributes	All `EGLConfig`s available on the specified display that match the attribute names and values should be returned in the buffer containing *num_config config_size* elements pointed to by *configs*
220	eglGetConfigAttrib	Call the function with a range of valid attributes	The valued of the specified attribute should be returned in *value*, and the function should return `EGL_TRUE` If *attribute* is not a valid attribute, `EGL_FALSE` should be returned
300	eglBindAPI	Pass the *api* `EGL_OPENVG_API` to the function	The supported client API `OPENVG` should be set as the current rendering API and `EGL_TRUE` should be returned
301	eglBindAPI	Pass the *api* `EGL_OPENGL_ES_API` to the function	The supported client API `OPENGL_ES` should be set as the current rendering API and `EGL_TRUE` should be returned
302	eglQueryAPI	Query the function before and after calling **eglBindAPI**	The initial value of the current rendering API should be set to `EGL_OPENGL_ES_API` if it is supported, otherwise it should be set to `EGL_NONE` If **eglBindAPI** has been called, the current rendering API should be returned
310	eglCreateContext	Call the function with a range of `EGLConfig`s *config* and `EGLContext`s *share_context*	The context should be initialised to the initial state for the current rendering API and a handle to it should be returned If *share_context* is not set to EGL_NO_CONTEXT, all shared data should by shared by *share_context*, all other contexts *share_context* already shares with, and the newly created context
320	eglDestroyContext	Pass a created context *ctx* to the function	All resources associated with the `EGLContext` *ctx* should be marked for deletion
330	eglMakeCurrent	Pass a range of contexts *ctx* and surfaces *draw* and *read* to the function	The `EGLContext` *ctx* should be bound to the current rendering thread and to the *draw* and *read* surfaces For an OpenGL ES context, *draw* should be used for OpenGL ES operations other than any pixel data read back, which will should be taken from the frame buffer values of *read* For an OpenVG context, *read* and *draw* must be the same `EGLSurface`

Test Case ID	Function/API Under Test	Description	Expected Result
331	eglGetCurrentContext	Create a context and make it current Query the function	The current context for the current rendering API should be returned
332	eglGetCurrentContext	Create a context and make it current with the rendering API `EGL_NONE` Query the function	`EGL_NO_CONTEXT` should be returned
333	eglGetCurrentContext	Query the function without creating a context	`EGL_NO_CONTEXT` should be returned
340	eglGetCurrentDisplay	Query the function	The `EGLDisplay` for the current context in the calling thread should be returned If there is no current context for the current rendering API, `EGL_NO_DISPLAY` should be returned
350	eglQueryContext	Call the function with an `EGLContext` and the *attribute* `EGL_CONFIG_ID`	The ID of the `EGLConfig` for the `EGLContext` *ctx* should be returned in *value*
351	eglQueryContext	Call the function with an `EGLContext` and the *attribute* `EGL_CONTEXT_CLIENT_TYPE`	The type of client API the `EGLContext` *ctx* supports should be returned in *value*
352	eglQueryContext	Call the function with an `EGLContext` and the *attribute* `EGL_CONTEXT_CLIENT_VERSION`	The version of the client API the `EGLContext` *ctx* supports should be returned in *value*
353	eglQueryContext	Call the function with an `EGLContext` and the *attribute* `EGL_RENDER_BUFFER`	The buffer which the client API rendering via the `EGLContext` *ctx* will used should be returned in *value*
400	eglCreateWindowSurface	Call the function with a range of `EGLConfig`s *config*, a platform-specific native window `NativeWindowType` and a range of *attribute*s	An onscreen `EGLSurface` should be created and a handle to it should be returned
410	eglCreatePbufferSurface	Call the function with a range of `EGLConfig`s *config* and a list of attributes that can include `EGL_WIDTH, EGL_HEIGHT, EGL_LARGEST_PBUFFER, EGL_TEXTURE_FORMAT, EGL_TEXTURE_TARGET, EGL_MIPMAP_TEXTURE, EGL_COLOR_SPACE, EGL_ALPHA_FORMAT`	A single pbuffer surface should be created and a handle to it should be returned
420	eglCreatePbufferFromClientBuffer	Call the function with a range of `EGLConfig`s *config* with *buftype* `EGL_OPENVG_IMAGE`, a valid VGImage *buffer*, and a list of attributes that can include `EGL_TEXTURE_FORMAT, EGL_TEXTURE_TARGET, EGL_MIPMAP_TEXTURE`	A single pbuffer surface should be created bound to the specified *buffer* for all of its buffer storage A handle to it should be returned

Test Case ID	Function/API Under Test	Description	Expected Result
430	eglCreatePixmapSurface	Call the function with a range of EGLConfigs *config*,a platform-specific native pixmap NativePixmapType and a range of *attribute*s	An offscreen EGLSurface should be created and a handle to it should be returned
440	eglDestroySurface	Pass a created surface to the function	All resources associated with the EGLSurface *surface* should be marked for deletion
450	eglSurfaceAttrib	Call the function with an EGLSurfaces *surface* and a range of *attribute*s and *value*s Currently *attribute* can only be EGL_MIPMAP_LEVEL	The specified *attribute* of *surface* should be set to *value* If the value of pbuffer attribute EGL_TEXTURE_FORMAT is EGL_NO_TEXTURE, the value of attribute EGL_TEXTURE_TARGET is EGL_NO_TEXTURE, or if *surface* is not a pbuffer, then EGL_MIPMAP_LEVEL will have no effect
460	eglQuerySurface	Query the function with a range of attributes	The value of *attribute* for *surface* should be returned in *value*
470	eglGetCurrentSurface	Query the function with *readdraw* EGL_READ	The read surface bound to the current context in the calling thread for the current rendering API should be returned
471	eglGetCurrentSurface	Query the function with *readdraw* EGL_DRAW	The draw surface bound to the current context in the calling thread for the current rendering API should be returned
500	eglSwapBuffers	Pass a back-buffered window surface to the function	The colour buffer should be copied to the native window associated with the surface
501	eglSwapBuffers	Pass single-buffered window, pixmap and pbuffer surfaces to the function	**eglSwapBuffers** should have no affect
510	eglCopyBuffers	Pass a surface and native pixmap handle to the function	The colour buffer for the surface should be copied to the NativePixmapType *target*
520	eglSwapInterval	Pass a range of *interval* values to the function	The minimum number of video frames per buffer swap for the window associated with the current context should be set to *interval*
600	eglBindTexImage	Create a pbuffer surface that supports EGL_BIND_TO_TEXTURE_RGB or EGL_BIND_TO_TEXTURE_RGBA Pass the *buffer* EGL_BACK_BUFFER to the function	A two-dimensional texture consisting of the image data in *buffer* with the attributes of the specified *surface* should be defined
610	eglReleaseTexImage	Call the function after calling **eglBindTexImage** with valid parameters	The specified colour buffer should be released back to the surface

Test Case ID	Function/API Under Test	Description	Expected Result
700	eglGetError	Call a range of functions with valid parameters Check the current error code	EGL_SUCCESS should be returned
701	eglGetError	Call a range of functions with a display that has not yet been initialised Check the current error code	EGL_NOT_INITIALIZED should be returned
702	eglGetError	Bind a context in a thread then attempt to access the context in another thread Check the current error code	EGL_BAD_ACCESS should be returned
703	eglGetError	Continue calling EGL functions until all of the available resources have been used Check the current error code	EGL_BAD_ALLOC should be returned
704	eglGetError	Pass an unrecognised attribute or attribute value to all functions that take an attribute list as a parameter Check the current error code	EGL_BAD_ATTRIBUTE should be returned
705	eglGetError	Pass an invalid context to all functions that take an EGLContext as a parameter Check the current error code	EGL_BAD_CONTEXT should be returned
706	eglGetError	Pass an invalid config to all functions that take an EGLConfig as a parameter Check the current error code	EGL_BAD_CONFIG should be returned
707	eglGetError	Create a window, pbuffer or pixmap surface and set it as the current surface. Destroy the surface and then attempt to use the surface Check the current error code	EGL_BAD_CURRENT_SURFACE should be returned
708	eglGetError	Pass an invalid display, or a display that has not been initialised, to all functions that take an EGLDisplay as a parameter Check the current error code	EGL_BAD_DISPLAY should be returned
709	eglGetError	Pass an invalid surface to all functions that take an EGLSurface as a parameter Check the current error code	EGL_BAD_SURFACE should be returned

Test Case ID	Function/API Under Test	Description	Expected Result
710	eglGetError	Pass inconsistent arguments to a function, such as a valid context that requires buffers that are not allocated by a valid surface Check the current error code	`EGL_BAD_MATCH` should be returned
711	eglGetError	Pass one or more invalid arguments to a function Check the current error code	`EGL_BAD_PARAMETER` should be returned
712	eglGetError	Pass an invalid native pixmap to all functions that take a `NativePixmapType` as a parameter Check the current error code	`EGL_BAD_NATIVE_PIXMAP` should be returned
713	eglGetError	Pass an invalid native window to all functions that take a `NativeWindowType` as a parameter Check the current error code	`EGL_BAD_NATIVE_WINDOW` should be returned
714	eglGetError	Setup some EGL contexts and then force a power management event to occur Check the current error code	`EGL_CONTEXT_LOST` should be returned
800	OpenGL ES 1.1	Exercise a range of EGL functions in multiple threads using OpenGL ES 1.1 as the rendering API	Each EGL function call in any thread should behave as specified in previous test cases
801	OpenGL ES 2.0	Exercise a range of EGL functions in multiple threads using OpenGL ES 2.0 as the rendering API	Each EGL function call in any thread should behave as specified in previous test cases
802	OpenVG 1.0	Exercise a range of EGL functions in multiple threads using OpenVG 1.0 as the rendering API	Each EGL function call in any thread should behave as specified in previous test cases
803	OpenGL ES 1.1 OpenGL ES 2.0 OpenVG 1.0	Exercise a range of EGL functions in multiple threads using all the supported rendering APIs	Each EGL function call in any thread should behave as specified in previous test cases
900	EGL_IMG_context_priority (extension)	Check the enums used by this extension by calling eglCreateContext() and eglQueryContext().	Each call using the enums might work correctly (no EGL_BAD_ATTRIBUTE)
1000	eglCreatePbufferSurface()	Check if a pbuffer can be created with a size of 0	Should work fine, even the call of the eglMakeCurrent() with the zero sized surface.

4.2. EGL_FN02 – MixedAPITest1

Test Groups

Test Group ID	Test Group Name
100	Shared rendering tests
200	EGLImage positive tests
300	EGLImage negative tests

Test Cases

Test Case ID	Function/API Under Test	Description	Expected Result
100	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Instruct OpenVG to clear the screen red. Then instruct OpenGL ES to clear the screen green. Then instruct OpenVG to draw a star shape. Then display the image.	Would expect to see a star overlayed on a green background.
101	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Use OpenGL ES to clear the screen green, then render a star with OpenVG.	Would expect to see a green background with a star in the centre of the screen.
102	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Use OpenVG to clear the screen blue, then render some primitives with OpenGL ES.	Would expect to see the OpenGL ES primitives displayed on a blue background.
103	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Use OpenGL ES to clear the screen green then render some OpenVG primitives, and then render some more primitives with OpenGL ES.	Would expect to see the OpenGL ES primitives overlayed on the OpenVG primitives, all displayed on a green background.
104	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Use OpenVG to clear the screen blue then render some OpenGL ES primitives and then render some more primitives with OpenVG.	Would expect to see the OpenVG primitives overlayed on the OpenGL ES primitives, all displayed on a blue background.
105	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Use OpenGL ES to render an image, then read the pixels back with OpenVG and check that they match.	Would expect an exact match.
106	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Use OpenVG to render an image, then read the pixels back with OpenGL ES and check that they match.	Would expect an exact match.
107	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Use OpenVG to clear the screen green, then render a blue semi transparent quad using OpenGL ES with blending.	Would expect to see a yellow quad on a green background.
108	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Setup scissoring region in OpenGL ES, then check the effect on OpenVG renderings.	Scissoring effects should be preserved across APIs.

Test Case ID	Function/API Under Test	Description	Expected Result
109	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Setup scissoring region in OpenVG, then check the effect on OpenGL ES renderings.	Scissoring effects should be preserved across APIs.
200	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Construct an EGLImage from an OpenGL ES texture. Then construct an OpenVG VGImage from the EGLImage, and display using OpenVG. Construct a new EGLImage from OpenVG (using pixels from the image just displayed). Then construct a GL texture from the EGLImage, and display using OpenGL ES.	The image data should be successfully passed between APIs and displayed without corruption.
300	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with an invalid value for the <dpy> parameter in order to generate an EGL_BAD_DISPLAY error.	Error caught and EGL_BAD_DISPLAY set.
301	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with a value of EGL_NO_DISPLAY for the <dpy> parameter in order to generate an EGL_BAD_DISPLAY error.	Error caught and EGL_BAD_DISPLALY set.
302	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with an invalid value for the <ctx> parameter in order to generate an EGL_BAD_CONTEXT error.	Error caught and EGL_BAD_CONTEXT set.
303	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with a value of EGL_NO_CONTEXT for the <ctx> parameter in order to generate an EGL_BAD_CONTEXT error.	Error caught and EGL_BAD_CONTEXT set.
304	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with an invalid value for the <target> parameter in order to generate an EGL_BAD_PARAMETER error.	Error caught and EGL_BAD_PARAMETER set.

305	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with the <target> parameter set to EGL_NATIVE_PIXMAP_KHR and an invalid value for the <buffer> parameter in order to generate the EGL_BAD_PARAMETER error.	Error caught and EGL_BAD_PARAMETER set.
306	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with the <target> parameter set to EGL_NATIVE_PIXMAP_KHR and a value for the <buffer> parameter for which there is no matching EGLConfig supporting native pixmaps of the correct colourr buffer format. Check that the error EGL_BAD_PARAMETER is generated.	Error caught and EGL_BAD_PARAMETER set.
307	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with an invalid value in the <attr_list> parameter list. Check that the error EGL_BAD_PARAMETER is generated.	Error caught and EGL_BAD_PARAMETER set.
308	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with a set of parameters identifying a resource with an off-screen buffer bound to it (by a call to eglBindTexImage). Check that the EGL_BAD_ACCESS error is generated.	Error caught and EGL_BAD_ACCESS set.
309	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with a set of parameters identifying a resource with an off-screen buffer bound to it (by a call to eglCreatePbufferFromClientBuffer). Check that the EGL_BAD_ACCESS error is generated.	Error caught and EGL_BAD_ACCESS set.
310	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with a set of parameters identifying a resource which is itself an EGLImage sibling. Check that the EGL_BAD_ACCESS error is generated.	Error caught and EGL_BAD_ACCESS set.

311	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with a <target> of EGL_NATIVE_PIXMAP_KHR and a context which is EGL_NO_CONTEXT. Check that the error EGL_BAD_PARAMETER is generated.	Error caught and EGL_BAD_PARAMETER set.
312	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR with a <target> of EGL_NATIVE_PIXMAP_KHR and a <buffer> value which is not a handle to a NativePixmapType object. Check that the EGL_BAD_PARAMETER is generated.	Error caught EGL_BAD_PARAMETER set.
313	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreateImageKHR when there is insufficient memory available to complete the operation. Check that the error EGL_BAD_ALLOC is generated.	Error caught and EGL_BAD_ALLOC set.
320	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglDestroyImageKHR with an invalid value for the <dpy> parameter. Check that the error EGL_BAD_DISPLAY is generated.	Error caught and EGL_BAD_DISPLAY generated.
321	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglDestroyImageKHR with a value of EGL_NO_CONTEXT for the <dpy> parameter. Check that the error EGL_BAD_DISPLAY is generated.	Error caught and EGL_BAD_DISPLAY generated.
322	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglDestroyImageKHR with a <dpy> parameter different to the one used to create the image. Check that the error EGL_BAD_MATCH is generated.	Error caught and EGL_BAD_MATCH generated.
323	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglDestroyImageKHR with an invalid value for the <image> parameter. Check that the error EGL_BAD_PARAMETER is generated.	Error caught and EGL_BAD_PARAMETER generated.
324	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function eglCreatePbufferFromClientBuffer with an EGLImage sibling as a source buffer. Check that the error EGL_BAD_ACCESS is generated.	Error caught and EGL_BAD_ACCESS generated.

330	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_GL_TEXTURE_2D_KHR for the <target> parameter, and an invalid value for the <dpy> parameter. Check that the error EGL_BAD_DISPLAY is generated.	Error caught and EGL_BAD_ACCESS generated.
331	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_GL_TEXTURE_2D_KHR for the <target> parameter, and an invalid value for the <ctx> parameter. Check that the error EGL_BAD_CONTEXT is generated.	Error caught and EGL_BAD_CONTEXT generated.
332	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_GL_TEXTURE_2D_KHR for the <target> parameter, and a non-matching value for the <ctx> parameter (i.e. an OpenVG context). Check that the error EGL_BAD_MATCH is generated.	Error caught and EGL_BAD_MATCH generated.
333	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_GL_TEXTURE_2D_KHR for the <target> parameter, and an incomplete texture for the <buffer> parameter. Check that the error EGL_BAD_PARAMETER is generated.	Error caught and EGL_BAD_PARAMETER generated.
334	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_GL_TEXTURE_2D_KHR for the <target> parameter, and a value of 0 for the <buffer parameter> (i.e. the default object). Check that the error EGL_BAD_PARAMETER is generated.	Error caught and EGL_BAD_PARAMETER generated.
335	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_GL_TEXTURE_2D_KHR for the <target> parameter, and have an invalid mipmap level defined in the <attr_list>. Check that the error EGL_BAD_MATCH is generated.	Error caught and EGL_BAD_MATCH generated.

340	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_VG_PARENT_IMAGE_KHR for the <target> parameter, and an invalid value for the <dpy> parameter. Check that the error EGL_BAD_DISPLAY is generated.	Error caught and EGL_BAD_DISPLAY generated.
341	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_VG_PARENT_IMAGE_KHR for the <target> parameter, and an invalid value for the <ctx> parameter. Check that the error EGL_BAD_CONTEX is generated.	Error caught and EGL_BAD_CONTEXT generated.
342	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_VG_PARENT_IMAGE_KHR for the <target> parameter, and an non-matching context for <ctx> parameter (i.e. an OpenGL ES context). Check that the error EGL_BAD_MATCH is generated.	Error caught and EGL_BAD_MATCH generated.
343	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_VG_PARENT_IMAGE_KHR for the <target> parameter, and a buffer object from a different OpenVG context to the one passed to the current function call. Check that the error EGL_BAD_PARAMETER is generated.	Error caught and EGL_BAD_PARAMETER generated.
344	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call eglCreateImageKHR with a value of EGL_VG_PARENT_IMAGE_KHR for the <target> parameter and a buffer object which is a child image (i.e. created by vgChildImage). Check that the error EGL_BAD_ACCESS is generated.	Error caught and EGL_BAD_ACCESS generated.
350	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function glEGLImageTargetTexture2DOES with a value for the <image> parameter which cannot be used to create GL texture object (i.e. a multisampled EGLImage). Check that the error GL_INVALID_OPERATION is generated.	Error caught and GL_INVALID_OPERATION generated.

351	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function glEGLImageTargetTexture2DOES with an invalid enum for the <target> parameter. Check that the error GL_INVALID_ENUM is generated.	Error caught and GL_INVALID_ENUM generated.

Test Case ID	Function/API Under Test	Description	Expected Result
361	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function glEGLImageTargetTexture2DOES with an invalid value for the <image> parameter. Check that the error GL_INVALID_VALUE is generated.	Error caught and GL_INVALID_VALUE generated.

Test Case ID	Function/API Under Test	Description	Expected Result
371	EGL, OpenGL ES 1.1, OpenVG 1.0.1	Call function vgCreateEGLImageTargetKHR with an invalid value for the <image> parameter. Check that the error VG_ILLEGAL_ARGUMENT_ERROR is generated.	Error caught and VG_ILLEGAL_ARGUMENT_ERROR generated.

4.3. EGL_FN03 – MixedAPITest2

Test cases are identical to those described fro EGL_FN01 – MixedAPITest1, only using OpenGL ES 2 as the 3D rendering API rather than OpenGL ES 1.

4.4. EGL_FN<04-07> - EGL1_4_<OGL2 | OVG>

This set of applications each test the extensions added to the EGL specification in revision 1.4. Since it is impossible to test the functionality fully without the use of a client API, four versions of the test application have been created, one for each of OpenGL 2.0, OpenVG 1.0 and OpenVG 1.1.

The functionality under test is detailed in the following tables:

Test Group ID	Test Group Name
100	EGL 1.4 Extension Tests

Test Case ID	Function/API Under Test	Description	Expected Result
100	EGL 1.4 (OVG only)	Test that EGL multisampling is allowable for OpenVG	Multisampling should be supported
101	EGL 1.4 (all APIs)	Test the multisample resolution behaviour using the EGL_MULTISAMPLE_RESOLVE EGLSurface attribute.	Feature is optional according to the specification. Test should compile successfully and run without causing a crash.
102	EGL 1.4 (all APIs)	Test control of swap behaviour using the EGL_SWAP_BEHAVIOR bit in the EGL_SURFACE_TYPE EGLSurface attribute.	Feature is optional according to the specification. Test should compile successfully and run without causing a crash.
103	EGL 1.4 (all APIs)	Test the relaxed definition of the EGLNativeDisplayType which can now allow mappings to X & Win32 data structures (in eglGetDisplay)	Feature appears to be optional according to the specification, but the test should run without generating errors.

4.5. EGL_FN12 – eglchecksymbols

This is a shell script (unix platform only) that is checking the list of exported symbols visible from the API library file. If unnecessary symbols are visible the test will fail.

Appendix A. Functional Coverage

Feature	Function/API
Basic Management	eglGetDisplay eglInitialize eglTerminate
	eglQueryString
	eglWaitClient eglWaitGL eglWaitNative
	eglGetProcAddress eglReleaseThread
Configuration Management	eglGetConfigs eglChooseConfig eglGetConfigAttrib
Context Management	eglBindAPI eglQueryAPI eglCreateContext eglDestroyContext eglMakeCurrent eglGetCurrentContext eglGetCurrentDisplay eglQueryContext
Surface Management	eglCreateWindowSurface eglCreatePbufferSurface eglCreatePbufferFromClientBuffer eglCreatePixmapSurface eglDestroySurface eglSurfaceAttrib eglQuerySurface eglGetCurrentSurface
Buffer Management	eglSwapBuffers eglCopyBuffers eglSwapInterval
Texture Management	eglBindTexImage eglReleaseTexImage
Error Management	eglGetError
Thread Management	OpenGL ES 1.1 OpenGL ES 2.0 OpenVG 1.0 OpenVG 1.1 OpenGL 2.0