OpenWF Composition 1.0 API Quick Reference Card

OpenWF COMPOSITION

OpenWF Composition® is a standardized API for compositing and serves as a low-level interface for two-dimensional composition used in embedded and/or mobile devices. Target users are windowing systems, system integrators etc. The API is implementable on top of a wide range of hardware.
The header file to include is <WF/wfc.h>

- [n.n.n] refers to the section in the API Specification available at www.khronos.org/openwf/.
- Blue are datatypes defined in the WFC spec.
- (r/w) – read/writable (r) – read only
- Brown are constant values defined in the WFC spec.
- *Italic* are parameter names in function declarations

Errors [2.11] – of type WFCErrorCode

Errors codes and their numerical values are defined by the WFCErrorCode enumeration could be retrived by the following function:

WFCErrorCode **wfcGetError**(WFCDevice *dev*).

The possible values are as follows:

WFC_ERROR_NONE	WFC_ERROR_OUT_OF_MEMORY
WFC_ERROR_ILLEGAL_ARGUMENT	WFC_ERROR_UNSUPPORTED
WFC_ERROR_BAD_ATTRIBUT E	WFC_ERROR_IN_USE
WFC_ERROR_BUSY	WFC_ERROR_BAD_DEVICE
WFC_ERROR_BAD_HANDLE	WFC_ERROR_INCONSISTENCY

Functions that returns handles could return the following error:

WFC_INVALID_HANDLE [2.6]

Device - A *WFCDevice*[3] is an abstract device that is capable of performing composition operations, typically a unit of graphics hardware. Devices can vary in their support for specific input and output formats.

Device Attributes [4.1] of type WFCDeviceAttrib

WFC_DEVICE_CLASS	**(r) - supports on-screen or not.**
WFC_DEVICE_ID	**(r) – the ID of the device – could be WFC_DEFAULT_DEVICE_ID**

Device Class [4.1.1] of type WFCDeviceClass

WFC_DEVICE_CLASS_FULLY_CAPABLE	**Support both on- and off-screen rendering**
WFC_DEVICE_CLASS_OFF_SCREEN_ONLY	**No on-screen compositing**

WFCint **wfcEnumerateDevices**(WFCint **deviceIds*,
WFCint *deviceIdsCount*, const WFCint **filterList*)

Populate a list of available devices with respect to the filter-list (could be WFC_NONE).

WFCDevice **wfcCreateDevice**(WFCint *deviceId*,
const WFCint **attribList*)

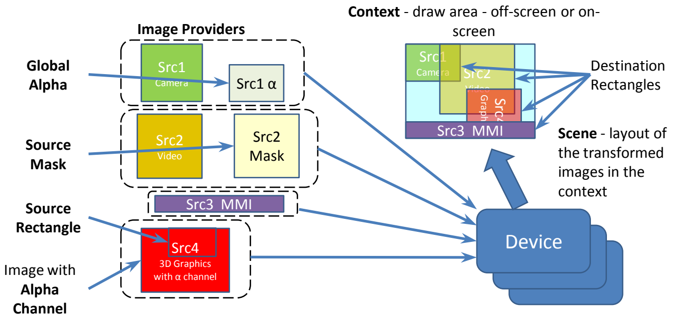
Create a device with a known ID - could use WFC_DEFAULT_DEVICE_ID.

WFCint **wfcGetDeviceAttribi**(WFCDevice *dev*,
WFCDeviceAttrib *attrib*)

Retrieve capabilities for a specific device.

WFCErrorCode **wfcDestroyDevice**(WFCDevice *dev*)

Delete a specific device.



Context - A WFCContext[5] stands for a visual scene description applied to either an on-screen or off-screen target. It represents the state required for a device to be used for composition of a scene. A scene consists of a stack of **Elements**, added on top of **WFC_CONTEXT_LOWEST_ELEMENT**. (See Element Ordering.). A Context is permanently bound to a target.

Context Attributes [5.1] of type WFCContextAttrib

WFC_CONTEXT_TYPE	**(r) On-screen or off-screen**
WFC_CONTEXT_TARGET_HEIGHT	**(r) Size of the destination in pixels**
WFC_CONTEXT_TARGET_WIDTH	**(r) Size of the destination in pixels**
WFC_CONTEXT_LOWEST_ELEMENT	**(r) Reference to bottom element**
WFC_CONTEXT_ROTATION	**(r/w) Rotation from src to dest**
WFC_CONTEXT_BG_COLOR	**(r/w) RGBA vector – 0 ≤ value ≤ 1**

Context type [5.1.1] of type WFCContextType

WFC_CONTEXT_TYPE_ON_SCREEN
WFC_CONTEXT_TYPE_OFF_SCREEN

Rotation [5.1.4] – also used for element rotation

WFC_ROTATION_0	**No rotation**
WFC_ROTATION_90	**Rotate 90 degrees clockwise**
WFC_ROTATION_180	**Rotate 180 degrees clockwise**
WFC_ROTATION_270	**Rotate 270 degrees clockwise**

Context Creation and Destruction [5.1], [5.3] and [5.7]

WFCContext **wfcCreateOnScreenContext**(WFCDevice *dev*,
WFCint *screenNumber*, const WFCint **attribList*)

WFCContext **wfcCreateOffScreenContext**(WFCDevice *dev*,
WFCNativeStreamType *stream*, const **WFCint** **attribList*)

The offscreen context requires a stream to render into.

void **wfcDestroyContext**(WFCDevice *dev*, WFCContext *ctx*)

Commit Context Attribute Changes [5.4]

void **wfcCommit**(WFCDevice *dev*, WFCContext *ctx*, WFCboolean *wait*)

NOTE -Changes in attributes will take effect when calling wfcCommit.

Query Context Attributes [5.5] – single value / vector of values

WFCint **wfcGetContextAttribi**(WFCDevice *dev*, WFCContext *ctx*,
WFCContextAttrib *attrib*)

void **wfcGetContextAttribfv**(WFCDevice *dev*, WFCContext *ctx*,
WFCContextAttrib *attrib*, WFCint *count*, WFCfloat **values*)

Set Context Attributes [5.6] – single value / vector of values

void **wfcSetContextAttribi**(WFCDevice *dev*, WFCContext *ctx*,
WFCContextAttrib *attrib*, WFCint *value*)

void **wfcSetContextAttribfv**(WFCDevice *dev*, WFCContext *ctx*,
WFCContextAttrib *attrib*, WFCint *count*, const WFCfloat **values*)

OpenWF Composition 1.0 API Quick Reference Card

Image Providers - input to composition.

No valid attributes defined in the spec.

Source inputs [6.1] - **WFCSource** image data – could contain alpha

```
WFCSource wfcCreateSourceFromStream(WFCDevice dev,
        WFCContext ctx, WFCNativeStreamType stream,
        const WFCint *attribList)

void wfcDestroySource(WFCDevice dev, WFCSource src)
```

Mask inputs [6.2] - **WFCMask** per-pixel opacity data

```
WFCMask wfcCreateMaskFromStream( WFCDevice dev,
        WFCContext ctx, WFCNativeStreamType stream,
        const WFCint *attribList)

void wfcDestroyMask(WFCDevice dev, WFCMask mask)
```

Synchronization [9] – compositing and other EGL client APIs could be synchronized using EGLSyncObjects

```
void  wfcFence(WFCDevice dev, WFCContext ctx,
          WFCEGLDisplay dpy, WFCEGLSync sync)
```

Composition Elements [7] – of type WFCElement

A scene consists of zero or more Elements stacked over a background plane. Composition is equivalent to blending each Element on top of the destination buffer according to the relative ordering of the Elements with respect to alpha or mask (WFCTransperencyType) . The result of composition is a 2D image. The source data, that is content of source rectangle, is transformed to match destination rectangle with respect to color format and size (using **WFC_ELEMENT_SOURCE_SCALE_FILTER**).

WFCElementAttrib [7.1]

WFC_ELEMENT_DESTINATION_ RECTANGLE	(r/w) Placement of transformed image in context coordinates
WFC_ELEMENT_SOURCE	(r/w) Handle to image provider
WFC_ELEMENT_SOURCE_RECTANGLE	(r/w) Sub area in source coordinates
WFC_ELEMENT_SOURCE_FLIP	(r/w) Flipping the source or not
WFC_ELEMENT_SOURCE_ROTATION	(r/w) Rotation in 90 degrees angles
WFC_ELEMENT_SOURCE_SCALE_FILTER	(r/w) Quality of scaling
WFC_ELEMENT_TRANSPARENCY_TYPES	(r/w) Blending type for this element
WFC_ELEMENT_GLOBAL_ALPHA	(r/w) Apply global alpha
WFC_ELEMENT_MASK	(r/w) Handle to mask source

WFCScaleFilter [7.1.6]

WFC_SCALE_FILTER_NONE	Nearest-neighbor replication (required)
WFC_SCALE_FILTER_FASTER	Low resource requirements (optional)
WFC_SCALE_FILTER_BETTER	High quality filtering (optional)

WFCTransparencyType [7.1.7] - bitfield

WFC_TRANSPARENCY_NONE	0 (default)
WFC_TRANSPARENCY_ELEMENT_GLOBAL_ALPHA	(1 << 0)
WFC_TRANSPARENCY_SOURCE	(1 << 1)
WFC_TRANSPARENCY_MASK	(1 << 2)

Only the following combinations of transparency are possible:

- WFC_TRANSPARENCY_ELEMENT_GLOBAL_ALPHA | WFC_TRANSPARENCY_SOURCE
- WFC_TRANSPARENCY_ELEMENT_GLOBAL_ALPHA | WFC_TRANSPARENCY_MASK

Rendering [8] – Note context inactive when created.

User driven compositing – call wfcCompose for every frame to render.

```
void  wfcCompose(WFCDevice dev, WFCContext ctx, WFCboolean wait)
```

Autonomous compositing – implementation decides when rendering is needed when context is active.

```
void  wfcActivate(WFCDevice dev, WFCContext ctx)

void  wfcDeactivate(WFCDevice dev, WFCContext ctx)
```

Renderer and extension information [10]

```
 WFCint wfcGetStrings(WFCDevice dev, WFCStringID name,
        const char **strings, WFCint stringsCount)

WFCboolean wfcIsExtensionSupported(WFCDevice dev,
        const char *string)
```

Attribute Creation and Destruction [7.1] and [7.6]

```
WFCElement  wfcCreateElement(WFCDevice dev, WFCContext ctx,
        const WFCint *attribList)

void wfcDestroyElement(WFCDevice dev, WFCElement element)
```

Querying Element Attributes [7.3] single value / vector of values

```
WFCint  wfcGetElementAttribi(WFCDevice dev, WFCElement element,
        WFCElementAttrib attrib)

WFCfloat  wfcGetElementAttribf(WFCDevice dev,
        WFCElement  element, WFCElementAttrib attrib)

void   wfcGetElementAttribiv(WFCDevice dev, WFCElement element,
        WFCElementAttrib attrib, WFCint count, WFCint *values)

void  wfcGetElementAttribfv(WFCDevice dev, WFCElement element,
        WFCElementAttrib attrib, WFCint count, WFCfloat *values)
```

Setting Element Attributes [7.4] single value / vector of values

```
 void  wfcSetElementAttribi(WFCDevice dev, WFCElement element,
        WFCElementAttrib attrib, WFCint value)

void  wfcSetElementAttribf(WFCDevice dev, WFCElement element,
        WFCElementAttrib attrib, WFCfloat value)

void  wfcSetElementAttribiv(WFCDevice dev, WFCElement element,
        WFCElementAttrib attrib, WFCint count, const WFCint *values)

void  wfcSetElementAttribfv(WFCDevice dev, WFCElement element,
        WFCElementAttrib attrib, WFCint count, const WFCfloat *values)
```

Element Ordering [7.5] – layering of images in the scene graph

wfcInsertElement() with a *subordinate* of **WFC_INVALID_HANDLE** inserts the element at the bottom of the scene

```
 void  wfcInsertElement(WFCDevice dev, WFCElement element,
        WFCElement subordinate)

void  wfcRemoveElement(WFCDevice dev, WFCElement element)

WFCElement  wfcGetElementAbove(WFCDevice dev,
        WFCElement element)

WFCElement  wfcGetElementBelow(WFCDevice dev,
        WFCElement element)
```

The Khronos Group is an industry consortium creating open standards for authoring and acceleration of parallell computing. Graphics and dynamic media on a wide variety of platforms and devices.

See www.khronos.org/openwf to learn more about the Khronos Group. And OpenWF