OpenWF Display® is a standardized API for compositing and output to display. It 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 implemented on top of a legacy display controller as well as specific hardware. The header file to include is <WF/wfd.h>

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

**Errors [2.11]** – of type WFDErrorCode

Error codes and their numerical values are defined by the WFDErrorCode enumeration and could be retrieved by the following function:

```
WFDErrorCode wfdGetError(WFDDevice device)
```

The possible values are as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFD_ERROR_NONE</td>
<td>0x0000</td>
</tr>
<tr>
<td>WFD_ERROR_OUT_OF_MEMORY</td>
<td>0x7510</td>
</tr>
<tr>
<td>WFD_ERROR_ILLEGAL_ARGUMENT</td>
<td>0x7511</td>
</tr>
<tr>
<td>WFD_ERROR_BAD_SUPPORTED</td>
<td>0x7512</td>
</tr>
<tr>
<td>WFD_ERROR_BAD_ACCESS</td>
<td>0x7514</td>
</tr>
<tr>
<td>WFD_ERROR_BUSY</td>
<td>0x7515</td>
</tr>
<tr>
<td>WFD_ERROR_BAD_HANDLE</td>
<td>0x7516</td>
</tr>
<tr>
<td>WFD_ERROR_INCONSISTENCY</td>
<td>0x7517</td>
</tr>
<tr>
<td>WFD_ERROR_IN_USE</td>
<td>0x7518</td>
</tr>
</tbody>
</table>

Functions that return handles could return the following error:

```
WFD_INVALID_HANDLE [2.6]
```

**Device** - A WFDDevice [3] is an abstraction of a display controller that supports one or more ports (WFDPort - display abstraction) and zero or more pipelines (a WFDPipe - manipulates source images).

**Device Creation and Destruction** [3.1], [3.2], [3.3]

```
WFDint wfdEnumerateDevices(WFDint *deviceIds,
                          WFDSend deviceIdsCount, const WFDint *filterList)
```

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

```
WFDDevice wfdCreateDevice(WFDint deviceId,
                         const WFDint *attribList)
```

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

```
WFDErrorCode wfdDestroyDevice(WFDDevice device)
```

Delete a specific device.

**Commit modifications** [3.4] Modifications are cached until committed.

```
void wfdDeviceCommit(WFDDevice device, WFDCommitType type,
                     WFDHandle handle)
```

Handle is a reference to the port or pipeline to commit – or WFD_INVALID_HANDLE when committing the entire device.

**Device Configuration** [3.5] – currently only WFD_DEVICE_ID is defined in the spec.

```
WFDint wfdGetDeviceAttrib(WFDDevice device,
                         WFDDeviceAttrib attrib)
```

```
void wfdSetDeviceAttrib(WFDDevice device,
                        WFDDeviceAttrib attrib, WFDValue value)
```

See WFDDeviceAttrib and WFDValue for details on these APIs.
OpenWF Display 1.0 API Quick Reference Card

**Port** - A WFDPort is the output from a WFDDevice (i.e. a display). It could be a CRT, a fixed LCD or an attachable TV for example. The API supports configuration of the display hardware.

```c
WFDInt wfdEnumeratePorts( WFDDevice device, WFDInt *portIds, WFDInt portIdsCount, const WFDInt *filterList )
Retrieve a list of numbers and IDs of available ports of a device. Note that ports with detached display hardware [4.5.1.3] will still be listed and possible to create. If ID = WFD_INVALID_PORT_ID an unfiltered list is returned.

WFDPort wfdCreatePort( WFDDevice device, WFDInt portId, const WFDInt *attribList )
If ID = WFD_DEFAULT_DEVICE_ID an integration specific default device is returned.

void wfdDestroyPort( WFDDevice device, WFDPort port )

**Port Modes** [4.4] – one or more mode supported for attached display hardware

WFDPortMode wfdGetPortModes( WFDDevice device, WFDPort port, WFDPortMode *modes, WFDInt modesCount )

WFDPortMode wfdGetCurrentPortMode( WFDDevice device, WFDPort port )

void wfdSetPortMode( WFDDevice device, WFDPort port, WFDPortMode mode )

WFDInt wfdGetPortModeAttrib( WFDDevice device, WFDPort port, WFDPortMode mode, WFDPortModeAttrib attrib )

**WFDPortConfigAttrib** [4.5.1]

WFD_PORT_ID 0x7620 (i) from wfdEnumeratePorts
WFD_PORT_TYPE 0x7621 (r) WFDPortType
WFD_PORT_DETACHABLE 0x7622 (r) WFD_TRUE or WFD_FALSE
WFD_PORT_ATTACHED 0x7623 (r) WFD_TRUE or WFD_FALSE
WFD_PORT_NATIVE_RESOLUTION 0x7624 (i) array (width, height) in pixels
WFD_PORT_PHYSICAL_SIZE 0x7625 (i) array (width, height) in mm
WFD_PORT_FILL_PORT_AREA 0x7626 (i) if WFD_TRUE whole area must be filled
WFD.PORT_BACKGROUND.COLOR 0x7627 (r/w) (r,g,b) in float [0 - 1]
WFD.PORT_FLIP 0x7628 (r/w) Dependent of hw support
WFD.PORT_MIRROR 0x7629 (r/w) Dependent of hw support
WFD.PORT_ROTATION 0x762A (r/w) in 90deg values if supported
WFD.PORT_POWER_MODE 0x762B (i) current powermode
WFD.PORT_GAMMA_RANGE 0x762C (i) array [min, max]
WFD.PORT_GAMMA 0x762D (r/w) min s value s max
WFD.PORT_PARTIAL_REFRESH_SUPPORT 0x762E (r) WFDPortPartialRefresh
WFD.PORT_PARTIAL_REFRESH_MAXIMUM 0x762F (i) array (width, height)
WFD.PORT_PARTIAL_REFRESH_ENABLE 0x7630 (i) WFD_TRUE or WFD_FALSE
WFD.PORT_PARTIAL_REFRESH_RECTANGLE 0x7631 (r/w) (offsetX, offsetY, width, height)
WFD.PORT_PIPELINE_ID_COUNT 0x7632 (i) Nbr of pipelines
WFD.PORTBindable_PIPELINE_IDS 0x7633 (i) List of pipelines
WFD.PORT_PROTECTION_ENABLE 0x7634 (r/w) WFD_TRUE or WFD_FALSE

**Get/Set Port Modes & Attributes**

**Power Mode** [4.5.1.11] WFDPowerMode – indicated but maybe not possible for a specific display hardware. Recovery time to ON decreases from OFF to SUSPEND to LIMITED_USE, and the power consumption will increase.

```
WFD_POWER_MODE_OFF 0x7666 No power –frames lost
WFD_POWER_MODE_SUSPEND 0x7667 Faster recovery then OFF
WFD_POWER_MODE_LIMITED_USE 0x7668 Frames maintained in hw
WFD_POWER_MODE_ON 0x7669 Fully operational
```

**Partial Refresh** [4.5.1.13]

WFD_PORT_PARTIAL_REFRESH_SUPPORT indicates what mode the display hw supports. WFD_PORT_PARTIAL_REFRESH_MAXIMUM defines the max size of the rectangle – (width, height). WFD_PORT_PARTIAL_REFRESH_RECT defines the actual size (offsetX, offsetY, width, height). WFD_PORT_PARTIAL_REFRESH_ENABLE activates the supported partial refresh mode from WFD_PORT_PARTIAL_REFRESH_SUPPORT.

WFDPortPartialRefresh – mode supported by the port

WFD_PARTIAL_REFRESH_NONE 0x7690
WFD_PARTIAL_REFRESH_VERTICAL 0x7691
WFD_PARTIAL_REFRESH_HORIZONTAL 0x7692
WFD_PARTIAL_REFRESH_BOTH 0x7693

Partial vertical – offsetY and height are used, partial horizontal – offsetX and width are used.

**Querying Port Attributes** [7.3] integer or float, single value / vector of values

WFDInt wfdGetPortAttrib( WFDDevice device, WFDPort port, WFDPortConfigAttrib attrib )

WFDfloat wfdGetPortAttribf( WFDDevice device, WFDPort port, WFDPortConfigAttrib attrib )

void wfdGetPortAttribv( WFDDevice device, WFDPort port, WFDPortConfigAttrib attrib, WFDInt count, WFDInt *value )

WFDfloat wfdGetPortAttribfv( WFDDevice device, WFDPort port, WFDPortConfigAttrib attrib, WFDInt count, WFDfloat *value )

**Assigning Port Attributes** [7.3] integer or float, single value / vector of values

void wfdSetPortAttrib( WFDDevice device, WFDPort port, WFDPortConfigAttrib attrib, WFDInt value )

void wfdSetPortAttribf( WFDDevice device, WFDPort port, WFDPortConfigAttrib attrib, WFDfloat value )

void wfdSetPortAttribv( WFDDevice device, WFDPort port, WFDPortConfigAttrib attrib, WFDInt count, const WFDInt *value )

void wfdSetPortAttribfv( WFDDevice device, WFDPort port, WFDPortConfigAttrib attrib, WFDInt count, const WFDfloat *value )

void wfdBindPipelineToPort( WFDDevice device, WFDPort port, WFDPipeline pipeline )

© 2010 Khronos Group

http://www.khronos.org/openwf/
Pipelines [5] – is an abstraction of the hardware that transforms and blends source images into the final compositized image on the display. Note that mask, rotation and scaling are optional to support.

WFDint wfdEnumeratePipelines( WFDDevice device, WFDint *pipelineIds, WFDInt pipelineIdsCount const WFDint *filterList )

WFDPipeline wfdCreatePipeline( WFDDevice device, WFDInt pipelineId, const WFDInt *attribList )

void wfdDestroyPipeline( WFDDevice device, WFDPipeline pipeline )

Pipeline Layering [5, 9] - retrieves the pipeline layering order without having to bind the port and pipeline

WFDInt wfdGetPipelineLayerOrder( WFDDevice device, WFDPort port, WFDPipeline pipeline )

Returns the same value as for the WFD_PIPELINE_LAYER attribute on success.

Display Data [4.7]

WFDDisplayDataFormat – format types that could be supported

WFDint wfdGetDisplayDataFormats( WFDDevice device, WFDPort port, WFDDisplayDataFormat *format, WFDint formatCount )

Check what dataformats the display supports.

WFDint wfdGetDisplayData( WFDDevice device, WFDPort port, WFDDisplayDataFormat format, WFDuint8 *data, WFDInt dataCount )

Retrieve display data in a specific format.

Get/Set Pipeline Attributes [5.7.2] & [5.7.3] integer or float, single value / vector of values

WFDInt wfdGetPipelineAttrib( WFDDevice device, WFDPipeline pipeline, WFDPipelineConfigAttrib attrib )

WFDfloat wfdGetPipelineAttribf( WFDDevice device, WFDPipeline pipeline, WFDPipelineConfigAttrib attrib )

void wfdGetPipelineAttribv( WFDDevice device, WFDPipeline pipeline, WFDPipelineConfigAttrib attrib, WFDint count, WFDint *value )

void wfdSetPipelineAttrib( WFDDevice device, WFDPipeline pipeline, WFDPipelineConfigAttrib attrib, WFDint count, WFDfloat *value )

void wfdSetPipelineAttribv( WFDDevice device, WFDPipeline pipeline, WFDPipelineConfigAttrib attrib, WFDint count, const WFDfloat *value )

Scaling [5.7.1.9]

WFDScaleFilter

| WFD_SCALE_FILTER.NONE | 0x7760 |
| WFD_SCALE_FILTER.FASTER | 0x7761 |
| WFD_SCALE_FILTER.BETTER | 0x7762 |

Transparency [5.8]

WFDTransparency – bit field denoting possible combinations of supported transparency

WFDint wfdGetPipelineTransparency( WFDDevice device, WFDPipeline pipeline, WFDInt transCount )

Query the supported transparency pixel formats.

WFDTSFormat – transparent source color type supported

WFDint wfdGetDisplayTSColor( WFDDevice device, WFDPipeline pipeline, WFDTSColorFormat colorFormat, WFDInt count, const void *color )

Set transparent color for the pipeline.
Image Sources [5.5.1] Content that can be used as input to display pipelines.

`WFDSource wfdCreateSourceFromImage(WFDDevice device, WFDLine pipeline, WFDEGLImage image, const WFInt *attribList)`

`WFDSource wfdCreateSourceFromStream(WFDDevice device, WFDLine pipeline, WFDEGLImage image, const WFInt *attribList)`

For streams see also [2.8].

`void wfdDestroySource(WFDDevice device, WFDSource source)`

Rendered and extension information [6]

<table>
<thead>
<tr>
<th>WFDStringID</th>
<th>Information about the runtime platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFD_VENDOR</td>
<td>0x7500</td>
</tr>
<tr>
<td>WFD_RENDERER</td>
<td>0x7501</td>
</tr>
<tr>
<td>WFD_VERSION</td>
<td>0x7502</td>
</tr>
<tr>
<td>WFD_EXTENSIONS</td>
<td>0x7503</td>
</tr>
</tbody>
</table>

`WFDSelf wfdGetStrings(WFDDevice device, WFDStringID name, const char **strings, WFInt stringsCount)`

`WFDSelf wfdIsExtensionSupported(WFDDevice device, const char *string)`

WFDRect – only relevant for EGLImage sources (offsetX, offsetY, width, height)

WFDTransition

<table>
<thead>
<tr>
<th>WFD_TRANSITION_INVALID</th>
<th>0x77E0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFD_TRANSITION_IMMEDIATE</td>
<td>0x77E1</td>
</tr>
<tr>
<td>WFD_TRANSITION_AT_VSYNC</td>
<td>0x77E2</td>
</tr>
</tbody>
</table>

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

See [www.khronos.org/openmf](http://www.khronos.org/openmf) to learn more about the Khronos Group. And OpenWF