



WiSeSLAp 1.0 'Baklava' Documentation

Generated with ROBODoc Version 4.99.38 (Sep 10 2009)

November 9, 2009

Index

Classes

- TWslDriver, 6
- TWslDriverDFB, 21
- TWslDriverSDL, 21
- TWslDriverXLIB, 21
- TWslSurface, 22
- TWslSurfaceDFB, 38
- TWslSurfaceSDL, 39
- TWslSurfaceXLIB, 39

Enumerations

- TWslBackend, 6
- TWslPixelFormat, 41

Functions

- WslCreateDriver, 40

Methods

- TWslDriver.Create, 9
- TWslDriver.Destroy, 10
- TWslDriver.FreeAllSurfaces, 11
- TWslDriver.Identify, 12
- TWslDriver.PollForEvent, 12
- TWslDriver.ProduceSurface(1), 13
- TWslDriver.ProduceSurface(2), 14
- TWslDriver.ReleaseSurface(1), 14
- TWslDriver.ReleaseSurface(2), 15
- TWslDriver.RequestSurface(1), 15
- TWslDriver.RequestSurface(2), 16
- TWslDriver.Start, 18
- TWslDriver.Stop, 19
- TWslDriver.WaitForEvent, 20
- TWslSurface.CairoStart, 24
- TWslSurface.CairoStop, 25
- TWslSurface.Create, 25
- TWslSurface.DecRefCount, 26
- TWslSurface.Destroy, 26
- TWslSurface.DoCacheLock, 27
- TWslSurface.Flip, 27
- TWslSurface.Flood, 28
- TWslSurface.GetPixel(1), 28
- TWslSurface.GetPixel(2), 28
- TWslSurface.IncRefCount, 30

- TWslSurface.Lock, 31
- TWslSurface.RectangleBlit(1), 33
- TWslSurface.RectangleBlit(2), 33
- TWslSurface.Release, 34
- TWslSurface.SetRefCount, 35
- TWslSurface.SimpleBlit, 36
- TWslSurface.SimpleRectangleBlit, 36
- TWslSurface.Unlock, 37
- TWslSurface.UpdateRect, 37

Properties

- TWslDriver.AllowXRender, 8
- TWslDriver.Backend, 8
- TWslDriver.DefaultPixelFormat, 9
- TWslDriver.DoubleBuffer, 10
- TWslDriver.FullScreen, 11
- TWslDriver.HardwareSurfaces, 12
- TWslDriver.Resolution, 17
- TWslDriver.Started, 18
- TWslDriver.VSync, 19
- TWslDriver.WindowTitle, 20
- TWslSurface.CacheLocked, 23
- TWslSurface.CairoCapable, 24
- TWslSurface.GUID, 29
- TWslSurface.Height, 30
- TWslSurface.IsPrimary, 31
- TWslSurface.PixFmt, 32
- TWslSurface.Rectangle, 32
- TWslSurface.RefCount, 34
- TWslSurface.Width, 38

Units

- WslBackends, 6
- WslFactory, 40
- WslTypes, 41
- WslWidgets, 42

unsorted

- TWslBackend, 6
- TWslDriver, 6
- TWslDriver.AllowXRender, 8
- TWslDriver.Backend, 8
- TWslDriver.Create, 9
- TWslDriver.DefaultPixelFormat, 9

TWslDriver.Destroy, 10
TWslDriver.DoubleBuffer, 10
TWslDriver.FreeAllSurfaces, 11
TWslDriver.FullScreen, 11
TWslDriver.HardwareSurfaces, 12
TWslDriver.Identify, 12
TWslDriver.PollForEvent, 12
TWslDriver.ProduceSurface(1), 13
TWslDriver.ProduceSurface(2), 14
TWslDriver.ReleaseSurface(1), 14
TWslDriver.ReleaseSurface(2), 15
TWslDriver.RequestSurface(1), 15
TWslDriver.RequestSurface(2), 16
TWslDriver.Resolution, 17
TWslDriver.Start, 18
TWslDriver.Started, 18
TWslDriver.Stop, 19
TWslDriver.VSync, 19
TWslDriver.WaitForEvent, 20
TWslDriver.WindowTitle, 20
TWslDriverDFB, 21
TWslDriverSDL, 21
TWslDriverXLIB, 21
TWslPixelFormat, 41
TWslSurface, 22
TWslSurface.CacheLocked, 23
TWslSurface.CairoCapable, 24
TWslSurface.CairoStart, 24
TWslSurface.CairoStop, 25
TWslSurface.Create, 25
TWslSurface.DecRefCount, 26
TWslSurface.Destroy, 26
TWslSurface.DoCacheLock, 27
TWslSurface.Flip, 27
TWslSurface.Flood, 28
TWslSurface.GetPixel(1), 28
TWslSurface.GetPixel(2), 28
TWslSurface.GUID, 29
TWslSurface.Height, 30
TWslSurface.IncRefCount, 30
TWslSurface.IsPrimary, 31
TWslSurface.Lock, 31
TWslSurface.PixFmt, 32
TWslSurface.Rectangle, 32
TWslSurface.RectangleBlit(1), 33
TWslSurface.RectangleBlit(2), 33
TWslSurface.RefCount, 34
TWslSurface.Release, 34
TWslSurface.SetRefCount, 35
TWslSurface.SimpleBlit, 36
TWslSurface.SimpleRectangleBlit, 36
TWslSurface.Unlock, 37
TWslSurface.UpdateRect, 37
TWslSurface.Width, 38
TWslSurfaceDFB, 38
TWslSurfaceSDL, 39
TWslSurfaceXLIB, 39
WslBackends, 6
WslCreateDriver, 40
WslFactory, 40
WslTypes, 41
WslWidgets, 42

Contents

I	WiSeSLAp/WslBackends	6
1	WslBackends/TWslBackend	6
2	WslBackends/TWslDriver	6
3	WslBackends/TWslDriver.AllowXRender	8
4	WslBackends/TWslDriver.Backend	8
5	WslBackends/TWslDriver.Create	9
6	WslBackends/TWslDriver.DefaultPixelFormat	9
7	WslBackends/TWslDriver.Destroy	10
8	WslBackends/TWslDriver.DoubleBuffer	10
9	WslBackends/TWslDriver.FreeAllSurfaces	11
10	WslBackends/TWslDriver.FullScreen	11
11	WslBackends/TWslDriver.HardwareSurfaces	12
12	WslBackends/TWslDriver.Identify	12
13	WslBackends/TWslDriver.PollForEvent	12
14	WslBackends/TWslDriver.ProduceSurface(1)	13
15	WslBackends/TWslDriver.ProduceSurface(2)	14
16	WslBackends/TWslDriver.ReleaseSurface(1)	14
17	WslBackends/TWslDriver.ReleaseSurface(2)	15
18	WslBackends/TWslDriver.RequestSurface(1)	15

<i>CONTENTS</i>	5
19 WslBackends/TWslDriver.RequestSurface(2)	16
20 WslBackends/TWslDriver.Resolution	17
21 WslBackends/TWslDriver.Start	18
22 WslBackends/TWslDriver.Started	18
23 WslBackends/TWslDriver.Stop	19
24 WslBackends/TWslDriver.VSync	19
25 WslBackends/TWslDriver.WaitForEvent	20
26 WslBackends/TWslDriver.WindowTitle	20
27 WslBackends/TWslDriverDFB	21
28 WslBackends/TWslDriverSDL	21
29 WslBackends/TWslDriverXLIB	21
30 WslBackends/TWslSurface	22
31 WslBackends/TWslSurface.CacheLocked	23
32 WslBackends/TWslSurface.CairoCapable	24
33 WslBackends/TWslSurface.CairoStart	24
34 WslBackends/TWslSurface.CairoStop	25
35 WslBackends/TWslSurface.Create	25
36 WslBackends/TWslSurface.DecRefCount	26
37 WslBackends/TWslSurface.Destroy	26
38 WslBackends/TWslSurface.DoCacheLock	27

<i>CONTENTS</i>	6
39 WslBackends/TWslSurface.Flip	27
40 WslBackends/TWslSurface.Flood	28
41 WslBackends/TWslSurface.GetPixel(1)	28
42 WslBackends/TWslSurface.GetPixel(2)	28
43 WslBackends/TWslSurface.GUID	29
44 WslBackends/TWslSurface.Height	30
45 WslBackends/TWslSurface.IncRefCount	30
46 WslBackends/TWslSurface.IsPrimary	31
47 WslBackends/TWslSurface.Lock	31
48 WslBackends/TWslSurface.PixFmt	32
49 WslBackends/TWslSurface.Rectangle	32
50 WslBackends/TWslSurface.RectangleBlit(1)	33
51 WslBackends/TWslSurface.RectangleBlit(2)	33
52 WslBackends/TWslSurface.RefCount	34
53 WslBackends/TWslSurface.Release	34
54 WslBackends/TWslSurface.SetRefCount	35
55 WslBackends/TWslSurface.SimpleBlit	36
56 WslBackends/TWslSurface.SimpleRectangleBlit	36
57 WslBackends/TWslSurface.Unlock	37
58 WslBackends/TWslSurface.UpdateRect	37

<i>CONTENTS</i>	7
59 WslBackends/TWslSurface.Width	38
60 WslBackends/TWslSurfaceDFB	38
61 WslBackends/TWslSurfaceSDL	39
62 WslBackends/TWslSurfaceXLIB	39
63 WslBackends/WslCreateDriver	40
II WiSeSLAp/WslFactory	40
III WiSeSLAp/WslTypes	41
64 WslTypes/TWslPixelFormat	41
IV WiSeSLAp/WslWidgets	42

Part I

WiSeSLAp/WslBackends

[Units]

NAME: WslBackends

DESCRIPTION: The unit which contains all relevant abstractions and concrete implementations of the three backends: DFB, SDL, XLIB.

MAINTAINER: Roland Schaefer <mail@rolandschaefer.net>

1 WslBackends/TWslBackend

[WslBackends] [Enumerations]

NAME: TWslBackend

VALUES:

```
wibDFB : Use DirectFB.
wibSDL : Use SDL.
wibXLIB : Use Xlib with core X or XRender rendering.
wibNone : Dummy backend, will planfully lead to an exception.
```

NOTES: The availability of values depends on which backends were compiled into WiSeSLAp.

SEE ALSO:

```
WslBackends/TWslDriver
WslBackends/WslCreateDriver
```

2 WslBackends/TWslDriver

[WslBackends] [Classes]

NAME: TWslDriver (The abstract WslBackend driver class)

ANCESTOR:

```
TObject
```

DESCENDANTS:

WslBackends/TWslDriverDFB
WslBackends/TWslDriverSDL
WslBackends/TWslDriverXLIB

DESCRIPTION: Never initialize this class directly, since it is abstract. Initialize it by calling WslBackends (l)/WslCreateDriver (63).

SEE ALSO:

WslBackends/WslCreateDriver

CONSTRUCTORS:

WslBackends/TWslDriver.Create

DESTRUCTORS:

WslBackends/TWslDriver.Destroy

PUBLIC METHODS:

WslBackends/TWslDriver.Start
WslBackends/TWslDriver.Stop
WslBackends/TWslDriver.ProduceSurface(1)
WslBackends/TWslDriver.ProduceSurface(2)
WslBackends/TWslDriver.FreeAllSurfaces
WslBackends/TWslDriver.RequestSurface(1)
WslBackends/TWslDriver.RequestSurface(2)
WslBackends/TWslDriver.ReleaseSurface(1)
WslBackends/TWslDriver.ReleaseSurface(2)
WslBackends/TWslDriver.PollForEvent
WslBackends/TWslDriver.Identify

PUBLIC PROPERTIES:

WslBackends/TWslDriver.Backend
WslBackends/TWslDriver.WindowTitle
WslBackends/TWslDriver.Started
WslBackends/TWslDriver.Resolution
WslBackends/TWslDriver.FullScreen
WslBackends/TWslDriver.DefaultPixelFormat
WslBackends/TWslDriver.HardwareSurfaces
WslBackends/TWslDriver.DoubleBuffer
WslBackends/TWslDriver.VSync
WslBackends/TWslDriver.AllowXRender

3 WslBackends/TWslDriver.AllowXRender

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslDriver.AllowXRender

TYPE:

Boolean

DESCRIPTION: When using the Xlib backend, this can be used to force XRender functionality off (false) for debugging. It is also set to false when the driver detects that there is no XRender support in the server connected to, which also generates an exception informing the creator of the driver about missing X extensions.

NOTES: Since TWslSurface (30) is just a shallow wrapper to structures offered by the backends, surfaces will always be X pixmaps when the backend is Xlib and the driver is forced not to use XRender. Independently of the pixel format a surface was requested to have, this means that there is no alpha blitting when XRender is off, since core X rendering only supports solid copy operations.

SETTER:

WslBackends/TWslDriver.SetAllowXRender

GETTER:

WslBackends/TWslDriver.FAllowXRender

4 WslBackends/TWslDriver.Backend

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslDriver.Backend

TYPE:

TWslBackend

DESCRIPTION: Retrieves information which type backend this object is an instance of.

SETTER:

(read-only)

GETTER:

WslBackends/TWslDriver.FBackend

SEE ALSO:

WslBackends/WslCreateDriver
WslTypes/TWslBackend

5 WslBackends/TWslDriver.Create

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
constructor Create; virtual;
```

DESCRIPTION: Creates a driver without starting it. This means that the graphics output isn't started and no primary surface is created. Always call the wrapper WslBackends (I)/WslCreateDriver (63) instead of this creator.

SEE ALSO:

WslBackends/TWslDriver.Start
WslBackends/TWslDriver.Stop
WslBackends/TWslDriver.Destroy

6 WslBackends/TWslDriver.DefaultPixelFormat

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslDriver.DefaultPixelFormat

TYPE:

```
TWslPixelFormat
```

DESCRIPTION: The generic pixel format used when surfaces are created with either TWslDriver.ProduceSurface(2) (15) or TWslDriver.RequestSurface(2) (19). Can be changed at any time.

SETTER:

```
WslBackends/TWslDriver.FDefaultPixelFormat
```

GETTER:

```
WslBackends/TWslDriver.FDefaultPixelFormat
```

SEE ALSO:

TWslDriver.RequestSurface(2)
TWslDriver.ProduceSurface(2)
WslTypes/TWslPixelFormat

7 WslBackends/TWslDriver.Destroy

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
destructor Destroy; override;
```

DESCRIPTION: Destroys the driver and first calls TWslDriver.Stop (23) if necessary.

8 WslBackends/TWslDriver.DoubleBuffer

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslDriver.DoubleBuffer

TYPE:

Boolean

DESCRIPTION: Whether the display should be double buffered. Currently only supported in DirectFB and leading to possibly undesired behavior. Should be set before TWslDriver.Start (21) is called.

SETTER:

```
WslBackends/TWslDriver.FDoubleBuffer
```

GETTER:

```
WslBackends/TWslDriver.FDoubleBuffer
```

SEE ALSO:

WslBackends/TWslSurface.Flip

9 WslBackends/TWslDriver.FreeAllSurfaces

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
function FreeAllSurfaces : Integer;
```

DESCRIPTION: Destroys all surfaces owned by this driver. Should only be called if the caller is really done doing anything with this driver or wants to completely reconstruct the graphics.

SEE ALSO:

```
WslBackends/TWslDriver.ReleaseSurface(1)  
WslBackends/TWslDriver.ReleaseSurface(2)  
WslBackends/TWslSurface.Release
```

10 WslBackends/TWslDriver.FullScreen

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslDriver.FullScreen

TYPE:

```
Boolean
```

DESCRIPTION: Whether the full screen should be used. Might override the resolution set with WslBackends (I)/TWslDriver.Resolution (20). Should be set before TWslDriver.Start (21) is called.

SETTER:

```
WslBackends/TWslDriver.FFullscreen
```

GETTER:

```
WslBackends/TWslDriver.FFullscreen
```

SEE ALSO:

```
WslBackends/TWslDriver.Resolution  
WslBackends/TWslDriver.Start
```

11 WslBackends/TWslDriver.HardwareSurfaces

[*WslBackends*] [*Properties*]

NAME: WslBackends (l)/TWslDriver.HardwareSurfaces

TYPE:

Boolean

DESCRIPTION: Whether hardware surfaces should be used if available. Should be set before TWslDriver.Start (21) is called. Currently unused.

SETTER:

WslBackends/TWslDriver.FHardwareSurfaces

GETTER:

WslBackends/TWslDriver.FHardwareSurfaces

12 WslBackends/TWslDriver.Identify

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
class function Identify : string; virtual; abstract;
```

DESCRIPTION: A class function which returns a string identifying the current driver and the highest version of the backend library (DirectFB, Xlib or SDL) for which WiSeSLAp has certified the functionality of TWslDriver (2) and TWslSurface (30).

RESULT:

Identify : A driver backend identifier with version in plain English.

13 WslBackends/TWslDriver.PollForEvent

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
function PollForEvent : PWslIOEvent; virtual; abstract;
```

DESCRIPTION: Returns an I/O event if there is one in the queue. Usually only reasonable to call after TWslDriver.Start (21) has been called on the driver.

NOTES: TWslDriver (2) objects do not have their own event queue, so expect different behavior across backends, esp. in as much as length of the event queue is concerned.

RESULT:

PollFor Event : A pointer to the TWslIOEvent which is a member of the TWslDriver object, or nil if there was no new event.

SEE ALSO:

WslBackends/TWslDriver.Start
WslBackends/TWslDriver.WaitForEvent
WslTypes/TWslIOEvent

14 WslBackends/TWslDriver.ProduceSurface(1)

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
function ProduceSurface(W,H : SmallInt; IsForCairo : Boolean) :
TWslSurface;
```

DESCRIPTION: Creates a surface which has no true reference counter and is not added to the cache pool. The surface receives a (pseudo-)GUID automatically and is owned by the driver. It is instantly destroyed on the first call to its Release method. The pixel format is the default one as set in the driver with TWslDriver.DefaultPixelFormat (6).

ARGUMENTS:

W : Width of the surface.
H : Height of the surface.
IsForCairo : Whether the surface should be Cairo-enabled.

RESULT:

ProduceSurface : A TWslSurface object, underlyingly a backend-specific object of TWslSurfaceXLIB etc.

SEE ALSO:

WslBackends/TWslSurface.Release

```

WslBackends/TWslDriver.DefaultPixelFormat
WslTypes/TWslPixelFormat
WslBackends/TWslDriver.ProduceSurface(2)
WslBackends/TWslDriver.RequestSurface(1)
WslBackends/TWslDriver.RequestSurface(2)

```

15 WslBackends/TWslDriver.ProduceSurface(2)

[WslBackends] [Methods]

PROTOTYPE:

```

function ProduceSurface(PixelFormat : TWslPixelFormat;
W,H : SmallInt; IsForCairo : Boolean) : TWslSurface;

```

DESCRIPTION: Like TWslDriver.ProduceSurface(1) (14), but with explicit request for a specific pixel format.

ARGUMENTS:

```

PixelFormat : The desired pixel format (implicitly also depth)
W : Width of the surface.
H : Height of the surface.
IsForCairo : Whether the surface should be Cairo-enabled.

```

RESULT:

```

ProduceSurface : A TWslSurface object, underlyingly a
backend-specific object of TWslSurfaceXLIB etc.

```

SEE ALSO:

```

WslBackends/TWslSurface.Release
WslTypes/TWslPixelFormat
WslBackends/TWslDriver.ProduceSurface(1)
WslBackends/TWslDriver.RequestSurface(1)
WslBackends/TWslDriver.RequestSurface(2)

```

16 WslBackends/TWslDriver.ReleaseSurface(1)

[WslBackends] [Methods]

PROTOTYPE:

```
function ReleaseSurface(const AGUID : Shortstring) : Integer;
```

DESCRIPTION: Decrements the reference counter of the surface. See WslBackends (I)/TWslSurface.Release (53).

ARGUMENTS:

AGUID : The GUID of the surface to be released.

SEE ALSO:

```
WslBackends/TWslDriver.RequestSurface(1)
WslBackends/TWslDriver.RequestSurface(2)
WslBackends/TWslDriver.ReleaseSurface(2)
WslBackends/TWslSurface.Release
```

17 WslBackends/TWslDriver.ReleaseSurface(2)

[WslBackends] [Methods]

PROTOTYPE:

```
function ReleaseSurface(ASurface : TWslSurface) : Integer;
```

DESCRIPTION: See WslBackends (I)/TWslDriver.RequestSurface(1) (18).

ARGUMENTS:

ASurface : The surface object to be released.

SEE ALSO:

```
WslBackends/TWslDriver.RequestSurface(1)
WslBackends/TWslDriver.RequestSurface(2)
WslBackends/TWslDriver.ReleaseSurface(1)
WslBackends/TWslSurface.Release
```

18 WslBackends/TWslDriver.RequestSurface(1)

[WslBackends] [Methods]

PROTOTYPE:

```
function RequestSurface(W,H : SmallInt; IsForCairo : Boolean;
const GUID : ShortString) : TWslSurface;
```

DESCRIPTION: This creates a surface and allows to give it a name which can be used to retrieve it from the cache pool (the (pseudo-)GUID). The pseudo-GUID for cachable primitives should be generated with WslPrm methods. If a surface with GUID already exists, it will just be returned from cache. Each such request leads to an increment of the reference counter. The pixel format is the default one as set in the driver with TWslDriver.DefaultPixelFormat (6).

ARGUMENTS:

W : Width of the surface.
H : Height of the surface.
IsForCairo : Whether the surface should be Cairo-enabled.
GUID : The pseudo-GUID string by which the surface is identified in the cache pool.

RESULT:

RequestSurface : A TWslSurface object, underlyingly a backend-specific object of TWslSurfaceXLIB etc.

SEE ALSO:

WslBackends/TWslDriver.RequestSurface(2)
WslBackends/TWslDriver.ProduceSurface(1)
WslBackends/TWslDriver.ProduceSurface(2)
WslBackends/TWslDriver.ReleaseSurface(1)
WslBackends/TWslDriver.ReleaseSurface(2)
WslBackends/TWslDriver.DefaultPixelFormat
WslTypes/TWslPixelFormat
WiSeSLAp/WslPrm

19 WslBackends/TWslDriver.RequestSurface(2)

[WslBackends] [Methods]

PROTOTYPE:

```
function RequestSurface(PixelFormat : TWslPixelFormat; W,H : SmallInt;
IsForCairo : Boolean; const GUID : ShortString) : TWslSurface;
```

DESCRIPTION: Like TWslDriver.RequestSurface(1) (18), but with explicit request for a specific pixel format.

ARGUMENTS:

PixelFormat : The desired pixel format (implicitly also depth)
 W : Width of the surface.
 H : Height of the surface.
 IsForCairo : Whether the surface should be Cairo-enabled.
 GUID : The pseudo-GUID string by which the surface is identified
 in the cache pool.

RESULT:

RequestSurface : A TWslSurface object, underlyingly a backend-specific object of TWslSurfaceXLIB etc.

SEE ALSO:

WslBackends/TWslDriver.RequestSurface(1)
 WslBackends/TWslDriver.ProduceSurface(1)
 WslBackends/TWslDriver.ProduceSurface(2)
 WslBackends/TWslDriver.ReleaseSurface(1)
 WslBackends/TWslDriver.ReleaseSurface(2)
 WslTypes/TWslPixelFormat
 WiSeSLAp/WslPrm

20 WslBackends/TWslDriver.Resolution

[WslBackends] [Properties]

NAME: WslBackends (l)/TWslDriver.Resolution

TYPE:

TWslResolution

DESCRIPTION: Informs the caller which resolution has been set on the creation of the driver. This is the resolution available to the current application. The actual screen might be larger. Should be set before TWslDriver.Start (21) is called.

NOTES: With a DirectFB backend, WiSeSLAp will ultimately support dynamic resolution changes of the framebuffer device, such that the resolution set here would actually set the device resolution. This has not been implemented yet, mostly due to our lack of testing equipment.

SETTER:

WslBackends/TWslDriver.FResolution

GETTER:

WslBackends/TWslDriver.FResolution

SEE ALSO:

WslTypes/TWslResolution

SEE ALSO:

WslBackends/TWslDriver.FullScreen

WslBackends/TWslDriver.Start

21 WslBackends/TWslDriver.Start

[WslBackends] [Methods]

PROTOTYPE:

```
function Start : TWslSurface; virtual; abstract;
```

DESCRIPTION: Opens the display, enables event input, creates a primary surface and possibly mallocs many underlying structures depending on the chosen backend.

RESULT:

Start : The primary surface for graphics I/O.

22 WslBackends/TWslDriver.Started

[WslBackends] [Properties]

NAME: WslBackends (I)/TWslDriver.Started

TYPE:

Boolean

DESCRIPTION: Informs the caller whether the driver is in a started state.

SETTER:

(read-only)

GETTER:

WslBackends/TWslDriver.FStarted

SEE ALSO:

WslBackends/TWslDriver.Start

23 WslBackends/TWslDriver.Stop

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure Stop; virtual; abstract;
```

DESCRIPTION: Stop goes back to a state in between Create and Start. The display shuts down, all surfaces created through this driver are freed.

24 WslBackends/TWslDriver.VSync

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslDriver.VSync

TYPE:

Boolean

DESCRIPTION: Whether the driver should wait for vertical sync before doing flipping operations. Only supported with DirectFB. Should be set before TWslDriver.Start (21) is called.

SETTER:

```
WslBackends/TWslDriver.FVSync
```

GETTER:

```
WslBackends/TWslDriver.FVSync
```

SEE ALSO:

WslBackends/TWslSurface.Flip

25 WslBackends/TWslDriver.WaitForEvent

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
function WaitForEvent : PWSLIOEvent; virtual; abstract;
```

DESCRIPTION: Waits infinitely for an I/O event to happen, then returns it. Usually only reasonable to call after TWslDriver.Start (21) has been called on the driver.

NOTES: TWslDriver (2) objects do not have their own event queue, so expect different behavior across backends, esp. in as much as length of the event queue is concerned.

RESULT:

WaitFor Event : A pointer to the TWslIOEvent which is a member of the TWslDriver object.

SEE ALSO:

```
WslBackends/TWslDriver.Start
WslBackends/TWslDriver.PollForEvent
WslTypes/TWslIOEvent
```

26 WslBackends/TWslDriver.WindowTitle

[*WslBackends*] [*Properties*]

NAME: WslBackends (l)/TWslDriver.WindowTitle

TYPE:

```
AnsiString
```

DESCRIPTION: Sets or gets the window title to be used in a windowed environment.

SETTER:

```
WslBackends/TWslDriver.FWindowTitle
```

GETTER:

```
WslBackends/TWslDriver.FWindowTitle
```

27 WslBackends/TWslDriverDFB

[*WslBackends*] [*Classes*]

NAME: TWslDriverDFB (An implemented WslBackend driver class)

ANCESTOR:

TWslDriver

DESCRIPTION: This class implements its ancestor with DirectFB-specific routines. Never initialize this class directly. Get instances by using the wrapper WslBackends (I)/WslCreateDriver (63)

SEE ALSO:

WslBackends/WslCreateDriver

28 WslBackends/TWslDriverSDL

[*WslBackends*] [*Classes*]

NAME: TWslDriverSDL (An implemented WslBackend driver class)

ANCESTOR:

TWslDriver

DESCRIPTION: This class implements its ancestor with SDL-specific routines. Never initialize this class directly. Get instances by using the wrapper WslBackends (I)/WslCreateDriver (63)

SEE ALSO:

WslBackends/WslCreateDriver

29 WslBackends/TWslDriverXLIB

[*WslBackends*] [*Classes*]

NAME: TWslSurfaceXLIB (62) (An implemented WslBackend driver class)

ANCESTOR:

TWslDriver

DESCRIPTION: This class implements its ancestor with XLIB-specific routines. Never initialize this class directly. Get instances by using the wrapper WslBackends (I)/WslCreateDriver (63)

SEE ALSO:

WslBackends/WslCreateDriver

30 WslBackends/TWslSurface

[*WslBackends*] [*Classes*]

NAME: TWslSurface (The abstract WslBackend surface class)

ANCESTOR:

TObject

DESCENDANTS:

WslBackends/TWslSurfaceDFB
 WslBackends/TWslSurfaceSDL
 WslBackends/TWslSurfaceXLIB

DESCRIPTION: This class provides the middle level access to graphics resources. Instances maintain actual SDL, DirectFB surfaces or Xlib pixmaps and XRender images. They are all accessed through the same blitting and editing methods defined by this class and implemented by the non-abstract descendants. Most prominently, TWslSurface provides easy access to Cairo contexts for drawing onto them. Create instances only through the TWslDriver (2).ProduceSurface and TWslDriver (2).RequestSurface methods!

SEE ALSO:

WslBackends/TWslDriver.ProduceSurface
 WslBackends/TWslDriver.RequestSurface

PUBLIC PROPERTIES:

WslBackends/TWslSurface.GUID
 WslBackends/TWslSurface.CacheLocked
 WslBackends/TWslSurface.RefCount
 WslBackends/TWslSurface.Rectangle
 WslBackends/TWslSurface.Width
 WslBackends/TWslSurface.Height
 WslBackends/TWslSurface.PixFmt
 WslBackends/TWslSurface.CairoCapable
 WslBackends/TWslSurface.IsPrimary

CONSTRUCTORS:

WslBackends/TWslSurface.Create

DESTRUCTORS:

WslBackends/TWslSurface.Destroy

PUBLIC METHODS:

WslBackends/TWslSurface.CairoStart
 WslBackends/TWslSurface.CairoStop
 WslBackends/TWslSurface.SimpleBlit
 WslBackends/TWslSurface.SimpleRectangleBlit
 WslBackends/TWslSurface.RectangleBlit(1)
 WslBackends/TWslSurface.RectangleBlit(2)
 WslBackends/TWslSurface.Lock
 WslBackends/TWslSurface.Unlock
 WslBackends/TWslSurface.Flip
 WslBackends/TWslSurface.UpdateRect
 WslBackends/TWslSurface.GetPixel(1)
 WslBackends/TWslSurface.GetPixel(2)
 WslBackends/TWslSurface.Flood
 WslBackends/TWslSurface.SetRefCount
 WslBackends/TWslSurface.IncRefCount
 WslBackends/TWslSurface.DecRefCount
 WslBackends/TWslSurface.DoCacheLock
 WslBackends/TWslSurface.Release

31 WslBackends/TWslSurface.CacheLocked

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslSurface.CacheLocked

TYPE:

Boolean

DESCRIPTION: Provides information whether a cachable surface has already been cache-locked and cannot be modified anymore. Functions from the WslPrm subsystem, for example, cache-lock the surface after they have drawn the primitive (a button etc.)

SETTER:

(read-only)

GETTER:

WslBackends/TWslSurface.FCacheLocked

SEE ALSO:

WslBackends/TWslSurface.DoCacheLock

32 WslBackends/TWslSurface.CairoCapable

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslSurface.CairoCapable

TYPE:

Boolean

DESCRIPTION: Retrieves information whether the surface was created as Cairo-capable.

SETTER:

(read-only)

GETTER:

WslBackends/TWslSurface.FIsCairo

SEE ALSO:

WslBackends/TWslDriver.RequestSurface
WslBackends/TWslDriver.ProduceSurface

33 WslBackends/TWslSurface.CairoStart

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
function CairoStart : PCairo_t; virtual; abstract;
```

DESCRIPTION: Use to get a cairo context on the Surface, if the surface was created as cairo-capable.

RESULT:

A Cairo context pointer (Pcairo_t), if the surface was created as Cairo-capable, else nil.

SEE ALSO:

WslBackends/TWslSurface.CairoStop
 WslBackends/TWslDriver.ProduceSurface
 WslBackends/TWslDriver.RequestSurface

34 WslBackends/TWslSurface.CairoStop

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure CairoStop; virtual; abstract;
```

DESCRIPTION: Usually called after a call to CairoStart and if the caller is done drawing with Cairo onto the surface. Frees the Cairo context.

SEE ALSO:

WslBackends/TWslSurface.CairoStart
 WslBackends/TWslDriver.ProduceSurface
 WslBackends/TWslDriver.RequestSurface

35 WslBackends/TWslSurface.Create

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
constructor Create(Owner: TObject; const OwnerGUID : ShortString;  

  const CacheId : ShortString; const PixelFormat : TWslPixelFormat;  

  const AsPrimary, AsCairo : Boolean); virtual;
```

DESCRIPTION: This is the constructor which should be overridden and inherited by all backend-specific surface classes. However, the caller never initializes this class directly, since it is abstract. Instances should be produced through TWslDriver (2).ProduceSurface or TWslDriver (2).RequestSurface on an active driver, which will create backend-specific surfaces which you can then access through the methods of the abstract TWslSurface (30) interface.

SEE ALSO:

```
WslBackends/TWslDriver/ProduceSurface
WslBackends/TWslDriver/RequestSurface
```

36 WslBackends/TWslSurface.DecRefCount

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
function DecRefCount(const OwnerGUID : ShortString) : Word;
```

DESCRIPTION: For cachable surfaces only (see TWslDriver (2).RequestSurface). It decreases the reference counter by one. Because the OwnerGUID is required for most reference count changes, only the owning driver can change the reference count.

ARGUMENTS:

OwnerGUID : The GUID of the owning driver without which the reference counter will not be modified.

SEE ALSO:

```
WslBackends/TWslSurface.SetRefCount
WslBackends/TWslSurface.IncRefCount
WslBackends/TWslSurface.Release
WslBackends/TWslDriver.RequestSurface
```

37 WslBackends/TWslSurface.Destroy

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
destructor Destroy; override;
```

DESCRIPTION: The standard destructor, which (in its descendant implementations) frees all low-level resources (SDL, DirectFB, Xlib), which were allocated during the object's lifetime. It need not be explicitly called if, as recommended, surfaces are always created through an active TWslDriver (2), since the driver owns all surfaces and frees them (and the low-level resources) when it dies.

SEE ALSO:

```
WslBackends/TWslDriver.Destroy
```

38 WslBackends/TWslSurface.DoCacheLock

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure DoCacheLock;
```

DESCRIPTION: For cachable surfaces only (see TWslDriver (2).RequestSurface). It sets a hard lock on the whole surface, after which nothing can ever be changed on the pixel array of the surface in the whole of its lifetime. This is implemented because surfaces in the cache pool must not have contents changing under uncontrollable circumstances.

ARGUMENTS:

OwnerGUID : The GUID of the owning driver without which the reference counter will not be modified.

SEE ALSO:

```
WslBackends/TWslSurface.CacheLocked  
WslBackends/TWslDriver.RequestSurface
```

39 WslBackends/TWslSurface.Flip

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure Flip; virtual; abstract;
```

DESCRIPTION: Copies the backbuffer of the surface to the front buffer or actually flips the buffers (if WslBackends (1)/TWslDriver.DoubleBuffer (8) is set to true). This is only required on primary surfaces to make changes visible. In fact, all non-primary surfaces do nothing when their Flip method is called.

NOTES: Currently, true double buffering is only supported with DirectFB as a backend. De facto, it has been disabled, and there is always a copy operation involved in flipping.

SEE ALSO:

```
WslBackends/TWslSurface.UpdateRect  
WslBackends/TWslDriver.DoubleBuffer  
WslBackends/TWslSurface.IsPrimary
```

40 WslBackends/TWslSurface.Flood

[WslBackends] [Methods]

PROTOTYPE:

```
procedure Flood(R, G, B, A : Byte); virtual; abstract;
```

DESCRIPTION: Fills the whole surface with a solid color. Alpha information is used if the surface has an alpha channel. A Flood is recommended for surfaces on which the caller draws herself (as opposed to using the WslDyn or WslPrm subsystems) since some backends will deliver garbage-filled surfaces on create.

ARGUMENTS:

R : Red value [\$0..\$ff] for the fill.
 G : Green value [\$0..\$ff] for the fill.
 B : Blue value [\$0..\$ff] for the fill.
 A : Alpha value [\$0..\$ff] for the fill.

41 WslBackends/TWslSurface.GetPixel(1)

[WslBackends] [Methods]

PROTOTYPE:

```
procedure GetPixel(var APixel : TWslPixel; var AColor : TWslRGB);  
virtual;
```

DESCRIPTION: Retrieves an RGB value for a specific pixel. Pass the result container as the second argument (a var call).

ARGUMENTS:

APixel : A TWslPixel containing the X and Y of the desired pixel.
 AColor : The record which will be filled with the RGB values.

SEE ALSO:

WslBackends/TWslSurface.GetPixel(2)

42 WslBackends/TWslSurface.GetPixel(2)

[WslBackends] [Methods]

PROTOTYPE:

```
procedure GetPixel(var APixel : TWslPixel; var AColor : TWslRGBA); virtual;
```

DESCRIPTION: Retrieves an RGBA value for a specific pixel. Pass the result container as the second argument (a var call).

ARGUMENTS:

APixel : A TWslPixel containing the X and Y of the desired pixel.
 AColor : The record which will be filled with the RGBA values
 (as [\$0..\$ff]).

SEE ALSO:

```
WslBackends/TWslSurface.GetPixel(1)
```

43 WslBackends/TWslSurface.GUID

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslSurface.GUID

TYPE:

```
ShortString
```

DESCRIPTION: Gets the GUID of the surface which is set automatically when the surface is created by the driver. In case of cachable surfaces the GUID is a pseudo-GUID which encodes the identifier by which the surface can be retrieved from the cache pool.

NOTES: This is read-only since it trivially must never change after the surface's create time.

SETTER:

```
(read-only)
```

GETTER:

```
WslBackends/TWslSurface.GetGUID
```

SEE ALSO:

```
WslBackends/TWslDriver.RequestSurface  

WslBackends/TWslDriver.ProduceSurface
```

44 WslBackends/TWslSurface.Height

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslSurface.Height

TYPE:

SmallInt

DESCRIPTION: Retrieves the height of the surface.

SETTER:

WslBackends/TWslSurface.FHeight

GETTER:

WslBackends/TWslSurface.FHeight

SEE ALSO:

WslBackends/TWslSurface.Width

45 WslBackends/TWslSurface.IncRefCount

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
function IncRefCount(const OwnerGUID : ShortString) : Word;
```

DESCRIPTION: For cachable surfaces only (see TWslDriver (2).RequestSurface). It increases the reference counter by one. Because the OwnerGUID is required for most reference count changes, only the owning driver can change the reference count.

ARGUMENTS:

OwnerGUID : The GUID of the owning driver without which the reference counter will not be modified.

SEE ALSO:

WslBackends/TWslSurface.SetRefCount
WslBackends/TWslSurface.DecRefCount
WslBackends/TWslSurface.Release
WslBackends/TWslDriver.RequestSurface

46 WslBackends/TWslSurface.IsPrimary

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslSurface.IsPrimary

TYPE:

Boolean

DESCRIPTION: Retrieves information whether the surface is the primary (which is the one returned by TWslDriver.Start (21)).

SETTER:

(read-only)

GETTER:

WslBackends/TWslSurface.FIsPrimary

SEE ALSO:

WslBackends/TWslDriver.Start

47 WslBackends/TWslSurface.Lock

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure Lock; virtual; abstract;
```

DESCRIPTION: Locks the surface blocking manipulation of the pixel array.

NOTES: What actually happens when the surface gets locked might vary from backend to backend, and it might be that nothing happens at all. It might be wise to call a lock before any Cairo drawing operations and not leave the surface locked for blitting.

SEE ALSO:

WslBackends/TWslSurface.Unlock

48 WslBackends/TWslSurface.PixFmt

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslSurface.PixFmt

TYPE:

TWslPixelFormat

DESCRIPTION: Retrieves the pixel format of the surface.

SETTER:

(read-only)

GETTER:

WslBackends/TWslSurface.FPixelFormat

SEE ALSO:

WslTypes/TWslPixelFormat

49 WslBackends/TWslSurface.Rectangle

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslSurface.Rectangle

TYPE:

TWslRect

DESCRIPTION: Sets/gets the surface-internal convenience rectangle which can be used in blitting. The advantage is that the caller need not malloc and free rectangles and pass them to the blitting functions.

SETTER:

WslBackends/TWslSurface.SetRectangle

GETTER:

WslBackends/TWslSurface.GetRectangle

SEE ALSO:

WslBackends/TWslSurface.RectangleBlit(1)
 WslBackends/TWslSurface.RectangleBlit(2)

50 WslBackends/TWslSurface.RectangleBlit(1)

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure RectangleBlit(OtherSurface : TWslSurface); virtual;
abstract;
```

DESCRIPTION: Performs a blit of the other surface into itself using the part of the source defined in its internal TWslRect. The source rectangle defines the upper left X, Y and the width and height of the part of the source that will be used for blitting. Furthermore, the destination (this surface's rectangle) is used to determine the upper left X, Y of the destination where the blit occurs. Clipping might occur. A source alpha channel is used for alpha surfaces automatically.

ARGUMENTS:

OtherSurface : The source TWslSurface for the blit.

SEE ALSO:

WslBackends/TWslSurface.SimpleBlit
 WslBackends/TWslSurface.SimpleRectangleBlit
 WslBackends/TWslSurface.RectangleBlit(2)
 WslBackends/TWslDriver.ProduceSurface
 WslBackends/TWslDriver.RequestSurface

51 WslBackends/TWslSurface.RectangleBlit(2)

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure RectangleBlit(OtherSurface : TWslSurface;
  const SourceRect, DestRect : TWslRect); virtual; abstract;
```

DESCRIPTION: Like TWslSurface.RectangleBlit(1) (50), except that instead of using the two surfaces' internally defined rectangles, rectangles are passed explicitly.

ARGUMENTS:

OtherSurface : The source TWslSurface for the blit.
 SourcRect : Defines the blit area in source by X, Y, W, H.
 DestRect : Defines the upper left X, Y for blit in destination.

SEE ALSO:

WslBackends/TWslSurface.SimpleBlit
 WslBackends/TWslSurface.SimpleRectangleBlit
 WslBackends/TWslSurface.RectangleBlit(1)
 WslBackends/TWslDriver.ProduceSurface
 WslBackends/TWslDriver.RequestSurface

52 WslBackends/TWslSurface.RefCount

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslSurface.RefCount

TYPE:

Word

DESCRIPTION: Gets the reference count of a surface. If its in [1..65534], the reference count is an actual reference count. If it is 65535, the surface has no actual reference counter and is not in the cache pool. If RefCount<0 or RefCount>65535, please inform the WiSeSLAp developers.

SETTER:

(read-only)

GETTER:

WslBackends/TWslSurface.FRefCount

SEE ALSO:

WslBackends/TWslSurface.SetRefCount
 WslBackends/TWslSurface.IncRefCount
 WslBackends/TWslSurface.DecRefCount
 WslBackends/TWslDriver.RequestSurface
 WslBackends/TWslDriver.ProduceSurface

53 WslBackends/TWslSurface.Release

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure Release;
```

DESCRIPTION: As the only manual reference counter manipulation which can be called on the surface directly (not through the driver), this decrements the reference counter. When it reaches 0, the surface is destroyed automatically and removed from the owning TWslDriver (2).FOwned (TF-PHashObjectList). Non-cachable surfaces always have a reference count of 65535 and are destroyed immediately on the first call of Release.

SEE ALSO:

```
WslBackends/TWslSurface.SetRefCount
WslBackends/TWslSurface.IncRefCount
WslBackends/TWslSurface.DecRefCount
WslBackends/TWslDriver.RequestSurface
WslBackends/TWslDriver.ProduceSurface
WslBackends/TWslDriver.ReleaseSurface(1)
WslBackends/TWslDriver.ReleaseSurface(2)
```

54 WslBackends/TWslSurface.SetRefCount

[WslBackends] [Methods]

PROTOTYPE:

```
function SetRefCount(const OwnerGUID : ShortString;
  Value : Word) : Word;
```

DESCRIPTION: For cachable surfaces only (see TWslDriver (2).RequestSurface). It sets the reference counter to a specific value. Because the OwnerGUID is required for most reference count changes, only the owning driver can change the reference count.

ARGUMENTS:

```
OwnerGUID : The GUID of the owning driver without which the
  reference counter will not be modified.
Value : The new value for the reference counter
```

SEE ALSO:

```
WslBackends/TWslSurface.IncRefCount
WslBackends/TWslSurface.DecRefCount
WslBackends/TWslSurface.Release
WslBackends/TWslDriver.RequestSurface
```

55 WslBackends/TWslSurface.SimpleBlit

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure SimpleBlit(OtherSurface : TWslSurface;  
  const X, Y : SmallInt); virtual; abstract;
```

DESCRIPTION: Performs a blit of the full other surface into itself. Clipping might occur. A source alpha channel is used for alpha surfaces automatically.

ARGUMENTS:

OtherSurface : The source TWslSurface for the blit.
X : Blit upper left corner X in this surface (destination).
Y : Blit upper left corner Y in this surface (destination).

SEE ALSO:

WslBackends/TWslSurface.SimpleRectangleBlit
WslBackends/TWslSurface.RectangleBlit(1)
WslBackends/TWslSurface.RectangleBlit(2)
WslBackends/TWslDriver.ProduceSurface
WslBackends/TWslDriver.RequestSurface

56 WslBackends/TWslSurface.SimpleRectangleBlit

[*WslBackends*] [*Methods*]

PROTOTYPE:

```
procedure SimpleRectangleBlit(OtherSurface : TWslSurface;  
  const PosX, PosY, X, Y, W, H : SmallInt); virtual; abstract;
```

DESCRIPTION: Performs a blit of the other surface into itself using the part of the source defined by X, Y, W, H. These define the upper left X, Y and the width and height of the part of the source that will be used for blitting. The blit occurs at X, Y in the destination (this surface). Clipping might occur. A source alpha channel is used for alpha surfaces automatically.

ARGUMENTS:

OtherSurface : The source TWslSurface for the blit.
PosX : Blit upper left corner X in this surface (destination).
PosY : Blit upper left corner Y in this surface (destination).

X : Upper left corner X of the desired part of the source.
 Y : Upper left corner Y of the desired part of the source.
 W : Width of the desired part of the source.
 H : Height of the desired part of the source.

SEE ALSO:

WslBackends/TWslSurface.SimpleBlit
 WslBackends/TWslSurface.RectangleBlit(1)
 WslBackends/TWslSurface.RectangleBlit(2)
 WslBackends/TWslDriver.ProduceSurface
 WslBackends/TWslDriver.RequestSurface

57 WslBackends/TWslSurface.Unlock

[WslBackends] [Methods]

PROTOTYPE:

```
procedure Unlock; virtual; abstract;
```

DESCRIPTION: Puts the surface is unlocked state. Called after TWslSurface.Unlock and when the caller is done doing things like Cairo drawing.

NOTES: What actually happens when the surface gets locked might vary from backend to backend, and it might be that nothing happens at all. It might be wise to call a lock before any Cairo drawing operations and not leave the surface locked for blitting.

SEE ALSO:

WslBackends/TWslSurface.Lock

58 WslBackends/TWslSurface.UpdateRect

[WslBackends] [Methods]

PROTOTYPE:

```
procedure UpdateRect(const X, Y, W, H : Word); virtual; abstract;
```

DESCRIPTION: Copies an area defined by X, Y, W, H from the surfaces backbuffer to its front buffer. This is only required on primary surfaces to make changes visible if the caller know that only a certain part of the surface has changed (instead of a full Flip).

ARGUMENTS:

X : Upper left X of the area to be copied from back to front.
Y : Upper left Y of the area to be copied from back to front.
W : Width of the area to be copied from back to front.
H : Height of the area to be copied from back to front.

SEE ALSO:

WslBackends/TWslSurface.Flip

59 WslBackends/TWslSurface.Width

[*WslBackends*] [*Properties*]

NAME: WslBackends (I)/TWslSurface.Width

TYPE:

SmallInt

DESCRIPTION: Retrieves the width of the surface.

SETTER:

WslBackends/TWslSurface.FWidth

GETTER:

WslBackends/TWslSurface.FWidth

SEE ALSO:

WslBackends/TWslSurface.Height

60 WslBackends/TWslSurfaceDFB

[*WslBackends*] [*Classes*]

NAME: TWslSurfaceDFB (An implemented WslBackend driver class)

ANCESTOR:

TWslDriver

DESCRIPTION: This class implements its ancestor with DirectFB-specific routines. Never initialize this class directly. Get instances by calling `TWslDriver (2).ProduceSurface` or `TWslDriver (2).RequestSurface`.

SEE ALSO:

```
WslBackends/TWslDriver.ProduceSurface(1)
WslBackends/TWslDriver.ProduceSurface(2)
WslBackends/TWslDriver.RequestSurface(1)
WslBackends/TWslDriver.RequestSurface(2)
```

61 WslBackends/TWslSurfaceSDL

[WslBackends] [Classes]

NAME: `TWslSurfaceSDL` (An implemented `WslBackend` driver class)

ANCESTOR:

```
TWslDriver
```

DESCRIPTION: This class implements its ancestor with SDL-specific routines. Never initialize this class directly. Get instances by calling `TWslDriver (2).ProduceSurface` or `TWslDriver (2).RequestSurface`.

SEE ALSO:

```
WslBackends/TWslDriver.ProduceSurface(1)
WslBackends/TWslDriver.ProduceSurface(2)
WslBackends/TWslDriver.RequestSurface(1)
WslBackends/TWslDriver.RequestSurface(2)
```

62 WslBackends/TWslSurfaceXLIB

[WslBackends] [Classes]

NAME: `TWslDriverXLIB (29)` (An implemented `WslBackend` driver class)

ANCESTOR:

```
TWslDriver
```

DESCRIPTION: This class implements its ancestor with XLIB-specific routines. Never initialize this class directly. Get instances by calling `TWslDriver (2).ProduceSurface` or `TWslDriver (2).RequestSurface`.

SEE ALSO:

```
WslBackends/TWslDriver.ProduceSurface(1)
WslBackends/TWslDriver.ProduceSurface(2)
WslBackends/TWslDriver.RequestSurface(1)
WslBackends/TWslDriver.RequestSurface(2)
```

63 WslBackends/WslCreateDriver

[*WslBackends*] [*Functions*]

PROTOTYPE:

```
function WslCreateDriver(Typ : TWslBackend) : TWslDriver;
```

DESCRIPTION: This function should be used to get a valid driver object which is then accessed by the caller through the abstract TWslDriver interface.

ARGUMENTS:

Typ : A TWslBackend identifier which specifies the requested backend to be initialized.

RESULT:

A backend-specific driver TWslDriver object which the caller should treat as TWslDriver. For low-level access, it can be cast to TWslDriverDFB or similar.

SEE ALSO:

```
WslBackends/TWslDriver
WslBackends/TWslBackend
```

Part II

WiSeSLAp/WslFactory

[*Units*]

NAME: WslFactory

DESCRIPTION: This unit provides functionality for reading, writing, streaming, etc. WiSeSLAP widget trees, their properties, action links, etc. This functionality allows it (in principle) to create a widget tree (the GUI side of an application) from predefined non-procedural definitions, which come from static inline code (normal stand-alone applications), external (XML) files, or (via the future WslNet subsystem) from almost anywhere. NOTE The factory and builder classes defined here must be derived to custom descendants when defining additional components the class definitions of which cannot be seen from WslFactory.pas.

MAINTAINER: Roland Schaefer <mail@rolandschaefer.net>

Part III

WiSeSLAp/WslTypes

[Units]

NAME: WslTypes

DESCRIPTION: A unit which contains many type definitions which should be includable by all other units.

MAINTAINER: Michael Karg <mgor@gmx.net> Roland Schaefer <mail@rolandschaefer.net>

64 WslTypes/TWslPixelFormat

[WslTypes] [Enumerations]

NAME: TWslPixelFormat

DESCRIPTION: For our convenience, we allow only true color modes. The exact format for wpf16 might vary over backends. Most backends will not use wpf16 anyway, but use either RB24 or ARGB32. Furthermore, whether a surface has an alpha channel, is implicit in the pixel format.

VALUES:

wpf16: Non-alpha, most likely RGB16 (5-6-5) or RGB24 (8-8-8).
 wpf24 : Non-alpha, most likely RGB24 (8-8-8).
 wpf32 : Alpha, most likely RGBA32 (8-8-8-8).

SEE ALSO:

WslBackends/TWslDriver.ProduceSurface(2)
 WslBackends/TWslDriver.RequestSurface(2)
 WslBackends/TWslDriver.DefaultPixelFormat

Part IV

WiSeSLAp/WslWidgets

[Units]

NAME: WslWidgets

DESCRIPTION: This unit contains definitions and implementations of all non-custom widgets shipped with WiSeSLAp as well as the TWslApplication class which provides all functionality to run the widgets in an application.

MAINTAINERS: Michael Karg <mgor@gmx.net> Roland Schaefer <mail@rolandschaefer.net>