Gorba

imotion

Infomedia Documentation

Configuration

Technical Description

Version:	2.0
State:	Released
Classification:	Internal use only
Author:	RAN
Creation date:	2014-06-25
Repository:	$/Gorba/Main/Motion/Infomedia/Documents/ TD_InfomediaDocumentation.docx

Gorba AG
Sandackerstrasse
9245 Oberbüren
Switzerland

Table of contents

Modification management

Version	Date	Name	Dept.	Modifications	State
0.1	2014-06-25	RAN	SW	Initial version	draft
1.1	2014-10-23	RAN	SW	Added chapter 5 about example usages	draft
1.2	2014-11-05	RAN	SW	Updated minor changes in complete document based on DEL suggestions. Added chapter 5.2.2	draft

Review

Version	Date	Name	Dept.	Remarks
0.1	2014-06-26	WES	SW	Reviewed, added glossary, minor formatting changes
1.1	2014-10-30	DEL	PH	Reviewed,
1.2	2014-11-10	WES	SW	Reviewed, application icons added.

Release

Version	Date	Name	Dept.	Remarks
1.0	2014-06-26	WES	SW	Release matching Infomedia 2.2.1426
2.0	2014-11-10	WES	SW	Release matching Infomedia 2.4.1445

1 Introduction

1.1 Infomedia Version

This document covers Infomedia version **2.4.1445.**

1.2 Purpose

The goal of this document is to describe the functionality and requirement of each component in Infomedia. It also describes in detail each parameter of every component in detail in order to configure it properly.
At the end of this document, the reader will have all the details to configure every component of Infomedia.

1.3 Intended Audience

This document is addressed to product managers or customer project managers that are familiar with this product and are able to install, operate and maintain it.

2 System Overview

Infomedia is a set of applications which handle the management of the complete presentation and the rendering of the information to different mediums. Each of the component acts as an individual application within the complete system. The components have to be configured individually and started by the System Manager as separate applications.
The various components of Infomedia are:

- Composer
- DirectX Renderer
- Audio Renderer
- AHDLC Renderer

The Composer is a compulsory component to be used when any information is to be presented to the end user. The different Renderers can be used both individually and together in a system based on the requirements. In the following chapters, each of the components is described along with the association to other components within Infomedia.

3 Components of Infomedia

3.1 Composer

The Composer application is part of Infomedia. It handles the management of the complete presentation.

3.1.1 Requirements to run Composer

Composer must be configured to be started and handled by the System Manager. The Composer must have a separate folder for configuration under the "Config" folder and a separate folder for the binaries under "Progs" folder in the TFT system folder structure (refer to TD_TFT2.0System document for details of the folder structure).
Composer requires the presentation file (*.im2) and its associated files/folders to be present in the "Presentation" folder in the TFT system folder structure.

3.1.2 Interaction with other applications

Composer receives information via XIMPLE and translates it to information to be sent to the configured renderers based on the Presentation file (*.im2). Composer can receive the XIMPLE from Protran or any other source.
Composer sends the information to different Renderers if the Renderers are configured to run in System Manager and if they are configured to receive information in the Presentation file.

3.1.3 Shortcuts available

If Composer is started by System Manager, it can be triggered to restart by pressing "r" + Enter in the console window.

3.2 DirectX Renderer

The DirectX Renderer application is part of Infomedia. It handles the rendering of information received from the Composer on the TFT screen. The DirectX Renderer can also be configured to render information from the Composer to multiple TFT screens.
In order to be able to render the information to the TFT screen, the system must be configured in the following files:

- **DirectXRenderer.xml** – Configure the different screens used by the DirectX Renderer.
- **Presentation file (*.im2)** – The Physical screens, Virtual Displays, the resolution of the Layouts must all be configured in order to correctly render the information.

3.2.1 Requirements to run DirectX Renderer

The DirectX Renderer must be configured to be started and handled by the System Manager. The DirectX Renderer must have a separate folder for Configuration under the "Config" folder and a separate folder for the binaries under "Progs" folder in the TFT system folder structure (refer to TD_TFT2.0System document for details of the folder structure).
To use DirectX Renderer on a system other than InfoVision and Inform TFTs, DirectX 9c must be installed. If necessary, please install it from the following URL:

R:\Softwareserver_Release\SW\02_imotion\02_TFT\00_Basic_System (TFT 2.0)\Requirements for Basicsystem TFT\Directx und DotNet

3.2.2 Interaction with other applications

The DirectX Renderer receives the information to be rendered on the TFT screens from the Composer via Medi. The DirectX Renderer informs the Composer on some specific cases, for example, when a video has reached its end, via Medi.

3.2.3 Shortcuts available

3.2.3.1 Alt+Enter

The DirectX Renderer can be configured to show in the following window modes (the details are available in chapter 4.2.1):

- Windowed
- FullScreenWindowed
- FullScreenExclusive

Use Alt+Enter to toggle between the different window modes.
If the TFT screen cannot be shown in FullScreenExclusive, a message is shown.

3.2.3.2 F1

Press "F1" to toggle displaying the following information about the rendered screen:

- Screen adapter
- Screen resolution
- Screen coordinates
- The window mode
- The current frame rate in FPS

3.3 Audio Renderer

The Audio Renderer application is part of Infomedia. It handles the rendering of information received from the Composer onto an audio device like a speaker. The Audio Renderer can play MP3 audio files and uses Acapela to output text-to-speech (TTS). The Presentation file (*.im2) can be configured to select the audio files to be used, the text-to-speech voice, the amount of time to Pause between different audios etc.
In order to be able to render the information to an audio device, the system must be configured in the following files.

- **AudioRenderer.xml** – Configure the different IO ports to be used to control the volume and the Speaker operation (turn on or off). Configuration for the text-to-speech operation.
- **HardwareManager.xml** - Configure the IO pins on the hardware to be used for the volume and Speaker operation.
- **Presentation file (*.im2)** – The Physical screen, Virtual Display and the Layouts must all be configured in order to correctly render the information.

3.3.1 Requirements to run Audio Renderer

The Audio Renderer must be configured to be started and handled by the System Manager. The Audio Renderer must have a separate folder for Configuration under the "Config" folder and a separate folder for the binaries under "Progs" folder in the TFT system folder structure (refer to TD_TFT2.0System document for details of the folder structure).
If the Audio Renderer is configured to use Acapela for text-to-speech, then the binaries for Acapela must be placed in the exact location as configured in the AudioRenderer.xml.

3.3.2 Interaction with other applications

The Audio Renderer receives the information to be rendered on the audio device like a speaker from the Composer via Medi.

3.3.3 Shortcuts available

If Audio Renderer is started by System Manager, it can be triggered to restart by pressing "r" + Enter in the console window.

3.4 AHDLC Renderer

The AHDLC Renderer application is part of Infomedia. It handles the rendering of information received from the Composer to different Gorba LED signs. The AHDLC Renderer can render information to multiple LED signs at once.
In order to be able to render the information to different LED signs, the system must be configured in the following files.

- **AhdlcRenderer.xml** – Configure the serial port, the connected LED signs with the correct addresses, the type of sign and their resolutions.
- **Presentation file (*.im2)** – Each LED sign requires a Physical Screen, Virtual Display and its associated configuration. The resolution of the LED signs here must match the resolution configured in the AhdlcRenderer.xml.

3.4.1 Requirements to run AHDLC Renderer

The AHDLC Renderer must be configured to be started and handled by the System Manager. The AHDLC Renderer must have a separate folder for Configuration under the "Config" folder and a separate folder for the binaries under "Progs" folder in the TFT system folder structure (refer to TD_TFT2.0System document for details of the folder structure).

3.4.2 Interaction with other applications

The AHDLC Renderer receives the information to be rendered on the LED signs from the Composer via Medi.

3.4.3 Shortcuts available

If AHDLC Renderer is started by System Manager, it can be triggered to restart by pressing "r" + Enter in the console window.

4 Configuration

4.1 Composer

The Composer requires configuration in order to operate correctly with the selected hardware. The file for configuration is **"Composer.xml"**. Composer also requires "**NLog.config**" for logging and **"medi.config"** for the Medi configuration.

4.1.1 Example Configuration

This chapter was automatically generated from Composer.xml on 2014/11/10 by WES.

```
<?xml version="1.0" encoding="utf-8"?>
<Composer xsi="http://www.w3.org/2001/XMLSchema-instance" schemaLocation="file:///. Composer.xsd">
  <EnableTimeSync>false</EnableTimeSync>
  <XimpleInactivity Timeout="PT60S" AtStartup="true" />
</Composer>
```

Figure 1 Example Composer.xml

4.1.2 XML Structure

This chapter was automatically generated from Composer.xsd on 2014/11/10 by WES.
The root element is ComposerConfig.

4.1.2.1 ComposerConfig Structure

<EnableTimeSync>	0:1	string	Indicates whether time sync using Ximple should be enabled. If this is set to true, composer will change the system date and time (using Hardware Manager) to the date and time sent through Ximple (table SystemStatus, columns Date and Time).
<XimpleInactivity>	1:1	XimpleInactivityConfig	Defines the timeout behavior when not receiving Ximple data from Protran.

Table 1 XML elements inside ComposerConfig

Contains the entire configuration of the Infomedia Composer application.

4.1.2.2 XimpleInactivityConfig Structure

Timeout=	0:1	string, default: "PT60S"	Defines the amount of time no Ximple data was received from Protran before the remote computer status (table SystemStatus, column RemotePC) is set to 0.
AtStartup=	0:1	string, default: "false"	If this flag is set to true, the remote computer status (table SystemStatus, column RemotePC) is set to 0 at the start-up of Composer and only set to 1 once valid Ximple data was received.

Table 2 XML attributes of XimpleInactivityConfig

4.2 DirectX Renderer

The DirectX Renderer requires configuration in order to operate correctly with the selected hardware. The file for configuration is **"DirectXRenderer.xml"**. DirectX Renderer also requires "**NLog.config**" for logging and **"medi.config"** for the Medi configuration.

4.2.1 Example Configuration

This chapter was automatically generated from DirectXRenderer.xml on 2014/11/10 by WES.

```
<?xml version="1.0" encoding="utf-8"?>
<DirectXRenderer xsi="http://www.w3.org/2001/XMLSchema-instance" schemaLocation="file:///.
DirectXRenderer.xsd">
  <WindowMode>Windowed</WindowMode>
  <Screens>
    <!-- leave this tag empty for default behavior (running on Topbox) -->
    <Screen Adapter="0" Width="1920" Height="1080">
      <VisibleRegion Width="1368" Height="768" />
      <FallbackImage>D:\Presentation\Images\Fallback.jpg</FallbackImage>
    </Screen>
    <!--     define single screen with the default resolution:     <Screen />     -->
    <!--     define a special resolution on a single screen:     <Screen Adapter="0" Width="1440"
Height="900" />     -->
    <!--     define a special id for a single screen (otherwise Id = Adapter):     <Screen Adapter="0"
Id="MySpecialScreen" Width="1440" Height="900" />     -->
    <!--     define a special resolution on two screens:     <Screen Adapter="0" Width="1280" Height="800"
/>     <Screen Adapter="1" Width="1280" Height="800" />     -->
  </Screens>
  <Device>
    <MultiSample>None</MultiSample>
    <MultiSampleQuality>0</MultiSampleQuality>
  </Device>
  <Text>
    <!-- Possible TextMode values: Font, FontSprite, Gdi* (*= experimental only!) -->
    <TextMode>FontSprite</TextMode>
    <!-- Possible FontQuality values: Default, Draft, Proof, NonAntiAliased, AntiAliased, ClearType,
ClearTypeNatural -->
    <FontQuality>AntiAliased</FontQuality>
  </Text>
  <Image>
    <!-- 3600 = 1 hour -->
    <BitmapCacheTimeout>3600</BitmapCacheTimeout>
    <!-- 52428800 = 50 MB -->
    <MaxBitmapCacheBytes>52428800</MaxBitmapCacheBytes>
    <!-- 1000000 = 500x500 pixels (x4 bytes) -->
    <MaxCacheBytesPerBitmap>1000000</MaxCacheBytesPerBitmap>
  </Image>
  <Video>
    <!-- Possible VideoMode values: DirectShow, DirectXWindow, VlcWindow -->
    <VideoMode>DirectXWindow</VideoMode>
  </Video>
</DirectXRenderer>
```

Figure 2 Example DirectXRenderer.xml

4.2.2 XML Structure

This chapter was automatically generated from DirectXRenderer.xsd on 2014/11/10 by WES.

The root element is RendererConfig.

4.2.2.1 RendererConfig Structure

<WindowMode>	0:1	WindowMode, default: "FullScreenExclusive"	Defines the window mode in which to start the DirectX Renderer.
<FallbackTimeout>	0:1	duration, default: "PT30S"	The timeout after which the Fallback Screen (see below) or the splash screen is shown if no data is received from the Composer. A value of PT0S disables the fallback timeout. Do not set this value below 20 seconds since the keep-alive time of Composer is set to 15 seconds and you would see the screen constantly switching between fallback and regular content.
<Screens>	1:1	List of <Screen>	Definition of the screens to be used by the Renderer. If this element is left empty, the Renderer will automatically detect all screens attached to the system and show a render window on each of them.
<Device>	1:1	DeviceConfig	DirectX device specific settings. Do not change these settings without approval from Software Development.
<Text>	1:1	TextConfig	Settings related to the display of text. Do not change these settings without approval from Software Development.
<Image>	1:1	ImageConfig	Settings related to the display of images and the caching of images in RAM.
<Video>	1:1	VideoConfig	Settings related to the display of videos.

Table 3 XML elements inside RendererConfig

Configuration structure for Infomedia DirectX Renderer.

4.2.2.2 WindowMode Enumeration

Windowed	The window(s) is/are shown in a normal OS window that can be resized and moved. This mode should only be used for testing and not on productive configurations.
FullScreenExclusive	This mode is only possible if only a single screen is used. The window is in fullscreen on the given screen. If this mode is set while the renderer is configured to show multiple screens, it will automatically fall back to FullScreenWindowed mode. This mode is the most efficient mode as DirectX is directly handling the graphics buffer, but it is not compatible with the video modes DirectXWindow and VlcWindow (see below). Use this mode whenever you are not showing videos.
FullScreenWindowed	The window(s) is/are shown in fullscreen on the respective screens, but DirectX draws to a fully extended, frameless window. This mode is considerably slower than FullScreenExclusive but is the only option when using multiple screens and/or video modes DirectXWindow or VlcWindow.

Table 4 Enumeration values of WindowMode

Mode how the render window(s) is/are shown at start-up. The mode of the render window can also be changed at run-time by pressing Alt-Enter. The mode will cycle through the possible options.

4.2.2.3 ScreenConfig Structure

Adapter=	0:1	int	The adapter index (starting from 0) as the screens are defined in the operating system. If this attribute is omitted, the screens get their adapter index from the position in the config file.
Id=	0:1	string	The identifier for this screen towards Composer. This can be used to address same-size screens on the same adapter on different systems. If this attribute is omitted, the ID will be the same as the Adapter index.
Width=	0:1	int	The native width of the screen in pixels. This should always match the physical width of the screen (as well as the one configured in the OS settings). If this attribute is omitted (default), the width is determined from the adapter.
Height=	0:1	int	The native height of the screen in pixels. This should always match the physical height of the screen (as well as the one configured in the OS settings). If this attribute is omitted (default), the height is determined from the adapter. Important: for wide-screen, set this to 1080 since the screen is natively 1920x1080 even thought only 1920x630 pixels are visible. See VisibleRegionConfig for more information.

Table 5 XML attributes of ScreenConfig

<VisibleRegion>	0:1	VisibleRegionConfig	The area where the content of the DirectX Renderer is shown. If this element is not set (default), the entire screen size is used. Only set this for special cases when the Renderer should only use part of the screen (e.g. wide-screen).
<FallbackImage>	0:1	string	Relative or absolute path to the image to be shown on that screen in fallback mode (no data is received from Composer). See also FallbackTimeout above. If this element is not specified or the given file doesn't exist, the DirectX Renderer splash screen will be shown instead.

Table 6 XML elements inside ScreenConfig

Configuration of a single screen used by the Renderer.

4.2.2.4 VisibleRegionConfig Structure

X=	0:1	int	The horizontal position where the image should be rendered on the physical screen. Do not use this attribute without approval from Software Development. If this attribute is omitted (default), the image is rendered starting from the very left (x=0).
Y=	0:1	int	The vertical position where the image should be rendered on the physical screen. Do not use this attribute without approval from Software Development. If this attribute is omitted (default), the image is rendered starting from the very top (x=0).
Width=	0:1	int	The width of the region where the image should be rendered on the physical screen. Do not use this attribute without approval from Software Development. If this attribute is omitted (default), the image is rendered on the entire screen.
Height=	0:1	int	The width of the region where the image should be rendered on the physical screen. This parameter can be used e.g. on wide-screen TFTs that have a vertical resolution of 1080 pixels, but only the top most 630 pixels are visible (in this example this Height would be 630). If this attribute is omitted (default), the image is rendered on the entire screen.

Table 7 XML attributes of VisibleRegionConfig

4.2.2.5 DeviceConfig Structure

<SwapEffect>	0:1	SwapEffect, default: "Discard"	Definition how buffers are swapped.
<MultiThreaded>	0:1	boolean, default: "true"	Flag to set whether DirectX should be multi-threading safe or not.
<MultiSample>	0:1	MultiSampleType, default: "None"	The sampling mode.
<MultiSampleQuality>	0:1	int, default: "0"	The sampling quality.
<PresentationInterval>	0:1	PresentInterval, default: "One"	Value indicating how often the screen should be refreshed.
<PresentFlag>	0:1	PresentFlag, default: "None"	DirectX presentation flags.

Table 8 XML elements inside DeviceConfig

DirectX device specific settings. Do not change these settings without approval from Software Development.

4.2.2.6 SwapEffect Enumeration

Discard	The buffers are not swapped, but simply discarded.
Copy	The buffers are copied.
Flip	The buffers are flipped.

Table 9 Enumeration values of SwapEffect

Definition how buffers are swapped.

4.2.2.7 MultiSampleType Enumeration

SixteenSamples	Enables 16 levels of full-scene multisampling.
FifteenSamples	Enables 15 levels of full-scene multisampling.
FourteenSamples	Enables 14 levels of full-scene multisampling.
ThirteenSamples	Enables 13 levels of full-scene multisampling.
TwelveSamples	Enables 12 levels of full-scene multisampling.
ElevenSamples	Enables 11 levels of full-scene multisampling.
TenSamples	Enables 10 levels of full-scene multisampling.
NineSamples	Enables 9 levels of full-scene multisampling.
EightSamples	Enables 8 levels of full-scene multisampling.
SevenSamples	Enables 7 levels of full-scene multisampling.
SixSamples	Enables 6 levels of full-scene multisampling.
FiveSamples	Enables 5 levels of full-scene multisampling.
FourSamples	Enables 4 levels of full-scene multisampling.
ThreeSamples	Enables 3 levels of full-scene multisampling.
TwoSamples	Enables 2 levels of full-scene multisampling.
NonMaskable	Enables the multisample quality value.
None	Specifies no full-scene multisampling.

Table 10 Enumeration values of MultiSampleType

The sampling mode.

4.2.2.8 PresentInterval Enumeration

Default	Uses the default system timer resolution.
Immediate	The runtime updates the window client area immediately and might do so more than once during the adapter refresh period.
One	The driver will wait for the vertical retrace period. Present operations will not be affected more frequently than the screen refresh; the runtime will complete at most one Present operation per adapter refresh period.
Two	The driver will wait for the vertical retrace period. Present operations will not be affected more frequently than every second screen refresh.
Three	The driver will wait for the vertical retrace period. Present operations will not be affected more frequently than every third screen refresh.
Four	The driver will wait for the vertical retrace period. Present operations will not be affected more frequently than every fourth screen refresh.

Table 11 Enumeration values of PresentInterval

Screen refresh interval.

4.2.2.9 TextConfig Structure

<TextMode>	0:1	TextMode, default: "FontSprite"	Mode in which text is rendered.
<FontQuality>	0:1	FontQuality, default: "AntiAliased"	Quality of the font rendering.

Table 12 XML elements inside TextConfig

Settings related to the display of text. Do not change these settings without approval from Software Development.

4.2.2.10 TextMode Enumeration

Font	The text is rendered using DirectX fonts but without a sprite caching the rendered triangles.
FontSprite	The text is rendered using DirectX fonts using a sprite to cache the rendered triangles.
Gdi	The text is rendered using GDI to a bitmap which is then rendered using a texture. This mode is only experimental and should only be used when rotated fonts are required.

Table 13 Enumeration values of TextMode

Modes in which text can be rendered.

4.2.2.11 FontQuality Enumeration

Default	The font appearance does not matter.
ClearTypeNatural	Text is rendered using the Microsoft ClearType antialiasing (smoothing) method when possible. The font quality is given more importance than maintaining the text size; thus, the text width can change.
ClearType	Text is rendered using the ClearType antialiasing method when possible. The font quality is given less importance than maintaining the text size.
AntiAliased	Text is rendered using the antialiasing method when possible. The font quality is given less importance than maintaining the text size.
NonAntiAliased	The font is never antialiased.
Proof	Font's character quality is given more importance than exact matching of the logical-font attributes. For Windows Graphics Device Interface (GDI) raster fonts, scaling is disabled and the font closest in size is chosen. Although this size might not correspond exactly when Proof is used, the font's quality is high and its appearance is not distorted. Bold, italic, underline, and strikeout fonts are synthesized if necessary.
Draft	The font appearance is given less importance than when the Proof value is used. Scaling is enabled for Microsoft Windows Graphics Device Interface (GDI) raster fonts, which means that more font sizes are available, but their quality might be lower. Bold, italic, underline, and strikeout fonts are synthesized if necessary.

Table 14 Enumeration values of FontQuality

Quality of the font rendering.

4.2.2.12 ImageConfig Structure

<BitmapCacheTimeout>	0:1	string, default: "PT1H"	Bitmap cache timeout. A bitmap is kept unused in a RAM cache at maximum that amount of time.
<MaxBitmapCacheBytes>	0:1	long, default: "52428800"	Maximum size of the bitmap cache (in bytes). This is the approximate maximum amount of RAM used by DirectX Renderer to cache images. Once that limit is reached, the oldest (unused) bitmaps are removed from the cache. The default value is 50 MB.
<MaxCacheBytesPerBitmap>	0:1	int, default: "1000000"	Maximum size of a bitmap (in bytes) to keep in RAM. The size of the bitmap is simply calculated by width x height x 4 (32 bit color). If an image is bigger than the given attribute, the image is never kept in the cache. The default value is equivalent to 500 x 500 pixels, this means that many smaller images are cached, but background and other full-screen images are always loaded from disk.

Table 15 XML elements inside ImageConfig

Settings related to the display of images and the caching of images in RAM. DirectX Renderer will keep a certain amount of images in a RAM cache so it doesn't have to reload the images every time

from the disk when using them. The three parameters define how that caching is done and when cached bitmaps are discarded to free up memory.

4.2.2.13 VideoConfig Structure

<VideoMode>	0:1	VideoMode, default: "DirectShow"	The mode how videos are shown by DirectX Renderer.

Table 16 XML elements inside VideoConfig

Settings related to the display of videos.

4.2.2.14 VideoMode Enumeration

DirectShow	The renderer uses DirectShow on a DirectX texture. This mode allows for full-screen exclusive rendering as well as rendering other elements (images, text, ...) on top of a video. This mode does not support streaming videos. This mode is considerably slower than the other modes and should only be used in the above mentioned cases.
DirectXWindow	The video is shown in a borderless window on top of the rendering window. This mode is not compatible with full-screen exclusive mode. This mode does not support streaming videos. It is also not possible to draw on top of this video; other graphical elements will always appear below the video, independent of their relaitve z-index.
VlcWindow	The video is shown in a borderless window on top of the rendering window. The video is rendered using VideoLAN VLC media player; using this mode requires VLC to be installed on the target system. This mode is not compatible with full-screen exclusive mode. This mode supports streaming videos. It is also not possible to draw on top of this video; other graphical elements will always appear below the video, independent of their relaitve z-index.

Table 17 Enumeration values of VideoMode

The mode how videos are shown by DirectX Renderer.

4.3 Audio Renderer

The Audio Renderer requires configuration in order to operate correctly with the selected hardware. The file for configuration is **"AudioRenderer.xml"**. Audio Renderer also requires "**NLog.config**" for logging and **"medi.config"** for the Medi configuration.

4.3.1 Example Configuration

This chapter was automatically generated from AudioRenderer.xml on 2014/11/10 by WES.

```
<?xml version="1.0" encoding="utf-8"?>
<AudioRenderer xsi="http://www.w3.org/2001/XMLSchema-instance" schemaLocation="file:///. AudioRenderer.xsd">
  <IO>
    <VolumePort Name="SystemVolume" />
  </IO>
  <AudioChannels>
    <AudioChannel Id="1">
      <SpeakerPort Name="Speaker" />
      <SpeakerPort Name="Speaker1" />
    </AudioChannel>
    <AudioChannel Id="2">
      <SpeakerPort Name="Speaker1" />
    </AudioChannel>
  </AudioChannels>
  <TextToSpeech>
    <API>Acapela</API>
    <HintPath>D:\Progs\Acapela</HintPath>
  </TextToSpeech>
</AudioRenderer>
```

Figure 3 Example AudioRenderer.xml

4.3.2 XML Structure

This chapter was automatically generated from AudioRenderer.xsd on 2014/11/10 by WES.

The root element is AudioRendererConfig.

4.3.2.1 AudioRendererConfig Structure

<IO>	1:1	IOConfig	Configuration how digital I/O are being used by the Renderer.
<AudioChannels>	0:1	List of <AudioChannel>	List of all audio channels used by the renderer.
<TextToSpeech>	0:1	TextToSpeechConfig	Configuration of the text-to-speech component of Audio Renderer.

Table 18 XML elements inside AudioRendererConfig

The Audio Renderer configuration.

4.3.2.2 IOConfig Structure

<VolumePort>	0:1	IOPortConfig	If this element is present, it defines which GIOoM port to use to set the volume before playing any sounds. The volume to be set is configured in the .im2 file. The configured I/O needs to be writable, of type integer and have (at least) a range of 0..100. Usually the local "Volume" port from Hardware Manager is configured in this I/O.
<SpeakerPort>	0:1	IOPortConfig	If this element is present, it defines which GIOoM port to use to enable and disable the audio output. The port is enabled before playing any sounds and disabled once all playback has completed. The configured I/O needs to be writable and of type boolean. Usually this is a GPIO connected to an amplifier which is handled by Hardware Manager through one of the pins.

Table 19 XML elements inside IOConfig

Configuration how digital I/O are being used by the Renderer.

4.3.2.3 IOPortConfig Structure

Unit=	0:1	string	The unit name where to find the output (e.g. "TFT-01-23-45"). If this attribute is not set, the output will be searched on the local system only.
Application=	0:1	string	The application name where to find the output (e.g. "HardwareManager"). If this attribute is not set, the output will be searched in all applications.
Name=	1:1	string	The exact name of the GIOoM output (e.g. "Volume").

Table 20 XML attributes of IOPortConfig

4.3.2.4 AudioChannelConfig Structure

Id=	0:1	string	The identifier for the audio channel towards Composer. This can be used to address the correct speaker port on the hardware.

Table 21 XML attributes of AudioChannelConfig

<SpeakerPort>	0:*	IOPortConfig	List of all speaker ports available on this audio channel. If this element is present, it defines which GIOoM port to use to enable and disable the audio output. The port is enabled before playing any sounds and disabled once all playback has completed. The configured I/O needs to be writable and of type boolean. Usually this is a GPIO connected to an amplifier which is handled by Hardware Manager through one of the pins.

Table 22 XML elements inside AudioChannelConfig

Configuration of an audio channel. A channel must have unique id. It can have multiple speaker ports configured on it.

4.3.2.5 TextToSpeechConfig Structure

<API>	1:1	TextToSpeechApi	Defines which API (vendor) to use for text-to-speech.
<HintPath>	0:1	string	The hint path where text-to-speech engine related files are located. By default each engine searches in the most usual places for its file; this setting allows you to search first in the given path. For Acapela, this path is used to find the AcaTTS.dll.

Table 23 XML elements inside TextToSpeechConfig

Configuration of the text-to-speech component of Audio Renderer.

4.3.2.6 TextToSpeechApi Enumeration

Microsoft	The standard Microsoft TTS API from System.Speech.Synthesis.
Acapela	The Acapela API from Acapela Group.

Table 24 Enumeration values of TextToSpeechApi

Defines the possible APIs (vendors) usable for text-to-speech.

4.4 AHDLC Renderer

The AHDLC Renderer requires configuration in order to operate correctly with the selected hardware. The file for configuration is **"AhdlcRenderer.xml"**. Ahdlc Renderer also requires "**NLog.config**" for logging and **"medi.config"** for the Medi configuration.

4.4.1 Example Configuration

This chapter was automatically generated from AhdlcRenderer.xml on 2014/11/10 by WES.

```
<?xml version="1.0" encoding="utf-8"?>
<AhdlcRenderer xsi="http://www.w3.org/2001/XMLSchema-instance" schemaLocation="file:///. AhdlcRenderer.xsd">
  <Channel>
    <SerialPort>
      <ComPort>COM20</ComPort>
      <!-- Possible values: {1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200} Def. 38400-->
      <BaudRate>38400</BaudRate>
      <!-- Possible values: {5, 6, 7, 8} Def. 8 -->
      <DataBits>8</DataBits>
      <!-- Possible values: {None, One, Two, OnePointFive} Def. Two -->
      <StopBits>Two</StopBits>
      <!-- Possible values: {Odd, Even, Mark, Space, None} (case insensitive) Def. None -->
      <Parity>None</Parity>
      <!-- Possible values: { Default, Auto, Enabled, Disabled, EnableForTx, DisableForTx } -->
      <RTSMode>Default</RTSMode>
      <!-- Possible values: true (default, used for AHDLC over RS-485), false (used for AHDLC over Alphabus)
-->
      <IsHighSpeed>false</IsHighSpeed>
      <!-- Possible values: false (default), true -->
      <IgnoreFrameStart>false</IgnoreFrameStart>
      <!-- Possible values: false (default, wait for answers from signs), true (ignore answers and just send
data) -->
      <IgnoreResponses>false</IgnoreResponses>
      <!-- Possible values: positive integer, unit: milliseconds  -->
      <IgnoreResponseTime>10</IgnoreResponseTime>
    </SerialPort>
    <Signs>
      <Sign Address="1" Mode="Monochrome" Width="112" Height="16" />
      <Sign Address="7" Mode="Text" Width="96" Height="8" />
      <Sign Address="13" Mode="Color" Width="34" Height="20" />
    </Signs>
  </Channel>
</AhdlcRenderer>
```

Figure 4 Example AhdlcRenderer.xml

4.4.2 XML Structure

This chapter was automatically generated from AhdlcRenderer.xsd on 2014/11/10 by WES.
The root element is AhdlcRendererConfig.

4.4.2.1 ChannelConfig Structure

<SerialPort>	1:1	SerialPortConfig	Configuration of the serial (COM) port used by this channel.
<Signs>	1:1	List of <Sign>	List of all signs connected to this channel.

Table 25 XML elements inside ChannelConfig

Configuration of a channel. A channel is a single serial port connected to one or more signs. All signs on this channel must have unique addresses. This allows to use the AHDLC Renderer with multiple RS485 connections on a single unit. Usually there is exactly one channel configured.

4.4.2.2 SerialPortConfig Structure

<ComPort>	1:1	string	The serial port's name, e.g. "COM1".
<BaudRate>	0:1	int, default: "38400"	The baud rate to be used on the serial port.
<DataBits>	0:1	int, default: "8"	The number of data bits to be used on the serial port.
<StopBits>	0:1	StopBits, default: "Two"	The number of stop bits to be used on the serial port.
<Parity>	0:1	Parity, default: "None"	The parity to be used on the serial port.
<RTSMode>	0:1	RtsMode, default: "Default"	The behavior how to set the serial port's RTS signal when transmitting data.
<IsHighSpeed>	0:1	boolean, default: "true"	Flag indicating if the port is using AHDLC high-speed mode. If this flag is set to true, block numbers will occupy 2 bytes. Set this to true for signs that are running over RS-485 and to false if the signs are connected via an Alphabus converter.
<IgnoreFrameStart>	0:1	boolean, default: "false"	Flag value indicating whether to ignore the frame start. If the frame start is ignored, the AHDLC Renderer will still decode a frame even if its start marker is missing. This can be used in cases where the first bit(s) of the response are crippled because of timing issues.
<IgnoreResponses>	0:1	boolean, default: "false"	Flag indicating whether to ignore responses. If responses are ignored, the AHDLC Renderer will just send data (estimating the wait time for possible responses, but not reading any responses).
<IgnoreResponseTime>	0:1	positiveInteger, default: "10"	Time to wait in milliseconds when IgnoreResponses is set to true. This wait time will always be added to the estimated wait time.

Table 26 XML elements inside SerialPortConfig

Configuration of a serial (COM) port.

4.4.2.3 StopBits Enumeration

None	No stop bits are used.
One	One stop bit is used.
OnePointFive	1.5 stop bits are used.
Two	Two stop bits are used.

Table 27 Enumeration values of StopBits

Specifies the number of stop bits used on the serial port.

4.4.2.4 Parity Enumeration

None	No parity check occurs.
Odd	Sets the parity bit so that the count of bits set is an odd number.
Even	Sets the parity bit so that the count of bits set is an even number.
Mark	Leaves the parity bit set to 1.
Space	Leaves the parity bit set to 0.

Table 28 Enumeration values of Parity

Specifies the parity bit for a serial port.

4.4.2.5 RtsMode Enumeration

Default	The default behavior for the given hardware platform. This is "EnableForTx" on InfoVision TFT products and "Auto" for VM.cu.
Auto	The RTS line is automatically toggled by the hardware. This is currently only supported on VM.cu.
Enabled	The RTS line is enabled at startup and never changed thereafter.
Disabled	The RTS line is disabled at startup and never changed thereafter.
EnableForTx	The RTS line is disabled at startup, then enabled by the software before sending data and disabled again after sending data.
DisableForTx	The RTS line is enabled at startup, then disabled by the software before sending data and enabled again after sending data.

Table 29 Enumeration values of RtsMode

The behavior how to set the serial port's RTS signal when transmitting data.

4.4.2.6 SignConfig Structure

Id=	0:1	string	The identifier for this sign towards Composer. This can be used to address same-size screens on the address on different systems. If this attribute is omitted, the ID will be the same as the address below.
Address=	1:1		The AHDLC address (1..15) of the sign.
Mode=	1:1	SignMode	The mode in which the sign is to be used.
Width=	1:1	int	The width in pixels of the sign (usually a multiple of 14 for monochrome exterior signs). This attribute has to be specified but is ignored for interior signs.
Height=	1:1	int	The height in pixels of the sign (usually a multiple of 8 for monochrome exterior signs). This attribute has to be specified but is ignored for interior signs.
Brightness=	0:1	Brightness, default: "Default"	The brightness with which to display content on the sign. This attribute is only considered for monochrome exterior signs.

Table 30 XML attributes of SignConfig

Configuration of a single exterior or interior sign connected to a serial port.

4.4.2.7 SignMode Enumeration

Monochrome	The sign should be sent monochrome bitmaps only. In this mode, colors are converted to black/white where anything not black is considered white. Use this mode for monochrome exterior signs.
Color	The sign should be sent color bitmaps only. In this mode, no scrolling texts are supported. Use this mode for color exterior signs.
Text	The sign should be sent texts only. In this mode, no bitmaps are supported and text is aligned by the sign, not according to the values defined in the .im2 file. Use this mode for interior signs.

Table 31 Enumeration values of SignMode

The mode in which a sign is to be used.

4.4.2.8 Brightness Enumeration

Default	The default value (0).
Brightness1	The first brightness level (darkest, 10).
Brightness2	The second brightness level (11).
Brightness3	The third brightness level (brightest, 12).

Table 32 Enumeration values of Brightness

The brightness of a Gorba exterior sign.

5 Example usages

5.1 DirectX Renderer example usages

5.1.1 Define visible region on the screen example usage

In the DirectXRenderer.xml, the user can define a screen and only make a certain region of it visible. In such a configuration, the image, text etc rendered by the DirectX Renderer is shown only in the visible region. The X and Y allow the user to specify the offset of the visible region. Below is an example for such a configuration.

```
<Screen Adapter="0" Width="1920" Height="1080">
  <VisibleRegion X="0" Y="0" Width="1920" Height="630"/>
  <FallbackImage>..\..\Presentation\Images\Fallback.jpg</FallbackImage>
</Screen>
```

The example shown above is for 29' HD display in which only the **Height** attribute is modified to value 630 in the **VisibleRegion**.

5.2 Audio Renderer example usages

5.2.1 Speaker setup example usages

The Audio Channel configuration allows the user to configure the Speaker port for a specific Hardware output. The configuration between the Hardware Manager and the Audio Renderer is given in an example in the TD_HardwareManager.pdf document.
The second part to setup the Speaker to be switched on at a specific time is to be defined in both the **AudioRenderer.xml** and the ***.im2** file used by the Composer application.
For each Physical Screen of type "Audio" configured in the *.im2, an "Audio Channel" must be configured in the AudioRenderer.xml.
The Audio Channel and the Physical Screen are mapped to each other by the "Id".

Below is an example of *.im2 with three Physical Screens of type Audio defined.

```
<PhysicalScreens>
  <PhysicalScreen Name="AudioScreen1" Type="Audio" Id="1" Width="0" Height="0" />
  <PhysicalScreen Name="AudioScreen2" Type="Audio" Id="2" Width="0" Height="0" />
  <PhysicalScreen Name="AudioScreen3" Type="Audio" Id="3" Width="0" Height="0" />
</PhysicalScreens>
```

Below is an example of the AudioRenderer.xml with three Audio Channels configured to work with the above *.im2.

```
<IO>
  <VolumePort Name="SystemVolume" />
</IO>
<AudioChannels>
  <AudioChannel Id="1">
    <SpeakerPort Name="SpeakerOne" />
  </AudioChannel>
  <AudioChannel Id="2">
    <SpeakerPort Name="SpeakerTwo" />
  </AudioChannel>
  <AudioChannel Id="3">
    <SpeakerPort Name="SpeakerOne" />
    <SpeakerPort Name="SpeakerTwo" />
  </AudioChannel>
</AudioChannels>
```

When a cycle for Physical Screen with Id = 1 is active/enabled, the audio configured for that cycle is played on SpeakerOne which is configured under AudioChannel with Id = 1.
When a cycle for Physical Screen with Id = 3 is active/enabled, the audio configured for that cycle is played on SpeakerOne and SpeakerTwo simultaneously which are configured under AudioChannel with Id = 3.

5.2.2 Acapela example usage

If the Audio Renderer is configured to use Acapela for text-to-speech, then the binaries for Acapela must be placed in the exact location as configured in the AudioRenderer.xml as seen in the example below.

```
<TextToSpeech>
  <API>Acapela</API>
  <HintPath>..\..\Progs\Acapela</HintPath>
</TextToSpeech>
```

As seen in the example above, the <API> must be configured as "Acapela" and the <HintPath> must be the location where the Acapela binaries are placed on the TFT system. In this example, the binaries are placed within the Progs directory in the directory called Acapela.

5.3 AHDLC Renderer example usages

5.3.1 Signs example usage

Each sign that is configured in the AHDLC Renderer must have an associated Physical Screen of type LED configured in the *.im2 file. The "Address" attribute of the Sign is mapped to the "Id" of the Physical Screen.
Below is an example of *.im2 with three Physical Screens of type Audio defined.

```
<PhysicalScreens>
  <PhysicalScreen Name="FrontLED" Type="LED" Id="1" Width="98" Height="16" />
  <PhysicalScreen Name="SideLED" Type="LED" Id="2" Width="98" Height="16" />
  <PhysicalScreen Name="ColorLED" Type="LED" Id="12" Width="34" Height="20" />
</PhysicalScreens>
```

Below is an example of the AhdlcRenderer.xml with three Signs configured to work with the above *.im2.

```
<Signs>
  <Sign Address="1" Mode="Monochrome" Width="98" Height="16" />
  <Sign Address="2" Mode="Monochrome" Width="98" Height="16" />
  <Sign Address="12" Mode="Color" Width="34" Height="20" />
</Signs>
```

6 Glossary

AHDLC	Alpha High-Level Data Link Control; Protocol used to communicate with Gorba LED signs
FPS	Frames per seconds; measurement of graphics refresh speed
GDI	Graphics Device Interface; a core component of Microsoft Windows operating systems responsible for simple graphical rendering
GIOoM	Gorba I/O over Medi; Protocol used to read and write digital inputs and outputs over the Medi protocol
GPIO	General-purpose input/output; digital input or output available on certain hardware
I/O	Input or output
Medi	Message Dispatcher protocol used to communicate between Gorba software components
RTS	Request To Send; output line on a serial port
TTS	Text-to-speech; technology to convert normal language text into speech audio
VLC	VideoLAN VLC media player
Ximple	Protocol used over Medi for communication between Protran and Composer