SharpMedia High Level Architecture

© Bojan Šernek – SharpMedia project, document version: DRAFT

Document History:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Author | Reviewed By | Revision | Version | Date |
| Bojan Šernek |  | ?? | 0.1 DRAFT | 12-Mar-07 |
| Bojan Šernek |  | 136 | 0.1 DRAFT | 24-Mar-07 |
| Bojan Šernek |  | 339 | 0.2 DRAFT | 02-Dec-07 |

Contents

[Introduction 4](#_Toc184325784)

[Deliverables 5](#_Toc184325785)

[Deliverables Matrix 5](#_Toc184325786)

[Specific Deliverables 6](#_Toc184325787)

[Bootstrap Loader 6](#_Toc184325788)

[Core and OS 6](#_Toc184325789)

[Configuration Tools 6](#_Toc184325790)

[Textual Shell 7](#_Toc184325791)

[Network Client 7](#_Toc184325792)

[Network Server 7](#_Toc184325793)

[Dedicated Server 7](#_Toc184325794)

[Physical Database 7](#_Toc184325795)

[Graphics Library 8](#_Toc184325796)

[Standard Materials 8](#_Toc184325797)

[Scene Library 8](#_Toc184325798)

[Physics Library 8](#_Toc184325799)

[Sound Library 9](#_Toc184325800)

[Static Structure 10](#_Toc184325801)

[Runtime Configuration 10](#_Toc184325802)

[Full Version Configuration 10](#_Toc184325803)

[SharpMedia System Static Structure 11](#_Toc184325804)

[Application Configuration 11](#_Toc184325805)

[Component Providers and Usage Scenarios 12](#_Toc184325806)

[Libraries 12](#_Toc184325807)

[Applications 12](#_Toc184325808)

[Services 13](#_Toc184325809)

[Using Libraries 13](#_Toc184325810)

[Using Applications 13](#_Toc184325811)

[Using Services 13](#_Toc184325812)

[Component Cross Reference 14](#_Toc184325813)

[Core Components 14](#_Toc184325814)

[Kernel Components 14](#_Toc184325815)

# Introduction

This document aims to provide a complete architectural overview of the SharpMedia project in all of its aspects, namely, the actual deliverables in regards to the deliverables configuration matrix, process and component breakdowns are presented in this single, master document, usable for reference and overview.

# Deliverables

The SharpMedia project is a multi-configuration solution, meaning that different components in different versions and flavors represent a complete SharpMedia system existing for a specific task. With regards this aspect, the following deliverables configuration matrix is presented to clarify the relationships.

## Deliverables Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Deliverable** | **Runtime** | **Student Version[[1]](#footnote-2)** | **Full Version** |
| Bootstrap Loader | Yes | Yes | Yes |
| Core and OS | Yes | Yes | Yes |
| Configuration Tools | Yes | Yes | Yes |
| Textual Shell | Yes | Yes | Yes |
| Network Client | Yes | Yes | Yes |
| Network Server[[2]](#footnote-3) | Yes | Yes | Yes |
| Dedicated Server | Yes | Yes | Yes |
| Physical Database | Yes | Yes | Yes |
| Graphics Library | Yes | Yes | Yes |
| Standard Materials | Compiled Only | Yes | Yes |
| Scene Library | Yes | Yes | Yes |
| Physics Library | Simple | Simple | Yes |
| Sound Library | Yes | Yes | Yes |
| Music Player | No | Yes | Yes |
| Graphical Desktop Shell | No | Yes | Yes |
| Networked Logon | No | No | Yes |
| Graphical Interface Library | Yes | Yes | Yes |
| Default GUI Theme | No | Yes | Yes |
| Additional GUI Themes | No | No | Yes |
| Game Library | Yes | Yes | Yes |
| Game Examples | No | FPS, RTS | Yes |
| Groupware Client | No | Yes | Yes |
| Groupware Server | No | Standalone | Yes |
| 3D Modeling Tool | No | Yes | Yes |
| Texturing Tool | No | Yes | Yes |
| Texture Compression Tool | No | DXTn | Yes |
| Animation Tool | No | Yes, no lip sync | Yes |
| UV Mapping Tool | No | Yes | Yes |
| Document Authoring Tool | No | Yes | Yes |
| Office Integration | No | No | Yes |
| Maya Integration | No | Import/Export | Yes |
| 3DStudio Max Integration | No | Import/Export | Yes |
| Cinema 4D Integration | No | Import/Export | Yes |
| Command Line Toolset | No | No | Yes |
| Defect and Task Tracking | No | Yes, no automation | Yes |
| Light Mapping Toolset | No | Simple | Yes |
| Ray Tracing Toolset | No | Simple | Yes |
| Gold Master Authoring | No | No | Yes |
| Audio Editing Tool | No | Yes | Yes |
| Video Editing Tool | No | Yes | Yes |
| Image Composition Toolset | No | No | Yes |
| Offline Rendering Tool | No | Yes, no automation | Yes |
| Support Request Tool | No | No | Yes |
| Web Browser | No | No | Yes |

## Specific Deliverables

This section will iterate through all of the deliverables and provide a short overview of how each deliverable is distributed and packaged, as well as a summary of the added value / features to the system.

### Bootstrap Loader

Delivered as a host operating system executable and written in pure .NET CLR, the bootstrap loader loads the appropriate System assemblies and sets up the basic system. That includes the root in-memory database and the system level isolation.

In the development SharpMedia version, this executable is most closely resembled by OSTest.exe

Files:

* $INSTALLATION/smboot.exe

### Core and OS

Delivered as a set of assemblies, Core and System are libraries linked by all of the processes and components in SharpMedia. They are the foundation for any SharpMedia development and provide important features like the physical database and I/O abstraction, mathematical library, process management and isolation and more.

The Core and System runtime are loaded into the GAC, while the Kernel is not.

Files:

* $INSTALLATION/System/SharpMedia.dll
* $INSTALLATION/System/SharpMedia.ComponentOS.dll
* /System/Kernel

### Configuration Tools

Provides a set of textual and graphical tools that can be invoked to manipulate the configuration store.

Files:

* /System/Applications/Base/ReadConfig
* /System/Applications/Base/WriteConfig
* /System/Applications/Base/EditConfiguration

### Textual Shell

Reads textual input from the input stream (or the user) and writes them to the output (or to the user). Commands may be names of applications, or a selection of internal commands that borrow loosely from POSIX.

Files:

* /System/Applications/TextShell

### Network Client

Delivered as a SharpMedia library, the Network Client provides services and routines to connect a SharpMedia workstation to a SharpMedia server of any kind. It would use a combination of special sinks and .NET CLR remoting to achieve its task.

Files:

* /System/Libraries/NetClient

### Network Server

Delivered as a SharpMedia library and executable, the Network Server provides services and routines so that other components, like the Game Library, can expose their services to Network Clients.

Files:

* /System/Libraries/NetServer
* /System/Applications/Base/NetServerHost

### Dedicated Server

Delivered as a SharpMedia executable the Dedicated Server sets up the Network Server and System in a special configuration. As a result the System is only doing Network Server and related tasks. This minimizes the memory and processing overheads.

Files:

* /System/Applications/Base/DedicatedNetServerHost

### Physical Database

Delivered as a set of assemblies to be loaded by the bootstrap loader in order to enable mounting of physical (persistent) databases from block devices.

All of the Physical Database libraries reside in the GAC.

Files:

* $INSTALLATION/System/SharpMedia.Database.Physical.dll
* $INSTALLATION/System/SharpMedia.Database.Physical.Windows.dll

### Graphics Library

Delivered as a SharpMedia library and service, it deals with interfacing to the graphics rendering hardware and abstracts away most of the graphics objects like geometry, images and materials.

Files:

* /System/Libraries/Graphics
* /System/Libraries/Graphics/Drivers/Direct3D10Driver
* /System/Libraries/Graphics/Drivers/Tao

### Standard Materials

Can be delivered with “source”, which means that the source Direct Acyclic Graphs full with documentation are provided as a reference which can be either extended, reused or modified to specific needs, or “compiled only”, which includes only the materials as they are converted to a collection of commands.

Files:

* /Applications/Graphics/MaterialEditor/Library

Source Files:

* /Applications/Graphics/MaterialEditor/Library/Source

### Scene Library

The Scene library is delivered as a SharpMedia library. It can be used to take control of the Graphics device, offloading the management of the graphics resources and their optimization to the scene library.

Files:

* /System/Libraries/SharpMedia/Scene

### Physics Library

The Physics library is delivered as a SharpMedia library. It holds physical representation of objects and animates them based on their physical properties. Scene library provides direct wrappers to extract data from physics representation into renderable form.

Files:

* /System/Libraries/SharpMedia/ Physics

### Sound Library

Delivered as a SharpMedia library, the Sound Library gives the user the ability to play, sequence, synthesize, record, mix and process sound clips.

Files:

* /System/Libraries/Sound

# Static Structure

The static structure of the SharpMedia depends on the configuration used. It varies most between the runtime configuration and the full configuration, with the student version structured exactly as the full version, unless noted otherwise

## Runtime Configuration

The runtime configuration is usually represented by three entry point processes:

* The runtime system executable to run the multimedia application
* The dedicated server executable to run the multimedia application as a dedicated server
* The world editor executable to run the world editor and/or other tools that were shipped with the multimedia application

Unless otherwise specified, the runtime system **does not** contain any parts of the bundled groupware solution, source resources and code files, as well as any parts of the multimedia application development kit.

## Full Version Configuration

The full version configuration is usually represented by four entry point processes

* The groupware server executable to run the dedicated groupware server
* The groupware client executable to run the groupware client against a groupware server
* The shell executable used by the administration staff to perform maintenance on the deployed configuration
* The automated self-test executable that performs automatic self-testing of the configuration

# SharpMedia System Static Structure

All of the executables, except for the self-test executable, bootstrap into a configuration of the SharpMedia System.

It is important to note that while the host operating system static structure is very simple (only one SharpMedia process per configuration may run at a time); the internal static structure inside the SharpMedia System is quite more complex, especially in the Full Version.

The whole runtime system is based around components, which are objects that are connected to other objects on demand based on their requirements and exposed capabilities. In OOP, the requirements translate to settable public properties and exposed capabilities correlate to implemented interfaces.

A SharpMedia System instance is composed of applications, each composed of a graph of components, rooted in an application component, implementing an ***IApplication*** interface. Some of the applications are working like servers and expose their components to other applications and some applications consume components from other applications. Applications made for the sole purpose of serving components to other applications are called Services.

## Application Configuration

Since an application is actually just a graph of components, a similar application to an existing one may be created by just changing a few components or their configuration values. In that aspect, any valid graph of components that has a root with an ***IApplication*** capability is called an application.

Physically, an application is usually made of one or more code locations combined with an application descriptor that defines what components to instantiate from them and how to link them together with possibly imported components to form a valid application. This is called an application configuration.

The philosophy is to deliver a set of assemblies (code locations) and then create multiple applications from them just by supplying different configurations. A derived feature is that an application may change itself by modifying its configuration. Applications that have settings that the user can personalize may easily store personalized information in the application configuration that is specific to the user. In fact, all SharpMedia tools and editors must follow this rule.

# Component Providers and Usage Scenarios

What follows in this chapter are guidelines and scenarios that should cover most of the design and implementation problems faced when writing Services for, or part of SharpMedia, or using them.

## Libraries

Libraries are component providers that do not take an active role in activation they simply provide implementers of various interfaces in form of classes. A simple logging library, for example, would contain several different ***ILogger*** implementations, and the user linking against it can choose which one to use.

When a component is required from a Library, there are no calls made to the Library code itself except for the instantiation of the component (constructor of the class) and assignments to its properties (configuration).

Since these types of component providers are easiest to write, use and maintain, they are the preferred type of implementing SharpMedia features. Components are run in the context of the caller and use the caller’s security context. Threads may be spawned, but Library writers are advised against it if possible.

Components that are imported from a library and must be configured are done so in the Environment of the caller. This implies that if a caller has already selected an ***ILogger*** implementation and then requires a component that requires a logger implementation, it is provided from the one already selected by the caller.

If the caller has the appropriate permissions, a Library is always available for linking. As a result, a Library has no lifetime, and any lifetime objects it creates (in the context of its caller) are associated with the time span of this caller and are treated as such when the caller terminates.

Since Library components used by different callers execute in different contexts, they may not share data without inter process communication. Static variables, threads and similar mechanisms are not guaranteed to work in all System configurations.

## Applications

Applications are like Libraries in that they contain classes that represent components, so all rules that apply to Libraries apply to Applications. The main difference is that an application is expected to do some work, probably some work that is executed in a sequential fashion, like execute a batch script, compile a program or react to user input.

Applications therefore select a component from their Environment (either own or imported) that acts as the Application component. A Run method on this component is invoked when the application is started, and the application is expected to do its work in this method. Spawning of Threads is allowed and components run in an Application security context, which may be modified by an appropriate security tool.

## Services

Services are Applications that are not expected to run a sequential series of tasks but rather run as long as they are required to by other application, while serving them components. A typical example is a database engine that exposes database manipulation components to its clients.

There are several differences in regards to the Library, which exists for a very similar purpose:

* A Service runs in its own context and lifetime objects it creates are associated with that context, terminating with the Service and not the caller
* A Service is available for as long as it runs. When it terminates, its components are not available anymore
* A Service is an active component provider and is optionally informed when a client requests a component and when it terminates
* Threading, as other lifetime objects is available and executes in the Service context. Limiting by ThreadPools is encouraged

## Using Libraries

To use a Library, simply define a property with a setter of the interface Type that you wish to link against. If you wish to receive the default implementer of that interface, do not define any binding (parameters) for that property. If you wish to link to a specific implementation from a specific library, include that information in your parameters section for the component that requires the library component.

## Using Applications

Applications are usually of limited availability and should be used with care, though that might not be the case with user interactive applications. If an application exposes a component (for example a remote controlling API for a music player program) treat it as you would a normal library component, but be prepared for the event that the component may suddenly change to null when an application terminates.

## Using Services

See *Using Libraries*.

# Component Cross Reference

This section of the HLA will focus on components and the libraries, applications or services providing them. This chapter is meant to be used as a cross reference to quickly find a component and the document that describes it in detail.

## Core Components

TBD, we must still convert many of the classes in the Core library to be component-aware, or at least create a number of simple to use wrappers in a Proxy library, for example, “CoreComponents”.

## Kernel Components

The Kernel components are documented in the “Components\Kernel.docx” document.

|  |  |
| --- | --- |
| Interface Name | Description |
| ApplicationsDirectory | Manages access to the running application instances and installed applications |
| DocumentManagement | Runs document verbs and manages bindings between verbs, documents and applications |
| IComponentManagement | Allows for manual or automatic configuration by code or verification of configuration for each configured component |
| IDatabaseManager | Mediates access to a persistent, hierarchical storage medium, a hybrid between an object database and a file system |
| InstallationService | A service that keeps track of packages and performs installations, uninstallations, upgrades and maintenance of packages |
| PackageAuthoring | Library to facilitate creation of installation packages, including issuing and managing digital signatures |
| ConfigurationManagement | Manages creation and modification of application configurations |

1. It is questionable whether this configuration will ever exist [↑](#footnote-ref-2)
2. Used to host a target game [↑](#footnote-ref-3)