







Helix DNA Client definition

- Building blocks for media service client applications

Mobile  	Desktop  	Living room  
<ul style="list-style-type: none"> • Mobile TV • Mobile Video • Mobile Music • Ring Tones 	<ul style="list-style-type: none"> • Utility Player • Desktop TV • Desktop Video • Desktop Music • PMC Manager 	<ul style="list-style-type: none"> • IP-TV • Video Center • Music Center

Relationship to other components

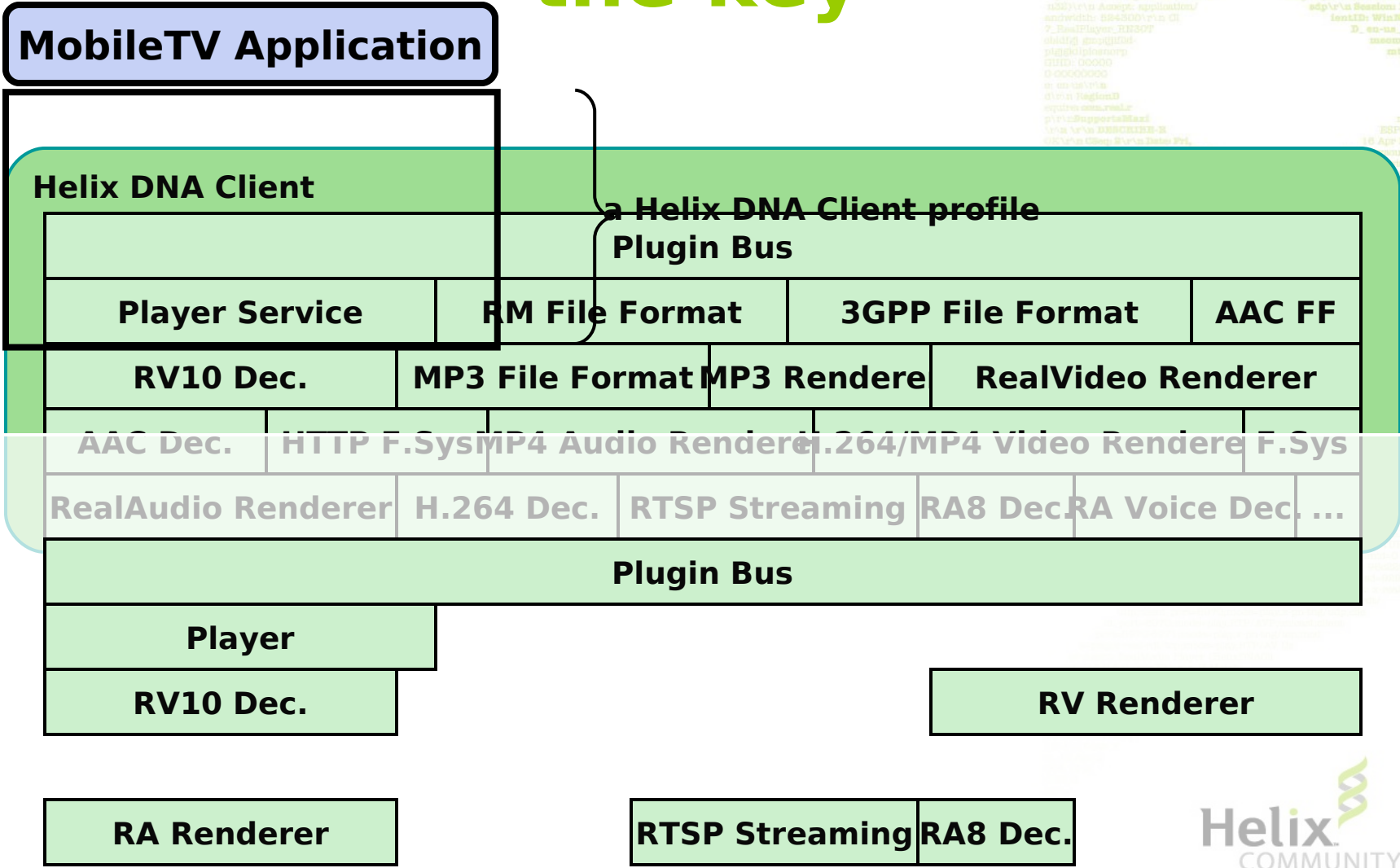
Application (GUI, application flow)

Helix DNA Client

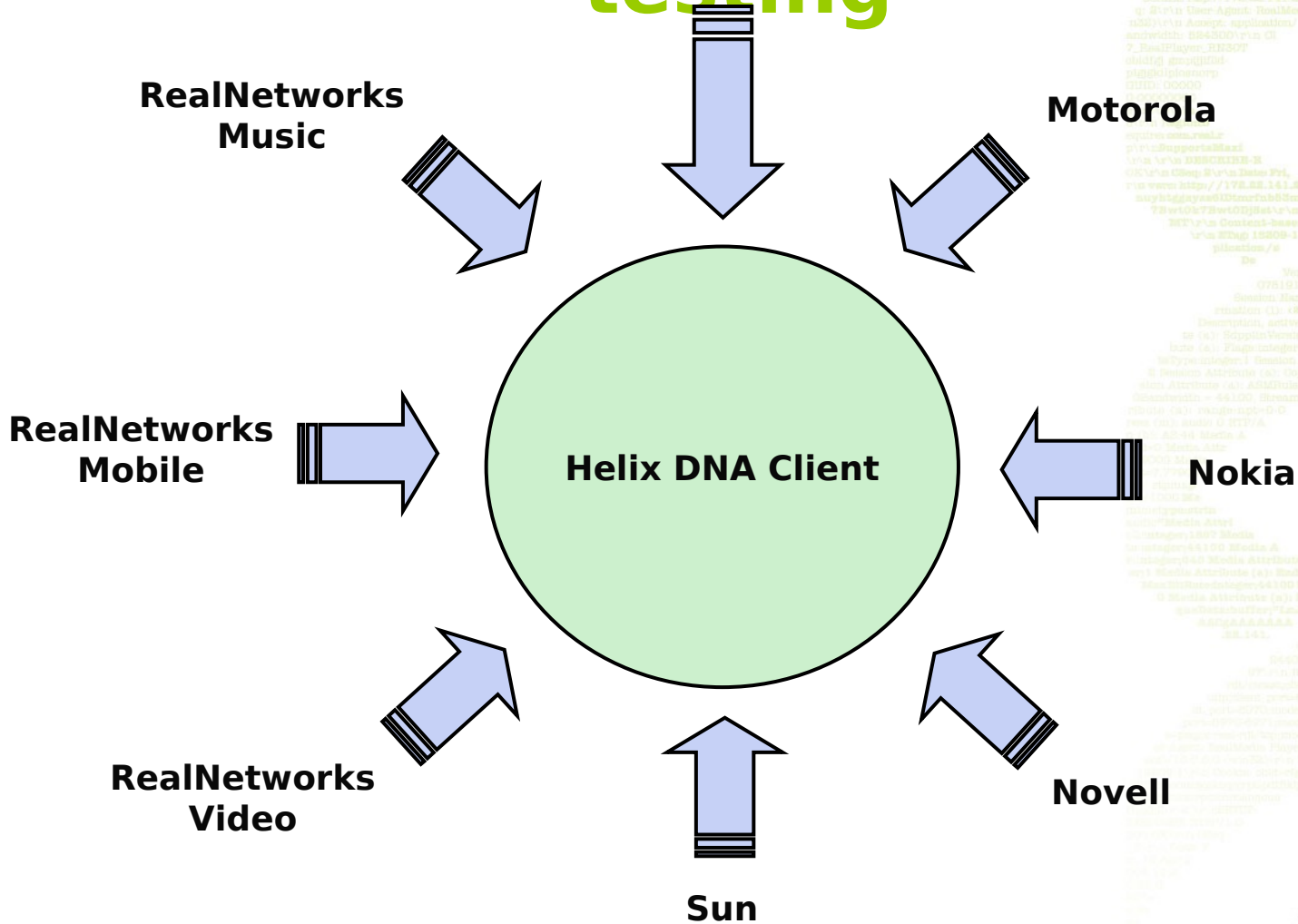
- Playback controls
- Synchronization & Timing
- Audio/Video Mixing
- Streaming protocols
- File formats
- Codecs
- DRM

Operating System

Modularity & Scalability are the key



Community development and testing



www.helixcommunity.org

Helix Community: Welcome - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

https://helixcommunity.org/

Customize Links Free Hotmail Windows Marketplace Windows Media Windows

Helix COMMUNITY

Search for: Submit

Home My Page Project Tree Code Snippets Project Openings

Welcome to the Helix™ Community!

The Helix Community is a collaborative effort among both leading technology companies and open-source developers to extend the Helix DNA™, the first open multi-format digital media platform.

Join the Helix Player 2.0 project!
This project is currently looking for developers and testers to assist in the completion of exciting new features such as Ad-Free radio, playlist support and Auto Bandwidth Detection. Visit the [Player project](#) for more info.

Key Helix Pages

- [Need help?](#)
- [Downloads](#)
- [Participating in Helix](#)
 - [Register](#)
 - [Mailing Lists & IRC](#)
 - [Report a bug!](#)
 - [Developer Info](#)
 - [Download Source](#)
 - [Browse CVS tree](#)
 - [Policies](#)
 - [Job Listings](#)
 - [Roadmap](#)
- [Project List](#)
- [Licenses](#)

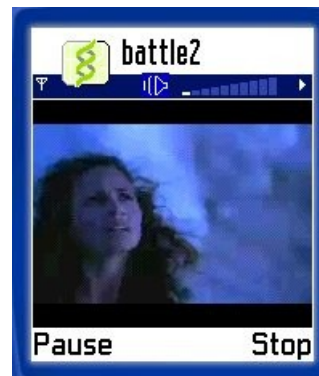
The Helix DNA platform is comprised of the following:

- [Helix DNA Client / Helix Player](#)
- [Helix DNA Producer](#)
- [Helix DNA Server](#)
- [Helix DNA Codecs and Formats](#)

- Tri-licensed: GPL, RPSL, RCSL
- Mailing lists, IRC
- Mail driven Code Reviews
- Bug Tracker
- Project pages
- Search Tool
- Code access via CVS + SSH
- Python based build system

Helix DNA Client Projects in the Community today

- OLPC – One Laptop Per Child Media Player
- Helix Player for Linux
- Helix Player for Symbian
- Helix DNA Client for 2008 (Brego)



Helix Client Architecture Stack

Application (GUI, application flow)

Middleware (platform API specialization):
HXClientKit, JSR135, MMF, ActiveX, NSPlugin...

Platform (technology primitives):
Player, Recorder, Auto Upgrade, Visualizations...

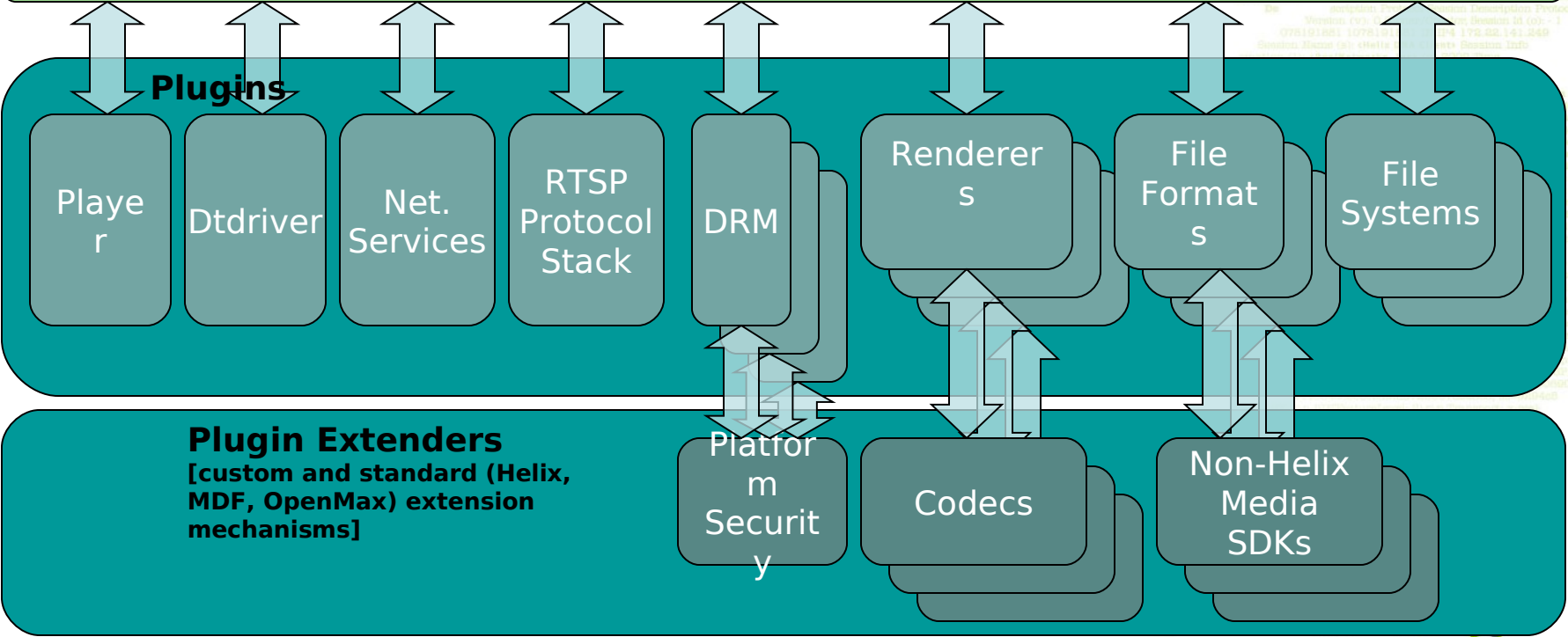
Atlas Platform Architecture

splay dtdrive

Helix DNA Client API

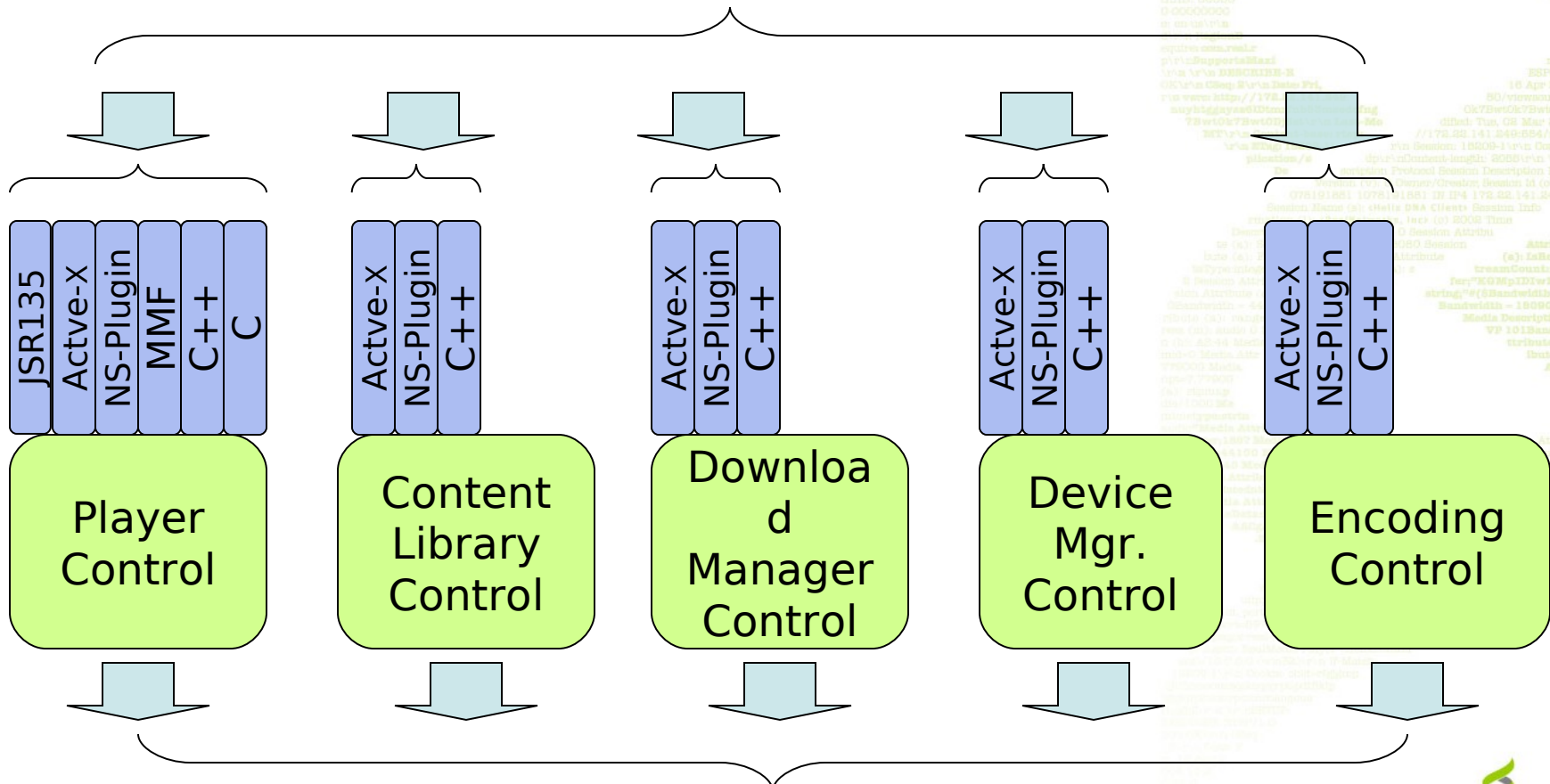


Plugin Bus + Essential Containers & OS Primitives



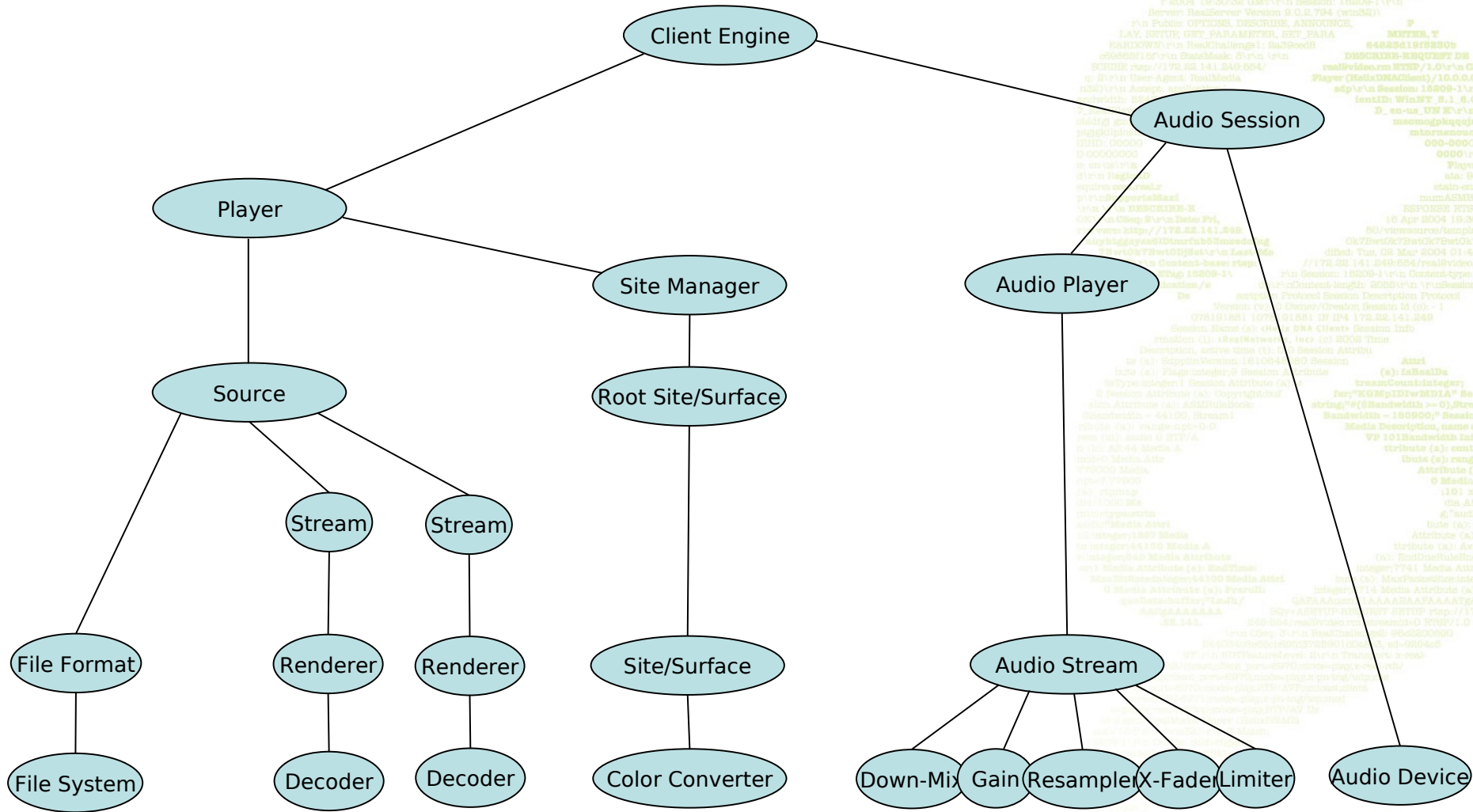
Middleware Architecture Concept

Easy to use application authoring APIs

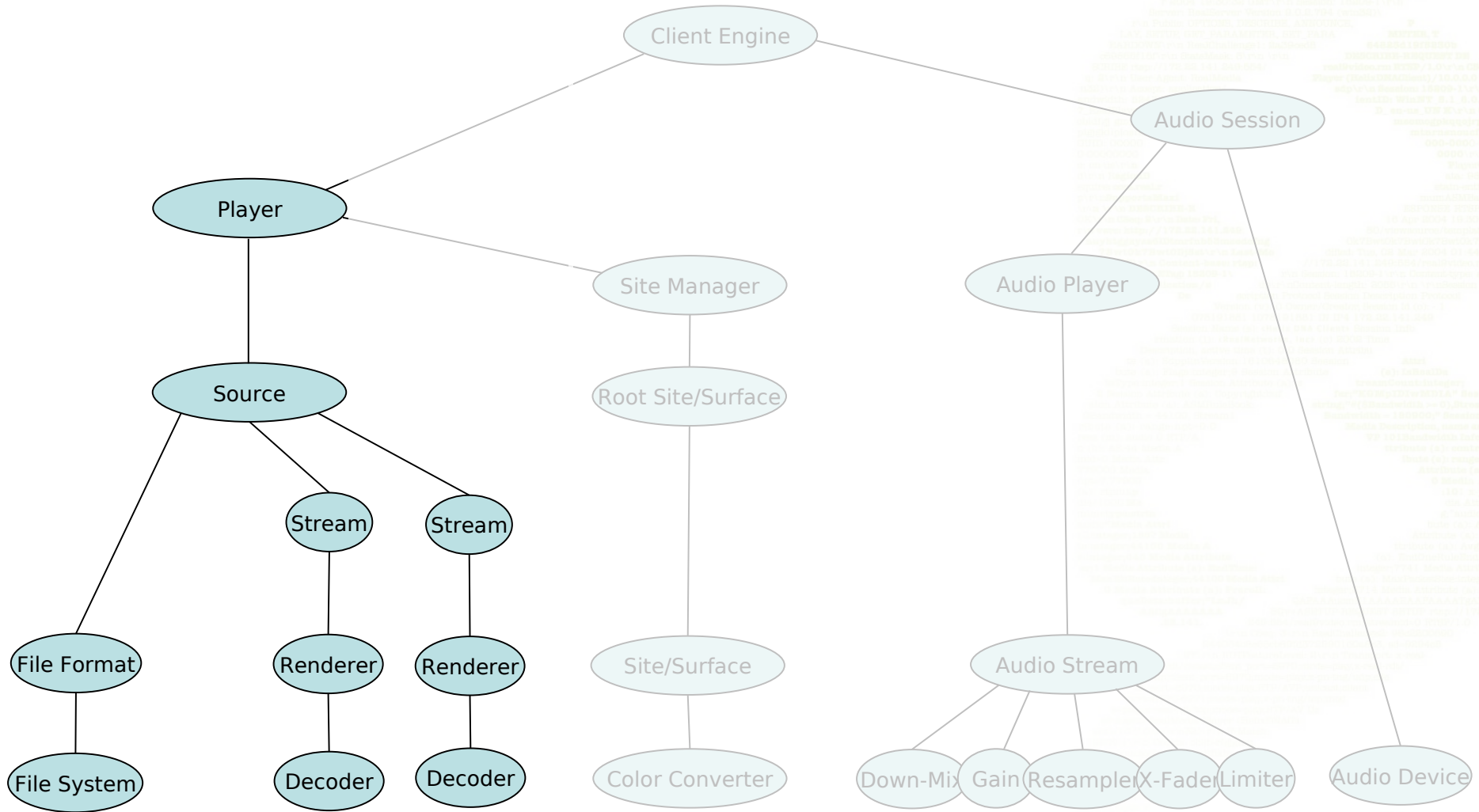


Helix DNA Client API

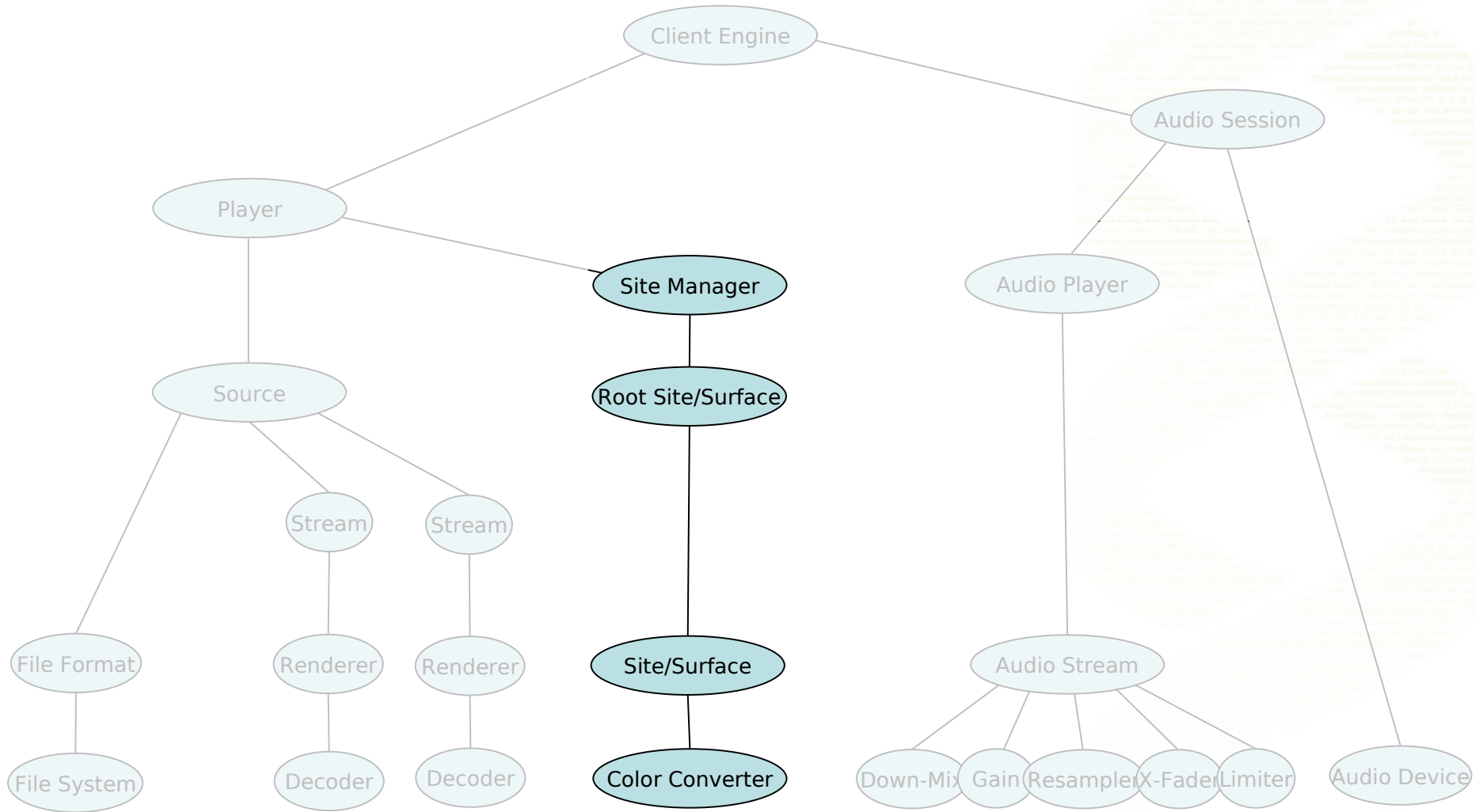
Atlas Playback Objects Control Hierarchy



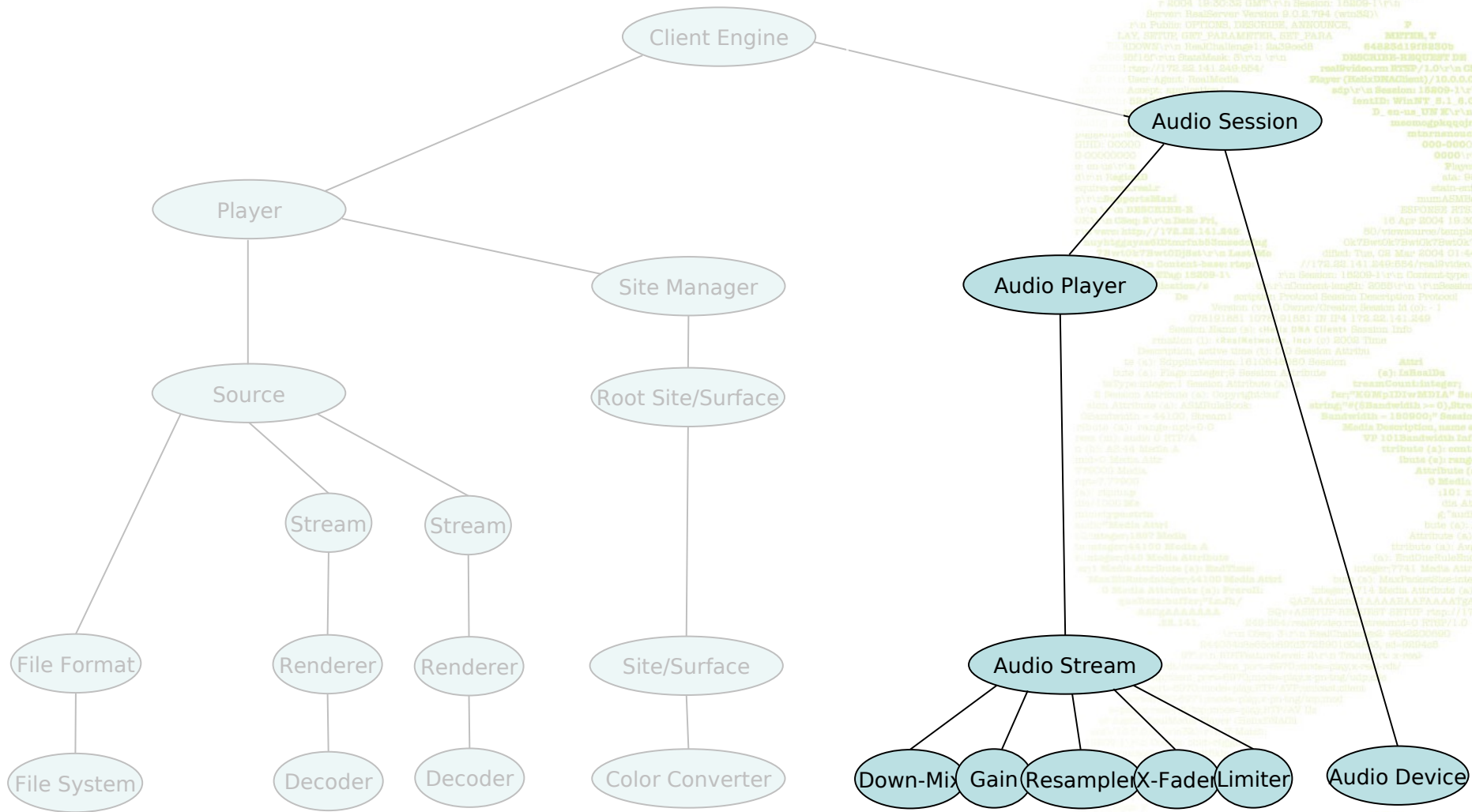
Atlas Playback Input and Rendering Hierarchy



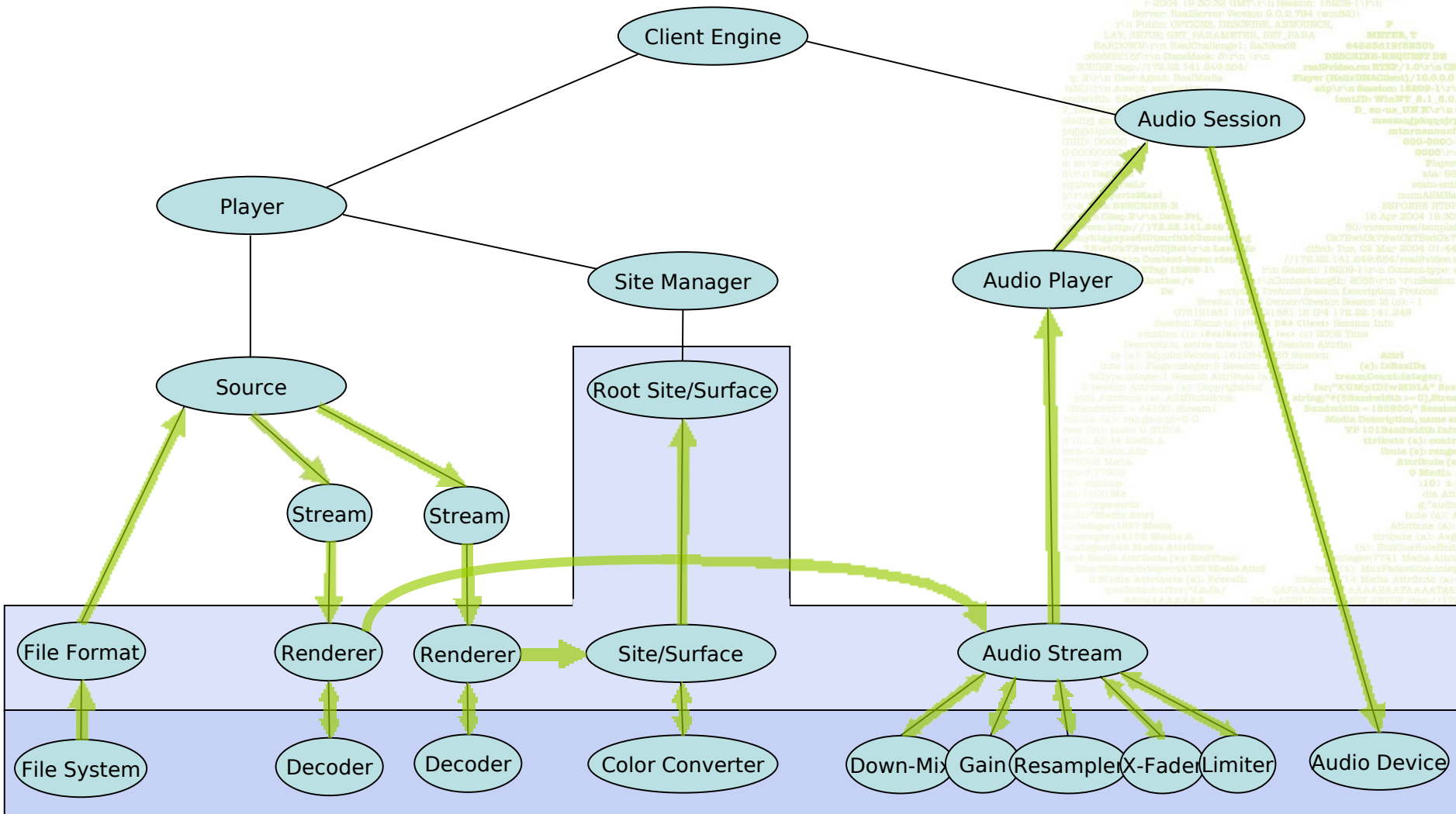
Atlas Playback Objects Control Hierarchy



Atlas Playback Objects Control Hierarchy



Atlas Playback Objects Control Hierarchy



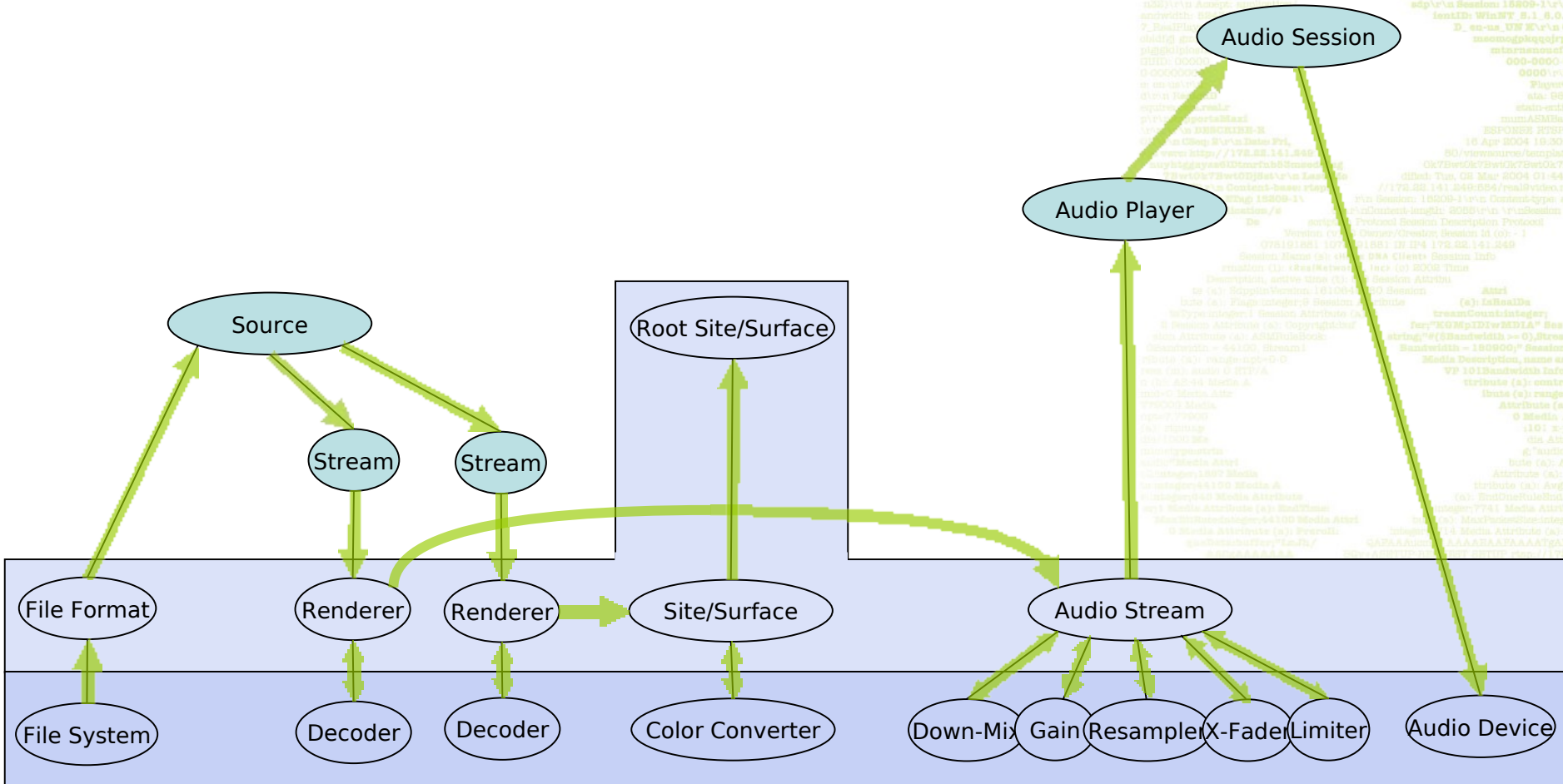
➔ Data Flow

Media objects in OpenMax domain easily replaceable by OpenMax components through OpenMax-IL interface

Media objects on the fringe of OpenMax domain but still replaceable by OpenMax components through OpenMax-IL interface



Atlas Playback Objects Control Hierarchy



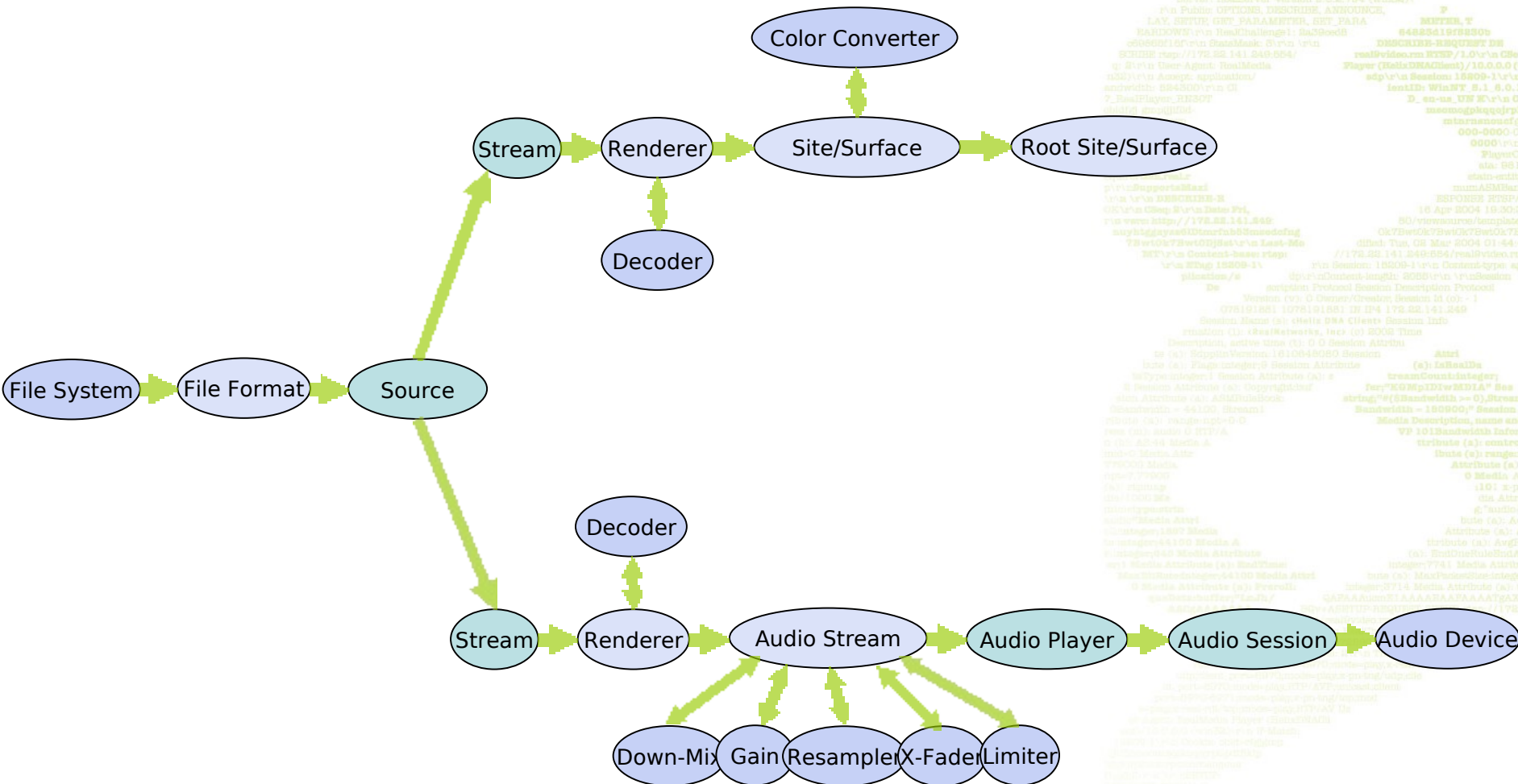
➔ Data Flow

Media objects in OpenMax domain easily replaceable by OpenMax components through OpenMax-IL interface

Media objects on the fringe of OpenMax domain but still replaceable by OpenMax components through OpenMax-IL interface



Media Data Flow through Atlas Playback Objects



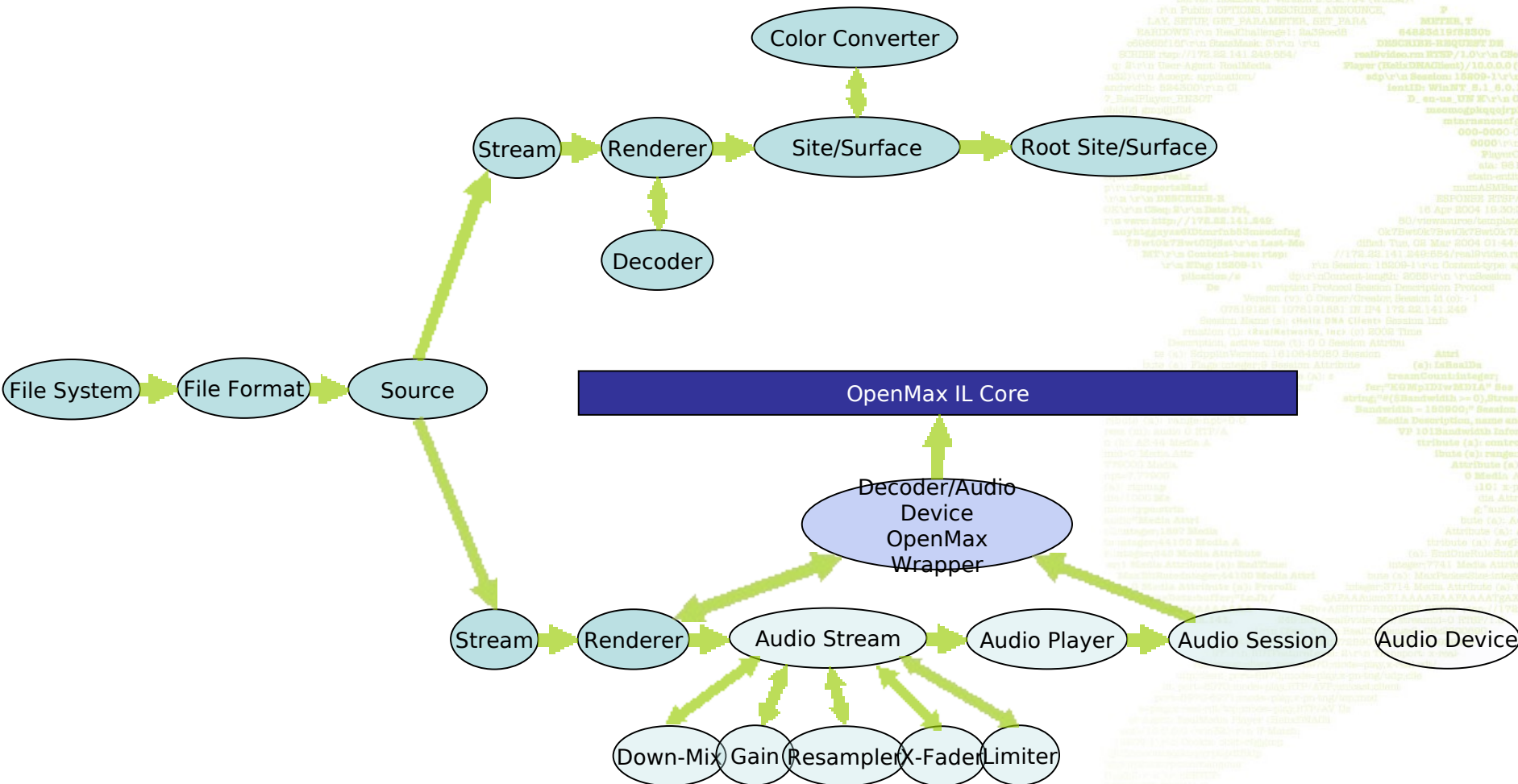
Data Flow

Media objects in OpenMax domain easily replaceable by OpenMax components through OpenMax-IL interface

Media objects on the fringe of OpenMax domain but still replaceable by OpenMax components through OpenMax-IL interface

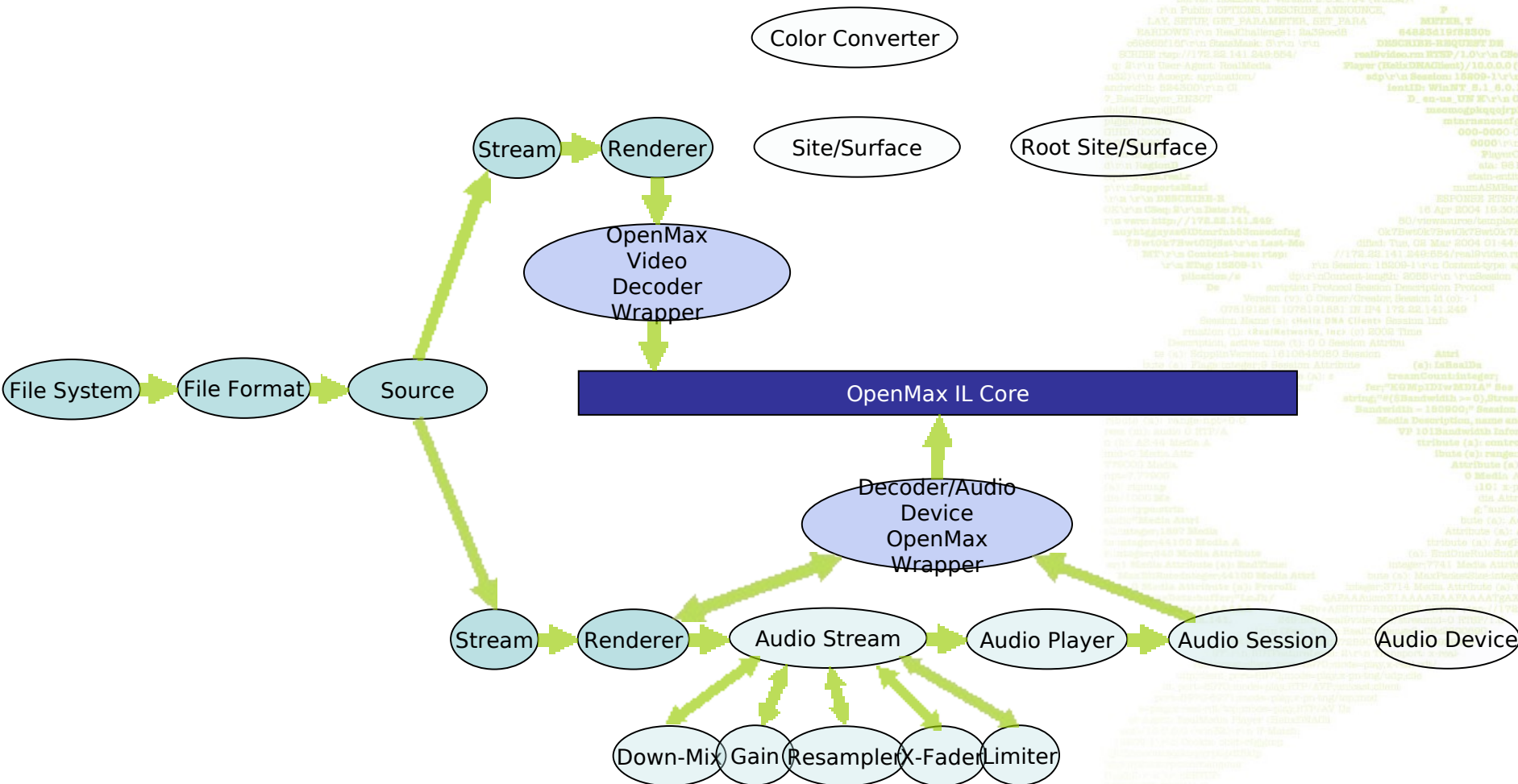


Atlas Playback Objects with OpenMax Audio Pipeline



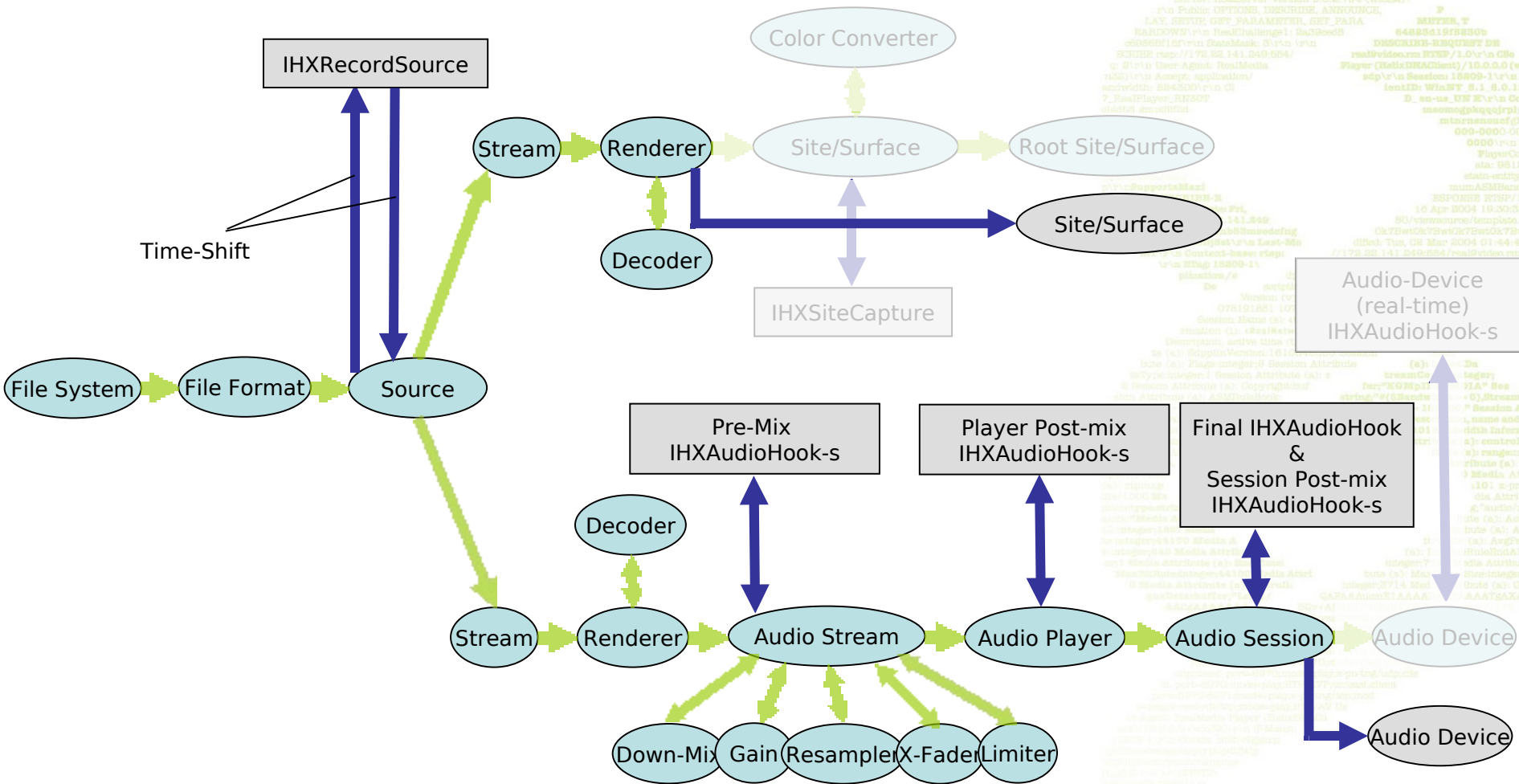
 **Data Flow**

Atlas Playback Objects with OpenMax Audio and Video Pipeline

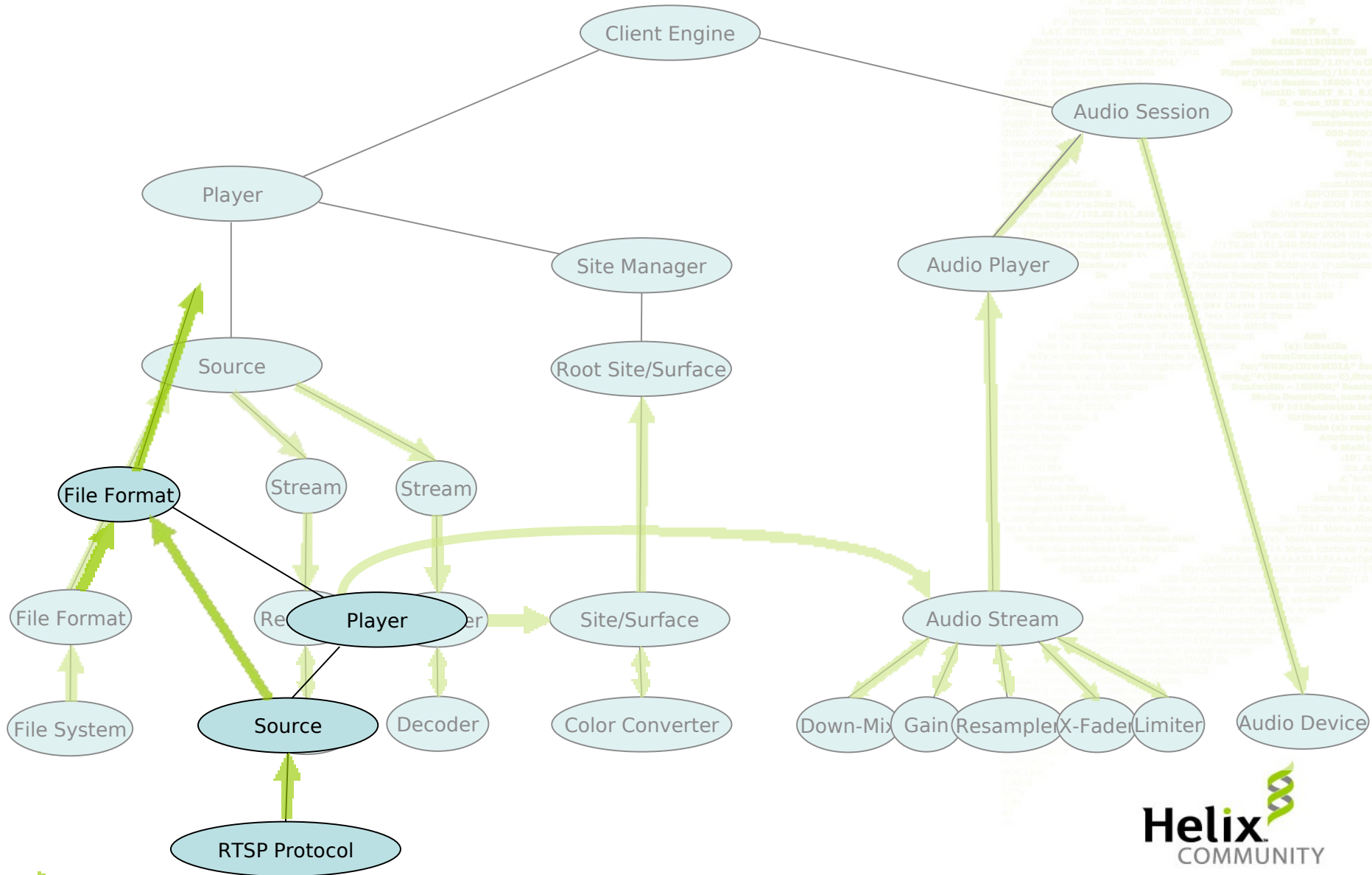


 Data Flow

Media Hooks in Playback path

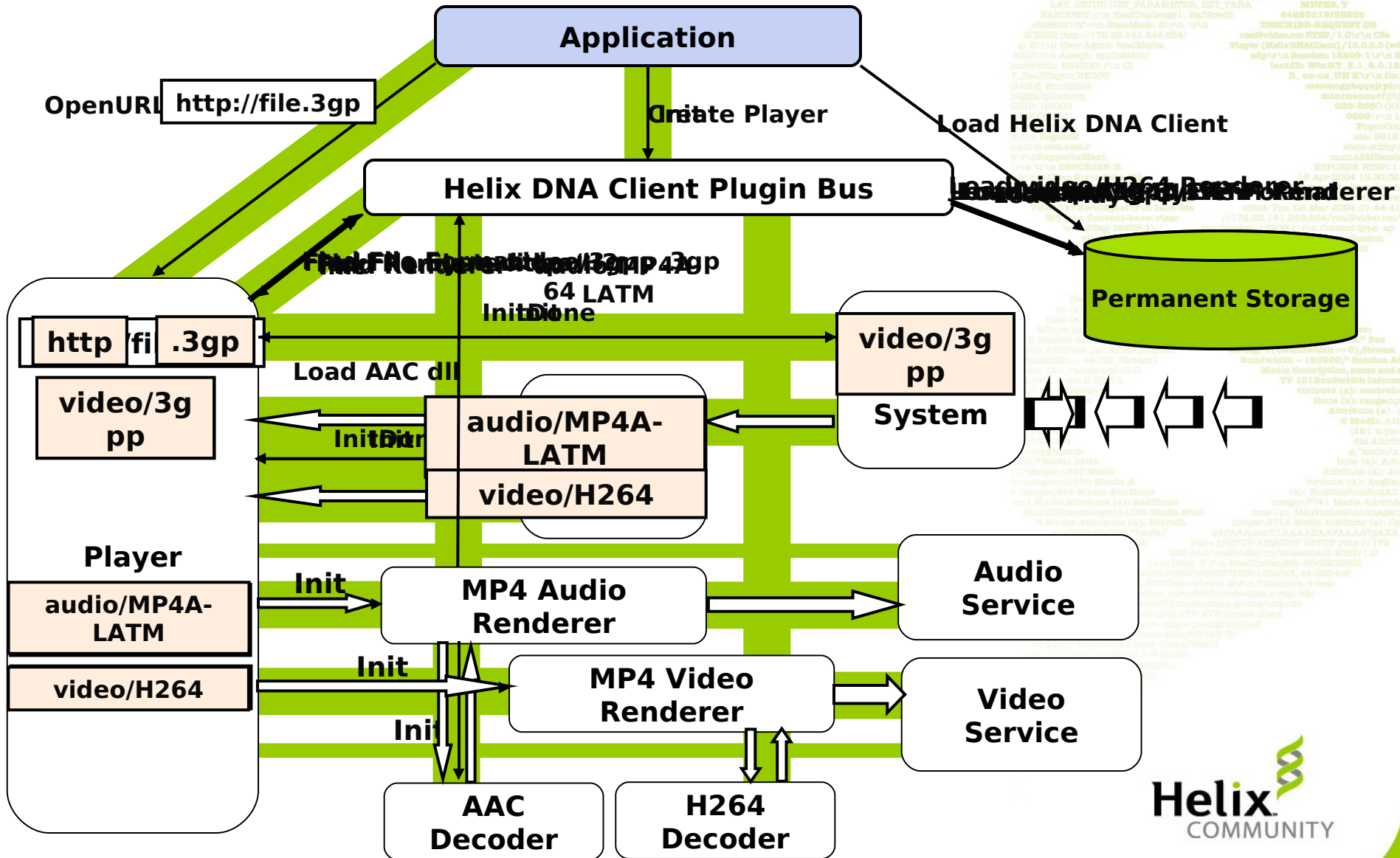


Repurposing the Control Hierarchy



➔ Data Flow

Media Path building



Implementation Foundation

- Object Oriented Approach
 - Framework is written in C++
 - COM-like model
- Interfaces are generally asynchronous
- Platform Adaptation Layer
 - Thread/Mutex and Timing Primitives
 - Audio Device, Video Surface Abstractions
 - Network Services Abstractions
 - DLL Access Abstraction
 - File I/O Abstraction



Threads

- Player Engine Threads:
 - Main Thread (application interface)
 - Core Thread (supplemental - engine only)
 - Optimized Scheduler Thread (high priority)
 - Network Thread
 - Plugin specific threads
 - Engine can run with Main Thread only

Plugin Buss

■ Primary Interface (IHXMediaPlatform):

```
DECLARE_INTERFACE_(IHXMediaPlatform, IUnknown)
{
    STDMETHOD(GetVersion)                (THIS_
        UINT32* pVersion) PURE;

    STDMETHOD(AddPluginPath)             (THIS_
        const char* pszName,
        const char* pszPath) PURE;

    STDMETHOD(Init)                      (THIS_
        IUnknown* pContext) PURE;

    STDMETHOD(Close)                    (THIS)    PURE;

    STDMETHOD(Reset)                    (THIS_
        IUnknown* pContext,
        HXBOOL    bPlatformOnly) PURE;

    STDMETHOD(Purge)                    (THIS)    PURE;

    STDMETHOD(CreateChildContext)        (THIS_
        IHXMediaPlatform** ppChildContext) PURE;
};
```

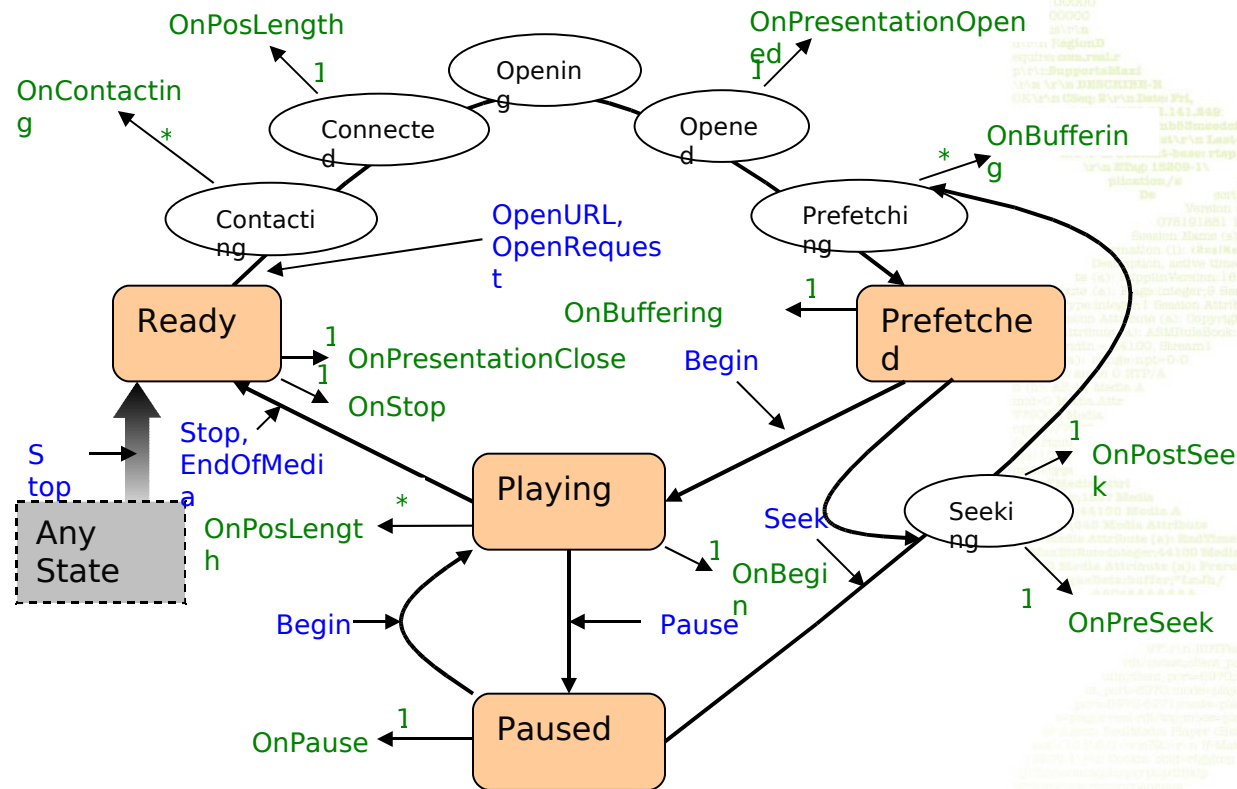
Plugin Buss - Services

- Intrinsic
 - IHXMediaPlatform
 - IHXMediaPlatformKicker
 - IHXPluginHandler
- Extendable
 - IHXCommonClassFactory
- Replaceable Services
 - IHXScheduler
 - IHXMutex
 - IHXOptimizedScheduler
 - IHXPreferences
 - IHXRegistry
 - IHXNetServices
 - IHXSiteEventHandler

Player Controls

- Player Management: IHXClientEngine
 - CreatePlayer(), ClosePlayer()
- Playback Control: IHXPlayer
 - Begin(), Pause(), Seek(), Stop(), SetVelocity()
- Playback Notifications: IHXClientAdviseSink
 - OnBegin(), OnBuffering(), OnPause(), OnStop(), OnPosLegth()...
- Reference Clock distributed to all renderer plugins
 - OnTimeSync()

Player State Machine



File Format – direct data (source) supplier

```
DECLARE_INTERFACE_(IHXFileFormatObject, IUnknown)
```

```
{  
    STDMETHOD(GetFileFormatInfo) (THIS_  
        REF(const char**) /*OUT*/ pFileMimeTypes,  
        REF(const char**) /*OUT*/ pFileExtensions,  
        REF(const char**) /*OUT*/ pFileOpenNames  
    ) PURE;  
  
    STDMETHOD(InitFileFormat) (THIS_  
        IHXRequest* /*IN*/ pRequest,  
        IHXFormatResponse* /*IN*/ pFormatResponse,  
        IHXFileObject* /*IN*/ pFileObject  
    ) PURE;  
  
    STDMETHOD(GetFileHeader) (THIS) PURE;  
  
    STDMETHOD(GetStreamHeader) (THIS_  
        UINT16 unStreamNumber) PURE;  
  
    STDMETHOD(GetPacket) (THIS_  
        UINT16 unStreamNumber) PURE;  
  
    STDMETHOD(Seek) (THIS_  
        ULONG32 ulOffset) PURE;  
  
    STDMETHOD(Close) (THIS) PURE;  
};
```


Structure of Source Data - Headers

```

===== Begin File Header
=====
Duration = 30060
Height = 288
StreamCount = 2
Width = 352
Abstract = <Buffer>
Author = <Buffer>
Copyright = <Buffer>
Title = <Buffer>
===== End File Header
=====
----- Begin Stream Header = 0 -----
AvgBitRate = 150879
Duration = 29999
FramesPerMSecond = 25000833
Height = 288
RTTTPayloadType = 96
SamplesPerSecond = 90000
StreamNumber = 0
Width = 352
ASMRuleBook = Marker=0;Marker=1;
MimeType = video/X-HX-AVC1
StreamName = Video Track
OpaqueData = <Buffer>
----- End Stream Header = 0 -----
----- Begin Stream Header = 1 -----
AvgBitRate = 40000
Duration = 30060
MaxBitRate = 40000
RTTTPayloadType = 96
SamplesPerSecond = 22050
StreamNumber = 1
ASMRuleBook = Marker=0;Marker=1;
MimeType = audio/X-RN-MP4-RAWAU
StreamName = Audio Track
OpaqueData = <Buffer>
----- End Stream Header = 1 -----

```

```

=====
Duration = 30060
Height = 288
StreamCount = 2
Width = 352
Abstract = <Buffer>
Author = <Buffer>
Copyright = <Buffer>
Title = <Buffer>
===== End File Header
=====
----- Begin Stream Header = 0 -----
AvgBitRate = 150879
Duration = 29999
FramesPerMSecond = 25000833
Height = 288
RTTTPayloadType = 96
SamplesPerSecond = 90000
StreamNumber = 0
Width = 352
ASMRuleBook = Marker=0;Marker=1;
MimeType = video/X-HX-AVC1
StreamName = Video Track
OpaqueData = <Buffer>
----- End Stream Header = 0 -----
----- Begin Stream Header = 1 -----
AvgBitRate = 40000
Duration = 30060
MaxBitRate = 40000
RTTTPayloadType = 96
SamplesPerSecond = 22050
StreamNumber = 1
ASMRuleBook = Marker=0;Marker=1;
MimeType = audio/X-RN-MP4-RAWAU
StreamName = Audio Track
OpaqueData = <Buffer>
----- End Stream Header = 1 -----

```

Structure of Source Data - Packets

..... Start Packets

Strm#	Time	Rule#	Lst	Flgs	BufferSize	t=
__RTPTTime__						
S= 0	T= 0	R= 1	L=F	F=03	B= 2859	t= 0
S= 1	T= 0	R= 1	L=F	F=03	B= 6	t= 0
S= 0	T= 40	R= 1	L=F	F=02	B= 657	t= 3597
S= 1	T= 46	R= 1	L=F	F=03	B= 230	t= 1024
S= 0	T= 80	R= 1	L=F	F=02	B= 698	t= 7197
S= 1	T= 93	R= 1	L=F	F=03	B= 237	t= 2048
S= 0	T= 120	R= 1	L=F	F=02	B= 1013	t= 10797
S= 1	T= 139	R= 1	L=F	F=03	B= 230	t= 3072
S= 0	T= 160	R= 1	L=F	F=02	B= 1201	t= 14397
S= 1	T= 186	R= 1	L=F	F=03	B= 230	t= 4096
S= 0	T= 200	R= 1	L=F	F=02	B= 1527	t= 17997
S= 1	T= 232	R= 1	L=F	F=03	B= 236	t=



Renderer – datatype powerhouse

```

DECLARE_INTERFACE_(IHXRenderer, IUnknown)
{
    STDMETHOD(GetRendererInfo) (THIS_
                                REF(const char**) /*OUT*/ pStreamMimeTypes,
                                REF(UINT32) /*OUT*/ unInitialGranularity) PURE;

    STDMETHOD(StartStream) (THIS_
                            IHXStream* pStream,
                            IHXPlayer* pPlayer) PURE;

    STDMETHOD(EndStream) (THIS) PURE;
    STDMETHOD(OnHeader) (THIS_
                        IHXValues* pHeader) PURE;

    STDMETHOD(OnPacket) (THIS_
                        IHXPacket* pPacket,
                        LONG32 lTimeOffset) PURE;

    STDMETHOD(OnTimeSync) (THIS_
                           ULONG32 ulTime) PURE;

    STDMETHOD(OnPreSeek) (THIS_
                          ULONG32 ulOldTime,
                          ULONG32 ulNewTime) PURE;

    STDMETHOD(OnPostSeek) (THIS_
                            ULONG32 ulOldTime,
                            ULONG32 ulNewTime) PURE;

    STDMETHOD(OnPause) (THIS_
                       ULONG32 ulTime) PURE;

    STDMETHOD(OnBegin) (THIS_
                       ULONG32 ulTime) PURE;

    STDMETHOD(OnBuffering) (THIS_
                            ULONG32 ulFlags,
                            UINT16 unPercentComplete) PURE;

    STDMETHOD(GetDisplayType) (THIS_
                               REF(HX_DISPLAY_TYPE) ulFlags,
                               REF(IHXBuffer*) pBuffer) PURE;

    STDMETHOD(OnEndofPackets) (THIS) PURE;
}
    
```


A/V Controls

- Audio Controls
 - Volume, Mute
 - Sound level offset (level normalization)
 - Advise Sink (Observer)
- Video Controls
 - Size, Position
 - Z-Order
 - Alpha-blending
 - Brightness, Hue, Saturation, Contrast, Sharpness
 - Site Watcher (Observer)

Code Layout: inter-module organization

common	<ul style="list-style-type: none"> Include Runtime Container System Fileio Util Debug Unittest 		<ul style="list-style-type: none"> Interface Definitions Stdlib insulation Building blocks Low level OS abstractions Common utilities Debug & Test facilities 	
client	<ul style="list-style-type: none"> Common Core medpltfm Netwksvc Videosvc Audiosvc 		<ul style="list-style-type: none"> Multimedia framework Framework Services 	
datatype	<ul style="list-style-type: none"> Rm Mp4 Mp3 Amr ... 	<ul style="list-style-type: none"> Fileformat Audio Video 	<ul style="list-style-type: none"> Renderer Codec Renderer Codec 	<ul style="list-style-type: none"> Format readers & sources File format writers Media Renderers Codecs
filesystem	<ul style="list-style-type: none"> Local http 			<ul style="list-style-type: none"> Data readers & sources
protocol	<ul style="list-style-type: none"> Common Rtsp Sdp Transport 			<ul style="list-style-type: none"> Protocol primitives & tools
audio	<ul style="list-style-type: none"> include fixptutil device resampler limiter mixer gaintool 			<ul style="list-style-type: none"> Audio device (sink) Audio tools
video	<ul style="list-style-type: none"> include vidutil site colorconverter 			<ul style="list-style-type: none"> Video site (sink) Video tools



Code Layout: intra-module organization

include { *.h	Interface & object code free definitions
pub { *.h	Library exported header files
*.cpp, *.c, *.h <Feature1 dir.> : <FeatureN dir.> :	Source code optionally subdivided into feature delineating subdirectories
Umakefil <target1makefil> : <targetNmakefil> :	Build instructions for One or more products of any type: Lib, dll, exe, res
Symbian.pcf sunos5.pcf wince-300-ppc.pcf ...	Platform specific build Instruction and per module configuration
platform { mac symbian unix win ... } → { *.cpp *.c, *.asm *.h *.cpp *.c, *.asm *.h	Platform specific source code
test	Unit test harness module
armi-rel32 thumb-rel32 armi-dbg32 thumb-dbg32 ...	Target platform separated output directories holding final products and intermediate object files



Build System Configuration

Helix
Media
Engine
Modules

ribosome



Build System Home

<p>build/bif-cvs/.../*.bif</p>	<p>Build information files:</p> <ul style="list-style-type: none"> -describe build targets -describe target dependencies -describe target inclusion/exclusion on per platform and profile basis -can be derived to allow customization without duplication (inheritance and overriding is supported)
<p>build/umakecf/*.cf</p>	<p>Build Configuration files:</p> <ul style="list-style-type: none"> -describe platform specific build tools -describe platform specific custom tools (e.g. installer package creation) -describe code base configuration macros (e.g. HELIX_CONFIG_FIXEDPOINT) -define platform keywords (e.g. symbian, win32, linux...) -allow composition through inclusion
<p>build/umakepf/*.pf build/umakepf/*.pfi</p>	<p>Profile & Profile Include Files:</p> <ul style="list-style-type: none"> -define a feature set -List code base feature macros (e.g. HELIX_FEATURE_VIDEO) -allow composition through inclusion

Profiles and Modularization

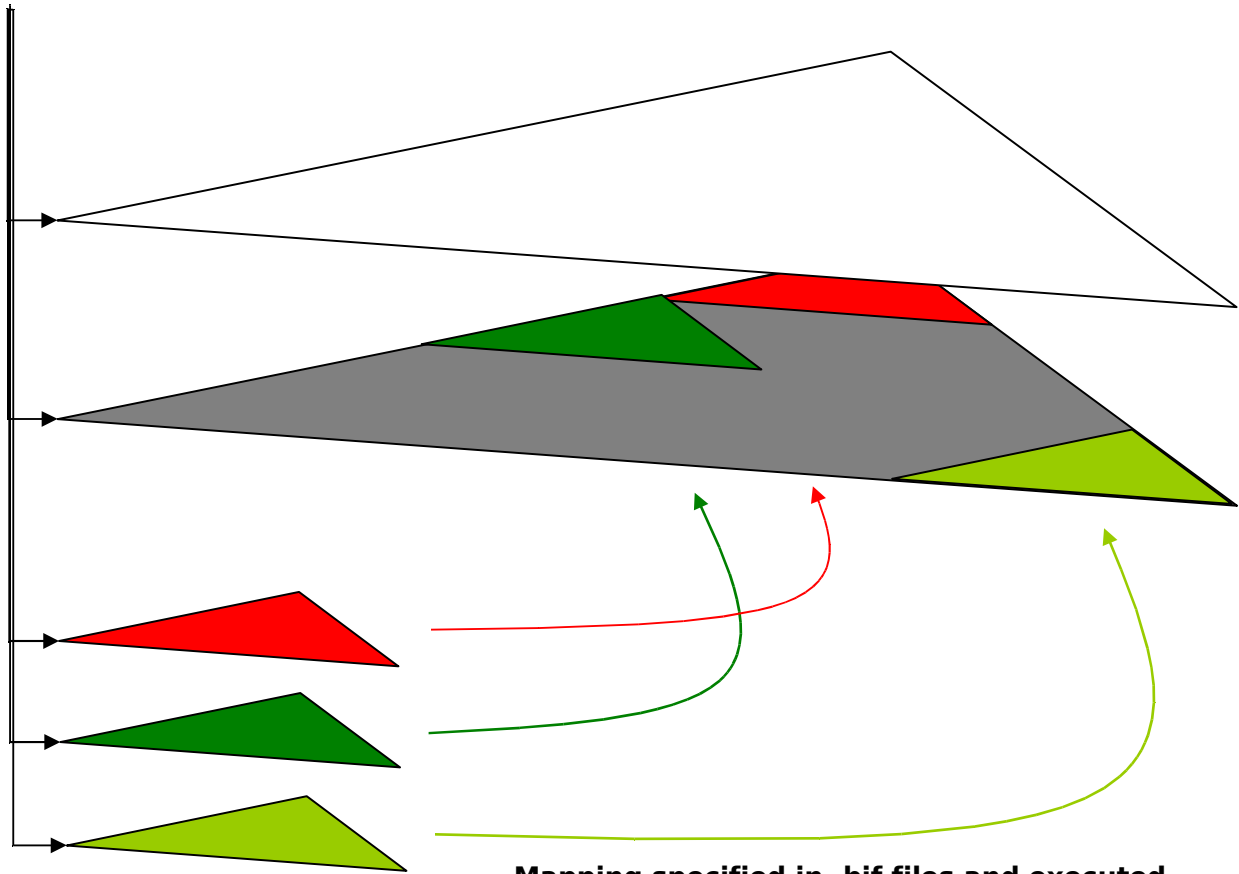
- Modularization – coding pattern solutions:
 - Abstract Interfaces
 - Derivation
 - Inline code selection (only where necessary)
- Module selection:
 - Target definitions in Build Information Files
- Module scalability
 - Conditional macros
 - Macro definitions are packaged into profile subsets (.pfi files) which are combined to form profiles (.pf files)

Restricted Modules

Inheritance
Direction



Source Tree Roots



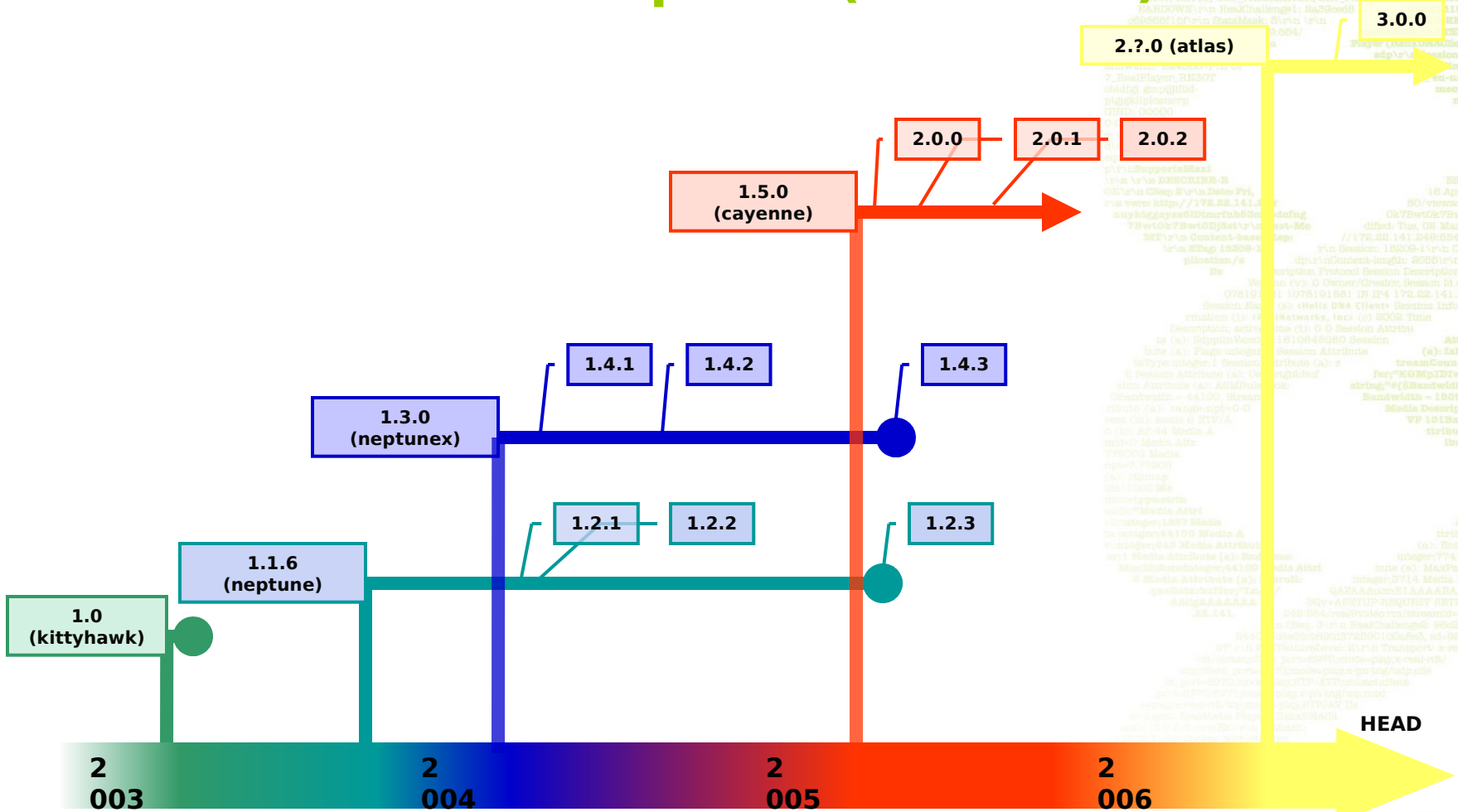
Public Tree:
common
datatype
protocol
...

Restricted Tree:
Container for Subtrees
of various access
Privileges:
common-restricted
datatype-restricted
protocol-restricted
...

**Subtrees of configurable
access privileges:**
rarvcode-audio
rarvcode-video
rarvcode-formprot
...

Mapping specified in .bif files and executed
by the build system upon CVS check-out

Helix code paths (branches):



Incremental Features in '05 and '06:

Helix DNA Client '05 - Cayenne

- 3GPP-Rel6
- PlayNow 1.0
- TrueLive
- IPv6
- TrickPlay
- Auto Bandwidth Detection
- Progressive Download Enhancements

Helix DNA Client '06 - Atlas

- Atlas Architecture
- H.264
- JSR135 support
- PlayNow2.0
- Windows Media

Incremental Features in '07:

Helix DNA Client '07 - Brego

- WMDRM
- Flash Integration
- Rhapsody Module
- Capture – Helix Producer integration
- PlayNow 3.0 (3GPP-PSS standard based)

