CELF Meeting
January 26th

UHAPI

John Vugts

Agenda

- Introduction

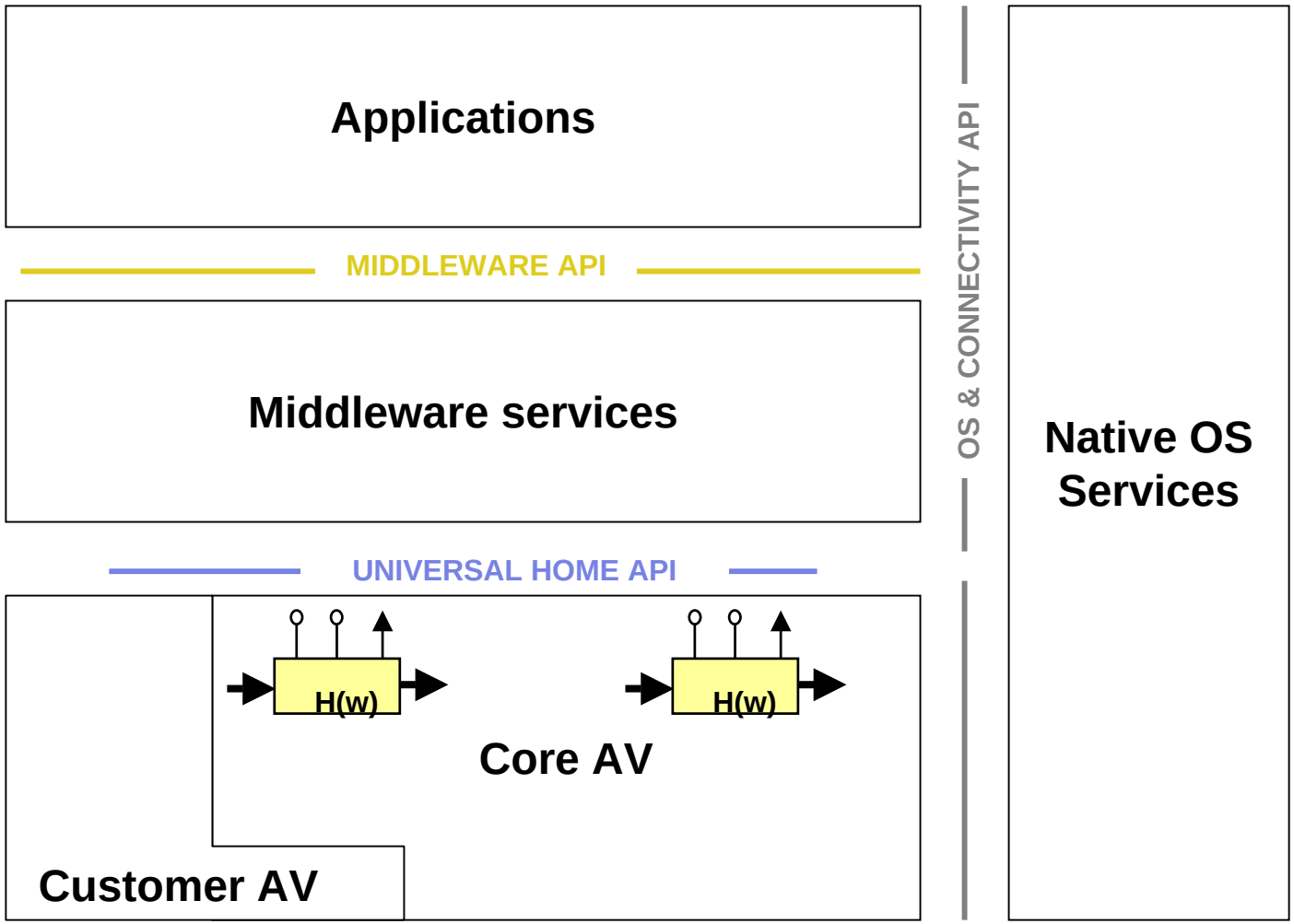
- UHAPI Scope
- UHAPI Characteristics
- UHAPI Concepts
- UHAPI Specification structure

- Overview of available Logical Components
- Walk through Logical Component
- Example Application code snippets

- Summary

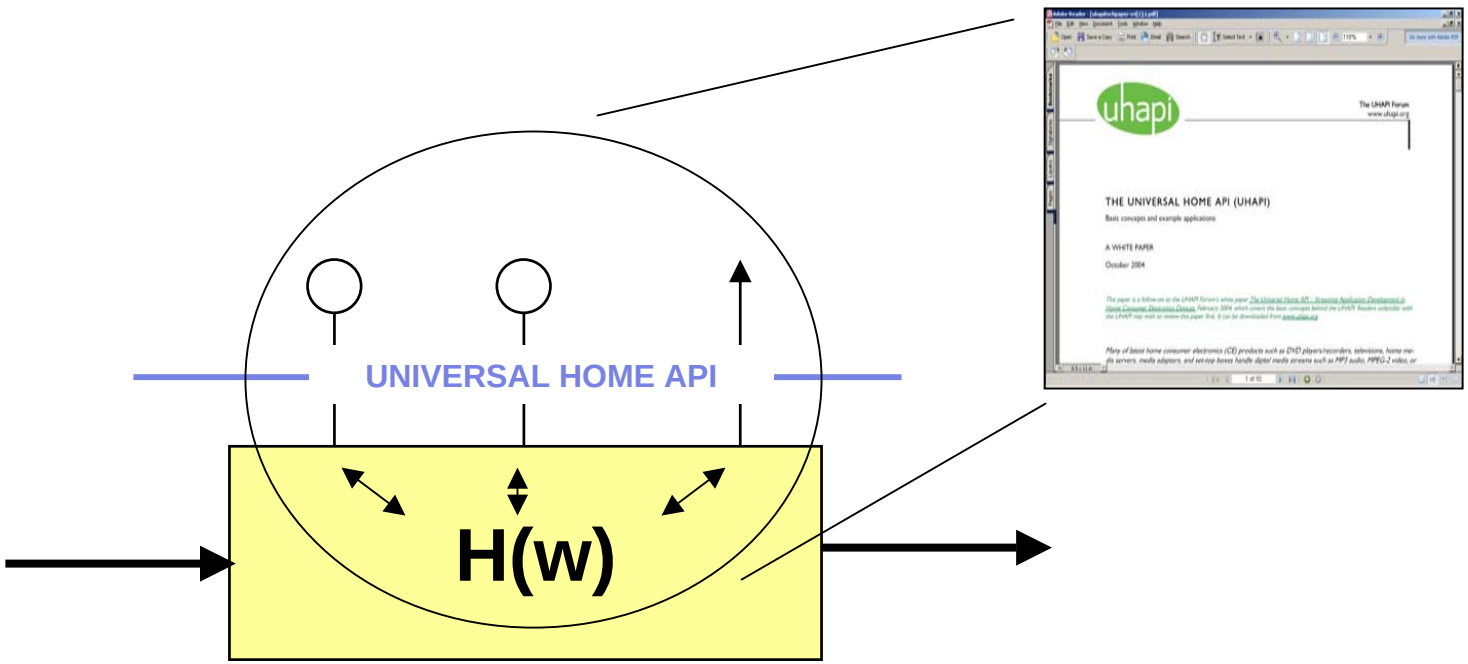
UHAPI Scope

UH Scope core and non core AV



UHAPI is an API to the MW

- Focus on runtime control by the Middleware (ISV).
- It does **not** specify e.g. **streaming interfaces.**
- Specification structure deals with diversity.



UHAPI Characteristics

UHAPI Characteristics (1)

- **API family for AV functionality (analog, digital)**
- **Designed with a middleware view in mind**
  - Domain model supports a large set of middlewares
- **Provides a consistent, orthogonal, coherent set of interfaces**
  - Well defined (syntax and semantics)
  - Easy to understand / debug by the users (functional interface)
  - Minimizes support costs
    - Fewer people involved (aspects touched) if one aspect changes
    - Reduced maintenance costs
  - Subset of provided interfaces can be used to manage diversity
  - Dependency between middleware and platform is very explicit
- **Binary Interface**
  - Binary releases
  - Enables partial downloads (dynamic binding)

UHAPI Characteristics (2)

- **Hardware and implementation independent interface**
  - Allows freedom in implementation and evolution
  - Support both HW and SW streaming
  - Support both on and off chip peripherals
  - Does not expose the physical software component architecture
- **Processor independent**
- **Used processor transparent to client**
  - Support efficient RPC implementation

UHAPI Characteristics (3)

- **Uses standard mechanisms**
  - Notification (runtime binding)
  - Error handling
  - Connection management (simple to program)
- **Uses standard COM like mechanisms**
  - IUnknown
    - QueryInterface
    - AddRef & Release
  - uhCom_CreateInstance
  - v-tables
  - GUIDs
- **“Interface grouping” and diversity mechanisms**
  - Interfaces
  - Roles (group of interfaces)
  - Logical Components (group of roles)
  - Platform instance (group of logical components) “Product”

UHAPI Concepts

Concepts Agenda

- **Logical components**
- **Connection management**
- **Framework v.s. platform instance**
  - Diversity elements
- **Interface technology**
  - 3rd party binding
- **Interface navigation**
- **Notifications**
- **Error handling**
  - Strong typing
- **Execution architecture**

Concepts Agenda

- **Logical components**
  - Logical v.s. physical
- **Connection management**
  - 3rd party binding
- **Framework v.s. platform instance**
  - Diversity elements
- **Interface technology**
- **Interface navigation**
- **Notifications**
- **Error handling**
- **Execution architecture**

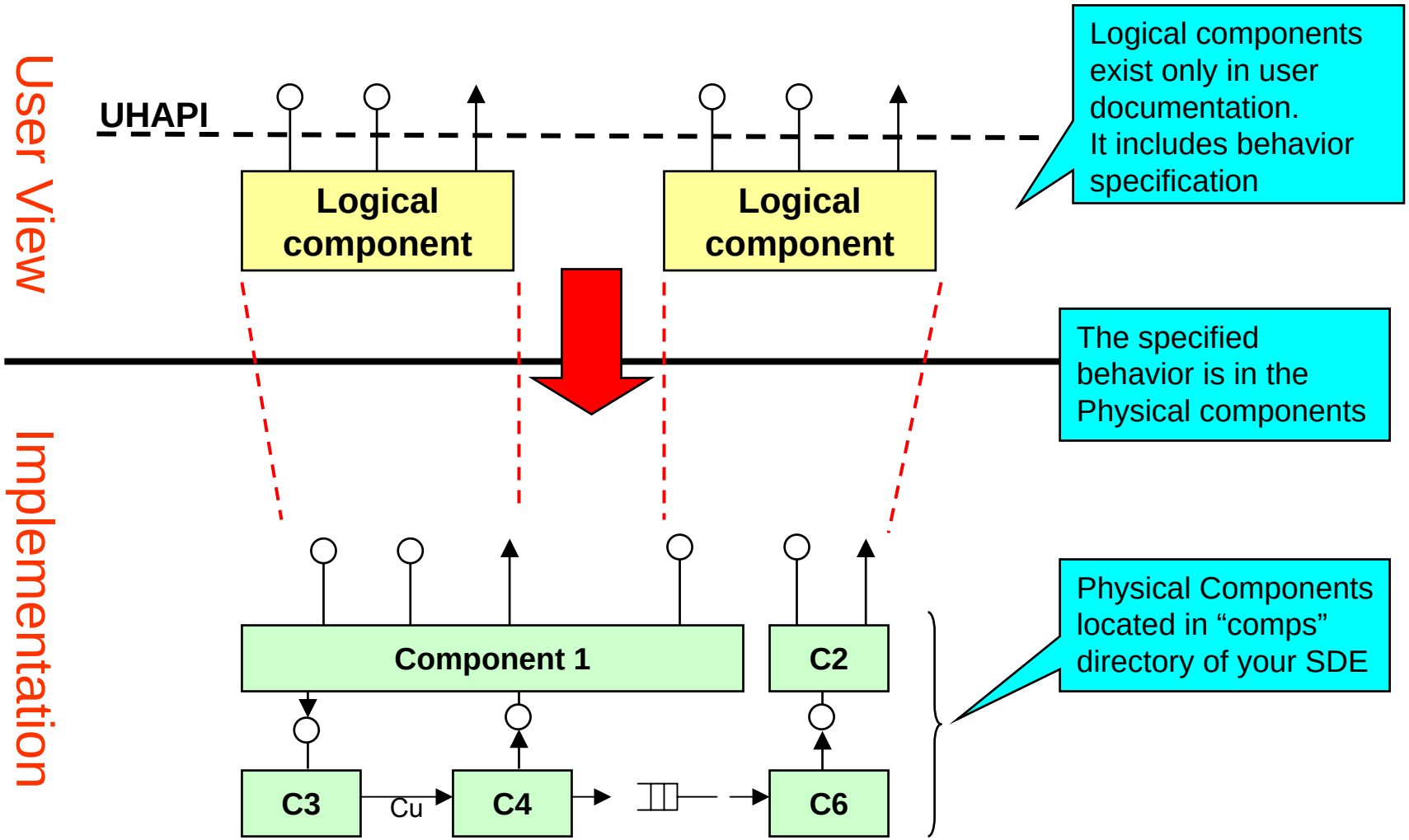
Logical v.s. Physical components

- **Logical components (part of the UHAPI spec):**
  - Specification entities.
  - Used to compose a logical model.
  - Terminology familiar to the clients of the UHAPI.
  - Related to interface suites.
  - One control aspect for the MW (e.g. Tuner, ATSC dec, Video feat.)

- **Physical components (NOT part of the UHAPI spec):**
  - Implementation entities.
  - Used to implement the functionality required.
  - Terminology familiar to domain experts
  - Mapped onto logical components, but not necessarily one to one.
  - One implementation aspect for the platform (e.g. SW streaming FW).

- **This to enable HW independence**

Logical v.s. Physical components



Concepts Agenda

- Logical components
  - Logical v.s. physical
- **Connection management**
  - 3rd party binding
- Framework v.s. platform instance
  - Diversity elements
- Interface technology
- Interface navigation
- Notifications
- Error handling
- Execution architecture

Connection Management

- A UHAPI client selects the streaming setup of the platform instance by selecting a use-case.

```
err = gpConnMgr->SelectUseCase(uhConnMgrSdkDemo_UcSingleWindow);
```

- The connection manager creates all streaming components and connects them (initializes system).

- A client initially obtains a logical component interface.

```
err = gpConnMgr->GetInstance(uhConnMgrSdkDemo_CVmix, UHIID_uhIVmix,
        &gpVmix);
```

- It abstracts the client from the Physical Components that need to be connected

Connection Management

- It relieves the client from difficult and HW specific task of setting up components (priorities, buffer sizes etc.)
- It handles transients in initializing the platform.

- The connection manager is platform instance specific.
- Use cases are NOT defined by UHAPI
- Systems will exist that have only one use case (initialize).

```
err = gpConnMgr->SelectUseCase(uhConnMgrSdkDemo_UcSingleWindow);
err = gpConnMgr->GetInstance(uhConnMgrSdkDemo_CVmix, UHIID_uhIVmix,
        &gpVmix);
```

Concepts Agenda

- Logical components
  - Logical v.s. physical
- Connection management
  - 3rd party binding
- **Framework v.s. platform instance**
  - Diversity elements
- Interface technology
- Interface navigation
- Notifications
- Error handling
- Execution architecture

UHAPI is an API framework to the MW

Platform independent code

Platform dependent code

UHAPI specification

Platform Instance specification A

Platform Instance specification B

Products A

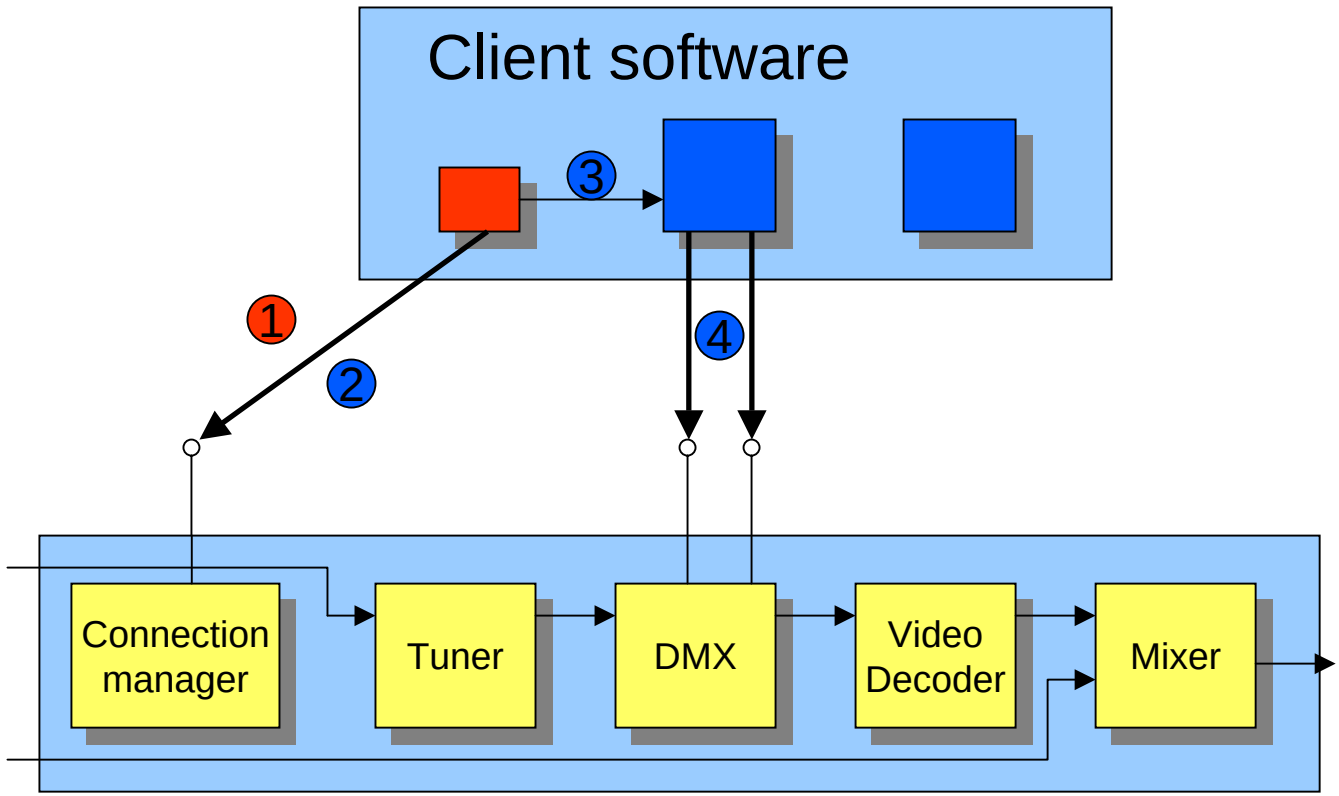
Products B

UHAPI Specification Diversity

- **Platform Instance document specifies:**
  - Which logical components (instances) are supported
  - Which use cases are supported
  - Diversity options of the Logical Components are specified:
    - Availability of optional interfaces
    - Parameter ranges
    - Available resources (e.g. number of section filters)
    - Other e.g. which standards are supported
  - Resource usage figures (CPU cycles, memory, memory bandwidth, algorithms used, ...)
- **This all supports different hardware, and make the interfaces hardware independent.**

Portability: Isolate platform dependencies

```
SelectUseCaseYyyy(p1, …);
GetInstance(UHCMG_DEMUX1,UHIID_uhIDmxCapability, &pDmx)
Bind(idx, UHIID_uhIDmxCapability, pDmx);
SetFilterParam(…)
```

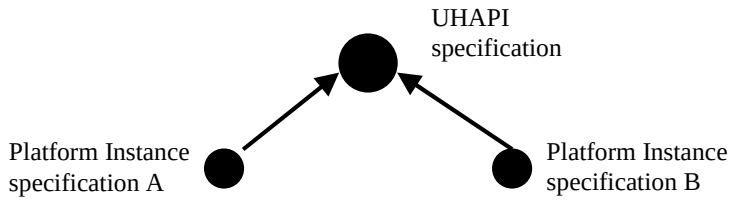


Diversity, platform independence

```
// Get current brightness value
err = gpVfeatBcs->GetBrightness(&gBrightness);
assert (err == UH_OK);

// Get brightness range
err = gpVfeatBcs->GetBrightnessRange(&gBrightnessMin, &gBrightnessMax);
assert (err == UH_OK);

// interprete the value
// (gBrightness - gBrightnessMin) / (gBrightnessMax - gBrightnessMin)

// increase brightness by 10%, assume at least 10 steps
assert(gBrightnessMax - gBrightnessMin + 1 >= 10);
gBrightness = gBrightness + (gBrightnessMax - gBrightnessMin) / 10;

// clip the value
if (gBrightness > gBrightnessMax)
    gBrightness = gBrightnessMax;

// set the value
err = gpVfeatBcs->SetBrightness(gBrightness);
assert (err == UH_OK);
```

Get the value

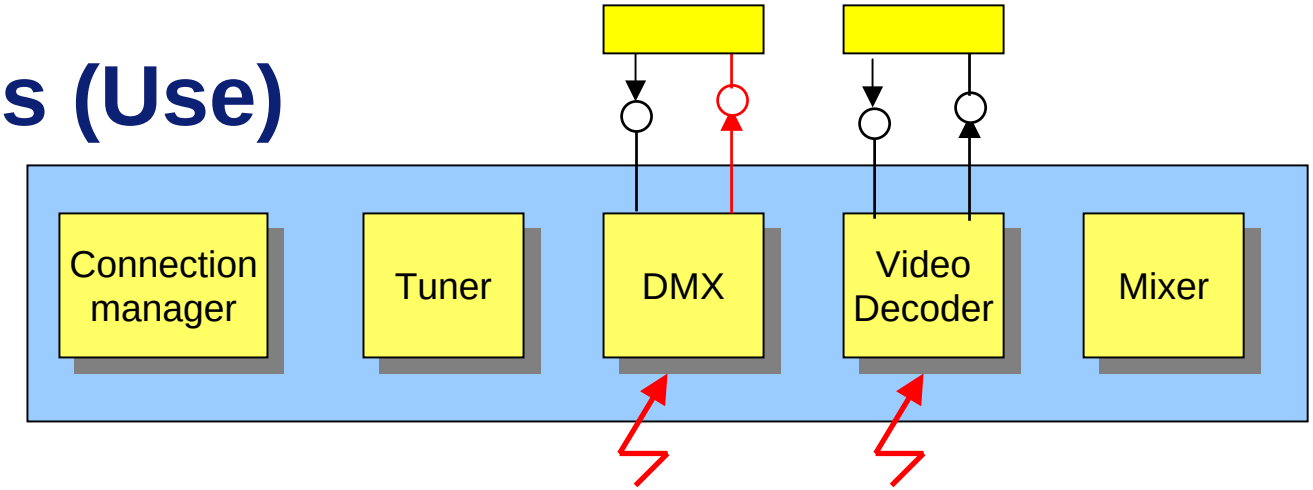
Deal with diversity



Concepts Agenda

- Logical components
  - Logical v.s. physical
- Connection management
  - 3rd party binding
- Framework v.s. platform instance
  - Diversity elements
- Interface technology
- Interface navigation
- **Notifications**
- Error handling
- Execution architecture

Notifications (Use)



- A client subscribes a notification interface using the ***Subscribe*** methods on the corresponding control interface (dynamic binding).
- Subscriptions are changed using bit masks.
- Multiple clients can subscribe to a control interface.
- A client can subscribe its notification interface at multiple control interfaces. A cookie can be used to distinguish between the various notifications.
- Notification interfaces have a separate method for each event.
- This enables efficient use of HW in CE devices.

Concepts Agenda

- Logical components
  - Logical v.s. physical
- Connection management
  - 3rd party binding
- Framework v.s. platform instance
  - Diversity elements
- Interface technlogy
- Interface navigation
- Notifications
- Error handling
- **Execution architecture**

Execution Architecture

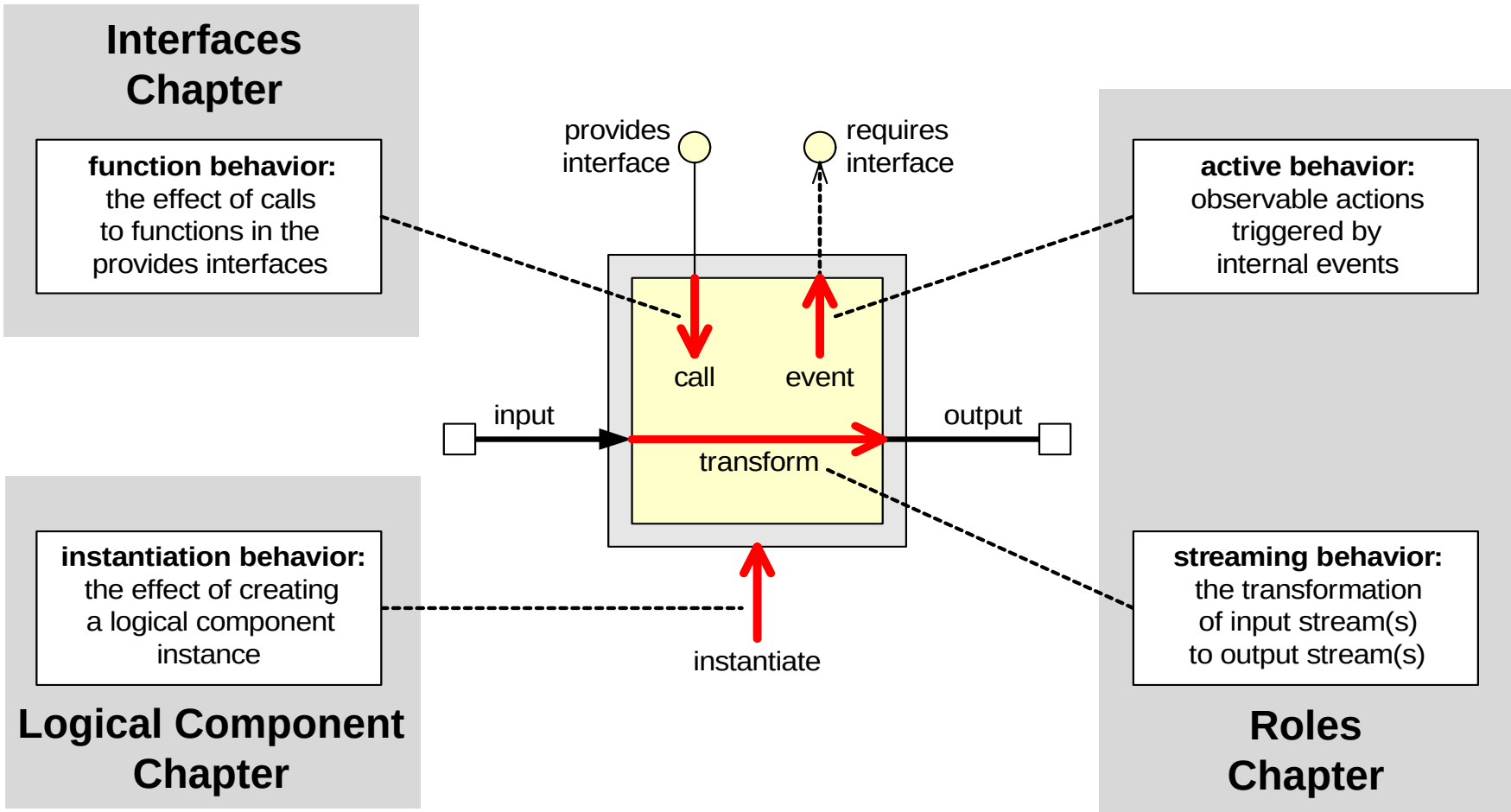
- **Logical Components are in principle single threaded**
  - Allows for simple implementation
  - In general it is not required to make everything thread safe
  - Where required it is multi-thread safe
- **Get methods are by default thread-safe even when the interface they are part of is not thread-safe.**
  - Easy to implement (stop scheduling or interrupts)
- **Access to an interface is by default single-threaded.**
  - To keep it as simple as possible
- **Notification methods are serialized**
  - This makes the client implementation simpler (single threaded)
- **Notification methods are not allowed to block.**

Execution Architecture (2)

- **Thread safeness can be specified on:**
  - Method level
  - Interface level
  - Role level
  - Logical Component level

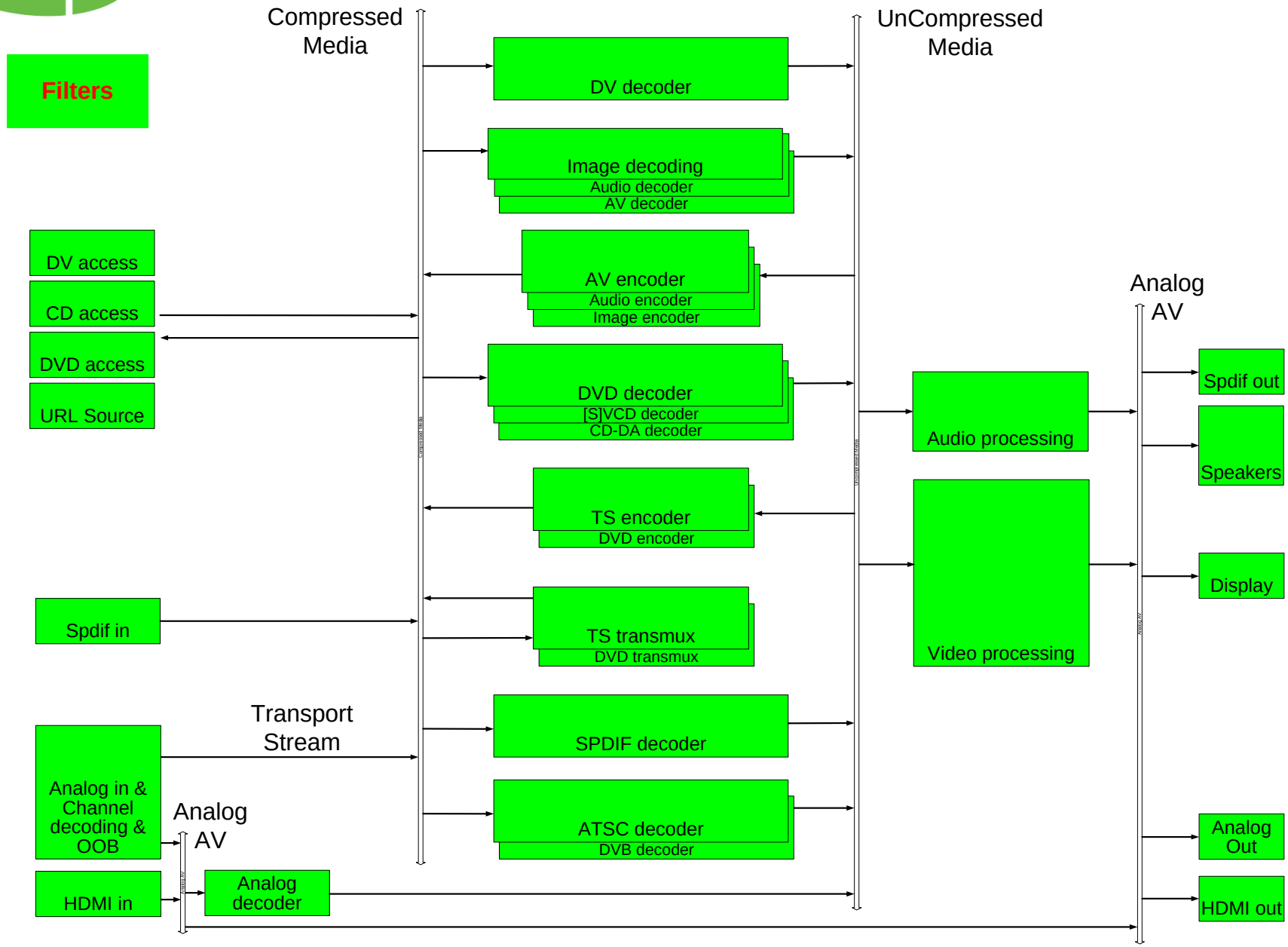
- **Single client view requires client to synchronize, but this is required anyway**
  - What would be the semantics of multiple threads calling a set method?

Specification views



Overview of available Logical Components

Universal Home Application Programming Interface



UHAPI 1.0 contents

General documents (7) :

API Specification Reader's Guide
API Naming Conventions
Error Handling
Execution Architecture
Notification
Qualifiers Quick Reference
API Evolution Rules

Type specifications (2) :

Basic Types
Global Types

API specifications (50) :

Front End Components (12)

Analog Audio & Video Demodulation
Analog AV Input
Anti Aging
Analog Audio Decoding
Channel Decoding
RF Amplification
Out Of Band Tuning & Demodulation
Signal Strength
Tuning
HdmiIn
SPDIF-in
VBI Slicing

Decoders/Encoders (5)

ATSC Decoder
Image Decoding
SPDIF Decoding
STC Decoding
Transport Stream Demultiplexing

Video Processing Components (15)

Ambient Level
Analog Video Decoding
Analog Video Encoding
Analog Video Encryption
Basic Video Featuring
Black Bar Detection
Color Transient Improvement
Dynamic Noise Reduction
Histogram Modification
Noise Measurement
Scan Rate Conversion
Sharpness Enhancement
Sharpness Measurement
Video Color Enhancement
Video Mixing

Audio Processing Components (10)

Audio Automatic Volume Leveling
Audio Bass Enhancements
Audio Dynamic Range Control
Audio Mixing
Audio Noise Generation
Audio Program Selection
Audio Volume Control
Equalizing
Speaker Set /Headphones
Output Configuration

Generic (8)

Analog AV Output
SPDIF-out

Connection Management
Fatal Error Handling
I am Alive
Pin Objects
Unknown
URL Source

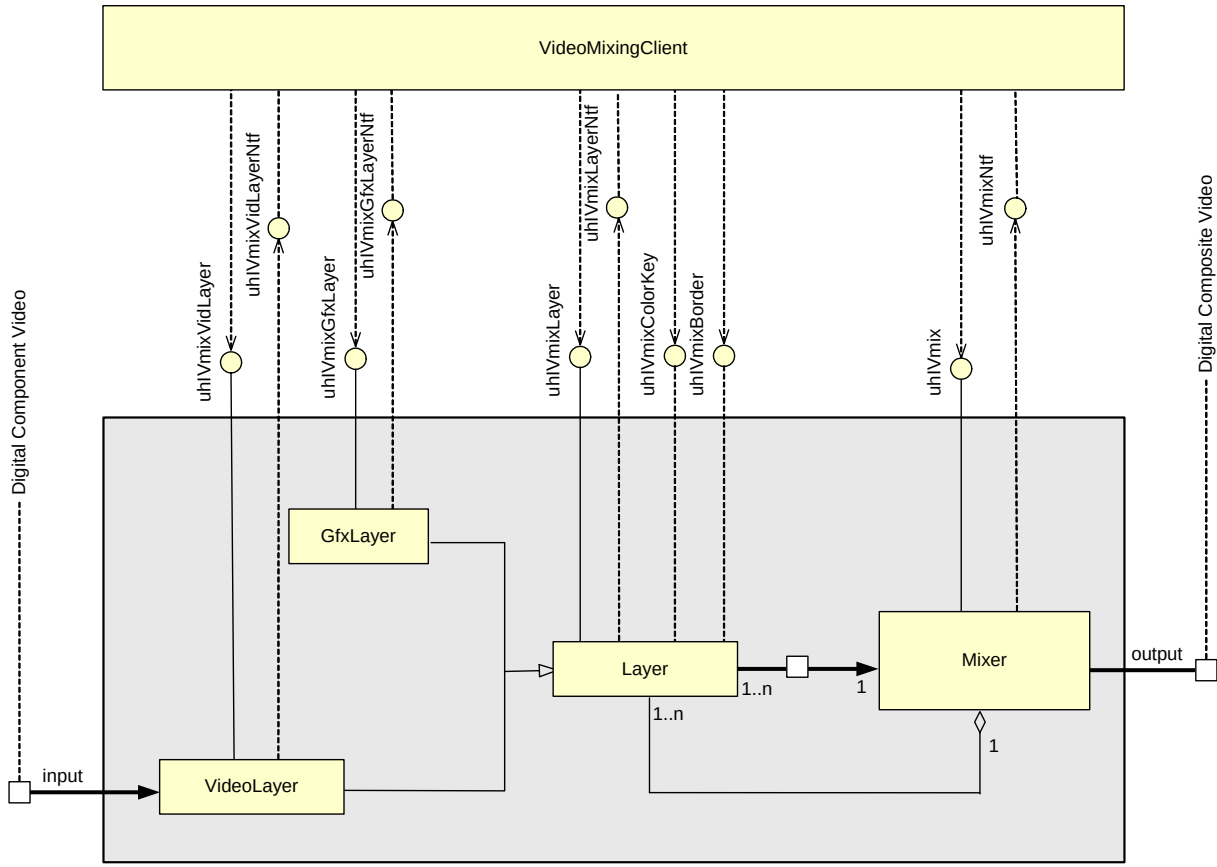
Walk through Logical Components

ATSC Decoder
Video Mixer

Example Application code snippets

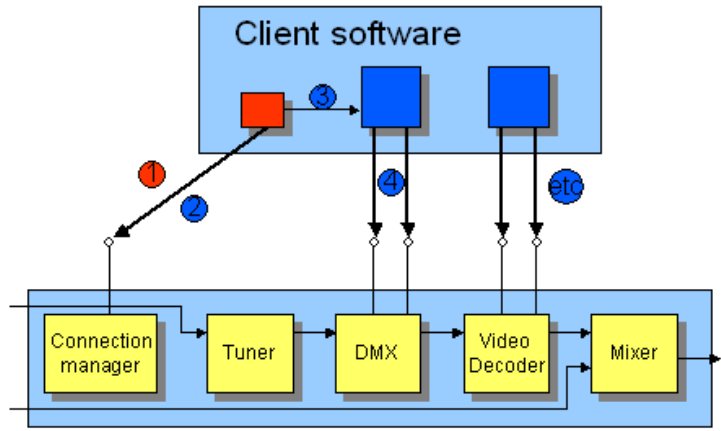
C++ example
Main usage is in plain C
Any language is possible

Explicit dependencies on interfaces



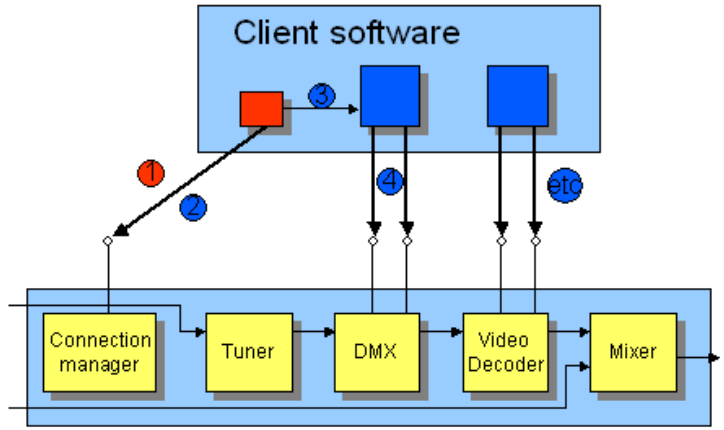

```
#include "uhIVmix.h"
#include "uhIVmixLayer.h"
#include "uhIVmixVidLayer.h"
#include "uhIVmixGfxLayer.h"
```

Bring up the platform



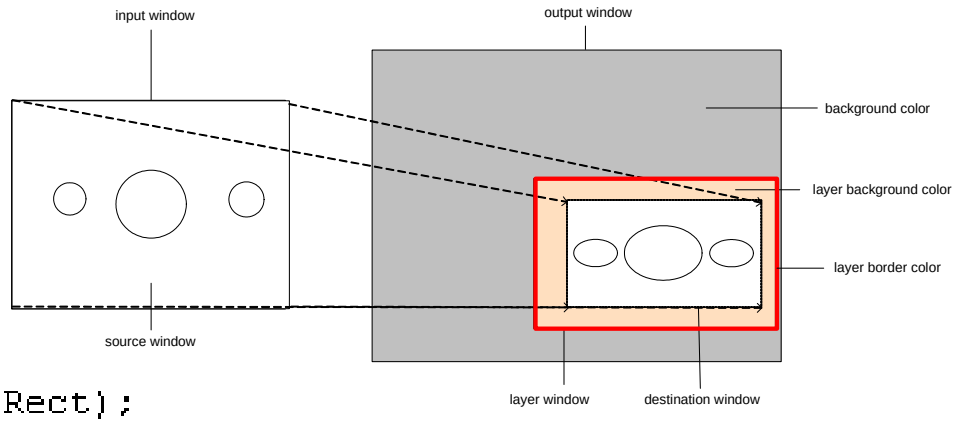

```
err = uhPlatformInit();
assert (err == UH_OK);

// Create a connection manager instance
std::cout << "Creating Connection Manager Instance" << std::endl;
err = uhComCreateInstance(UHCLSID_uhConnMgrSdkDemo, UHIID_uhConnMgrSdkDemo,
            &gpConnMgr);
assert (err == UH_OK);

// Select the "single window" Use Case
std::cout << "Selecting Single Window Use Case" << std::endl;
err = gpConnMgr->SelectUseCase(uhConnMgrSdkDemo_UcSingleWindow);
assert (err == UH_OK);
```

Get Video Mixer, and set the background




```
// Get the current Use Case's Vmix interface
err = gpConnMgr->GetInstance(uhConnMgrSdkDemo_CVmix, UHIID_uhIVmix, &gpVmix);
assert (err == UH_OK);

// Set the background colour of the Vmix
colour.redOrY = 30;
colour.greenOrU = 144;
colour.blueOrV = 255;
err = gpVmix->SetBgColor(colour);
```

Set the windows of the Video Mixer



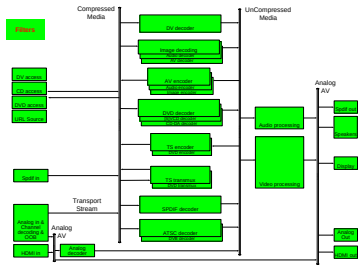

```
// Select entire input
inputRect.ul.x = 0;
inputRect.ul.y = 0;
inputRect.lr.x = gVidLayerMaxWidth;
inputRect.lr.y = gVidLayerMaxHeight;
err = gpVmixLayerV->SetSrcWindow(inputRect);
assert (err == UH_OK);

...

err = gpVmixLayerV->SetLayerWindow(layerRect);
assert (err == UH_OK);

err = gpVmixLayerV->SetDstWindow(destRect);
assert (err == UH_OK);

err = gpVmixLayerV->ActivateNewWindowSettings(True, 1000);
assert (err == UH_OK);
```

Set a PIP layer

Set a PIP destination

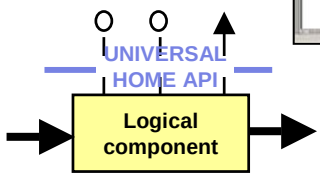
Smooth zoom within 1 Second

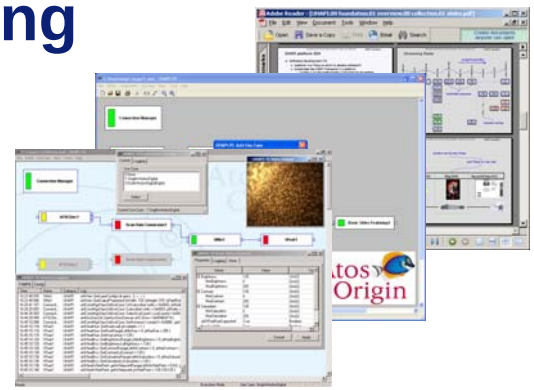
Summary



Summary

- UHAPI is complete, consistent and supports entire CE products
- UHAPI documentation is of high quality
- UHAPI is platform independent
- Member companies add tools and training
- Download UHAPI at www.uhapi.org





Questions?

Thanks for your attention!