PHILIPS

sense and simplicity

SPACE Porting Guide – Public Release

jointSPACE v1.0

October, 2009

joint space

split application architecture

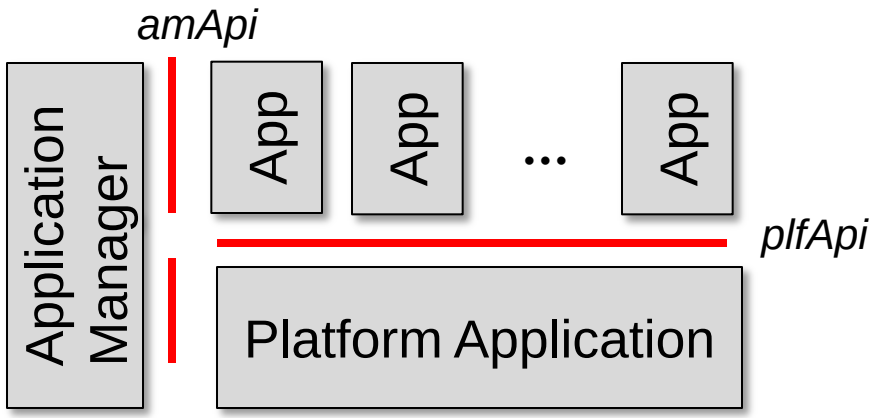
PHILIPS

joint space
split application architecture

Table of Contents

SPACE Concept

SPACE Decomposition



- System consists of applications, each a separate process
  - Using a shared Linux and DirectFB infrastructure
- Application Manager
  - System level responsibility
- Platform Application
  - FrontEnd (Tuner, Demux), A/V decoding and rendering, USB, IP, etc
  - Accessed by applications via plfApi

Linux

- The open source operating system used
  - Application support: processes, shared libraries, memory management
  - Hardware support: device drivers
  - Delivered by the supplier, including infrastructure (USB, IP, …)

- Linux available under GNU license conditions
  - Precautions needed to avoid IP infringement or loss of license income
  - Explicit exception to the GNU license for user-level applications

SPACE Terminology

- Library:
  - Maps to the well known definition of code libraries
  - Implements a set of functions, can call external functions
  - Format is typically ****.a*** or ****.so***

- Application:
  - Executable providing functionality (instantiated via a Linux process)
  - Format is typically ****.elf***

- Process:
  - A Linux process is a running instance of an application

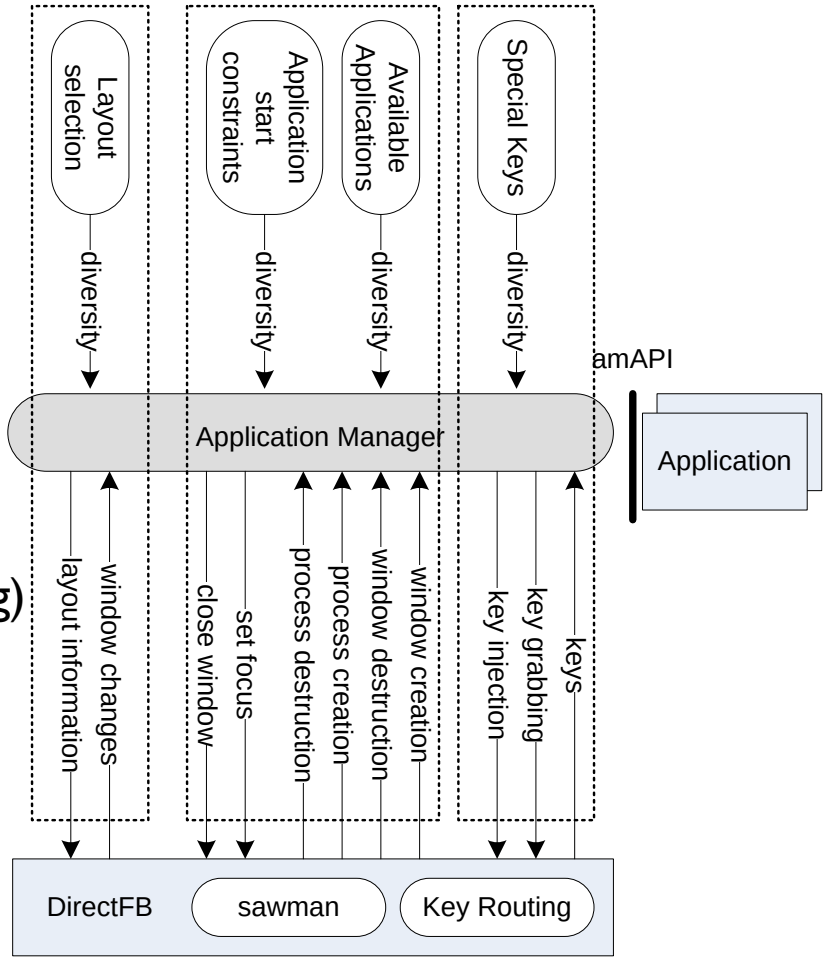
- Client
  - A process using the platform application and application manager

DirectFB

- Windowing framework
  - Key routing to the focused window and key grabbing
  - Window manager handles layout and focus of application windows
- Graphics engine
  - 2D accelerated graphics engine
  - Sharing graphics resources and video memory
- SaWMan
  - Callbacks to Application Manager for Linux process creation/destruction
  - Windows creation/destruction/reconfiguration
  - Sending window events directly to application
- Fusion
  - Base for IPC
- FusionDale
  - Event mechanism with subscription

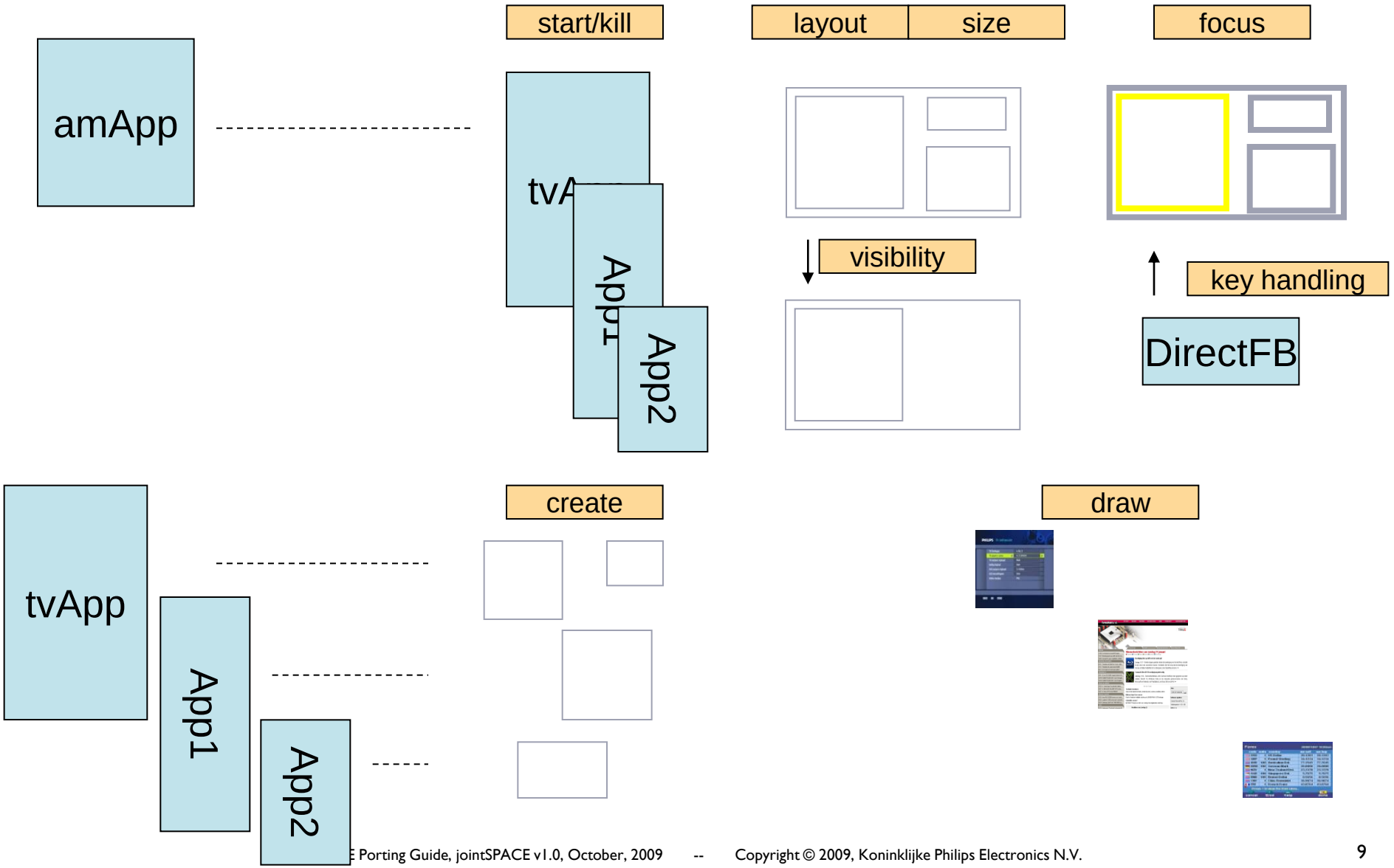
Application Manager

- Implements System Level behavior
  - User space start and shutdown
  - Window Layout and visibility
  - Multi-Window control logic
  - System level key management
    - For keys that are not routed to the window in focus (using key grabbing)
  - Setting focus to applications
  - Assigning A/V resources to applications
  - Sets the destination for A/V rendering



- Application Manager implements amApi
  - To communicate with applications

Application Responsibilities



Window Management

Layers | Z-order | Clients | Focus mgt

Application — SW

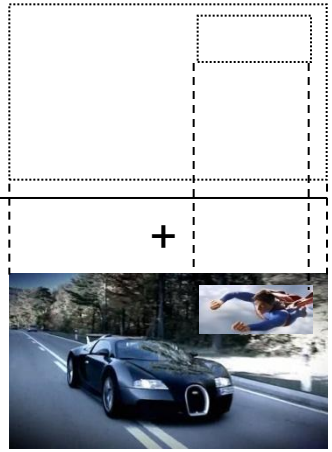
+

Border — SW

+

Video — HW

+



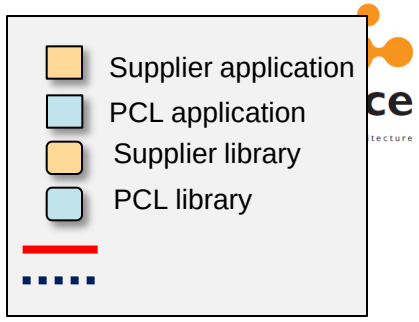
Platform Application

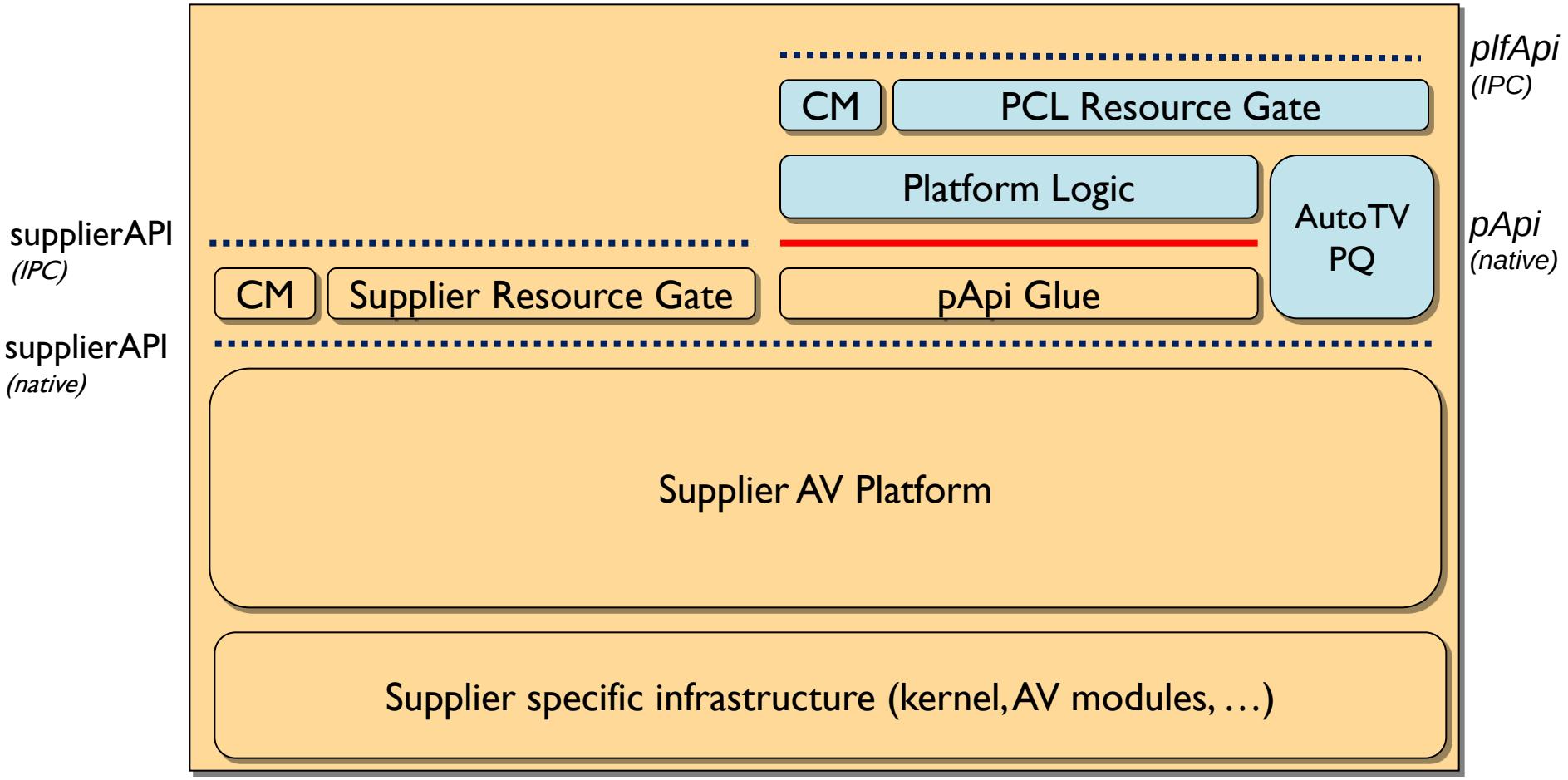
- Controls the hardware
  - Turns on hardware (power supplies, display, …)
  - Responsible for audio and video rendering
  - Can inject received RC keys into DirectFB

- Platform application has 2 distinct APIs
  - A control API to be used by clients
    - Implementation by PCL, used by all PCL SPACE applications
  - A clean porting API
    - Internal API towards HW suppliers to implement glue by suppliers

PHILIPS

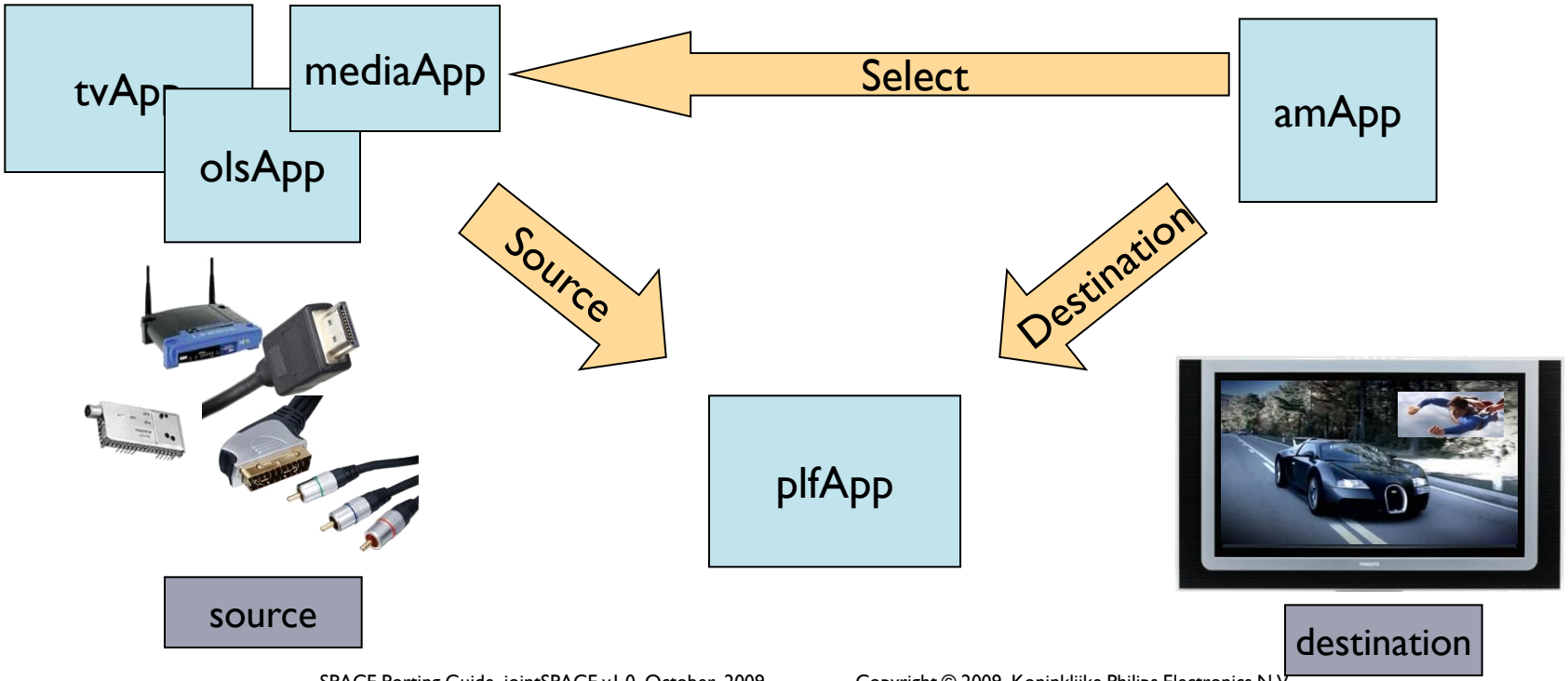
Joint Platform Application Architecture



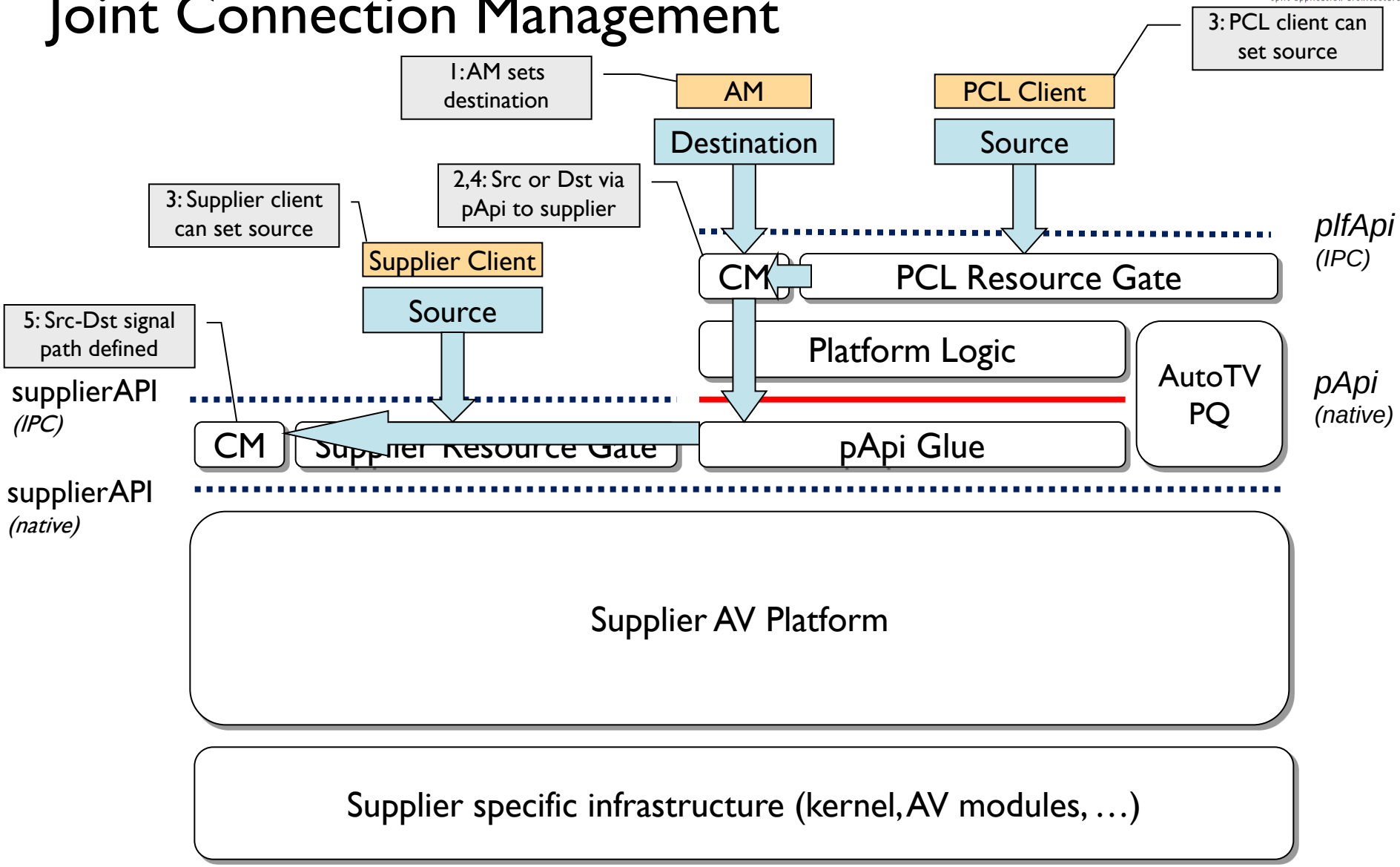


Connection Management (CM) Concept

- Application Manager sets destination, applications control source
  - Applications are not aware where the output is rendered (destination)
- The Platform Application compiles the optimal system use case
  - Based upon the destination and source parameters



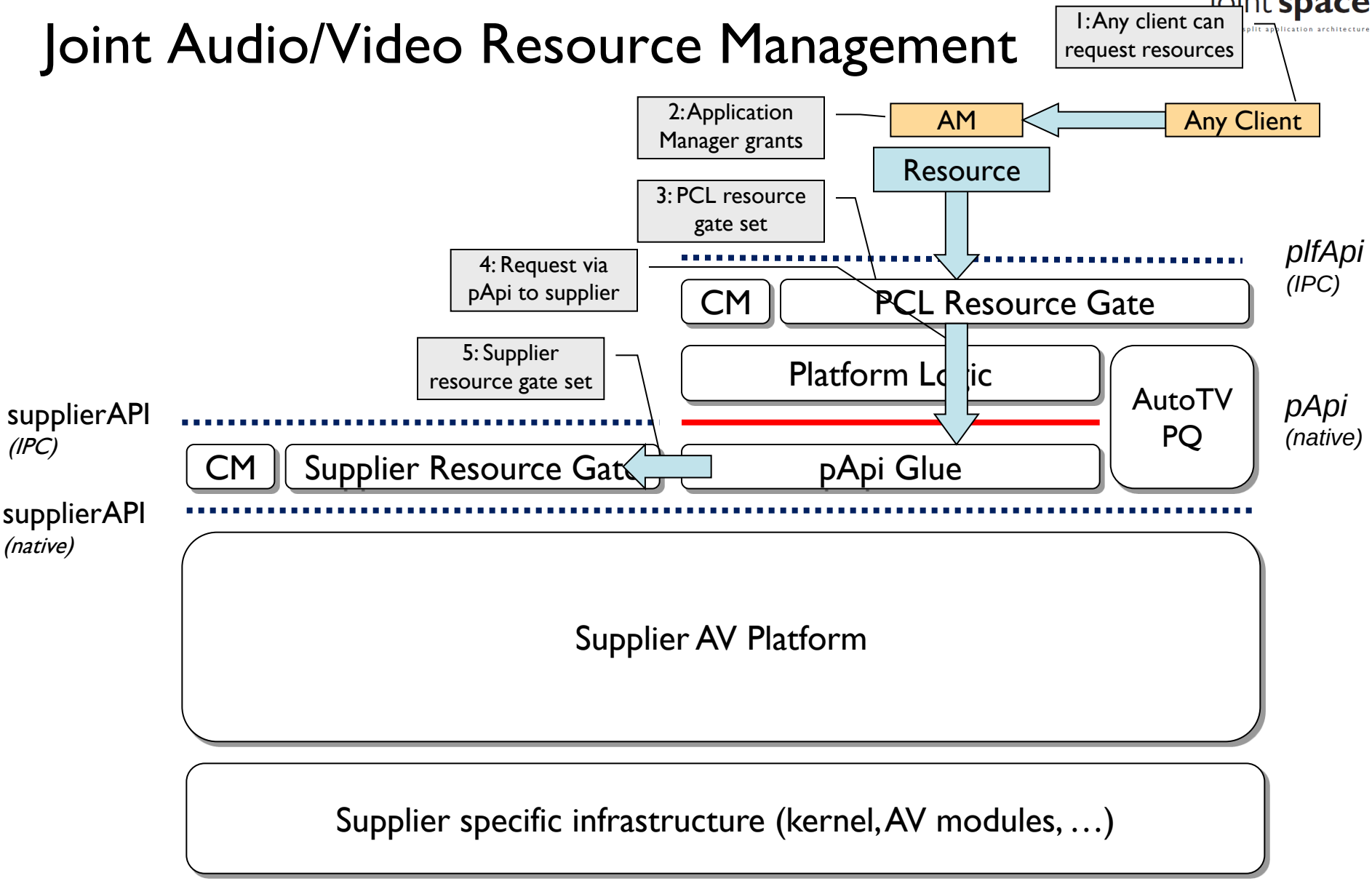
Joint Connection Management



Audio/Video Resource Management Concept

- The control API is partitioned in resource groups
  - A resource group is a set of interfaces
  - Access to the interfaces is guarded by a resource gate
- Applications can request a resource group dynamically
  - Application Manager informs all applications of changes in ownership
  - Applications must deal with unexpected loss of a resource
  - Applications can always listen to events from interfaces (subscription)
- Resources are assigned to a window ID
  - Windows created via DirectFB
  - Video windows are input only windows (border windows)

Joint Audio/Video Resource Management

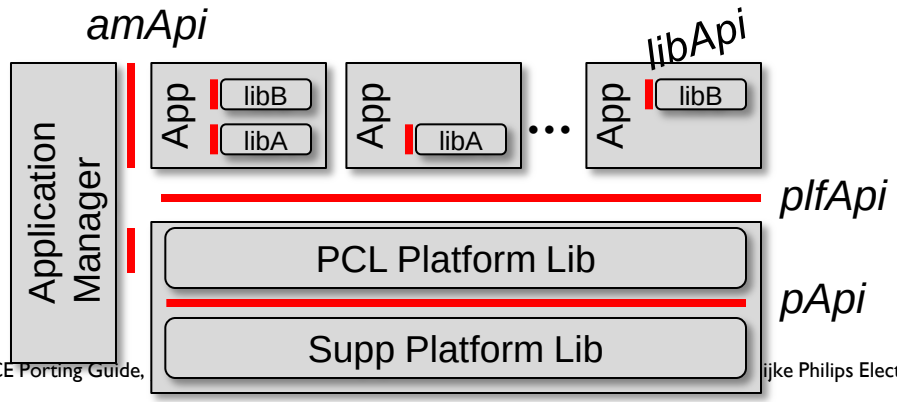


API Concepts

jointSPACE Porting API

- APIs between SPACE building blocks are called the "Porting API"
  - API from *supplier* platform application (supplierApi) not part of Porting API
  - API from *PCL* platform application (plfApi) not part of Porting API

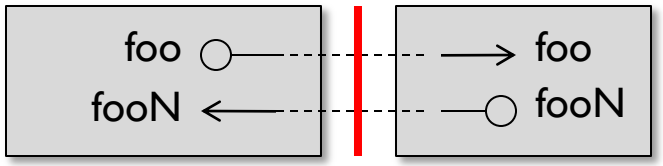
- Porting API consist of
  - amApi: between Application Manager and Applications
  - pApi: between PCL platform and Supplier platform library
  - libApi: APIs for shared data across Applications



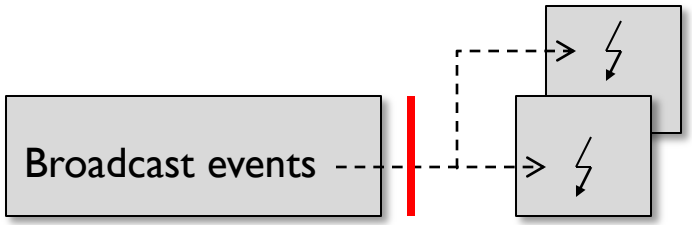
API Characteristics

- The Porting API is bi-directional
  - From both sides of the API a call can be done
  - Synchronous: callee blocked until return
  - Asynchronous: callee not blocked. Notification used on return (if needed)
  - In general cross process calls are asynchronous

- In some cases events are used instead of a call
  - Events are broadcasted to all subscribers
  - Notifications can also be multi-client. Part of the API definition.



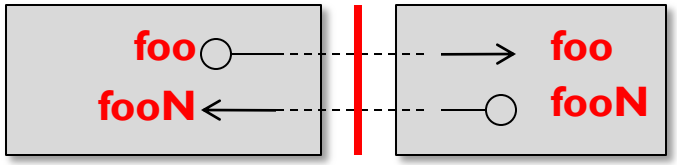
Application to application



Application to multiple applications

Interfaces

- API is partitioned in interfaces
  - An interface is a set of related functions
  - A short name (called prefix) is used to refer to an interface
- An interface is documented as a whole
  - Describing all functions and interactions
- Naming convention for functions of an interface <api>_<prefix>_<function>
  - Prefix of notification interfaces ends with an 'N'

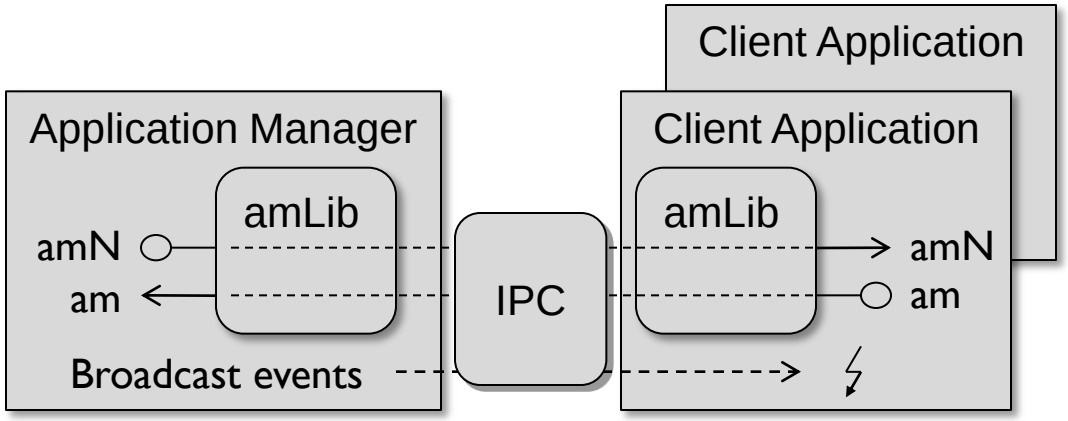


Application to application

amAPI

- Defines the interaction between Application manager and applications
  - Consist of a functional API and broadcast events
  - Asynchronous in both directions

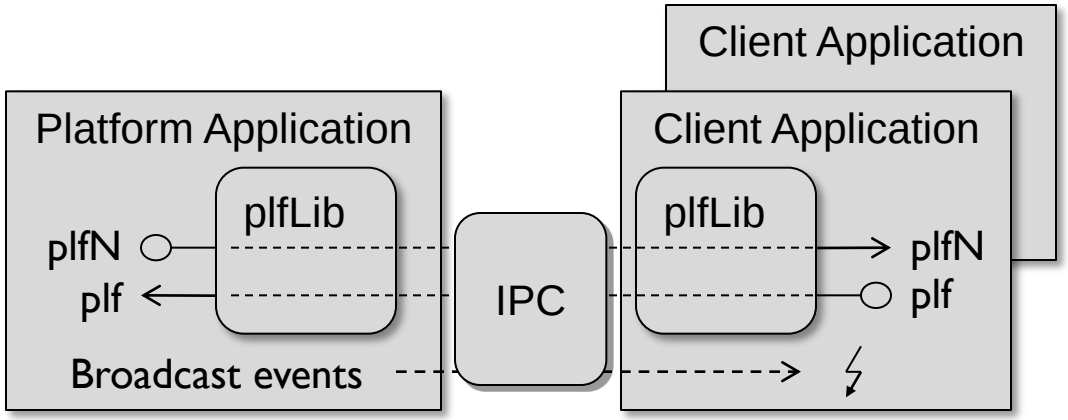
- The Application Manager can address individual applications
  - The Linux process ID (PID) will be used for this
  - The library (amLib) enables event broadcasts via IPC



plfAPI

- Defines the interaction between Platform Application and applications
  - Consist of a functional API and broadcast events
  - Can be synchronous from application to Platform Application
  - Always asynchronous from Platform Application to applications

- The library (plfLib) enables broadcast events via IPC
  - During initialization IPC connection is established



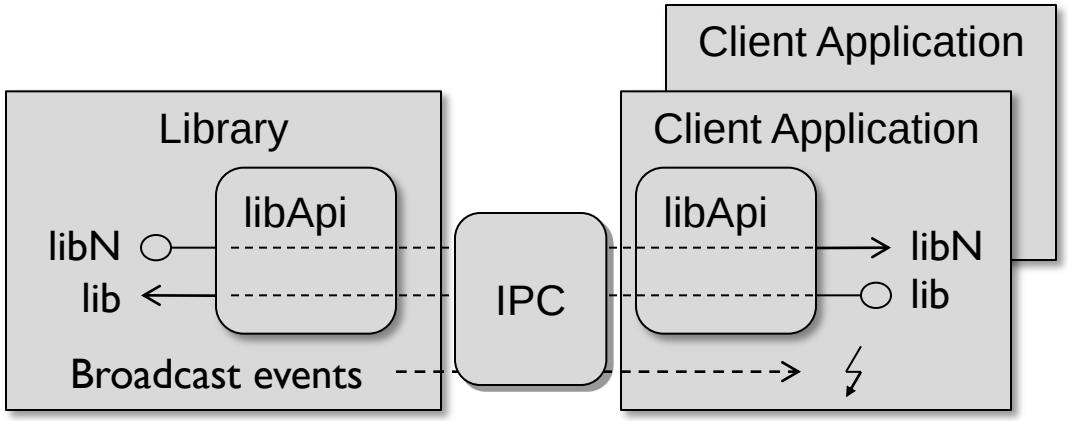
pAPI

- pApi defines the AV API as the platform porting layer
  - Derivative of plfApi (without resource management ID)
  - Must be implemented by suppliers
- Internal to Platform Application
  - No client application can access this API
  - Suppliers are the only users
- Must be used within process boundary
  - To avoid overload of IPC traffic between pApi and plfApi
  - Hence supplier glue and PCL platform code always in single process

Library API

- General concept of SPACE is that applications to not communicate
  - There is a design pattern to share data between applications
  - Implementation by suppliers or PCL

- Asynchronous in both directions
  - During initialization IPC connection is established
  - The library API (eg libApi) enables broadcast events via IPC

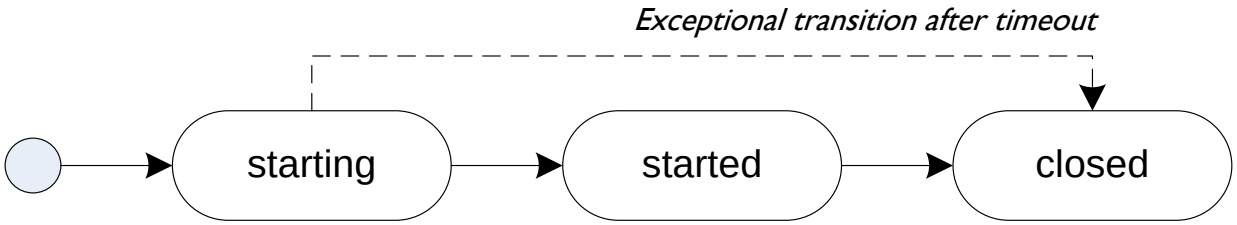


Application Lifecycle

Application Lifecycle

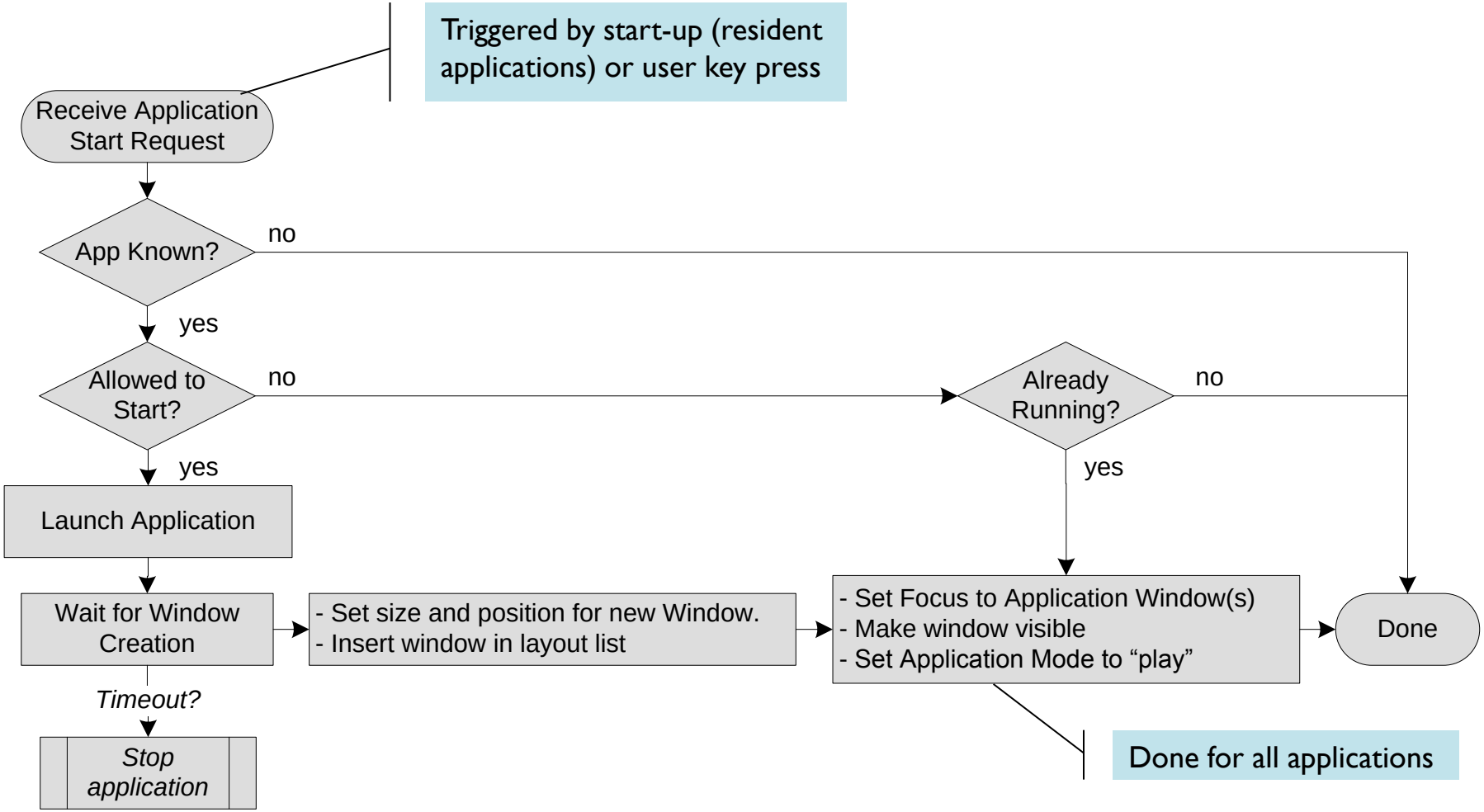
- The Application Manager controls the lifecycle of all applications
  - Application Manager starts and stops applications

- States and transitions



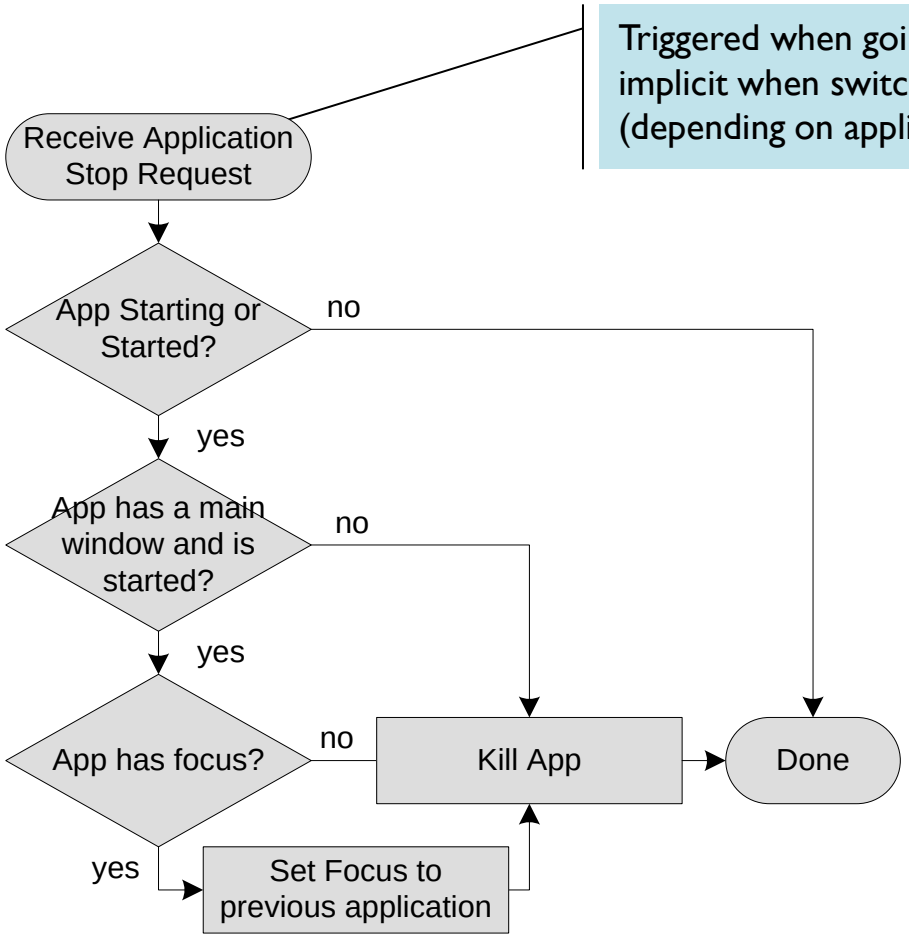
  - Starting: AM started the executable, PID known, awaiting DFB Window(s)
    - Application terminated if no windows are created (exception after timeout)
  - Started: DFB Window(s) created and known to AM
  - Closed: application terminated, i.e. Linux process exited.
    - AM is notified whenever a process exits (DirectFB/SaWMan)

Application Manager: Application Start



Application Manager: Application Stop

Triggered when going to standby (user key press), or implicit when switching to another application (depending on application manager policy)



Lifecycle Strategies

- There are two approaches to manage the lifecycle of applications
  - Starting and stopping on-demand
  - Keeping applications resident
- Starting on-demand takes start-up time when an application is needed
  - Can lead to poor system responsiveness
  - Only uses memory when the application is actually needed
- Resident applications consume memory when they are not active
  - Application manager sets visibility instead of starting/stopping → fast
  - Applications are started as a background activity at system boot
- Application Manager supports both strategies, diversity per application

Interfaces

- For starting and stopping, regular Linux functions are used
  - Starting: fork and execvp. Stopping: kill

- From DirectFB the following interfaces are used
  - IDirectFB
    - Main interface of DirectFB. Used to obtain all other interfaces.
  - IDirectFBWindow
    - Control appearance and focus, position and z-order, event and surface access
  - ISaWManManager
    - Installs notifications for application and windows changes
  - IFusionDale
    - Main interface to access other Fusion interfaces
  - IFusionDaleMessenger
    - Register events and listeners

Application States

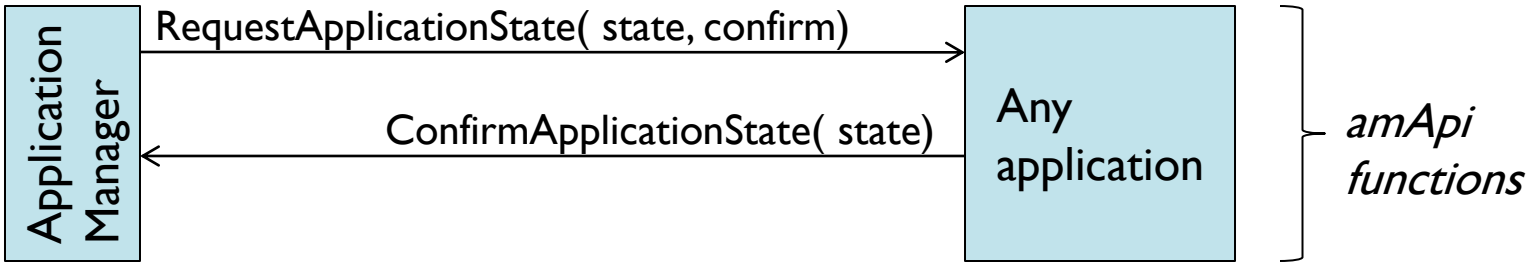
Application States

- Application States are introduced to manage the CPU and memory usage
  - Application Manager determines the state of all applications
  - Platform Application does not support application states

- Defined Application States
  - PlayState: Application has focus and is visible
    - Application is fully operational and controllable
  - PauseState: Application is not in focus, but could be visible
    - Limit CPU usage
    - Limit graphics animations
  - SuspendState: Application is not visible
    - This state is only invoked when resources are needed for other activities
    - Free up any possible memory (e.g. graphics)

Application State Transitions

- Applications receive a notification RequestApplicationState(state, confirm)
- When requested (confirm=true) applications call ConfirmApplicationState(state)



Interfaces

- From amApi
  - Interface am (called by applications). For Application Sate:
    - Confirm requested state is reached
      am_ConfirmApplicationState(AppStates state)
  - Interface amN (called by application manager). For Application Sate :
    - Set state for application
      amN_RequestApplicationState(AppStates state, Bool confirmation)

Resource Management

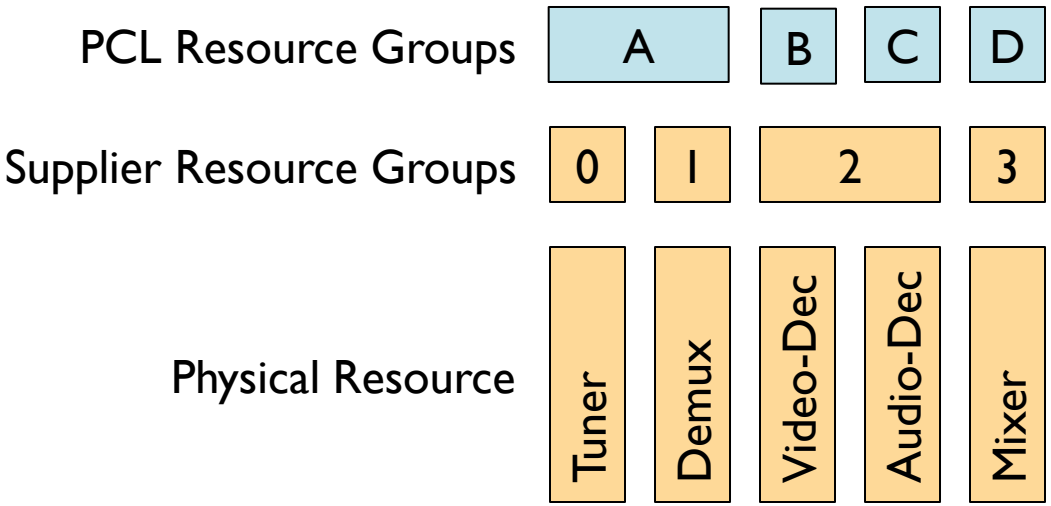
PCL Space Resource Groups

• Audio Featuring	Volume, Surround Sound, Audio Enhancement
• Video Featuring	Video improvements, view, video control
• Mute	Muting video when not owning the source resource
• Connectivity	Play non-broadcast media (mp3, jpg, video)
• Front-End	Select broadcast audio/video
• General	Support for service and alignment of parameters
• Infrastructure	Non AV peripherals (CEC, UART, I2C, USB, etc)
• Setup	Powering and destination setting
• Source	Controlling audio/video sources
• Graphics	Controlling the graphics scalar outside DirectFB
• Scale	Controlling video scaling

jointSPACE Resource Management

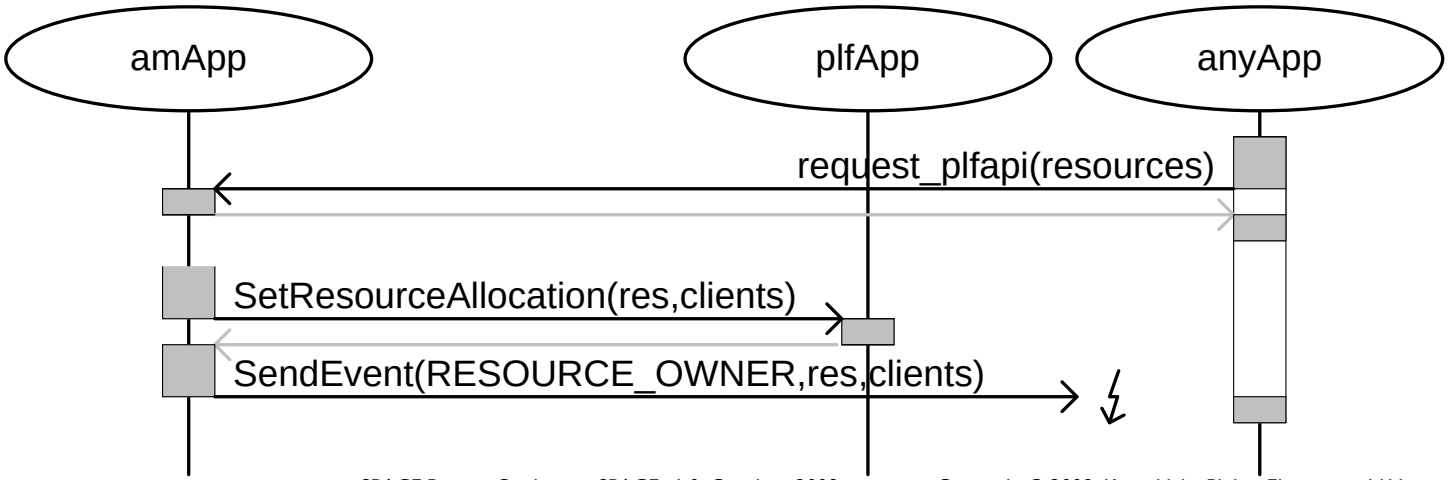
- Supplier can determine own resource groups
  - Optimally designed for supplier platform

- Supplier resource groups and PCL resource groups need to be mapped
  - Resource groups are not aligned



Resource Request

- Applications request resources from Application Manager
  - Platform Application allows only calls from application that owns resource
- Application manager decides on resource request granting
  - Decision can be based on different parameters (focus, break-in, …)
  - Default policy is to grant latest request
- Applications do not release resources
  - They are taken away by the Application Manager



Interfaces

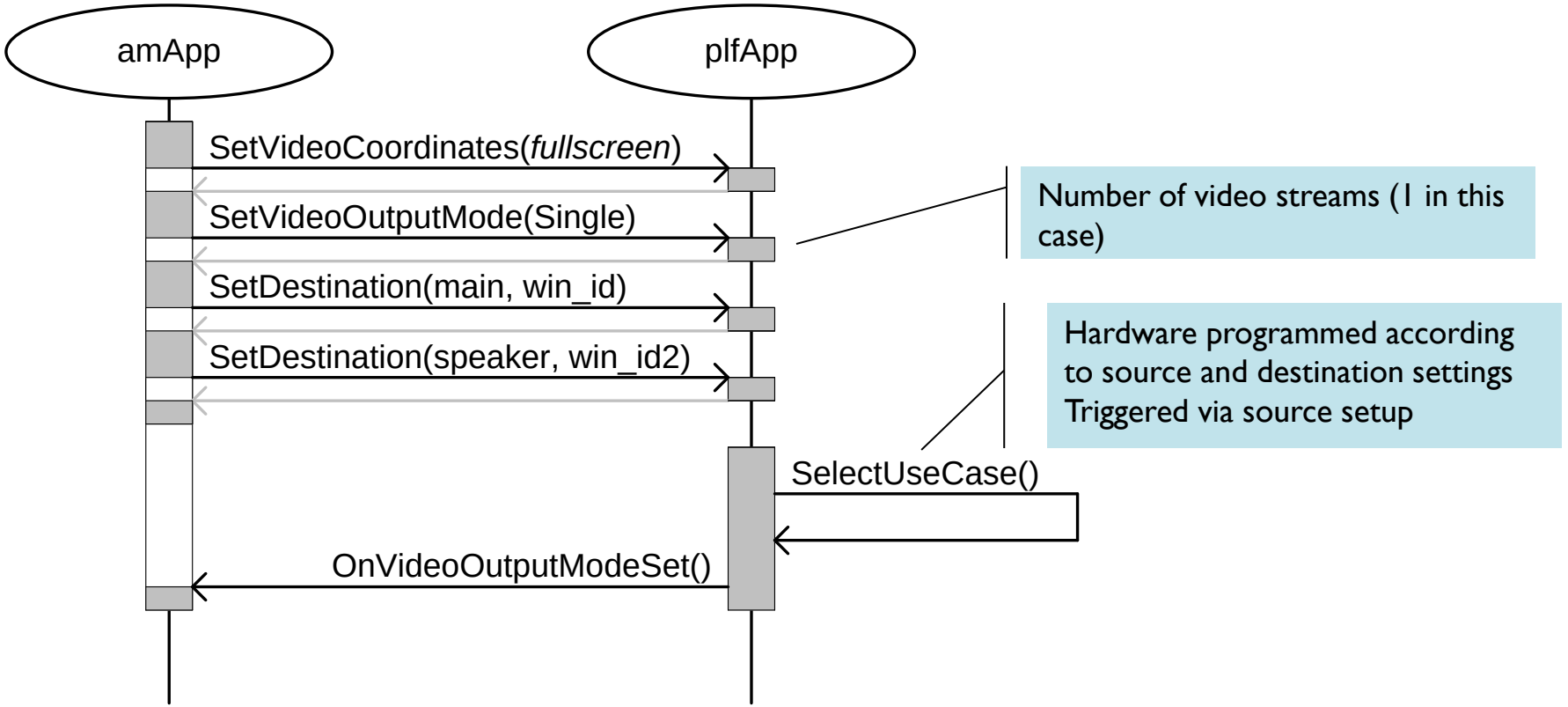
- From amApi
  - Interface am (called by applications).
    - To request a resource group: am_RequestPlfApi(PlfApiResources res)

- From plfApi
  - Interface IPlfApiResource
    - Used to set allocate platform resources
    - To set a list of clients for a resource (mostly for 1 client)
      SetResourceAllocation(int resid, int nbrclients, int* winid)
    - To add a list of clients to a resource
      SetClientResourceAllocation(int resid, int nbrclients, int* winid)

Connection Management

Destination Setup

- Application Manager sets window to be rendered on identified output
  - Below: one window (win_id) is rendered full screen



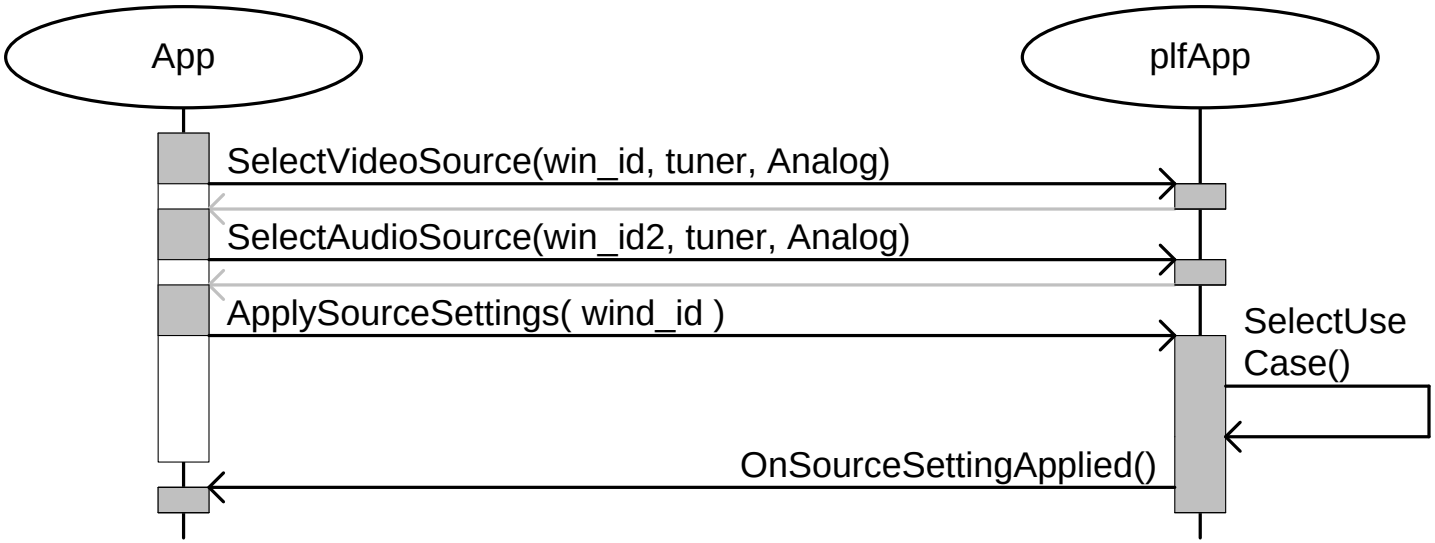
Interfaces

- From plfAPI: IPlfApiDestinationSetup and IPlfApiDestinationSetupNotify
  - Sets the number of used A/V output (windows), and their position/size

- Functions used from those interfaces
  - The Single or Dual selection is done via SetVideoOutputMode(int mode, Bool* retval)
    - Changes are notified via: OnVideoOutputModeSet(int mode)
  - Setting the application window for Main, Sub, Speaker, or Headphone via SetDestination(int destid, int windowid, Bool IsBorder)
  - Setting the output region for Stream1 and Stream2 is done via SetVideoCoordinates(int streamid, int ul_x, int ul_y, int lr_x, int lr_y, Bool* retval)
    - Changes notified via: OnVideoCoordinatesSet(int streamid)

Source Setup

- Application selects the video and audio source
  - With the earlier destination setup, the A/V platform is (re)configured



Interfaces

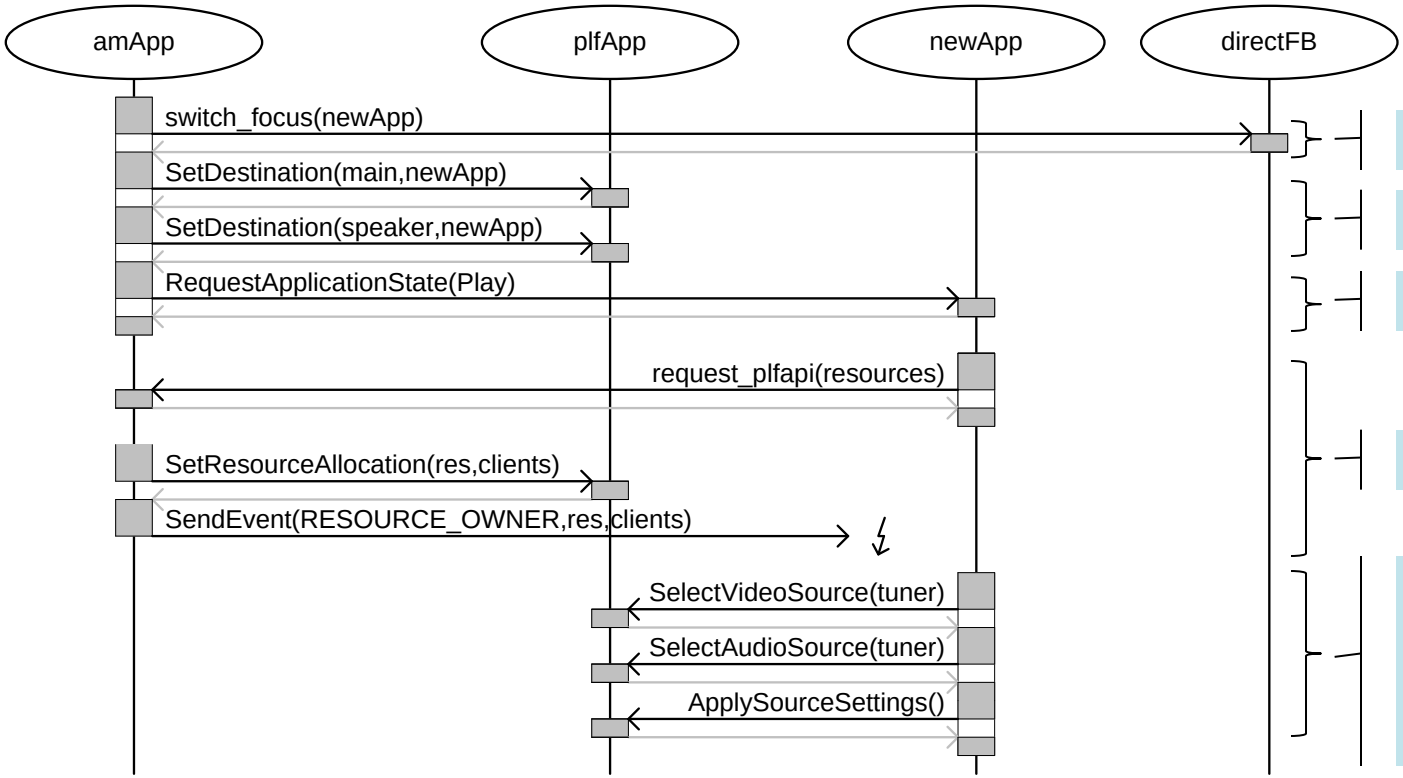
- From plfAPI (and papi): Interface IPlfApiSourceSetup
  - Selects the A/V source for the application (window)

- Functions used
  - SelectVideoSource(int winid, int source, Nat32 designator, Bool* retval)
  - SelectAudioSource(int audionodeid, int source, Nat32 designator, Bool* retval)
    - These select the source (Tuner, extension) for the audio/video
  - ApplySourceSettings(int winid, Bool* retval)
    - Only when this function is called, the earlier set values are applied
    - This selects and applies the use case using source and destination setup

Application Switch

Application Switching

- Switching between applications done by the Application Manager
  - Focus, source - destination setup, application state, resource ownership



Audio Video Control

Transient Handling

- Transients are undesired audio/video artifacts on the occurrence of an event
  - Event triggered by user action or autonomous (e.g. changed signal)
- A client application must deal with transients originating from its own action
  - Typically mutes audio and video to hide transient
- Transients intrinsic to hardware/signal conditions are handled within platform
  - E.g. unstable signal conditions, loss of signal, changing codec formats etc
- Typical management of transients involve muting the audio and video
  - Done via interfaces in the resource group “mute”

Examples of Transient Handling

- Transients a client application needs to handle
  - Program Selection
  - Program Installation
  - Switching across AV Input sources

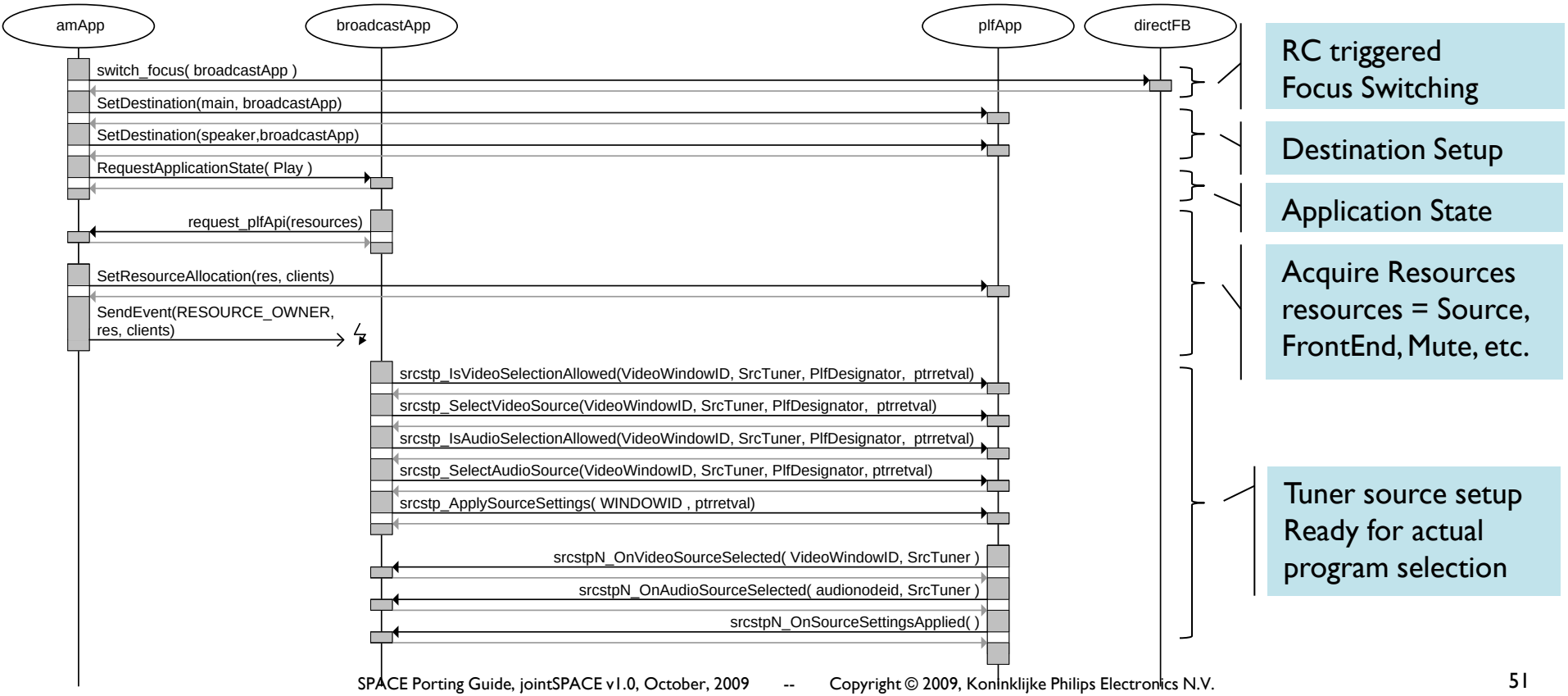
- Transients the platform needs to handle
  - Changing codec formats for e.g. MPEG2 to MPEG4
  - Loss of audio or video signal, low signal strength etc
  - Changing or applying View modes

Program Selection Example

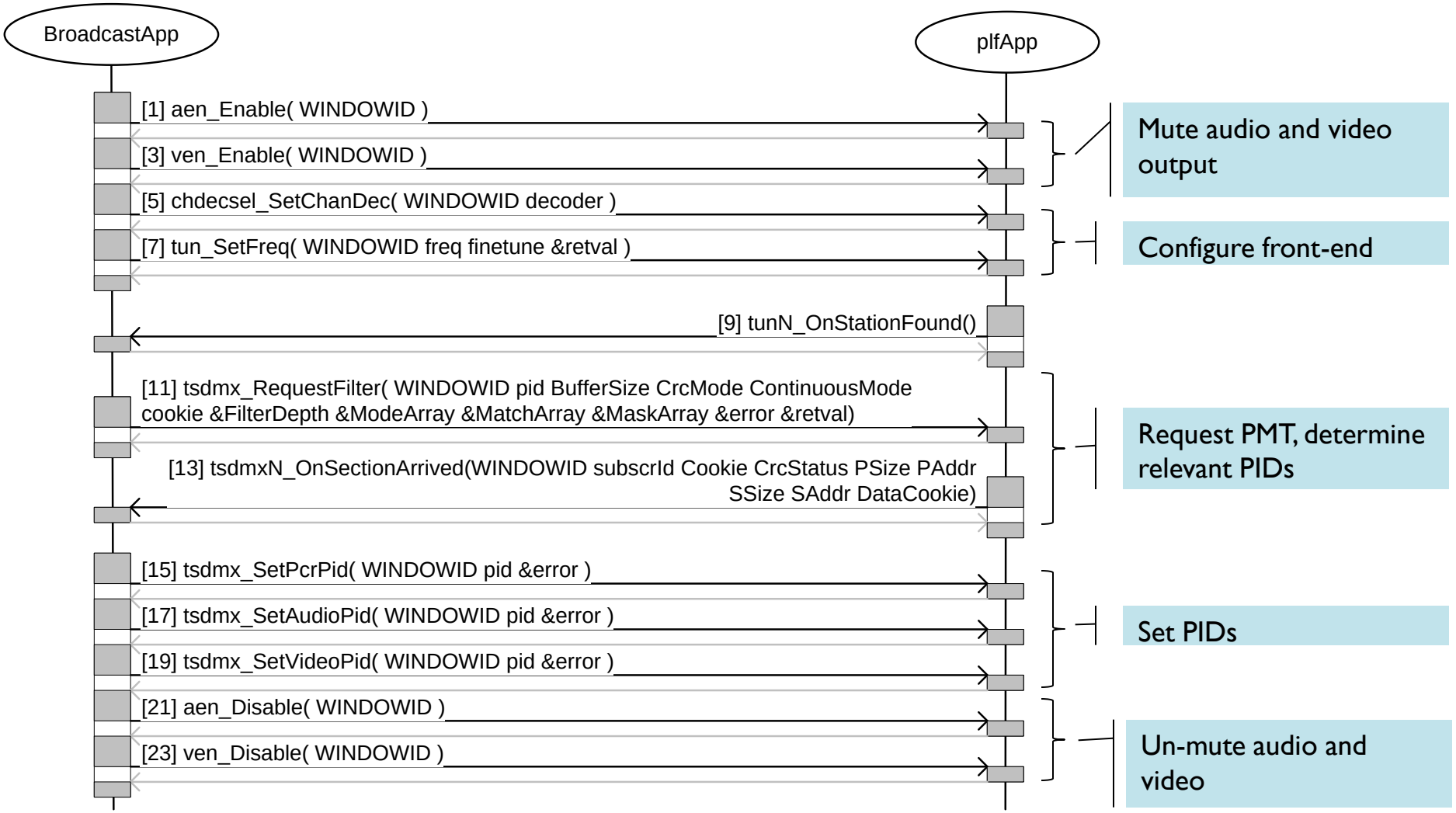
- The following steps are executed when zapping to a digital channel
  - Mute audio and video to avoid transients
  - Configure front-end (a.o. set frequency)
  - When a signal is detected, the PMT is requested
  - Extract the A/V and PCR PIDS from PMT
  - Set PIDs on the demux
  - Un-mute audio and video

Program Selection Overview

- Application switching and Source setup precedes program selection
  - When broadcast application is not in focus
  - When the tuner source is not yet setup



Program Selection Sequence Chart



Interfaces

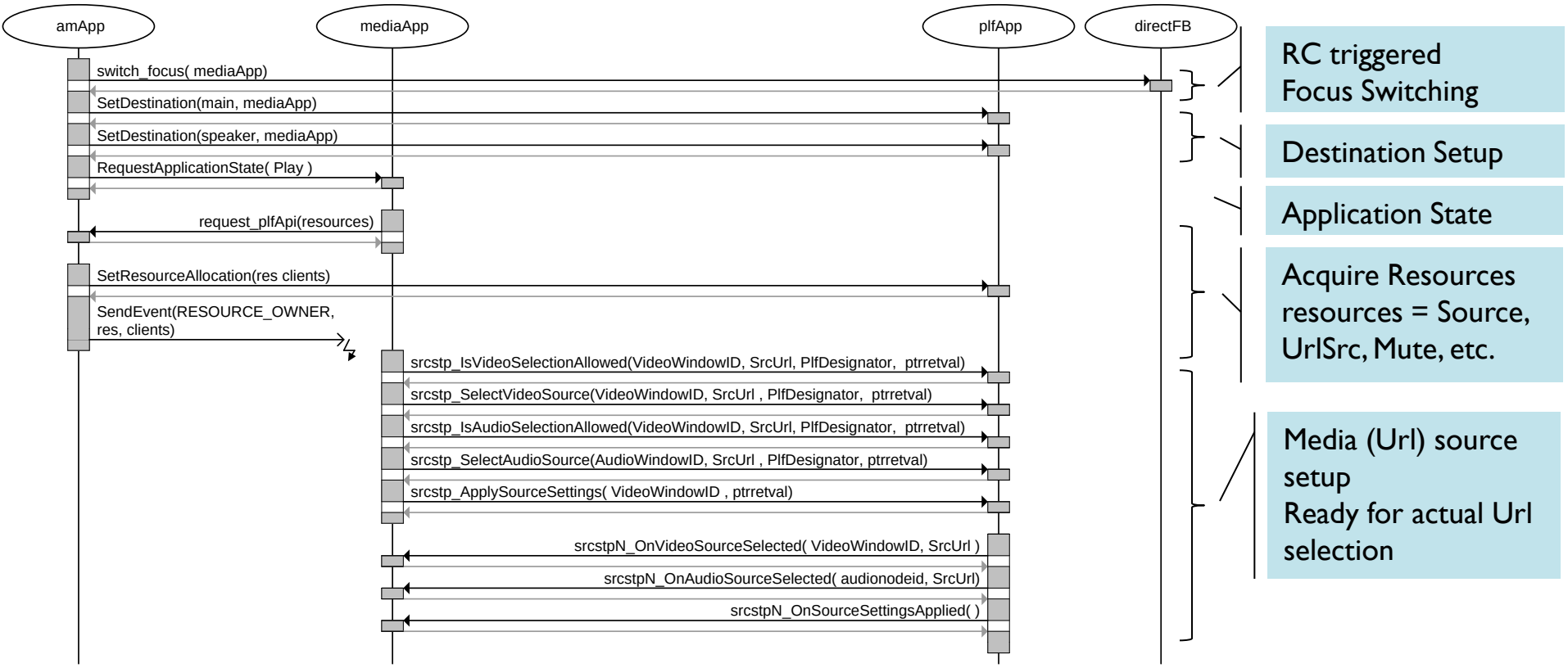
- From plfApi
  - IPlfApiSourceSetup (prefix src)
    - Sets the source and the format of the audio/video data
  - IPlfApiChanDecSelect (prefix chdec)
    - Selection of the channel decoder (DVB-T, DVB-C, ….)
  - IPlfApiTuning + Notify (prefix tun)
    - Tuning to a channel and notification of availability of transmission
  - IPlfApiTsDmxAbstract + Notify (prefix tsdmx)
    - Section filtering and PID setting
  - IPlfApiEnable(prefix aen)
    - Used to (un)mute audio
  - IPlfApiEnable(prefix ven)
    - Used to (un)blank video

Media Playback

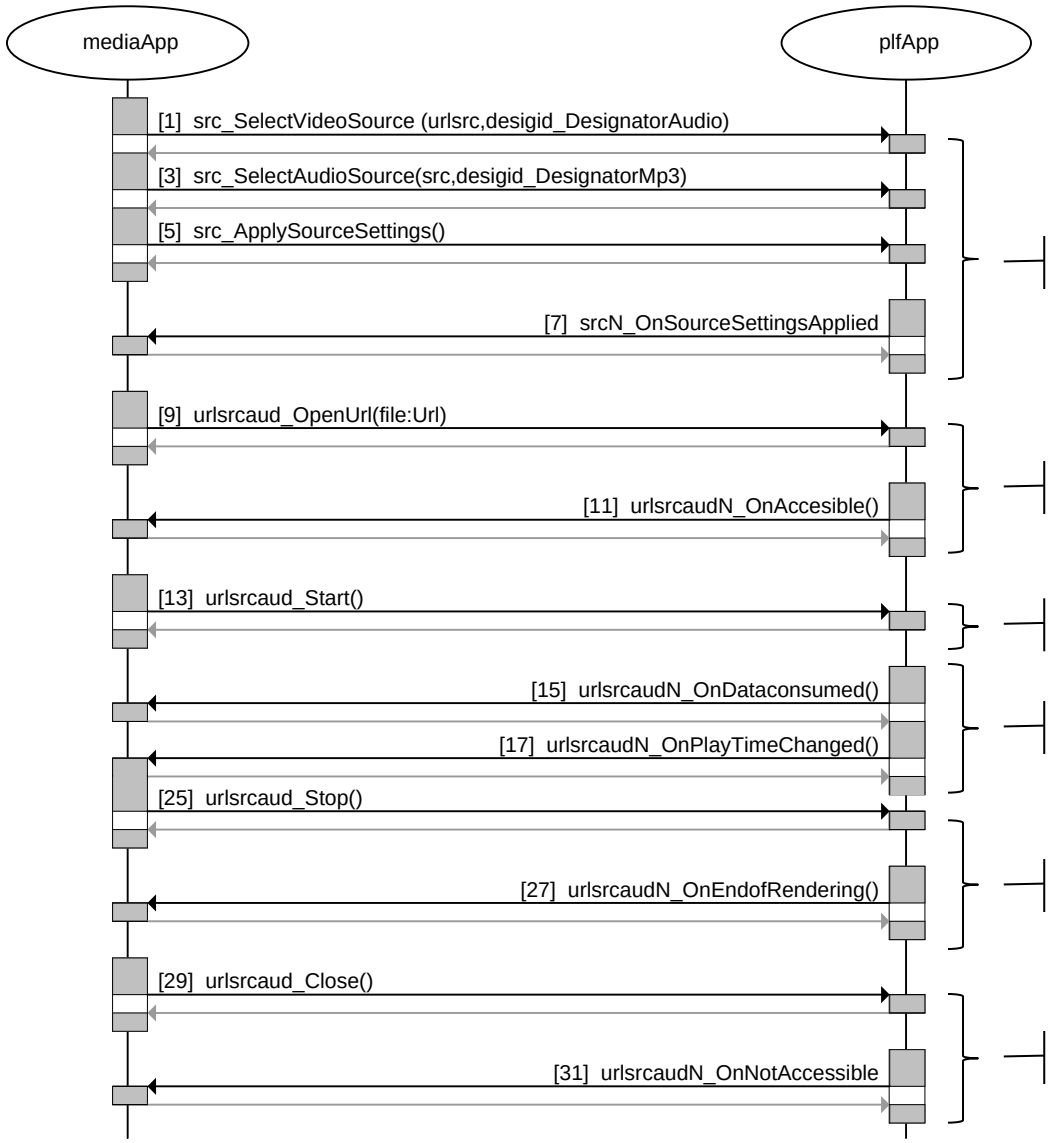
- For media playback, the following steps are needed
  - Set the audio/video source for the media format
    - E.g. mp3 playback from an URL
  - Open the actual URL
    - Which can be e.g. file:// or http://
  - Start playback
  - Use the progress notifications if desired (e.g. UI feedback)
  - Stop and close

Media Playback Overview

- Application switching and Source setup precedes program selection
  - When Media application is not in focus
  - When the tuner source is not yet setup



Media Playback Sequence Chart



Source settings

Open the URL

Start playback

Repeated notifications during playback

Stop playback

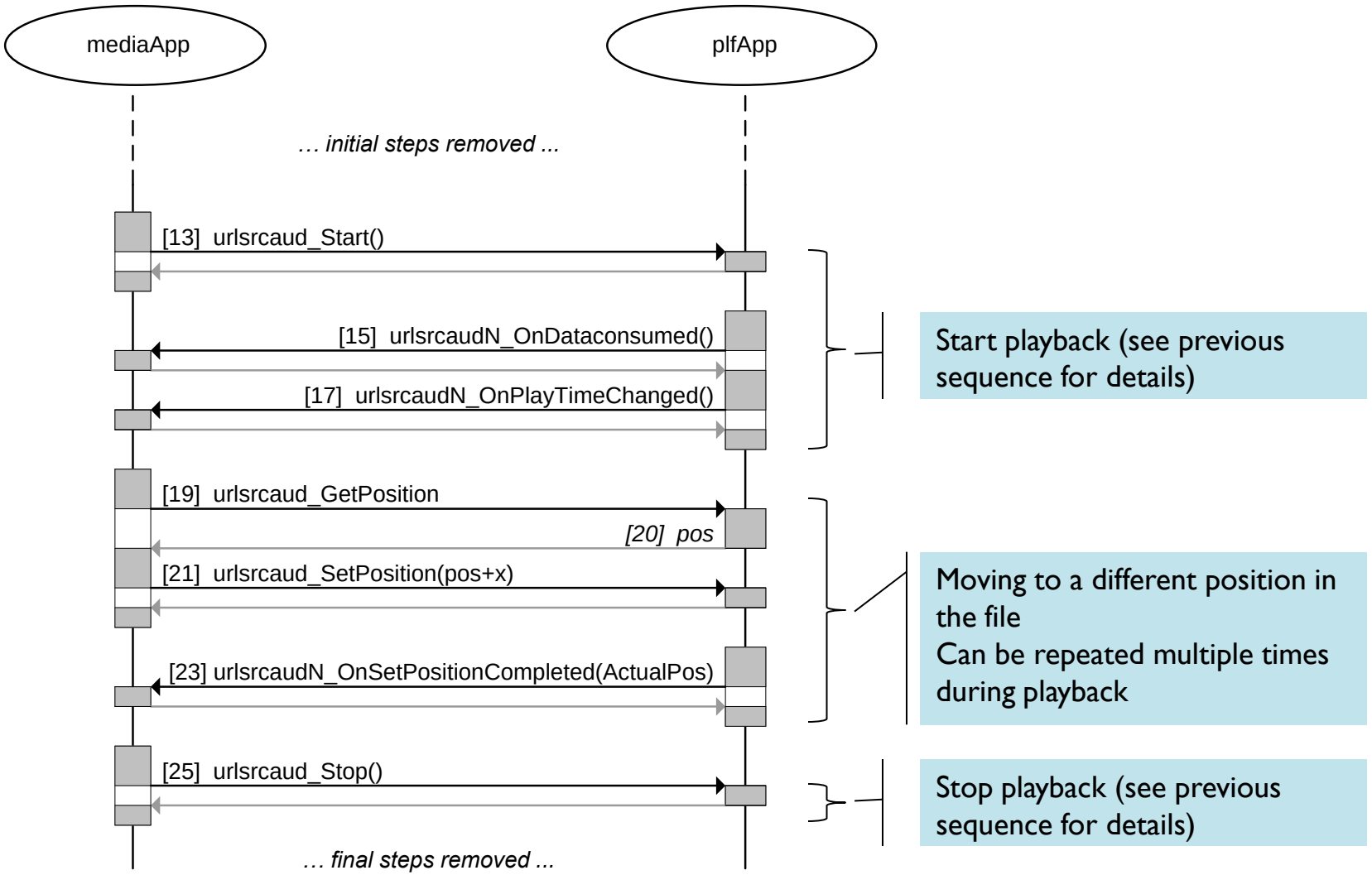
Close the URL

Trickplay

- During playback, the user can jump to a different location in the content
  - Jump forward or backward

- A direct position can be set for playback
  - Retrieving current position and adding delta to support a relative jump

Trickplay Sequence Chart



Interfaces

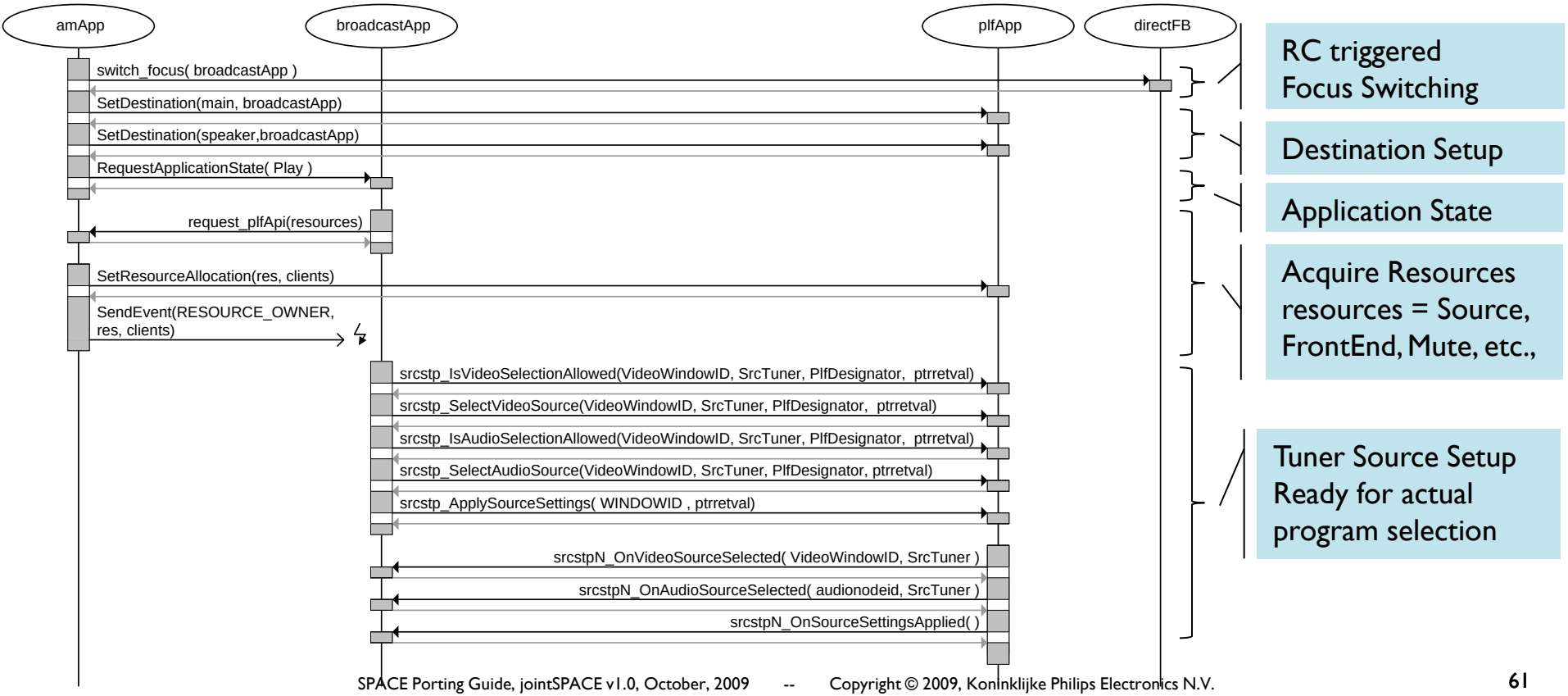
- The following interfaces are used from plfApi
  - IPlfApiSourceSetup (prefix src)
    - Sets the source and the format of the audio/video data
  - IPlfApiUrlSrc + Notify (prefix urlsrcaud)
    - Sets the location of the mp3 media and controls playback

Program Installation

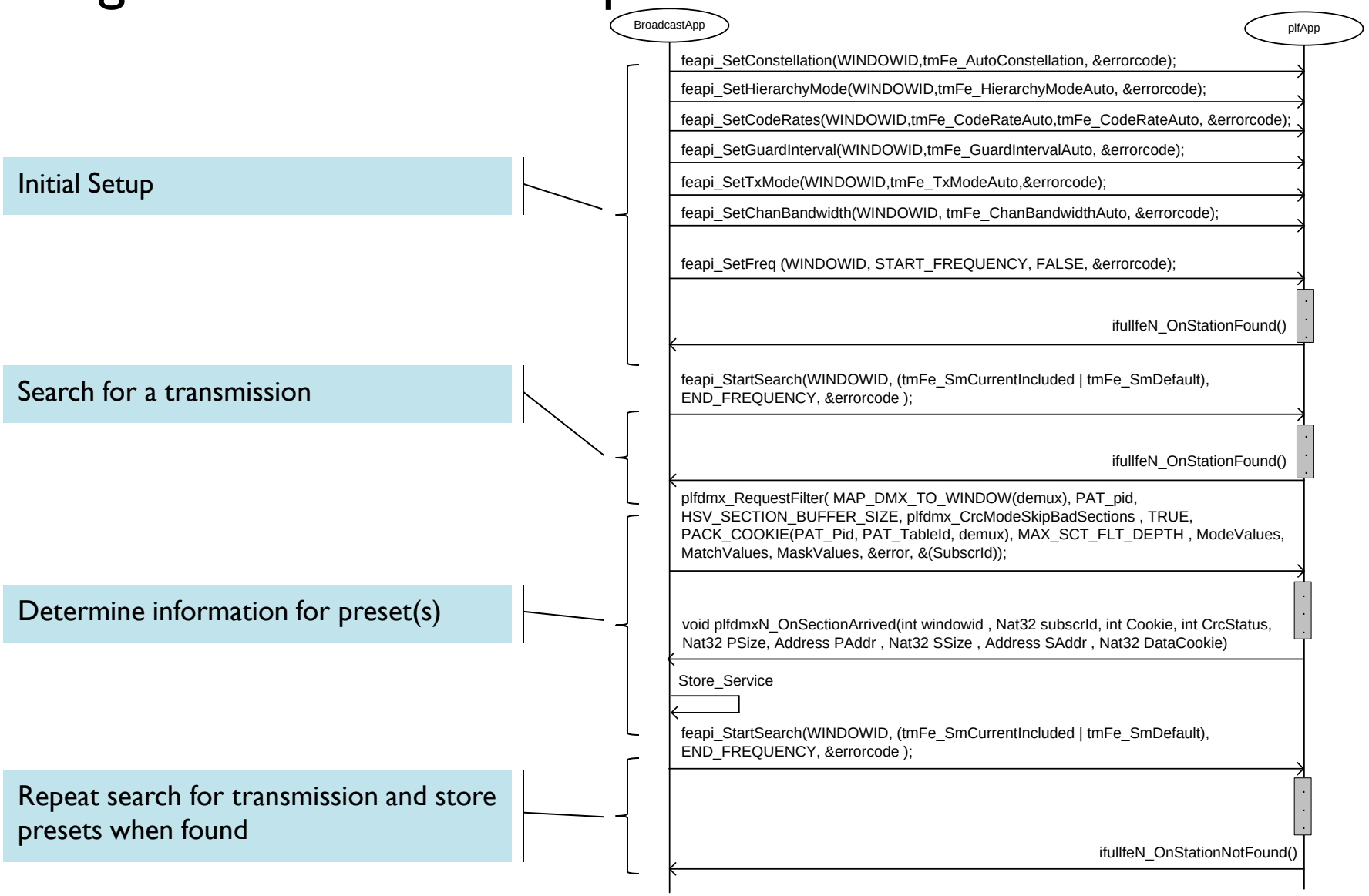
- Platform can search a frequency range
  - Until a transmission is found or end of the range is reached
- A complete channel installation is searching all frequencies
  - And storing found programs
- Steps done by client
  - Set the broadcast parameters (code rate, channel bandwidth, …)
  - Set starting frequency, and start search
  - When transmission found, retrieve tables and extract program info
  - Store the program info
  - Repeat search until all stations found

Program Installation Overview

- Typically Application switching and Source setup precedes Program Installation
  - When broadcast application is not in focus
  - When the tuner source is not yet setup



Program Installation Sequence Chart



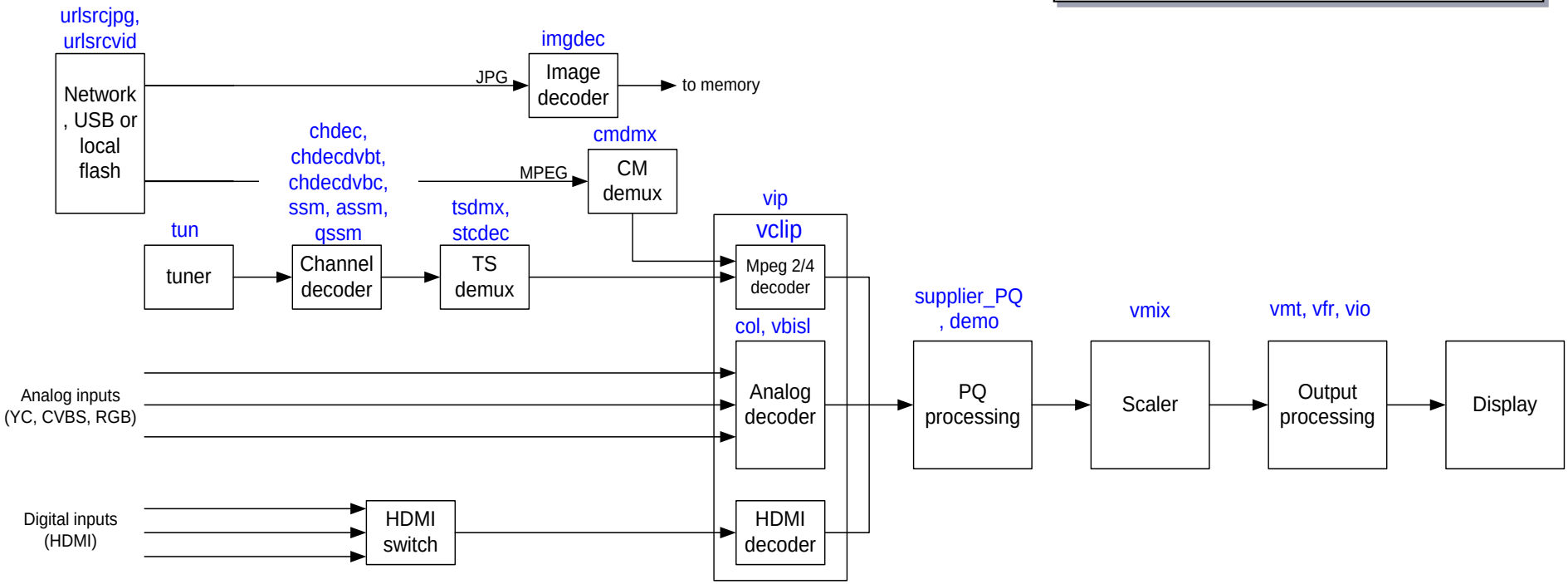
Interfaces

- From plfApi, the following interfaces are used
  - IPlfApiTuning + Notify (prefix tun)
    - Controlling the tuner frequency and searching for carrier
  - IPlfApiChanDec + Notify (prefix chdec)
    - Setting DVB parameters for a transmission
  - IPlfApiTsDmxAbstract + Notify (prefix tsdmx)
    - Control section filter and setting of PIDs

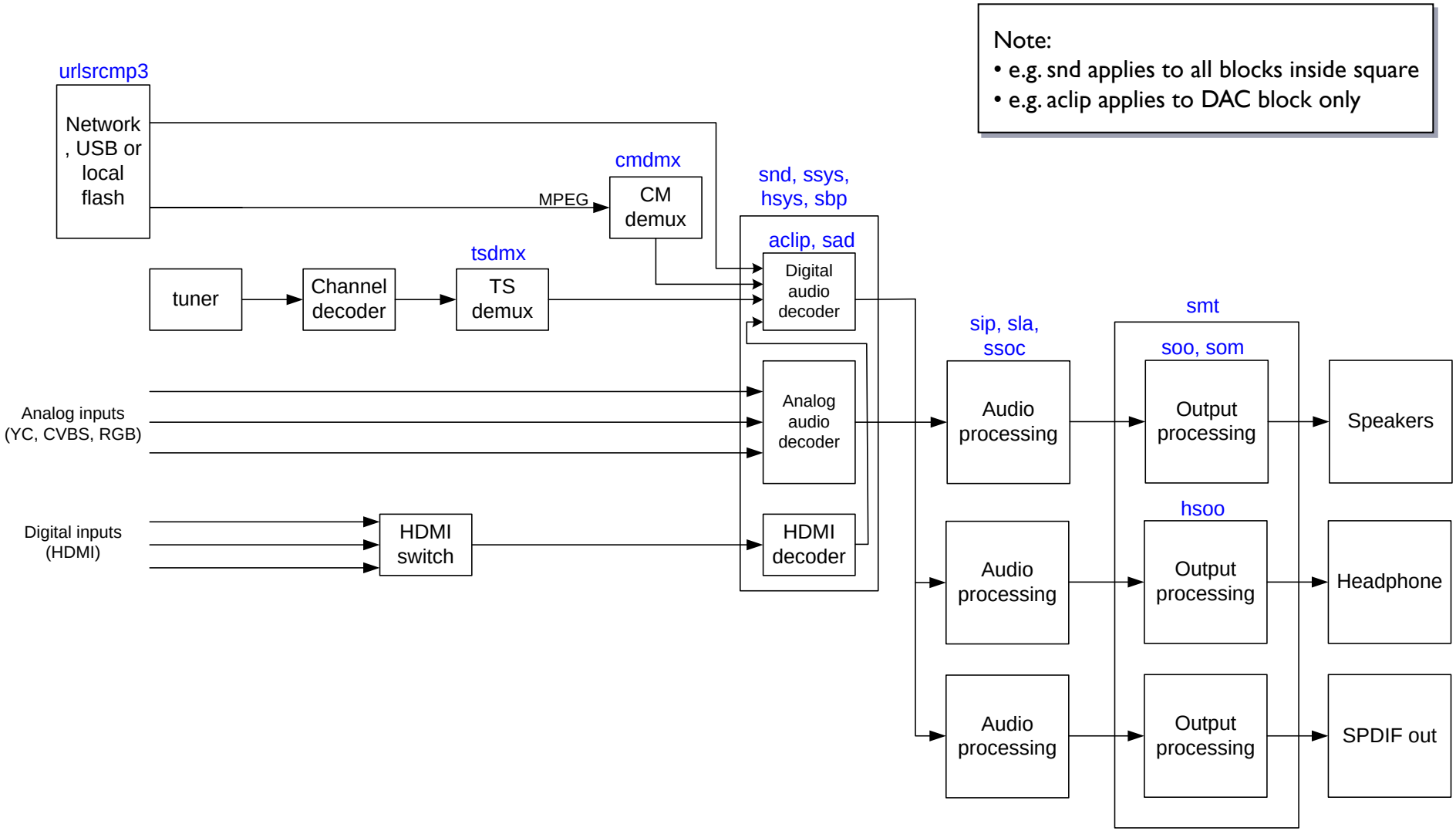
Audio Video Platform Interfaces

Video streaming logical model

Note:
- e.g. vip applies to all blocks inside square
- e.g. vclip applies to Mpeg block only



Audio streaming logical model



Overview of A/V Platform Interfaces

- The following slides list all interfaces
  - Organized per resource group
  - Indication if interface is part of papi.

- Syntax for plfapi interfaces
  - extern FResult <resgroup>_<prefix>_FunctionName(args …)
  - E.g extern FResult plfapivf_vio2_GetContrast(int winid, int * retval)

- Syntax for papi interfaces:
  - Main difference with plfapi is the removal of winid
  - extern FResult papi_<resgroup>_<prefix>_FunctionName(args …)
  - E.g: extern FResult papi_vf_vio2_GetContrast(int * retval)

PHILIPS

joint space
split application architecture

Resource Group: Source

Platform API	*prefix*	*pApi*	
IPlfApiSourceSetup	src	x	papi_src_src
IPlfApiSourceId	srcid	x	papi_src_srcid
IPlfApiSourceSetupNotify	srcN	x	papi_src_srcN
IPlfApiDesignatorId	dsgid	x	papi_src_dsgid
IPlfApiConnectionConfiguration	conf	x	papi_src_conf
IPlfApiBreakin	brk	x	papi_src_brk
IPlfApiBreakinNotify	brkN	x	papi_src_brkN
IPlfApiEnable	aen	x	papi_src_aen
IPlfApiEnable	ven	x	papi_src_ven
IPlfApiResourceAvailability	srcitf	x	papi_src_srcitf
IPlfApiResourceAvailabilityNotify	srcitfN	x	papi_src_srcitfN

PHILIPS

joint space
split application architecture

Resource Group: Setup

Platform API	*prefix*	*pApi*	
IPlfApiPower	pow	x	papi_setup_pow
IPlfApiPowerNotify	powN	x	papi_setup_powN
IPlfApiDestinationSetup	dst	x	papi_setup_dst
IPlfApiDestinationSetupNotify	dstN	x	papi_setup_dstN
IPlfApiKeyControl	key	-	
IPlfApiEnable	lgc	-	
IPlfApiBlank	blk	x	papi_setup_blk
IPlfApiBlankNotify	blkN	x	papi_setup_blkN
IPlfApiGfxScalerControl	gfxscctrl	-	
IPlfApiResource	res	x	papi_setup_res

PHILIPS

joint space
split application architecture

Resource Group: Audio featuring

Platform API	*prefix*	*pApi*	
IPlfApiSoundImprovement	sip	x	papi_af_sip
IPlfApiSoundImprovementNotify	sipN	x	papi_af_sipN
IPlfApiSoundLevelAdjust	sla	x	papi_af_sla
IPlfApiSoundLevelAdjustNotify	slaN	x	papi_af_slaN
IPlfApiSoundProfile	spr	-	
IPlfApiSurroundMode	som	x	papi_af_som
IPlfApiSurroundModeNotify	somN	x	papi_af_somN
IPlfApiSoundOutput	ssoo	x	papi_af_ssoo
IPlfApiSoundOutputNotify	ssooN	x	papi_af_ssooN
IPlfApiSoundOutput	hsoo	x	papi_af_hsoo
IPlfApiSoundOutputNotify	hsooN	x	papi_af_hsooN
IPlfApiSoundMute	smt	x	papi_af_smt
IPlfApiEnable	arc	-	
IPlfApiSoundDelay	snddelay	x	papi_af_snddelay
IPlfApiResourceAvailability	afitf	x	papi_af_afitf
IPlfApiResourceAvailabilityNotify	afitfN	x	papi_af_afitfN

Resource Group: Video featuring

Platform API	*prefix*	*pApi*	
IPlfApiCopyrightControl	cprc	x	papi_vf_cprc
IPlfApiDemoMode	demo	-	
IPlfApiDemoModeNotify	demoN	-	
IPlfApiPictureProfile	ppr	-	
IPlfApiPictureStyles	pst	-	
IPlfApiPictureStylesNotify	pstN	-	
IPlfApiSelfLearningTv	slt	-	
IPlfApiSelfLearningTvNotify	sltN	-	
IPlfApiVideoImprovement	vim	x	papi_vf_vim
IPlfApiVideoImprovementNotify	vimN	x	papi_vf_vimN
IPlfApiVideoOutput	vio	x	papi_vf_vio
IPlfApiVideoOutputNotify	vioN	x	papi_vf_vioN
IPlfApiViewPort	vp	-	
IPlfApiViewPortNotify	vpN	-	
IPlfApiVideoDelay	vdelay	x	papi_vf_vdelay
IPlfApiVideoDelayNotify	vdelayN	x	papi_vf_vdelayN
IPlfApiViewMode	vwm	-	
IPlfApiViewModeNotify	vwmN	-	
IPlfApiResourceAvailability	vfitf	x	papi_vf_vfitf
IPlfApiResourceAvailabilityNotify	vfitfN	x	papi_vf_vfitfN

Resource Group: Front-End

Platform API	*prefix*	*pApi*	
IPlfApiColorDecode	col	x	papi_fe_col
IPlfApiColorDecodeNotify	colN	x	papi_fe_colN
IPlfApiSoundDecode	snd	x	papi_fe_snd
IPlfApiSoundSystem	hsys	x	papi_fe_hsys
IPlfApiSoundSystem	ssys	x	papi_fe_ssys
IPlfApiSoundSystemNotify	hsysN	x	papi_fe_hsysN
IPlfApiSoundSystemNotify	ssysN	x	papi_fe_ssysN
IPlfApiVideoProperties	vip	x	papi_fe_vip
IPlfApiVideoPropertiesNotify	vipN	x	papi_fe_vipN
IPlfApiVbiInsert	vbiins	-	
IPlfApiSigStrengthMeas	ssm	x	papi_fe_ssm
IPlfApiSigStrengthMeas	assm	x	papi_fe_assm
IPlfApiSigStrengthMeas	qssm	x	papi_fe_qssm
IPlfApiSigStrengthMeasNotify	ssmN	x	papi_fe_ssmN
IPlfApiSigStrengthMeasNotify	assmN	x	papi_fe_assmN
IPlfApiSigStrengthMeasNotify	qssmN	x	papi_fe_qssmN
IPlfApiChanDec	chdec	x	papi_fe_chdec
IPlfApiChanDec	chdec2	x	papi_fe_chdec2
IPlfApiChanDecDvbC	chdecdvbc	x	papi_fe_chdecdvbc
IPlfApiChanDecDvbT	chdecdvbt	x	papi_fe_chdecdvbt
IPlfApiChanDecDvbT	chdecisdbt	x	papi_fe_chdecisdbt
IPlfApiChanDecNotify	chdecN	x	papi_fe_chdecN
IPlfApiChanDecNotify	chdec2N	x	papi_fe_chdec2N
IPlfApiChanDecSelect	chdecsel	x	papi_fe_chdecsel
IPlfApiChanTable	chantab	x	papi_fe_chantab
IPlfApiRfAmp	rfamp	x	papi_fe_rfamp
IPlfApiTuning	tun	x	papi_fe_tun
IPlfApiTuningNotify	tunN	x	papi_fe_tunN
IPlfApiTsDmxAbstract	tsdmx	x	papi_fe_tsdmx
IPlfApiTsDmxAbstractNotify	tsdmxN	x	papi_fe_tsdmxN
IPlfApiStcDec	stcdec	x	papi_fe_stcdec
IPlfApiStcDecNotify	stcdecN	x	papi_fe_stcdecN

Resource Group: Front-End continued

Platform API	*prefix*	*pApi*	
IPlfApiVbiSlice	vbisl	x	papi_fe_vbisl
IPlfApiVbiSliceNotify	vbislN	x	papi_fe_vbislN
IPlfApiPreset	prs	-	
IPlfApiDiseqc	diseq	x	papi_fe_diseq
IPlfApiDvbs	dvbs	x	papi_fe_dvbs
IPlfApiCiPlusAI	ciplusai	-	
IPlfApiCiPlusAmmi	ciplusammi	-	
IPlfApiCiPlusAmmiNotify	ciplusammintf	-	
IPlfApiCiPlusCa	ciplusca	-	
IPlfApiCiPlusCaNotify	cipluscantf	-	
IPlfApiCiPlusmmi	ciplusmmi	-	
IPlfApiCiPlusmmiNotify	ciplusmmintf	-	
IPlfApiCiPlusHcNotify	ciplushcntf	-	
IPlfApiCiPlusHc	ciplushc	-	
IPlfApiCiPlusHlc	ciplushlc	-	
IPlfApiCiPlusInit	ciplusinit	-	
IPlfApiCiPlusInitNotify	ciplusinitntf	-	
IPlfApiCiPlusStatus	ciplusstatus	-	
IPlfApiCiPlusStatusNotify	ciplusstatusntf	-	
IPlfApiCiPlusAINotify	ciplusaintf	-	
IPlfApiCiPlusUpgr	ciplusupgr	-	
IPlfApiSoundAudioDescription	sad	x	papi_fe_sad
IPlfApiSoundChannel	soc	x	papi_fe_soc
IPlfApiSoundChannelNotify	socN	x	papi_fe_socN
IPlfApiAudioClip	aclip	x	papi_fe_aclip
IPlfApiAudioClipNotify	aclipN	x	papi_fe_aclipN
IPlfApiVideoClip	vclip	x	papi_fe_vclip
IPlfApiVideoClipNotify	vclipN	x	papi_fe_vclipN
IPlfApiVideoFreeze	vfr	x	papi_fe_vfr
IPlfApiVideoFreezeNotify	vfrN	x	papi_fe_vfrN
IPlfApiResourceAvailability	feitf	x	papi_fe_feitf
IPlfApiResourceAvailabilityNotify	feitfN	x	papi_fe_feitfN

PHILIPS

joint space
split application architecture

Resource Group: Connectivity

Platform API	*prefix*	*pApi*	
IPlfApiCmDmx	cmdmx	x	papi_conn_cmdmx
IPlfApiCmDmxNotify	cmdmxN	x	papi_conn_cmdmxN
IPlfApiImageDec	imgdec	x	papi_conn_imgdec
IPlfApiImageDecNotify	imgdecN	x	papi_conn_imgdecN
IPlfApiUrlSrc	urlsrcaud	x	papi_conn_urlsrcaud
IPlfApiUrlSrc	urlsrcjpg	x	papi_conn_urlsrcjpg
IPlfApiUrlSrc	urlsrcvid	x	papi_conn_urlsrcvid
IPlfApiUrlSrcNotify	urlsrcaudN	x	papi_conn_urlsrcaudN
IPlfApiUrlSrcNotify	urlsrcjpgN	x	papi_conn_urlsrcjpgN
IPlfApiUrlSrcNotify	urlsrcvidN	x	papi_conn_urlsrcvidN
IPlfApiNetschemeConfig	netschemecfg	-	
IPlfApiDigitalAudioDecoderNotify	digadecN	x	papi_conn_digadecN
IPlfApiDigitalVideoDecoderNotify	digvdecN	x	papi_conn_digvdecN
IPlfApiDcfParser	dcf	x	papi_conn_dcf
IPlfApiMediaDetect	md	x	papi_conn_md
IPlfApiMediaDetectNotify	mdN	x	papi_conn_mdN
IPlfApiResourceAvailability	connitf	x	papi_conn_connitf
IPlfApiResourceAvailabilityNotify	connitfN	x	papi_conn_connitfN

Resource Group: General

Platform API	*prefix*	*pApi*	
IPlfApiSoundBeeper	sbp	x	papi_gen_sbp
IPlfApiCabinetAttributes	cabattr	-	
IPlfApiScreenAttributes	scrattr	-	
IPlfApiContrastReduction	cr	-	
IPlfApiContrastReductionNotify	crN	-	
IPlfApiTxtModeControl	tmc	-	
IPlfApiBasicControl	bcont	-	
IPlfApiVersion	ver	x	papi_gen_ver
IPlfApiVideoGeneratorTestPattern	vgtp	x	papi_gen_vgtp
IPlfApiVideoGeneratorTestPatternIds	vgtpid	-	
IPlfApiAlign	aln	x	papi_gen_aln
IPlfApiAlignIds	alnid	-	
IPlfApiAmbientLightControl	ambl	-	
IPlfApiDatabagTransfer	dbtf	x	papi_gen_dbtf
IPlfApiDatabagTransfer	dbtfci	-	
IPlfApiPowerMeter	pom	-	
IPlfApiAmbientLight	amb	-	
IPlfApiAmbientLightNotify	ambN	-	
IPlfApiService	serv	-	
IPlfApiServiceNotify	servN	-	
IPlfApiSplashScreen	spla	-	
IPlfApiTemperatureControl	tmpctrl	-	
IPlfApiTemperatureControlNotify	tmpctrlN	-	
IPlfApiHwError	hwerr	x	papi_gen_hwerr
IPlfApiHwErrorIds	hwerrids	-	
IPlfApiHwErrorNotify	hwerrN	x	papi_gen_hwerrN
IPlfApiFileSplit	filesplit	-	
IPlfApiResourceAvailability	genitf	x	papi_gen_genitf
IPlfApiResourceAvailabilityNotify	genitfN	x	papi_gen_genitfN

PHILIPS

joint space
split application architecture

Resource Group: Infrastructure

Platform API	*prefix*	*pApi*	
IPlfApiAudioOnly	audonly	x	papi_infra_audonly
IPlfApiCecLinkControl	cecctrl	x	papi_infra_cecctrl
IPlfApiCecLinkReceiveAcknowledge	cecack	x	papi_infra_cecack
IPlfApiCecLinkReceiveNotify	cecrxntf	x	papi_infra_cecrxntf
IPlfApiCecLinkTransmit	cectx	x	papi_infra_cectx
IPlfApiCecLinkTransmitStatusNotify	cectxntf	x	papi_infra_cectxntf
IPlfApiGeneralIo	gio	x	papi_infra_gio
IPlfApiI2cControl	i2cc	x	papi_infra_i2cc
IPlfApiI2cMaster	i2cm	x	papi_infra_i2cm
IPlfApiNetwork2ExEx	net2	-	
IPlfApiNetworkExEx	net	-	
IPlfApiNetworkNotify	netN	-	
IPlfApiNetworkNotify2	net2N	-	
IPlfApiNetworkNotify2Ex	net2exN	-	
IPlfApiNetworkNotifyEx	netexN	-	
IPlfApiNetworkNotifyExEx	netexexN	-	
IPlfApiPoint2PointAppIds	p2papid	-	
IPlfApiPoint2PointRx	p2pac	-	
IPlfApiPoint2PointRxEx	p2pacx	-	
IPlfApiPoint2PointRxNotify	p2pacN	-	
IPlfApiPoint2PointTxEx	p2pah	-	
IPlfApiPoint2PointTxNotify	p2pahN	-	
IPlfApiClock	clk	x	papi_infra_clk
IPlfApiClockNotify	clkntf	-	
IPlfApiClockNotifyEx	clkntfx	x	papi_infra_clkntfx
IPlfApiNoClearData	ncd	-	

PHILIPS

joint space
split application architecture

Resource Group: Infrastructure continued

Platform API	*prefix*	*pApi*	
IPlfApiSharedMemory	shmem	-	papi_infra_shmem
IPlfApiUpgradeInfo	upginf	-	
IPlfApiUpgradeInfo2	upginf2	-	
IPlfApiUsbDevice	usbdev	-	
IPlfApiUsbDevice2Ex	usbdev2	-	
IPlfApiUsbDeviceNotify	usbdevN	-	
IPlfApiUsbDeviceNotifyEx	usbdevexN	-	
IPlfApiKeyInject	keyi	-	
IPlfApiEventLogControl	evlogctr	-	
IPlfApiDebugDumpControl	dmpctr	-	
IPlfApiBootFlashFileSystem	bffs	-	
IPlfApiCrypt	crypt	-	
IPlfApiEnable	uartprint	-	
IPlfApiSharedResourceManager	resmgr	-	
IPlfApiFlashOperations	fops	-	
IPlfApiFlashOperationsNotify	fopsN	-	
IPlfApiLedControl	led	x	papi_infra_led
IPlfApiDatabagBrowser	dbbrwse	-	
IPlfApiAuthentication	auth	-	
IPlfApiUpgradeTooling	upgtool	-	
IPlfApiUpgradeToolingNotify	upgtoolN	-	
IPlfApiBootLoader	bootloader	-	
IPlfApiVideoStoreSD	vidstor	-	
IPlfApiVideoStoreSDNotify	vidstorN	-	
IPlfApiResourceAvailability	infraitf	x	papi_infra_infraitf
IPlfApiResourceAvailabilityNotify	infraitfN	x	papi_infra_infraitfN

Resource Group: Mute

Platform API	*prefix*	*pApi*	
IPlfApiVideoMute	vmt	x	papi_mute_vmt
IPlfApiVideoMuteNotify	vmtN	x	papi_mute_vmtN
IPlfApiVideoMute	vvmt	-	
IPlfApiVideoMuteNotify	vvmtN	-	
IPlfApiResourceAvailability	muteitf	-	papi_mute_muteitf
IPlfApiResourceAvailabilityNotify	muteitfN	-	papi_mute_muteitfN

Resource Group: Graphics

Platform API	*prefix*	*pApi*	
IPlfApiGfxControl	gfxctrl	-	
IPlfApiResourceAvailability	gfxitf	x	papi_gfx_gfxitf
IPlfApiResourceAvailabilityNotify	gfxitfN	x	papi_gfx_gfxitfN

Resource Group: Scale

Platform API	*prefix*	*pApi*	
IPlfApiVideoScale	vsc	-	
IPlfApiVideoScaleNotify	vscN	-	
IPlfApiResourceAvailability	scaleitf	x	papi_scale_scaleitf
IPlfApiResourceAvailabilityNotify	scaleitfN	x	papi_scale_scaleitfN

PHILIPS

joint space
split application architecture

pApi specific interfaces (non-plfapi)

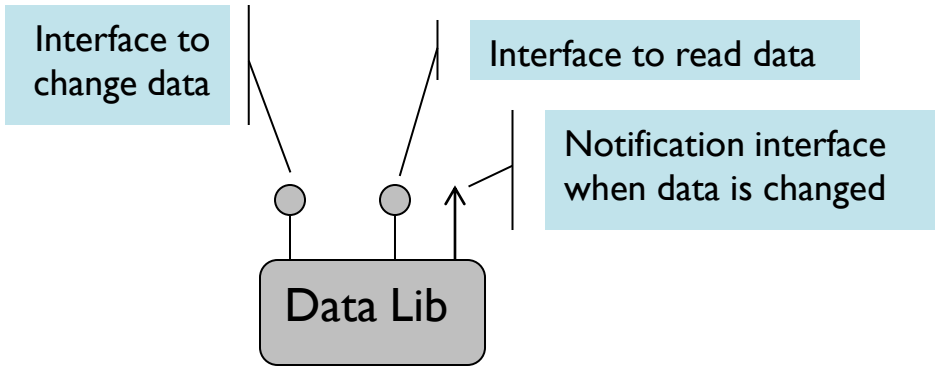
API	*prefix*	*pApi*	
tmIVmix	vmix	x	papi_papionly_vmix
tmIVmixVidLayer	vmixvid	x	papi_papionly_vmixvid
IPlfApiEnable	lvds	x	papi_papionly_lvds
IPlfApiEnable	key	x	papi_papionly_key
IPlfApiEnable	lkb	x	papi_papionly_lkb
IPapiWakeup	wakeup	x	papi_papionly_wakeup
tmIVmixNtf	vmixntf	x	papi_papionly_vmixntf
tmIVmixVidLayerNtf	vmixvidntf	x	papi_papionly_vmixvidntf
IPapiKeyNotify	keyntf	x	papi_papionly_keyntf
IPapiLocalKeyboardNotify	lkbntf	x	papi_papionly_lkbntf
IPapiHdmiGamutNotify	gamutntf	x	papi_papionly_gamutntf

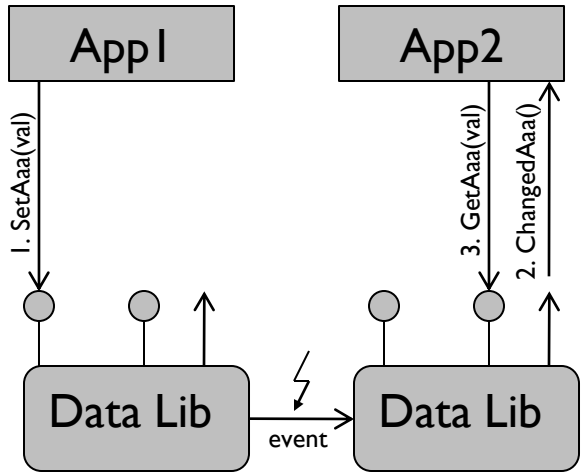
Application Data Sharing

Using Data Libraries

- Any application can gather data needed by other applications
  - Data must be shared, without creating application dependencies

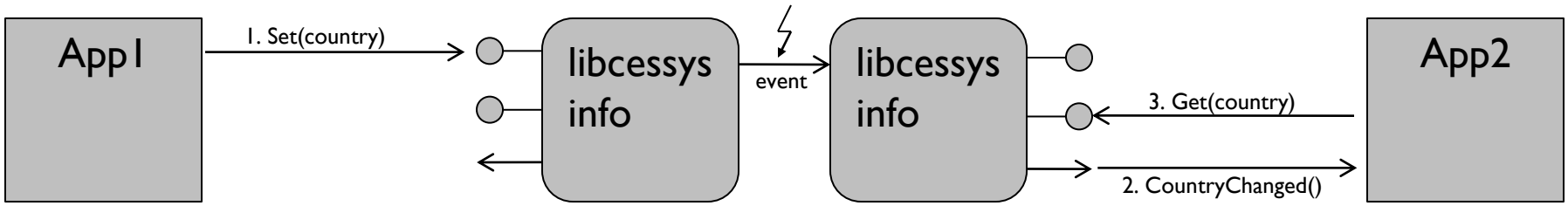
- Concept used are (shared) libraries
  - Library relies on broadcast events to inform changes to applications
  - Provides a functional API, hiding communication channel





System Information Library (cessysinfolib)

- Global settings of user preferences common to all applications
  - E.g. preferred menu language, country, …
  - Accessed via "System Info library "
  - Can be PCL or supplier implemented

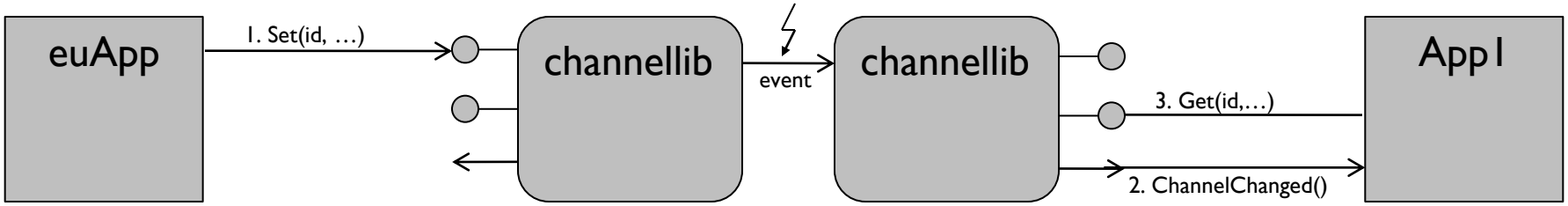


Interfaces

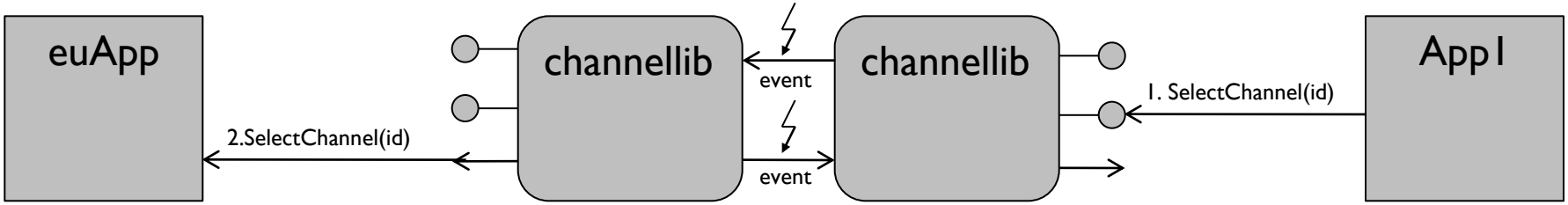
- The System Information Library implements the following interfaces
  - IAppApiSystemsettings
    - Set and retrieve user preferred country and menu language settings.
  - IAppApiCountryID's
    - Consists of enumerated country ID's.
    - Needs to be used along with interface IAppApiSystemsettings to set or retrieve the user preferred country settings

Channel Library (channellib<xx>)

- Access to installed broadcast channels
  - Can be from various applications (e.g. Antenna, Cable , Satellite)
  - All channel libraries have the same API.
  - Can be PCL or supplier implemented



- To trigger channel changes across applications
  - E.g., IP based EPG, factory
  - Requesting application remains in focus



Interfaces

- The Channel Library implements the following interfaces
  - IAppApiProgramData
    - Access to per channel attributes, like channel lock/ audio pid / video pid etc
  - IAppApiProgramDataControl
    - Access to current tuner preset, selected medium information etc.
    - Notifies updates to the installed channel data
  - IAppApiProgramIterator
    - Mechanism to filter the channel list based on the given criteria
      - Filter only digital channels / filter all favorite channels etc


PHILIPS