JTNC Standards Training – Volume 2

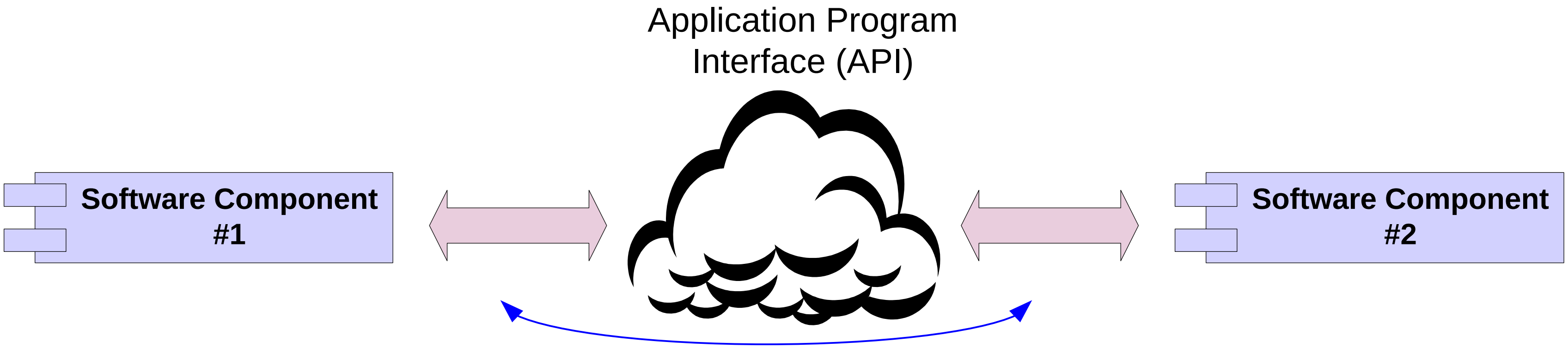
Department of Defense Waveform Standards, Compliance & Certification Directorate
19 May 2018

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. (15 May 2018)

Agenda

- **<u>Application Program Interfaces (API) Definition & Design Patterns</u>**
- **Modem Hardware Abstraction Layer (MHAL) API & MHAL On Chip Bus (MOCB) API**
- **JTRS Platform Adapter (JPA) Interface Spec.**
- **Timing Service API**
- **Vocoder Service API**
- **Wrap-up**

Definition of an API



An API is NOT software

It is a request format agreement

API Document

```
// Function to dial phone number
void dialPhoneNumber(long number)

// Function to setup three-way conference
void threeWayConference(long number, long number)
…
```

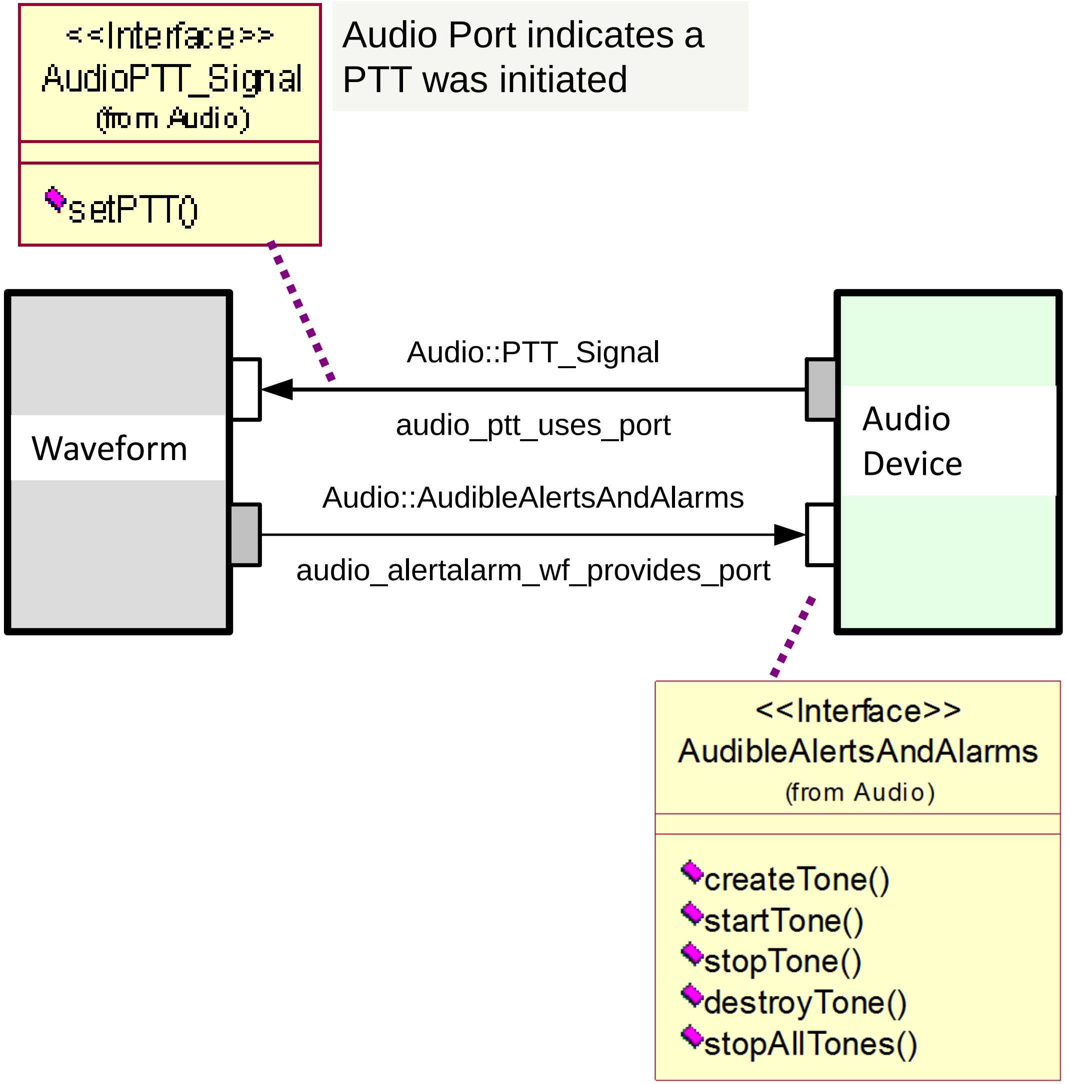
Typically the API is used within the user’s software code to request a service

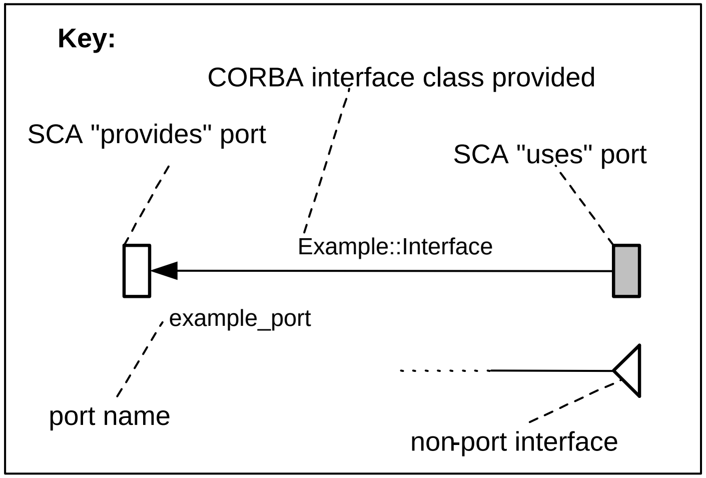
Software program

```
// Prompt operator for phone number
std::cout << “enter phone number!”;

std::cin >> phoneNumber;

// Dial the phone number
dialPhoneNumber(phoneNumber)

…
```

Audio Port Device API Port Diagram

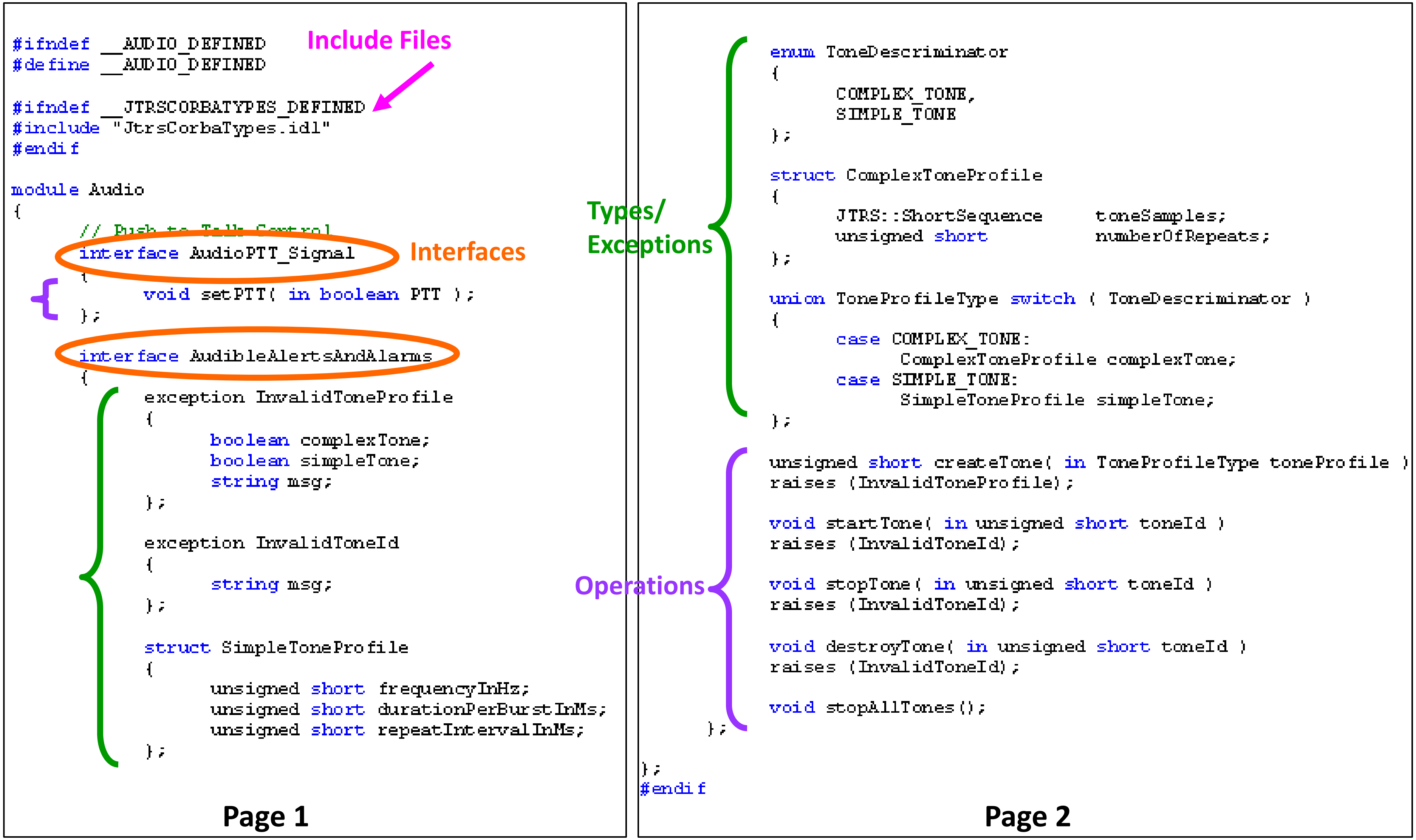
Audio Port indicates a PTT was initiated





Waveform requests the creation/destruction of audible alerts and alarms

Audio Port Device API Interface Description Language(IDL)



Include Files

Interfaces

Types/ Exceptions

Operations

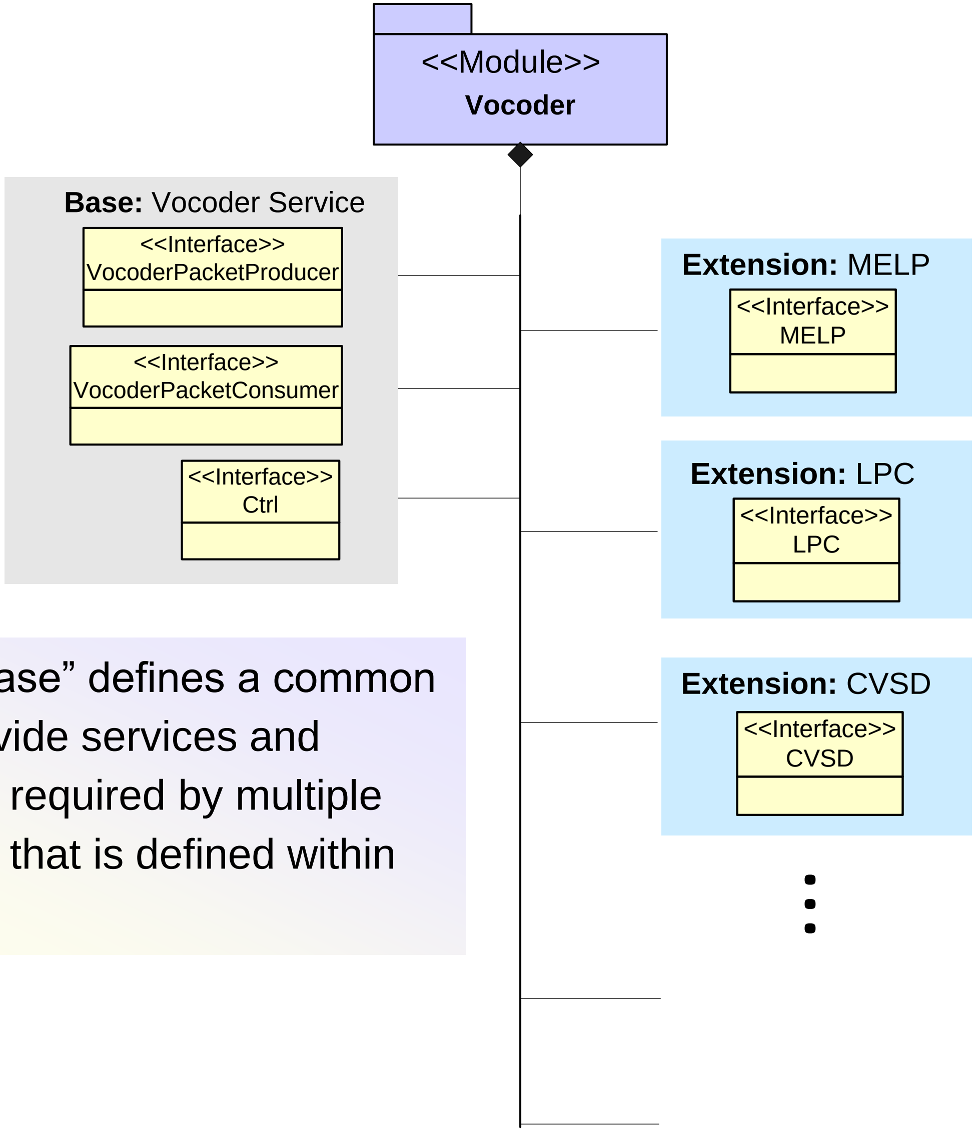
```
#ifndef __AUDIO_DEFINED
#define __AUDIO_DEFINED

#ifndef __JTRSCORBATYPES_DEFINED
#include "JtrsCorbaTypes.idl"
#endif

module Audio
{
      // Push to Talk Control
      interface AudioPTT_Signal
      {
            void setPTT( in boolean PTT );
      };

      interface AudibleAlertsAndAlarms
      {
            exception InvalidToneProfile
            {
                  boolean complexTone;
                  boolean simpleTone;
                  string msg;
            };

            exception InvalidToneId
            {
                  string msg;
            };

            struct SimpleToneProfile
            {
                  unsigned short frequencyInHz;
                  unsigned short durationPerBurstInMs;
                  unsigned short repeatIntervalInMs;
            };
```

Page 1

```
            enum ToneDescriminator
            {
                  COMPLEX_TONE,
                  SIMPLE_TONE
            };

            struct ComplexToneProfile
            {
                  JTRS::ShortSequence       toneSamples;
                  unsigned short            numberOfRepeats;
            };

            union ToneProfileType switch ( ToneDescriminator )
            {
                  case COMPLEX_TONE:
                        ComplexToneProfile complexTone;
                  case SIMPLE_TONE:
                        SimpleToneProfile simpleTone;
            };

            unsigned short createTone( in ToneProfileType toneProfile )
            raises (InvalidToneProfile);

            void startTone( in unsigned short toneId )
            raises (InvalidToneId);

            void stopTone( in unsigned short toneId )
            raises (InvalidToneId);

            void destroyTone( in unsigned short toneId )
            raises (InvalidToneId);

            void stopAllTones();
      };

};
#endif
```

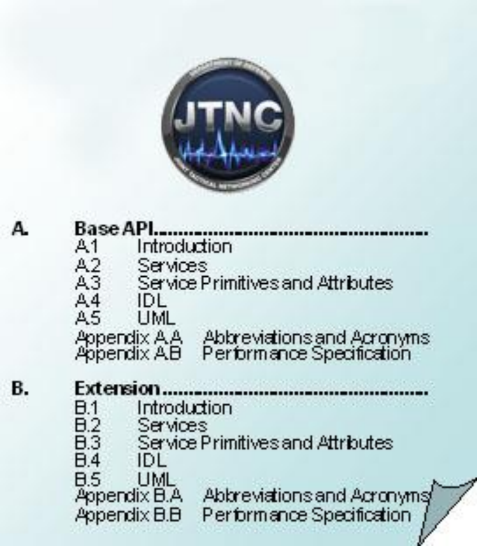
Page 2

Generating Flexible APIs for Tactical Radios



An API "extension" extends the API "base" and defines additional interfaces and operations that may not be needed by multiple radios. There may exist one or more API "extensions" for a specific API.

An API "base" defines a common set of provide services and interfaces required by multiple radio sets that is defined within each API.



Agenda

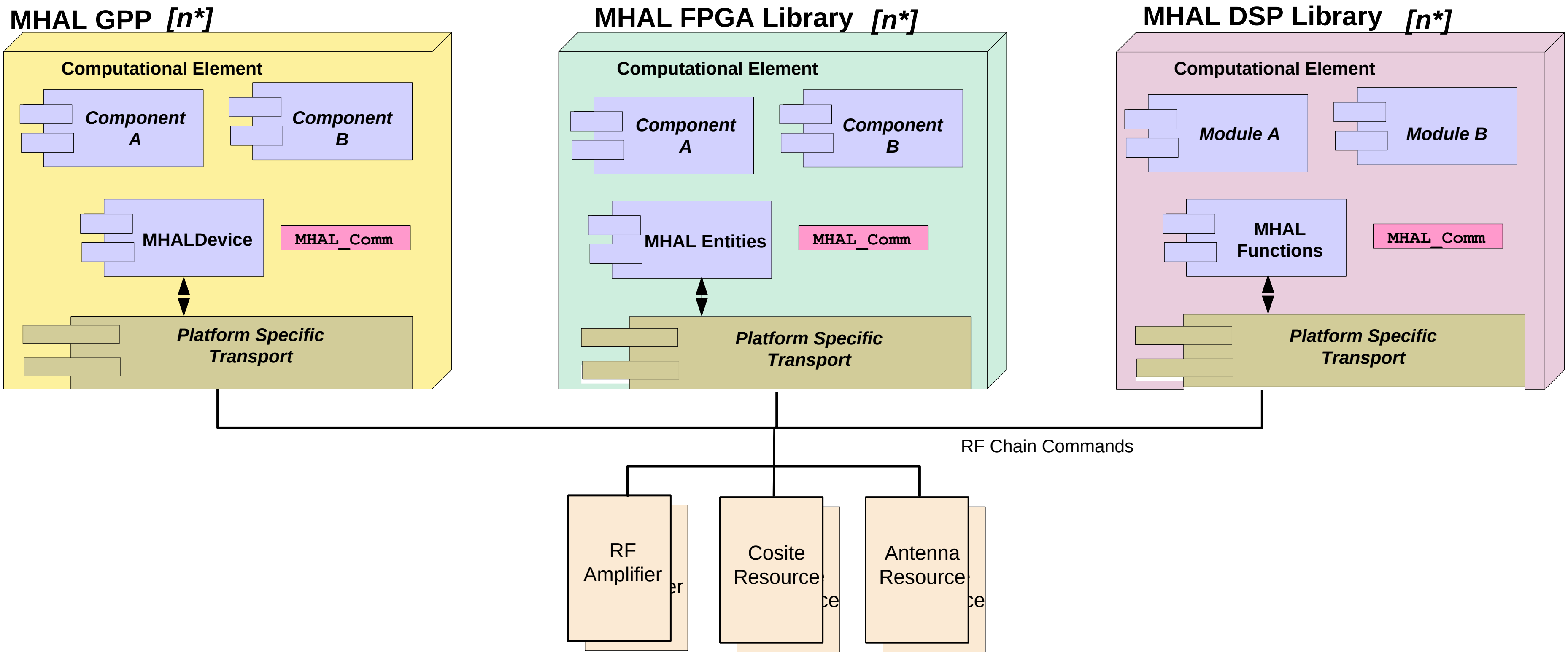
- **API Definition & Design Patterns**
- **MHAL & MOCB APIs**
- **JTRS Platform Adapter (JPA) Interface Spec.**
- **Timing Service API**
- **Vocoder Service API**
- **Wrap-up**

Modem Hardware Abstraction Layer (MHAL) API

- MHAL supports communication between application components hosted on
  - General purpose processors (GPPs)
    - Common Object Request Broker Architecture (CORBA) capable
  - Digital signal processors (DSPs)
    - C capable, no CORBA
  - Field-programmable gate arrays (FPGAs)
    - Hardware description language (HDL) capable, no CORBA

MHAL Communication Service

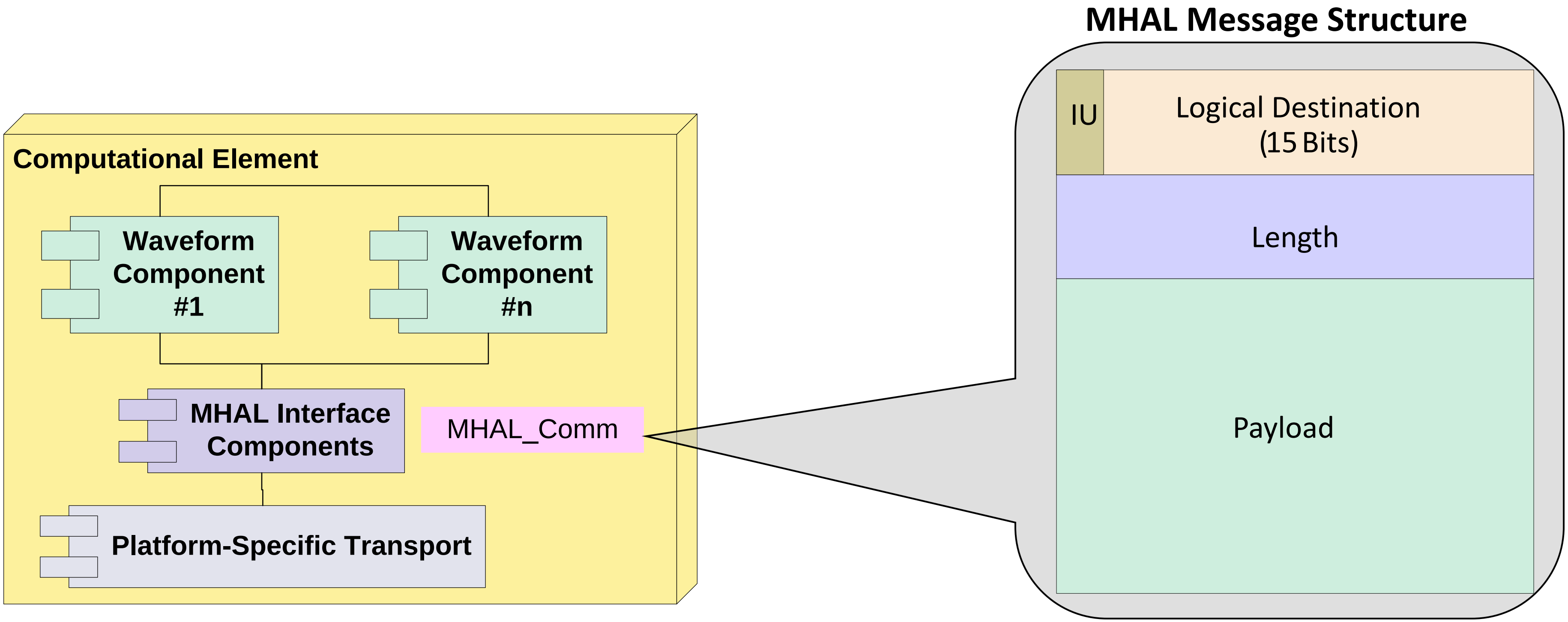
- Enables one Computational Element (CE) to access any of the other CEs (push model)
- Provides the message transport and an abstract message routing function
- Does not specify the platform specific transport, implementation or hardware architecture



MHAL Communication Service: MHAL Message Structure

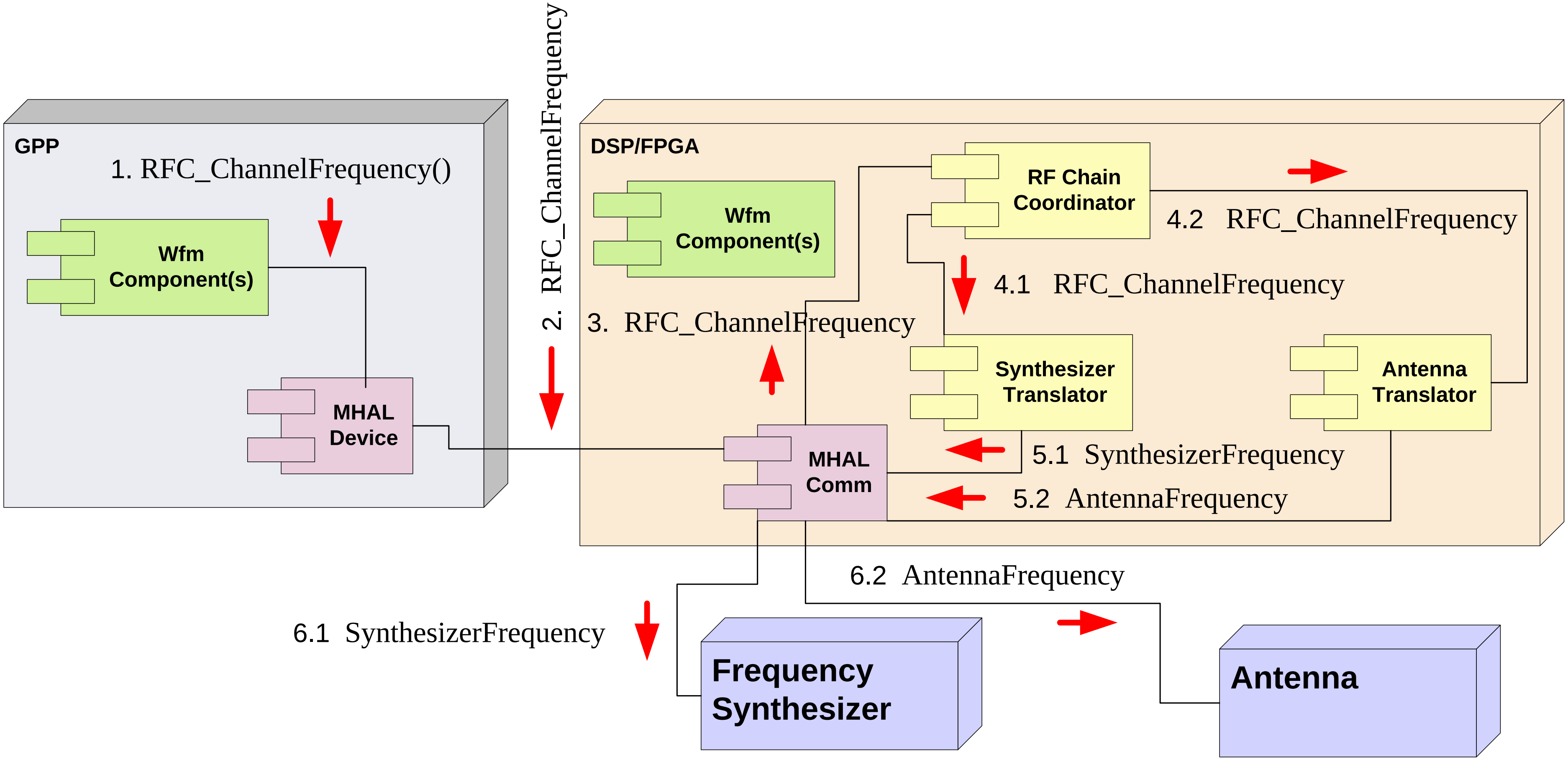
The MHAL API defines MHAL Message Structure:

- For commands that are sent via MHAL Communications Function (i.e. MHAL_Comm)
- Includes information required to maintain orderly processing of message buffers



MHAL RF Chain Coordinator API

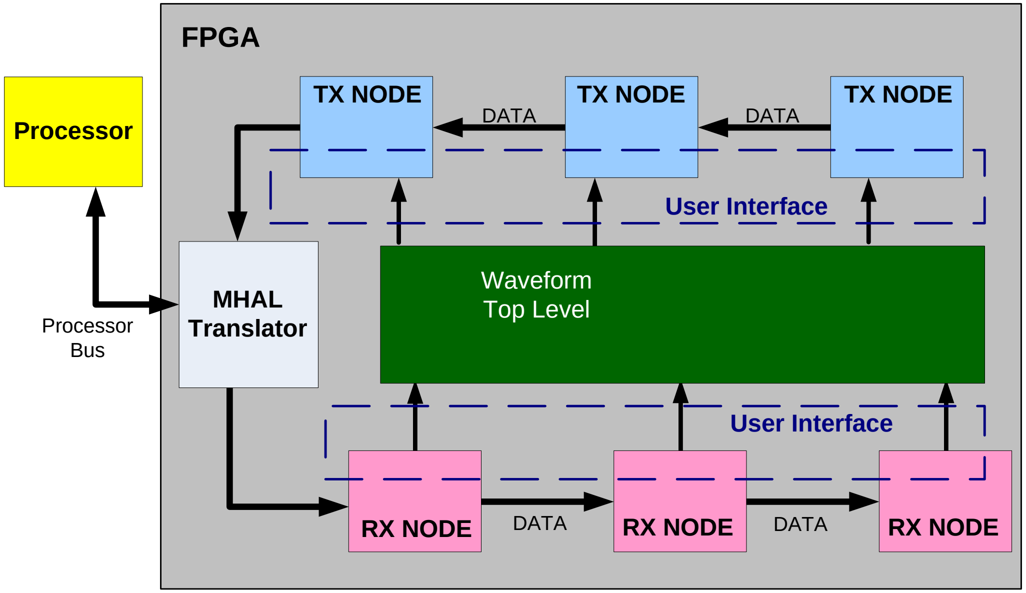
Consists of a set of sink and source functions that provide coordinated control of a communications channel’s RF resources.



Note: RF Chain Coordinator and Translators are optional

MHAL API Lessons Learned

- Well adapted to push packet interfaces
- Not well adapted to memory mapped interfaces
  - Integration and Test
    - ➢ Reduced Visibility & Flexibility
  - Implementation
    - ➢ Additional latency associated with data retrieval
    - ➢ Increased FPGA resource requirements

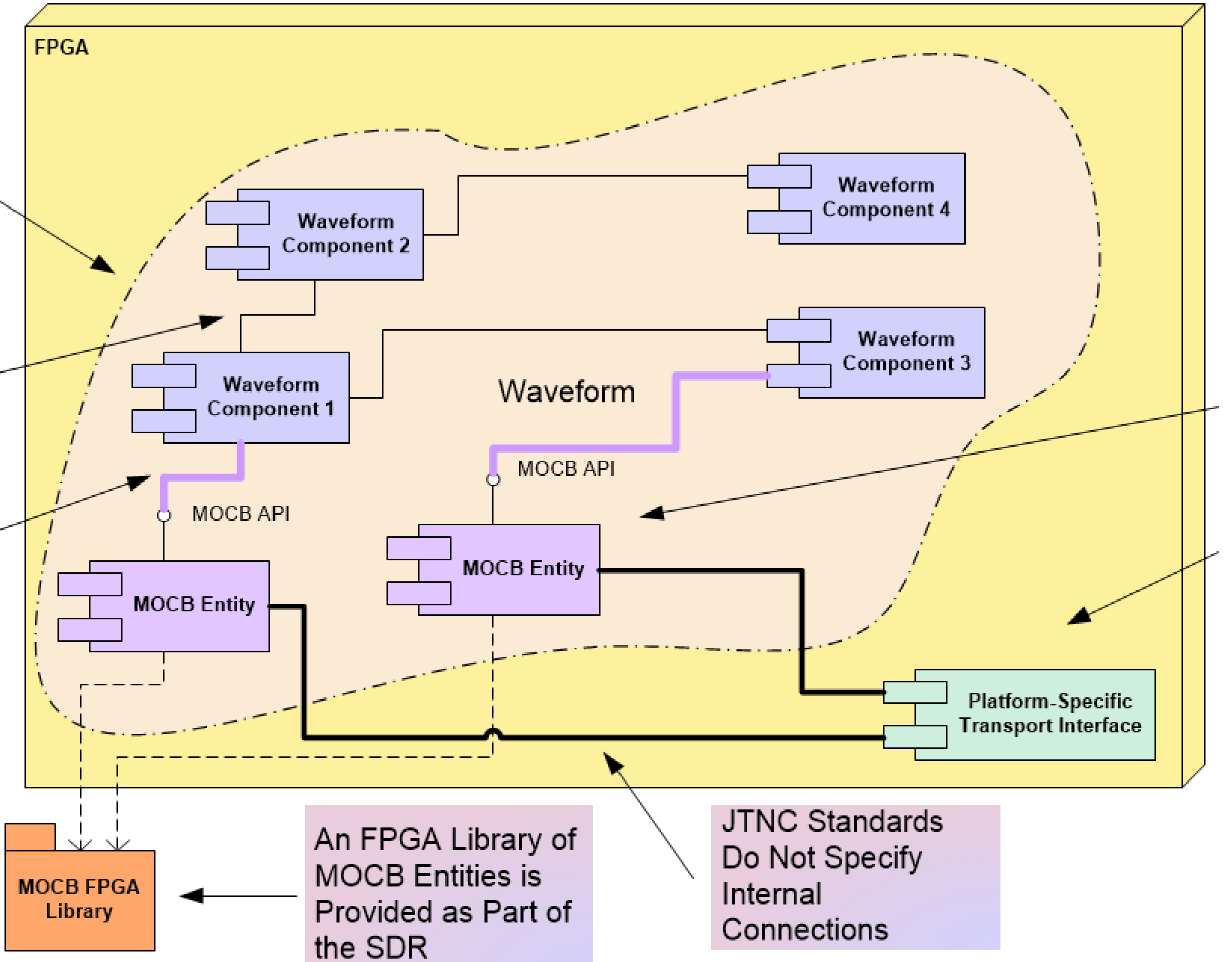


JTNC Interface for FPGAs

Concept is to Provide a Common Host Platform for the Waveform/ Applications on any SDR

JTNC Standards do not Specify Internal Waveform Interfaces

JTNC Standards Specifies the User Interfaces Between the Waveform and the Hardware



JTNC Standards do not Specify the Implementation

JTNC Standards Do Not Specify the Transport

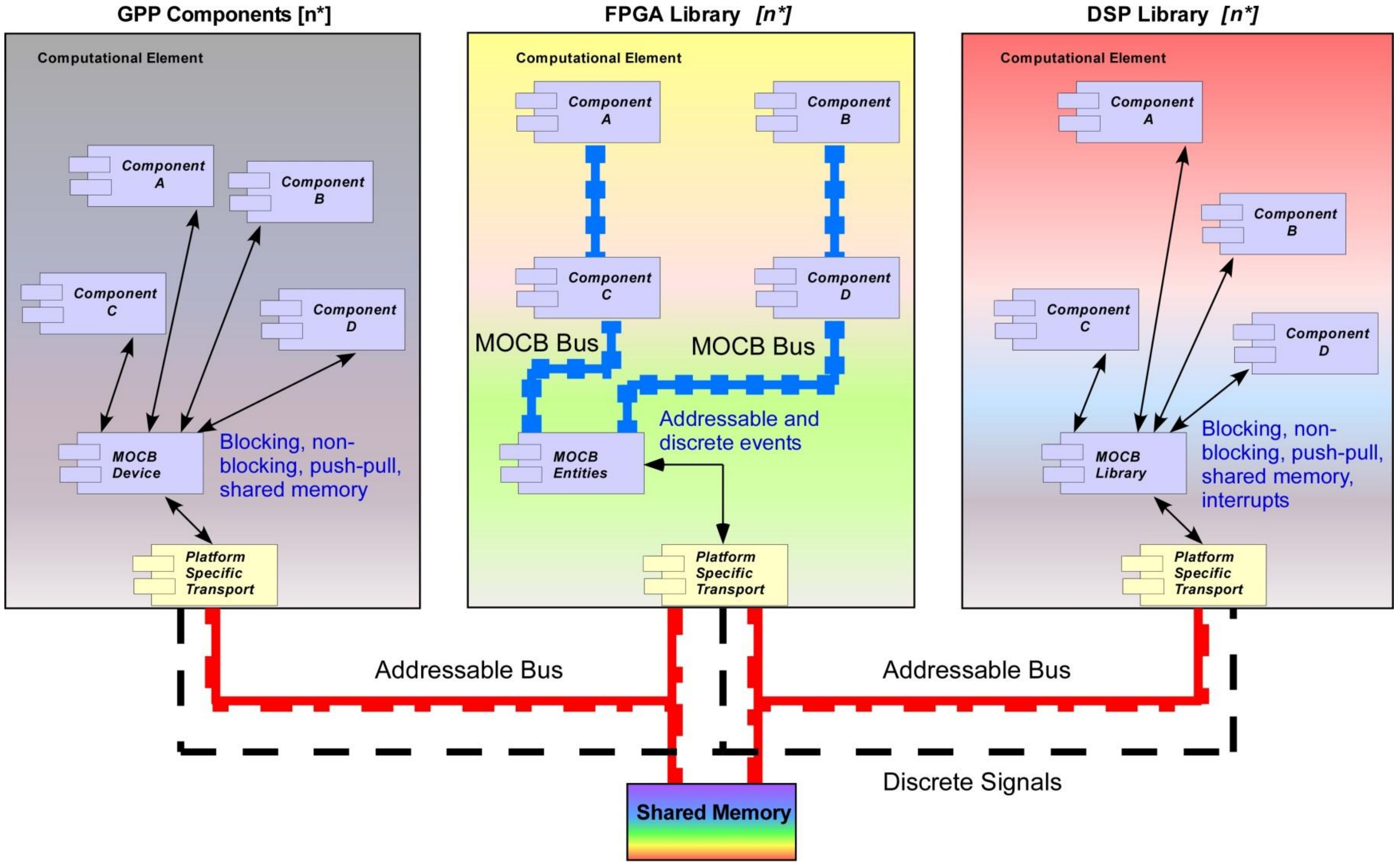
An FPGA Library of MOCB Entities is Provided as Part of the SDR

JTNC Standards Do Not Specify Internal Connections

MHAL on Chip Bus (MOCB) API defines parallel interfaces between the modem interfaces from the application software.

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. (15 May 2018)

JTNC API - MHAL On Chip Bus (MOCB)



MOCB API introduces interrupts, discrete signals, shared memory, and supports time-critical waveforms.

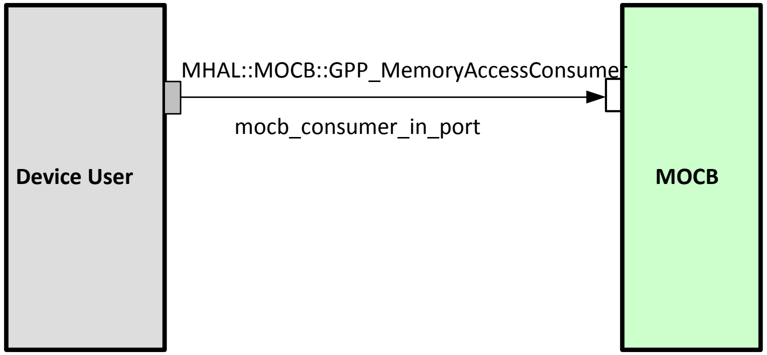
MOCB API Composition

- MOCB GPP API Extension
  - SCA CF::Device based interface
- MOCB DSP API Extension
  - Library of standardized components linked into the waveform code at build time
- MOCB FPGA API Extension
  - Waveform HDL to form a single loadable FPGA image (entity library) linked into a waveform build
- MOCB RF Chain Coordinator API Extension

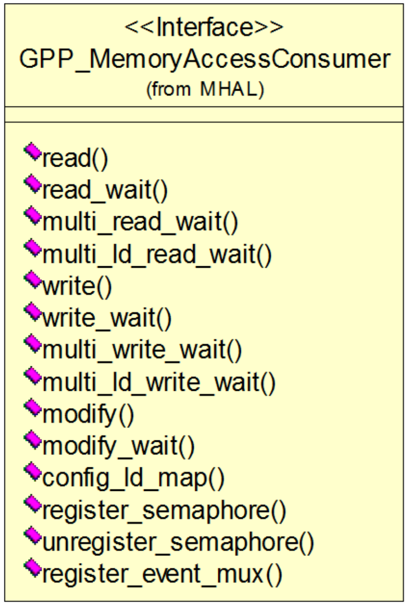
MOCB GPP API Extension Overview

- Extends MOCB base API and supports methods and attributes that are specific to GPP modem hardware
- Provides ability to synchronously and asynchronously read/write/modify data to/from a service user/provider's shared memory
- Defines memory mapped interfaces that specify a read (pull) capability between components
- Defines a structure for mapping a logical destination (LD) and address

MOCB Port Diagram



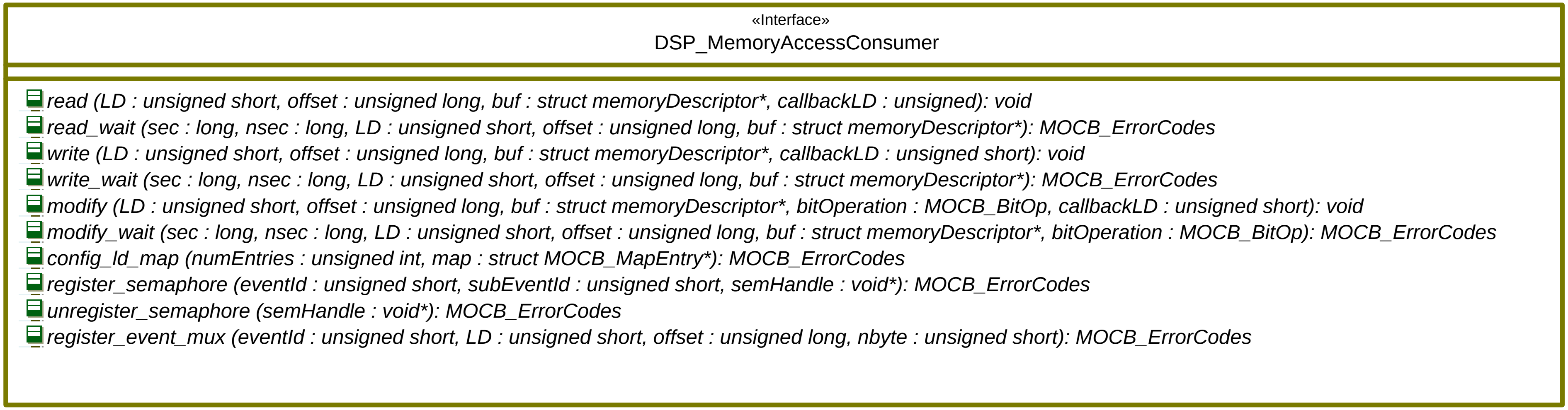
MOCB Class Diagram



MOCB DSP API Extension Overview

- Extends the MHAL DSP API Extension
- Collection of C function definitions
- Specify the ability to synchronously and asynchronously read/write/modify data to/from a service user/provider’s shared memory and control platform defined events

MOCB Interface Diagram

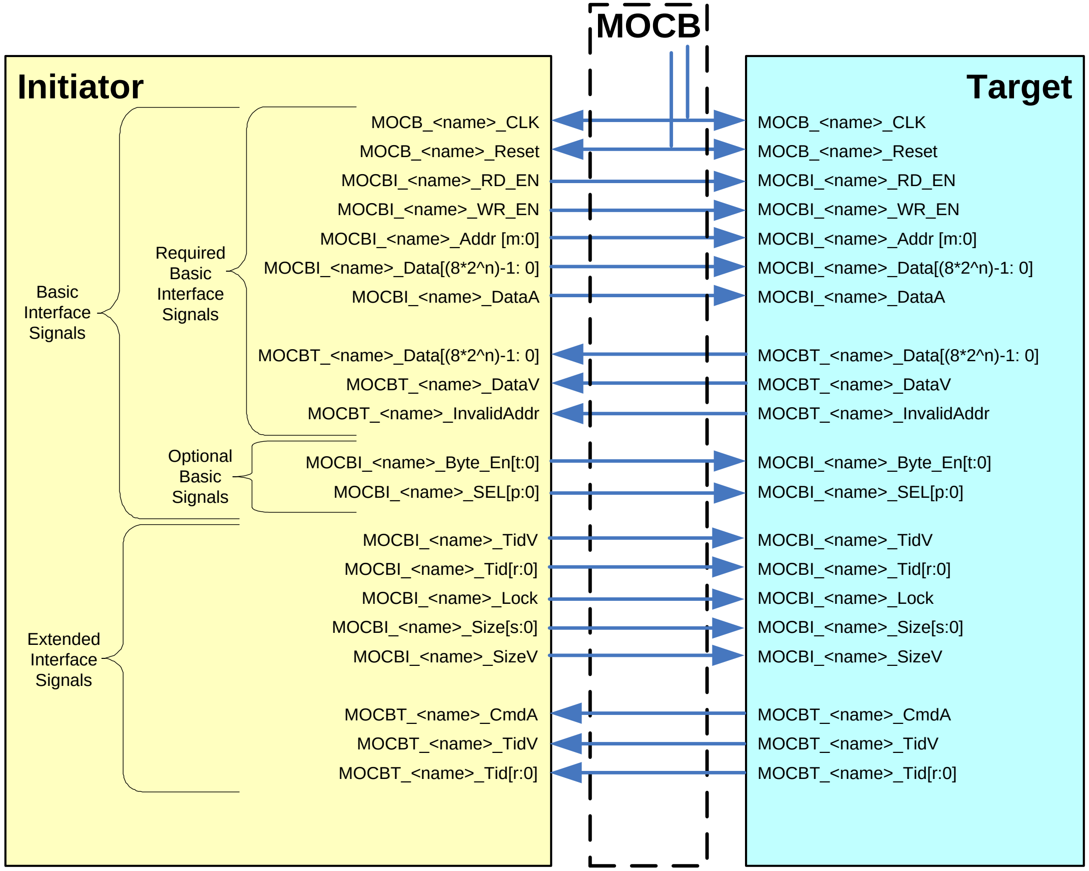


«Interface»
DSP_MemoryAccessConsumer

read (LD : unsigned short, offset : unsigned long, buf : struct memoryDescriptor, callbackLD : unsigned): void*
read_wait (sec : long, nsec : long, LD : unsigned short, offset : unsigned long, buf : struct memoryDescriptor): MOCB_ErrorCodes*
write (LD : unsigned short, offset : unsigned long, buf : struct memoryDescriptor, callbackLD : unsigned short): void*
write_wait (sec : long, nsec : long, LD : unsigned short, offset : unsigned long, buf : struct memoryDescriptor): MOCB_ErrorCodes*
modify (LD : unsigned short, offset : unsigned long, buf : struct memoryDescriptor, bitOperation : MOCB_BitOp, callbackLD : unsigned short): void*
modify_wait (sec : long, nsec : long, LD : unsigned short, offset : unsigned long, buf : struct memoryDescriptor, bitOperation : MOCB_BitOp): MOCB_ErrorCodes*
config_ld_map (numEntries : unsigned int, map : struct MOCB_MapEntry): MOCB_ErrorCodes*
register_semaphore (eventId : unsigned short, subEventId : unsigned short, semHandle : void): MOCB_ErrorCodes*
unregister_semaphore (semHandle : void): MOCB_ErrorCodes*
register_event_mux (eventId : unsigned short, LD : unsigned short, offset : unsigned long, nbyte : unsigned short): MOCB_ErrorCodes

MOCB FPGA API Extension Overview

- Interface Categories
  - Address / Data bus interface
  - Event interface
- Signal Categories
  - Required basic interface signals
  - Optional basic signals
  - Extended interface signals



MOCB

Initiator

Basic Interface Signals

Required Basic Interface Signals

MOCB_<name>_CLK

MOCB_<name>_Reset

MOCBI_<name>_RD_EN

MOCBI_<name>_WR_EN

MOCBI_<name>_Addr [m:0]

MOCBI_<name>_Data[(8*2^n)-1: 0]

MOCBI_<name>_DataA

MOCBT_<name>_Data[(8*2^n)-1: 0]

MOCBT_<name>_DataV

MOCBT_<name>_InvalidAddr

Optional Basic Signals

MOCBI_<name>_Byte_En[t:0]

MOCBI_<name>_SEL[p:0]

Extended Interface Signals

MOCBI_<name>_TidV

MOCBI_<name>_Tid[r:0]

MOCBI_<name>_Lock

MOCBI_<name>_Size[s:0]

MOCBI_<name>_SizeV

MOCBT_<name>_CmdA

MOCBT_<name>_TidV

MOCBT_<name>_Tid[r:0]

Target

MOCB_<name>_CLK

MOCB_<name>_Reset

MOCBI_<name>_RD_EN

MOCBI_<name>_WR_EN

MOCBI_<name>_Addr [m:0]

MOCBI_<name>_Data[(8*2^n)-1: 0]

MOCBI_<name>_DataA

MOCBT_<name>_Data[(8*2^n)-1: 0]

MOCBT_<name>_DataV

MOCBT_<name>_InvalidAddr

MOCBI_<name>_Byte_En[t:0]

MOCBI_<name>_SEL[p:0]

MOCBI_<name>_TidV

MOCBI_<name>_Tid[r:0]

MOCBI_<name>_Lock

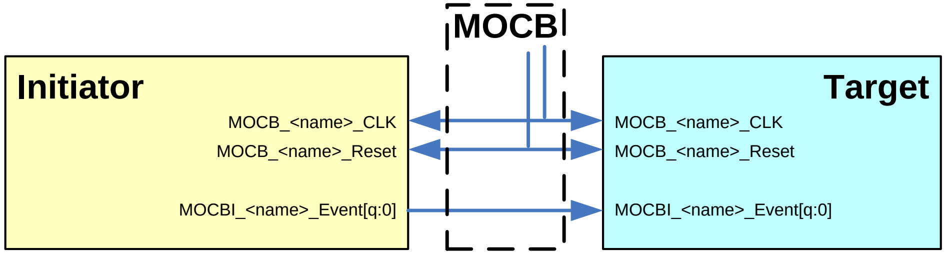
MOCBI_<name>_Size[s:0]

MOCBI_<name>_SizeV

MOCBT_<name>_CmdA

MOCBT_<name>_TidV

MOCBT_<name>_Tid[r:0]



MOCB

Initiator

MOCB_<name>_CLK

MOCB_<name>_Reset

MOCBI_<name>_Event[q:0]

Target

MOCB_<name>_CLK

MOCB_<name>_Reset

MOCBI_<name>_Event[q:0]

MHAL API and MOCB API Summary

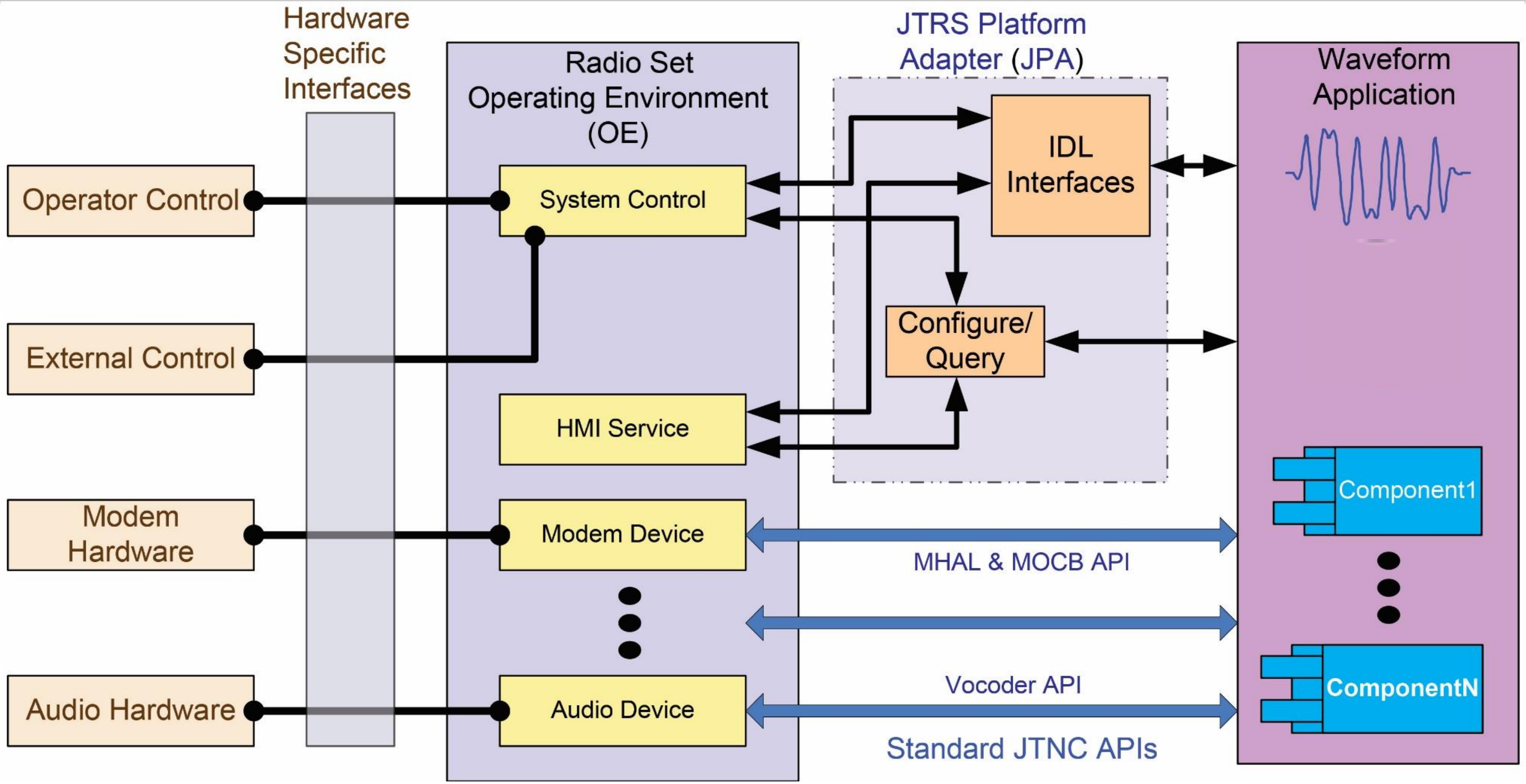
- MHAL API utilizes a push model approach
- MOCB API uses a memory mapped (pull) model approach
- MOCB API is the recommended approach going forward
  - Applicable to new and existing waveform
  - Provides support for higher speed and legacy waveforms
    - Increases visibility
    - Increases predictability
    - Decreases resource usage

➢ Still room for improvement – we are always receptive to better ideas!

Agenda

- API Definition & Design Patterns
- MHAL & MOCB APIs
- JTRS Platform Adapter (JPA) Interface Spec.
- Timing Service API
- Vocoder Service API
- Wrap-up

JTRS Platform Adapter (JPA) Adapts Waveform Interface



How to present a single command & control to the waveform baseline?

- JPA is an interface that isolates the configuration and management of the waveform from the radio set

Agenda

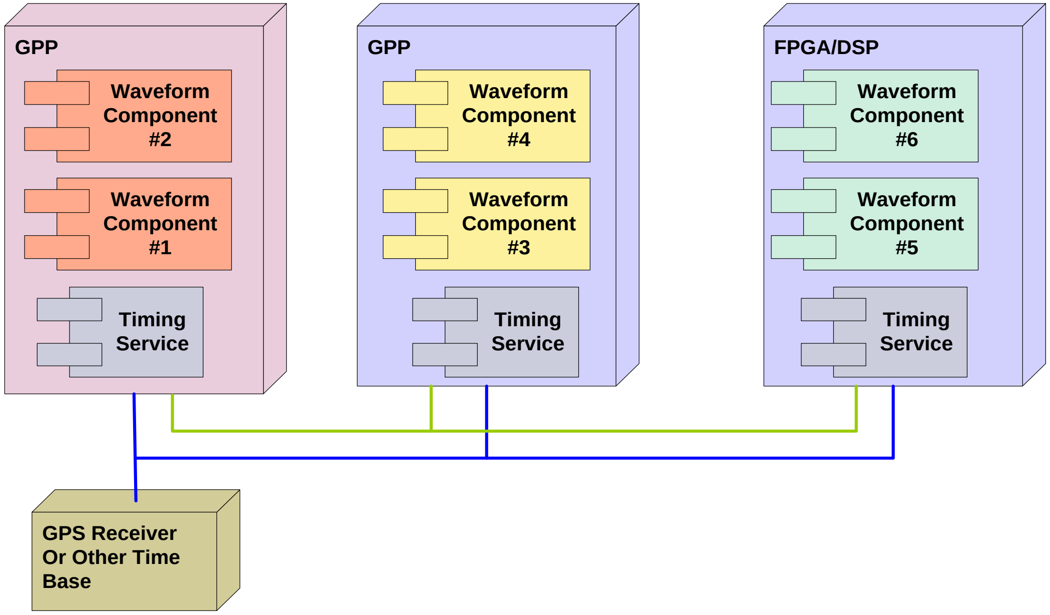
- API Definition & Design Patterns
- MHAL & MOCB APIs
- JTRS Platform Adapter (JPA) Interface Spec.
- <u>Timing Service API</u>
- Vocoder Service API
- Wrap-up

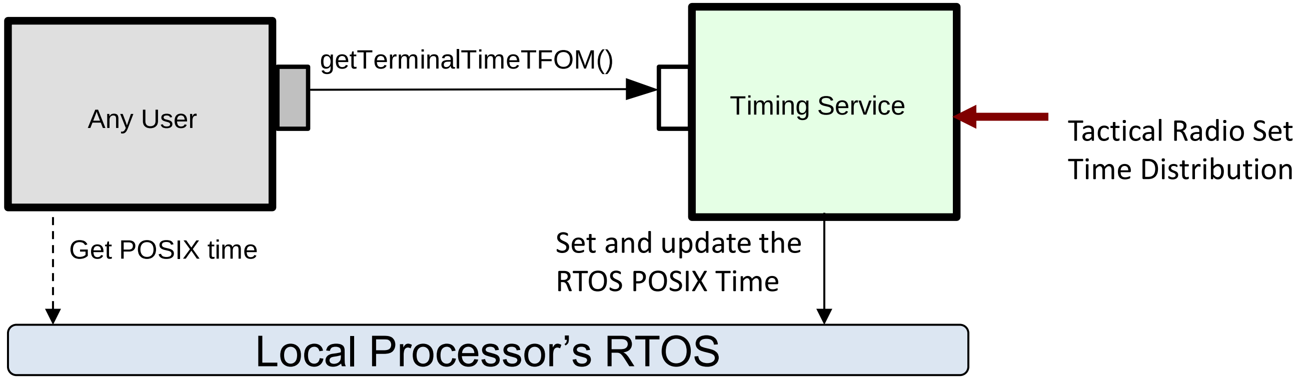
Concept of the Timing Service

How to distribute timing within a multi-channel radio with non-synchronized network times?

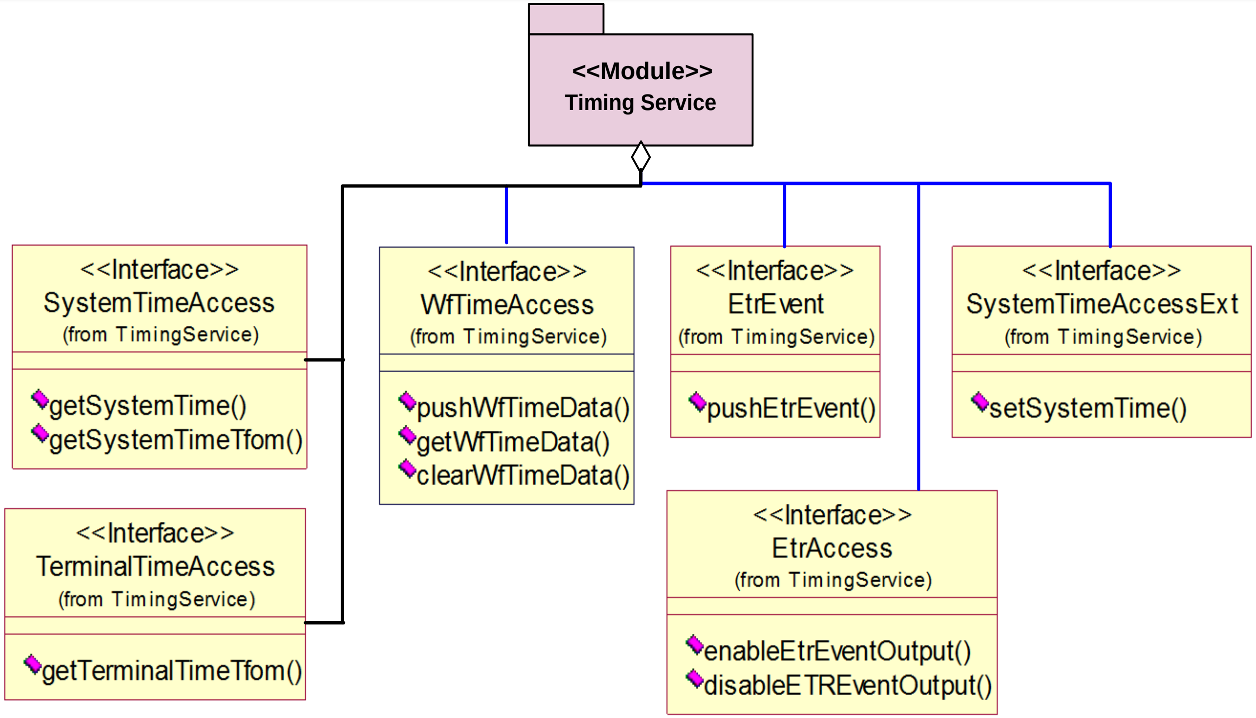
Almost all networking waveforms have their network time, which may or may not be synched to GPS time.

The use cases become complicated when there is multiple networking radio running on the radio – which is the reference time?





JTNC API - Timing Service API



- Terminal Time is the time returned from POSIX time and is monotonic increasing
- The *Timing Service* synchronizes the Terminal Time between distributed components within the terminal
- The *Timing Service* controls the local processor’s POSIX clock

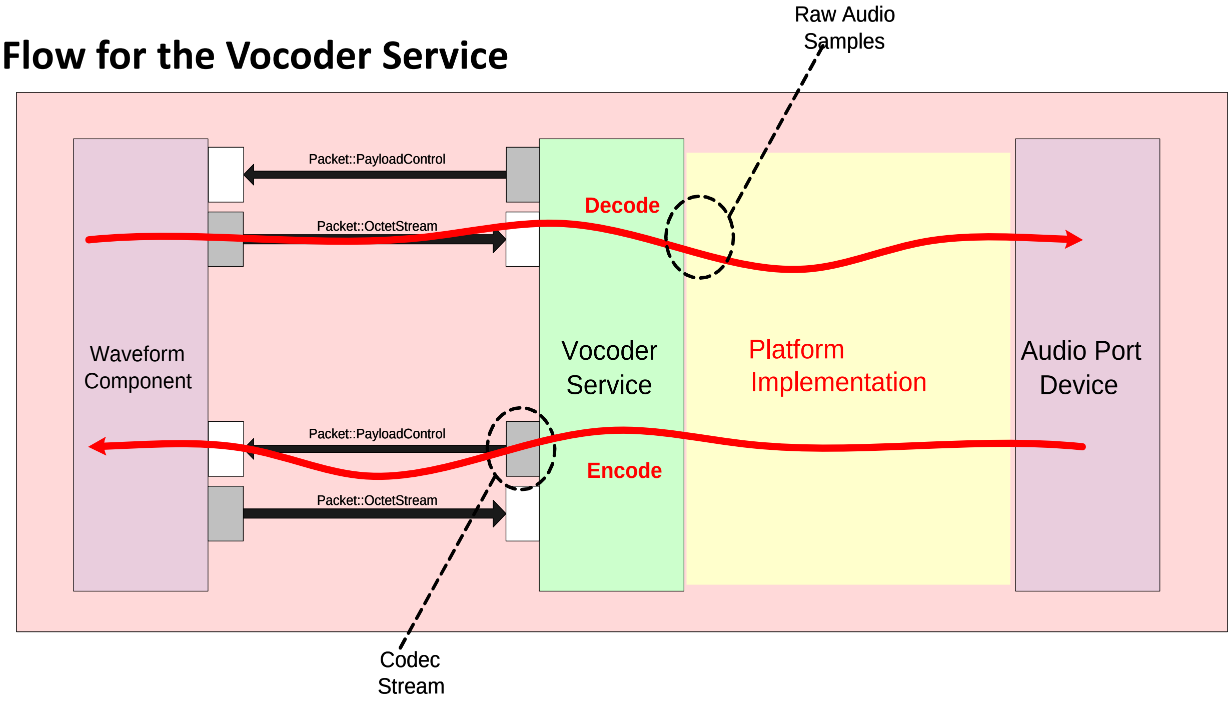
Agenda

- API Definition & Design Patterns
- MHAL & MOCB APIs
- JTRS Platform Adapter (JPA) Interface Spec.
- Timing Service API
- Vocoder Service API
- Wrap-up

Vocoder Service API

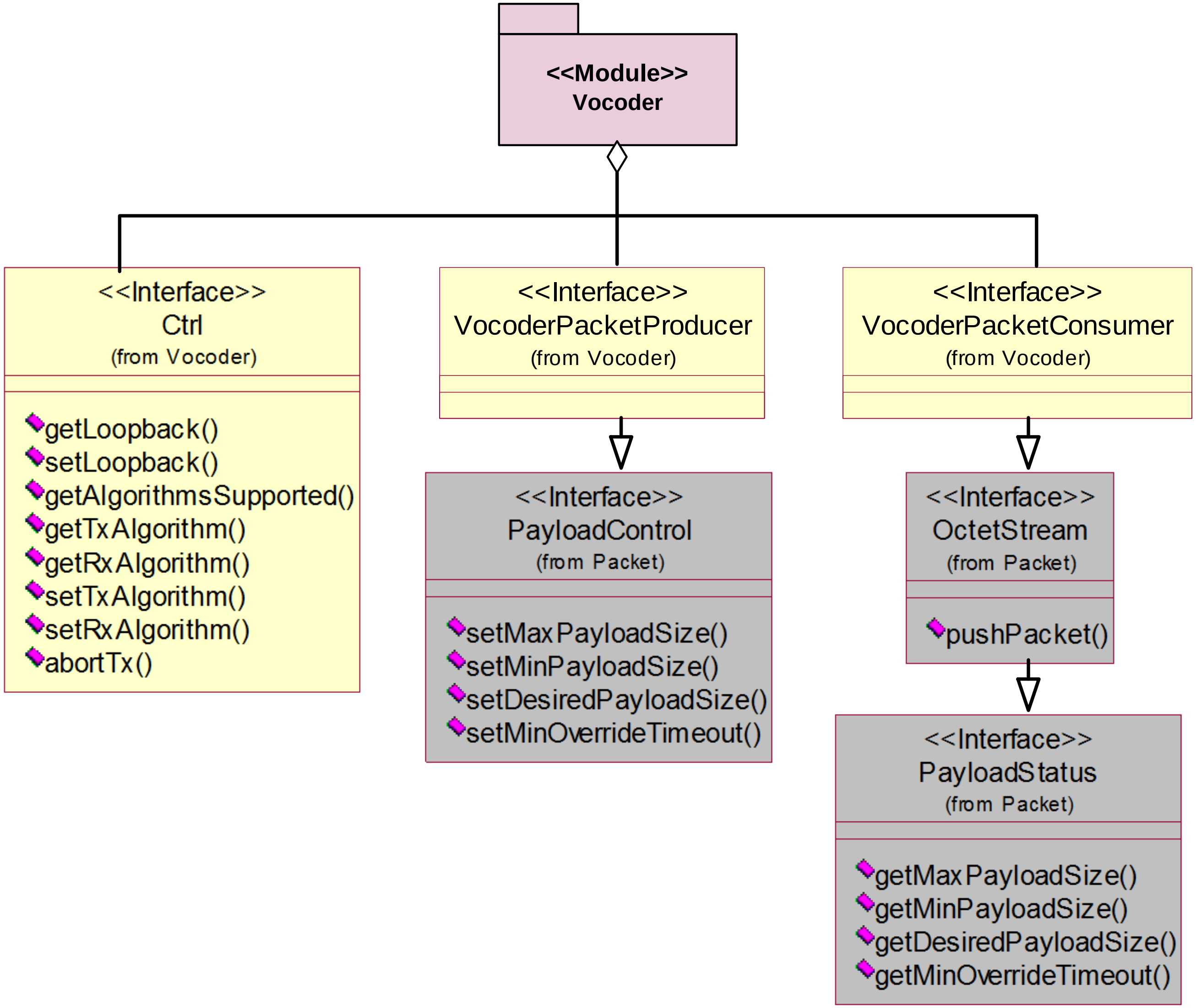
How to add vocoder algorithms to a radio without breaking all of the software?

Signal Flow for the Vocoder Service

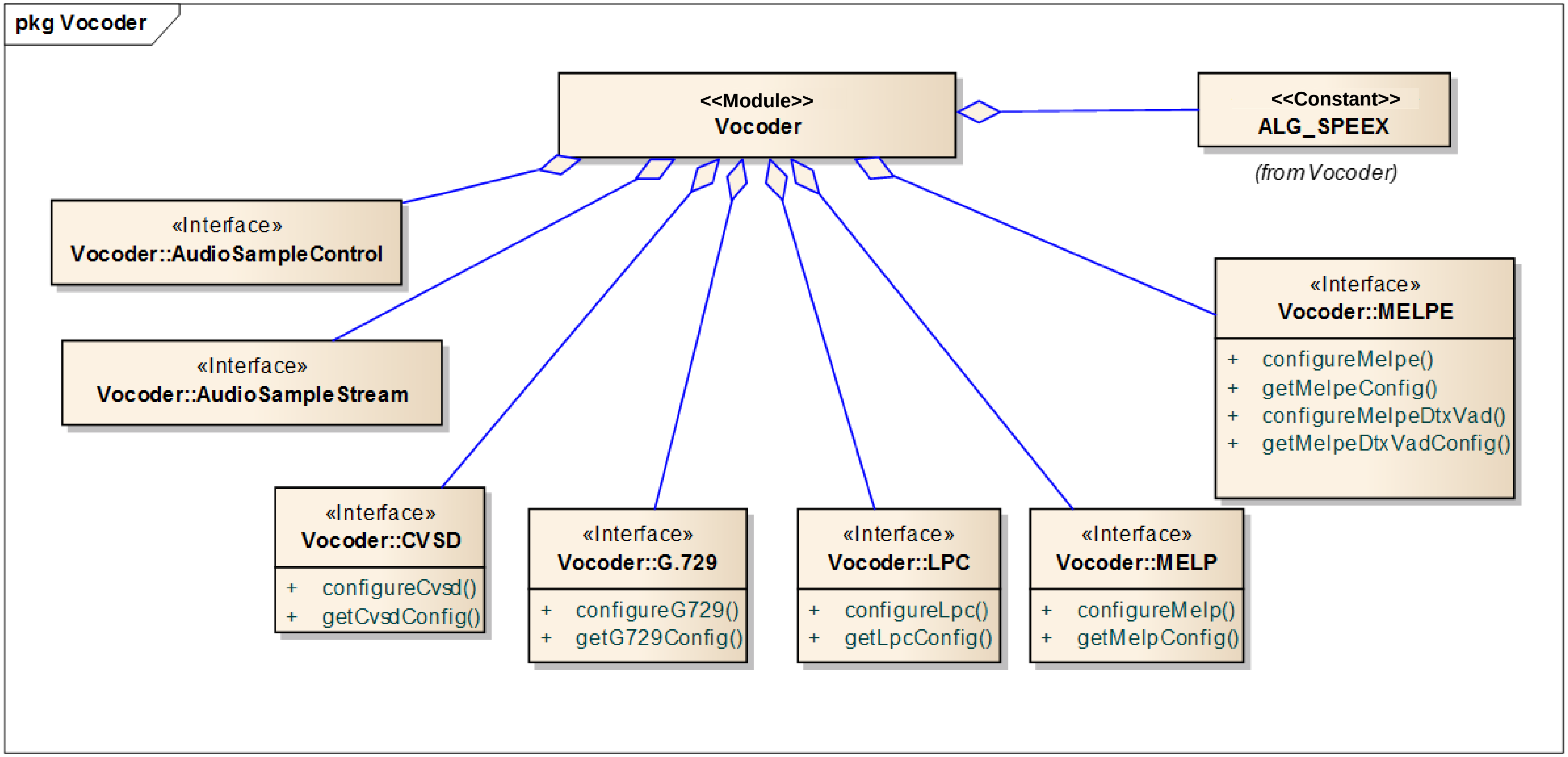


- Vocoder Service has a companion API – the Audio Port Device
- Platform can provide an implicit connection between the Vocoder Service as shown, or SCA port connections can be applied by the application factory
- Audio Port Device provides the ability to inject tones and other signals in-band with the audio stream

Base Interface for the Vocoder Service API



Extensions for the Vocoder Service API



- Radio set needs only implement the vocoders required by their waveform set

Agenda

- **API Definition & Design Patterns**
- **MHAL & MOCB APIs**
- **JTRS Platform Adapter (JPA) Interface Spec.**
- **Timing Service API**
- **Vocoder Service API**
- **Wrap-up**

Tactical Radio Standards in the DISR

Many Tactical Radio Standards have been registered in the DoD Information Standards Registry (DISR)

Standard Identifier	Standard Identifier
SCA 4.1:2015	Tactical Radio API - MHAL on Chip Bus (MOCB) 1.1.5
Tactical Radio API - Device IO 1.0.2	Tactical Radio API - Packet 2.0.2
Tactical Radio API - Audio Port Device 1.3.4	Tactical Radio API - Platform Adapter 1.3.3
Tactical Radio API - CORBA Types 1.0.2	Tactical Radio API - Serial Port Device 2.1.4
Tactical Radio API - Device IO Control	Tactical Radio API - Timing Svce 1.4.4
Tactical Radio API - Device Message Control 1.1.3	Tactical Radio API - Vocoder Svce 1.3.3
Tactical Radio API - Device Packet	
Tactical Radio API - Device Packet Signals	
Tactical Radio API - Ethernet Device 1.2.2	
Tactical Radio API - Frequency Reference Device	
Tactical Radio API - GPS Device 2.1.4	
Tactical Radio API - IO Signals	
Tactical Radio API - M1553 Specification	
Tactical Radio API - MHAL 3.0	

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. (15 May 2018)

Thank you

Contact Info:

Joint Tactical Networking Center
Department of Defense Waveform Standards, Compliance & Certification Directorate
JTNC_Standadards@navy.mil
http://www.public.navy.mil/jtnc