



Joint Program Executive Office Joint Tactical Radio System

Introducing the JTRS Public APIs

**JTRS Standards
5 June 2007**

JPEO JTRS

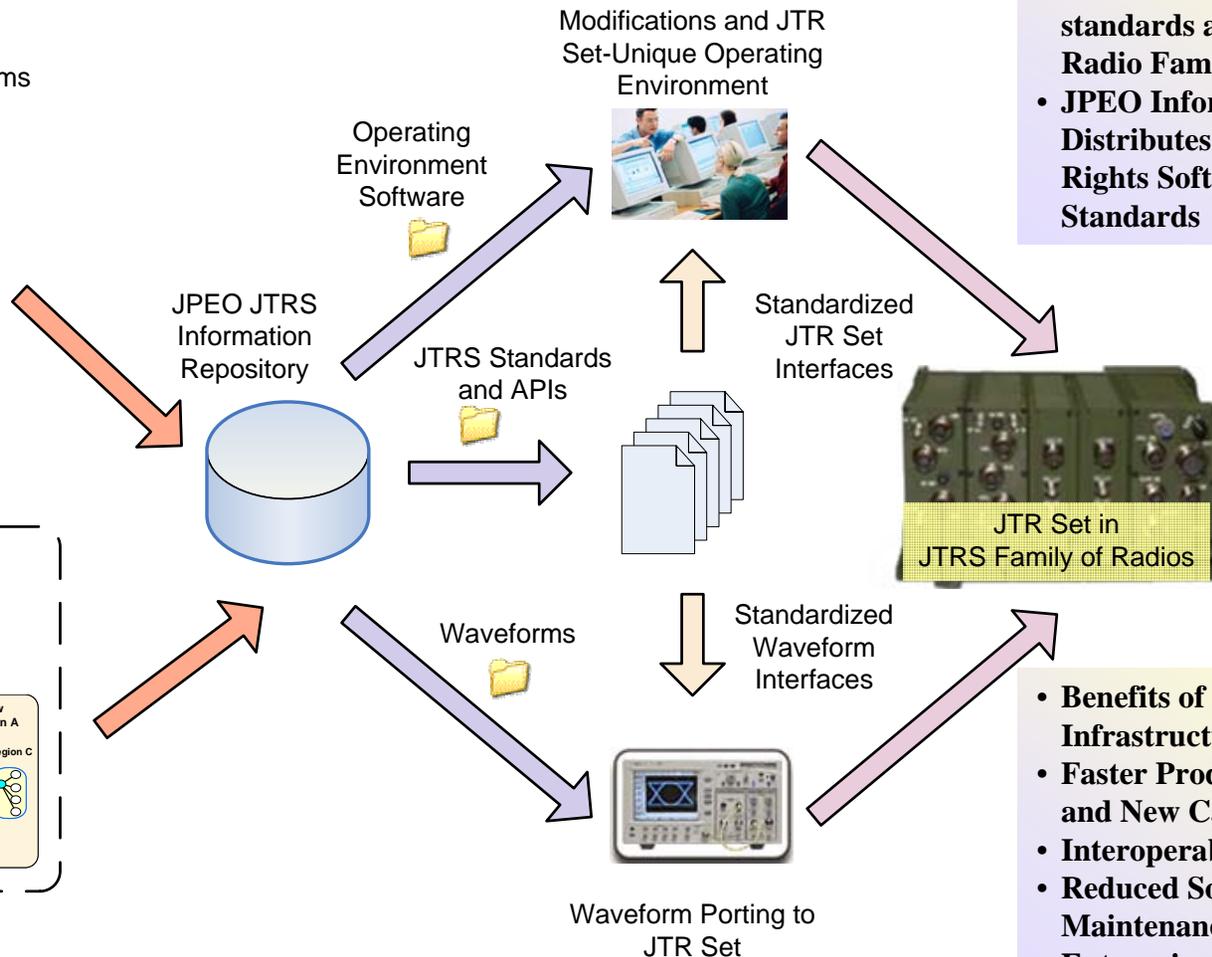
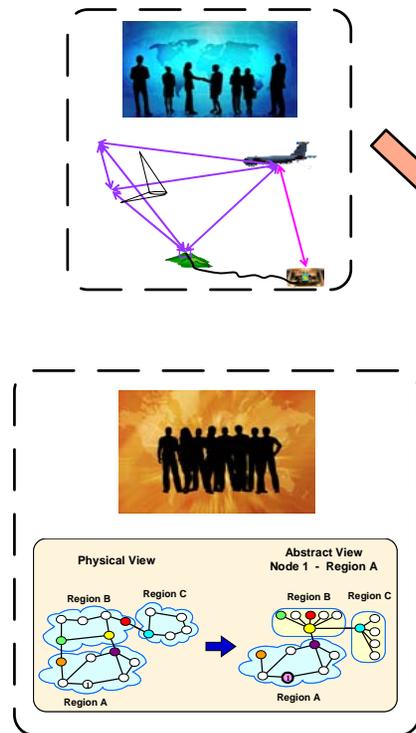
Topics

- ▶ The purpose of JTRS APIs
- ▶ How the JTRS APIs were developed
- ▶ Sampling of the public release JTRS APIs
 - Packet API
 - Vocoder
 - MHAL



Role of the JTRS Infrastructure

Waveform Development Teams

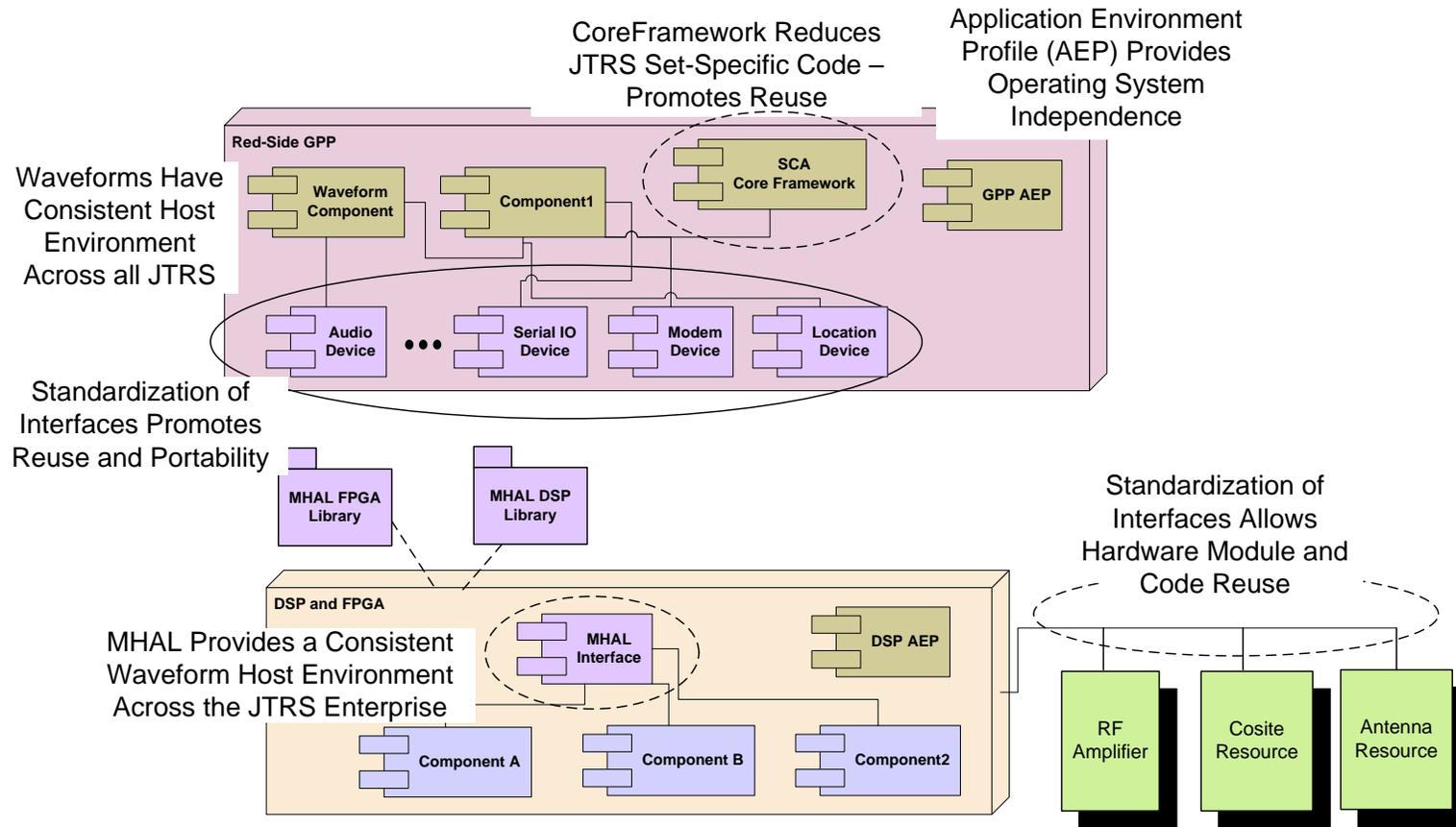


- Reuse is possible because of consistent interface and standards across the JTRS Radio Family
- JPEO Information Repository Distributes Government Rights Software and JTRS Standards

- Benefits of JTRS Infrastructure:
- Faster Product Development and New Capability Fielding
- Interoperability
- Reduced Software Maintenance
- Enterprise-Wide Documentation



Deployment of the JTRS Infrastructure

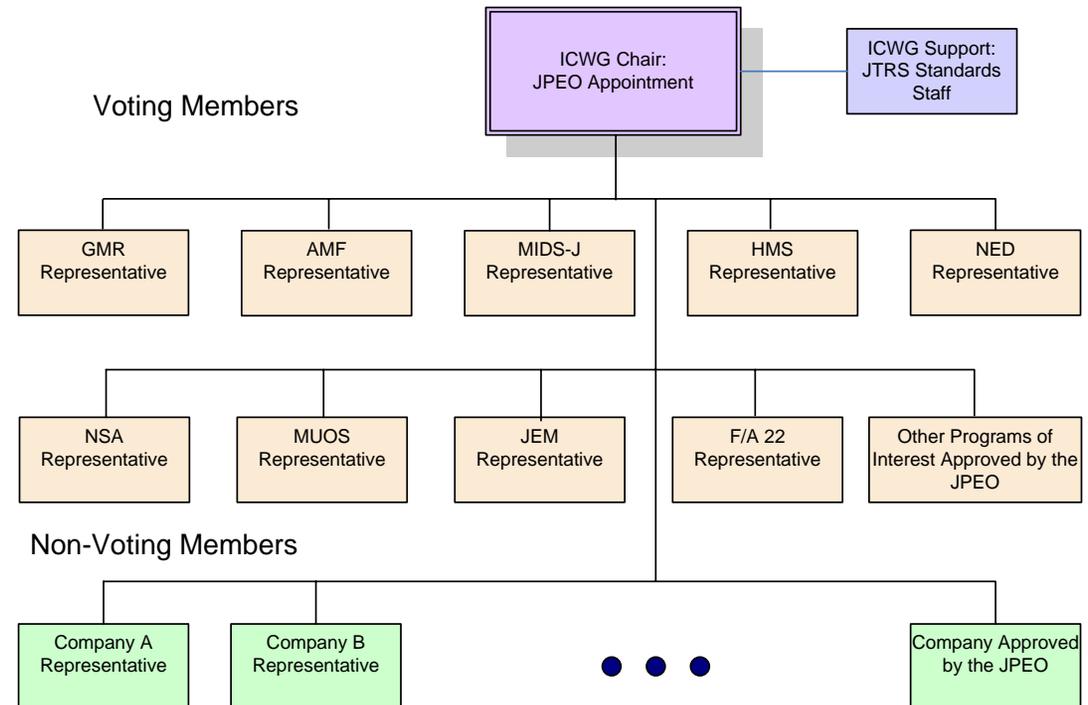


- APIs and Standards Approved by the JTRS Interface Control Working Group
- APIs Include Interfaces to Radio Hardware and Services such as Ethernet Device or Serial Port Device
- SCA 2.2.2 with extensions



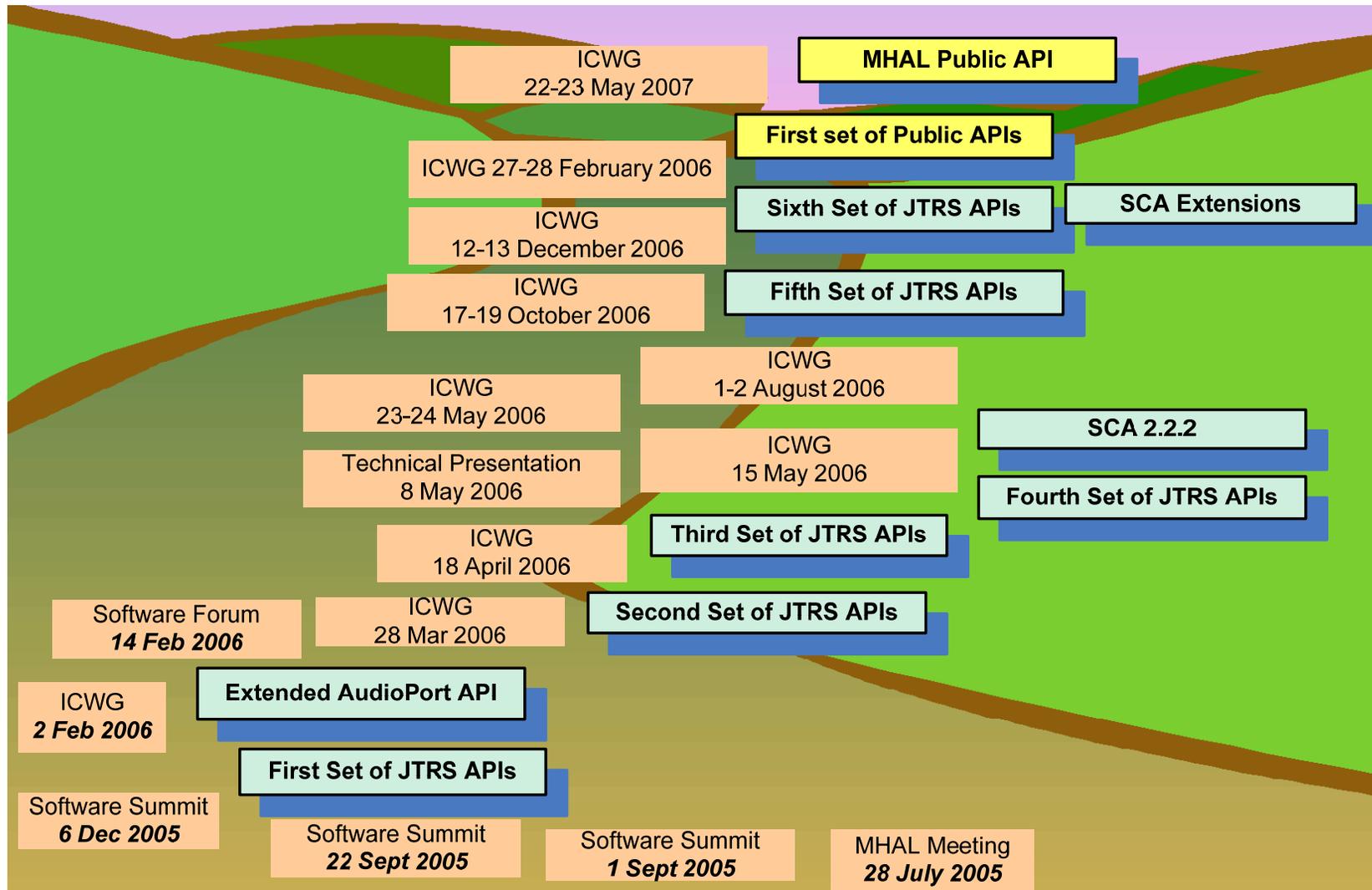
JPEO Interface Control Working Group (ICWG) Organization

- The ICWG Voting Members are responsible for:
 - Reviewing/evaluating APIs and SARs for technical, cost, schedule, impacts across the entire JTRS enterprise;
 - Participating in scheduled meetings to provide representation required to support disposition of changes;
 - Serving as a voting member on the ICWG.
- The ICWG Non-Voting Members are responsible for:
 - Reviewing/evaluating APIs and SARs for technical, cost, schedule, impacts across their representative programs;
 - Participating in scheduled meetings to provide representation required to discuss disposition of changes;





How the JTRS Infrastructure is Being Defined

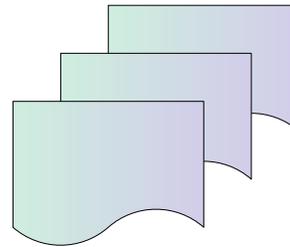




Public JTRS APIs

- Primitive APIs provide messaging and signaling interfaces used by the more complex APIs.
- Complex APIs define radio devices and services

JTRS Public APIs



Primitive APIs



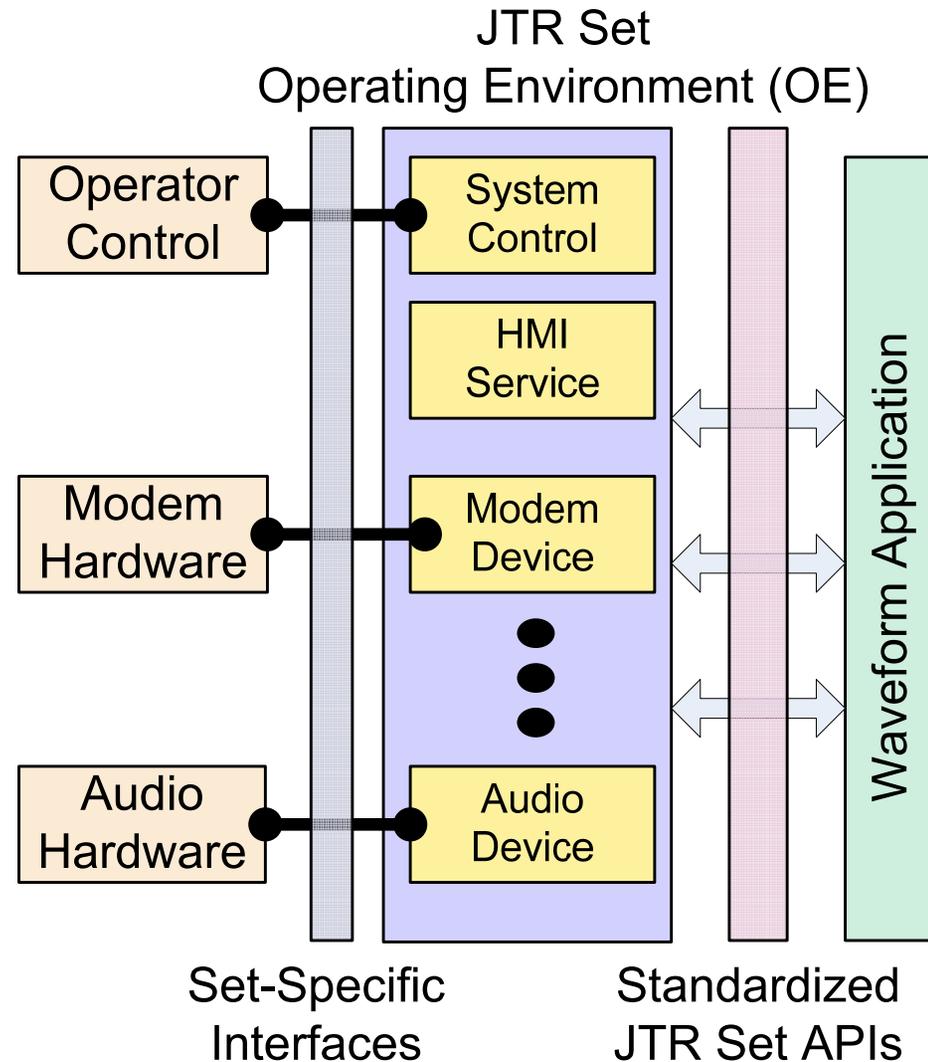
Complex APIs





Scope of JTRS APIs

- JTRS APIs are primarily defined from a waveform/application perspective.
- Intent is to enable portability of waveforms and applications, not the radio operating environment (OE).
- Interfaces which do not touch applications are permitted to be set-specific.
- Goal is to allow the radio set to be developed without restrictions on transport and hardware design, but facilitate reuse of waveforms and applications.

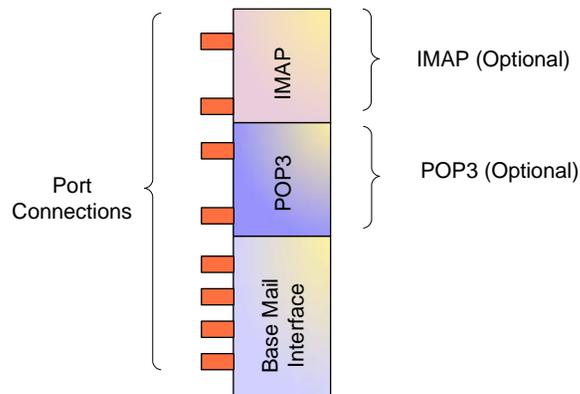




Format of JTRS APIs

- Goal is to enable portability and reusability across a family of radios differing in size, mission, and capabilities.
- APIs have a base or mandatory interface for a radio service or device.
- Optional or extended capabilities are defined in aggregated API extensions.
- A radio is permitted to any or none of the extensions as required to support a waveform/application.

Example of API with Base and Extension



Joint Tactical Radio System (JTRS) Standards Application Program Interface (API)



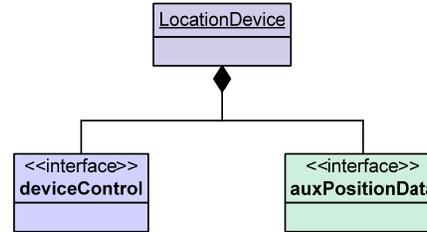
- A. Base API.....**
- A.1 Introduction
 - A.2 Services
 - A.3 Service Primitives and Attributes
 - A.4 IDL
 - A.5 UML
 - Appendix A.A Abbreviations and Acronyms
 - Appendix A.B Performance Specification
- B. Extension**
- B.1 Introduction
 - B.2 Services
 - B.3 Service Primitives and Attributes
 - B.4 IDL
 - B.5 UML
 - Appendix B.A Abbreviations and Acronyms
 - Appendix B.B Performance Specification



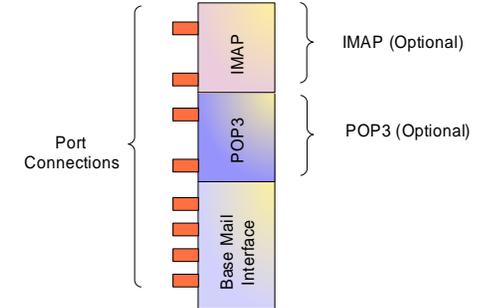
Design Patterns and Strategies of JTRS APIs

- The JTRS community developed several design patterns and strategies for the APIs.
- First consideration is separation of radio operating environment (services and devices) from the waveform applications.
- Second consideration was the strict limitation of visibility/control.
- The other design patterns and strategies support scalability and extensibility, enabling portability and reuse across a family of radios.

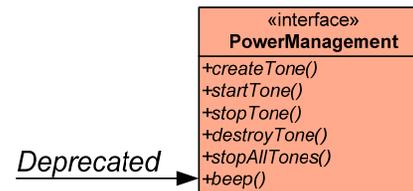
Aggregation



Extensions



Deprecation

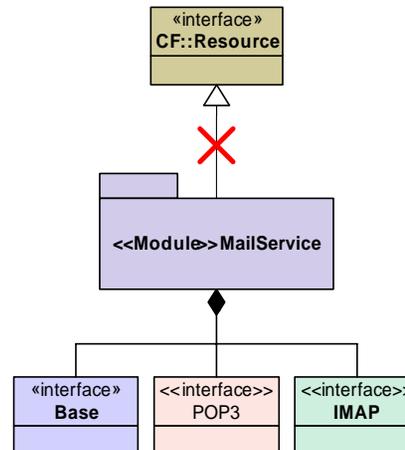


Explicit Enumeration

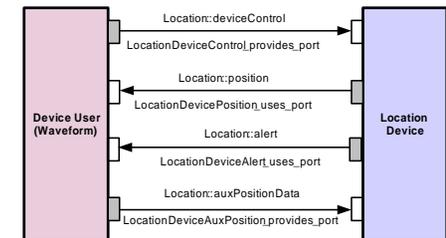
```

module MailService {
  const ExtEnum MAIL_POP = MAIL_BASE + 1;
};
  
```

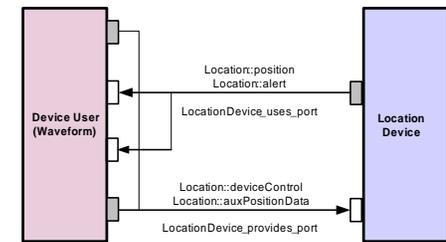
Least Privilege



Port Overloading



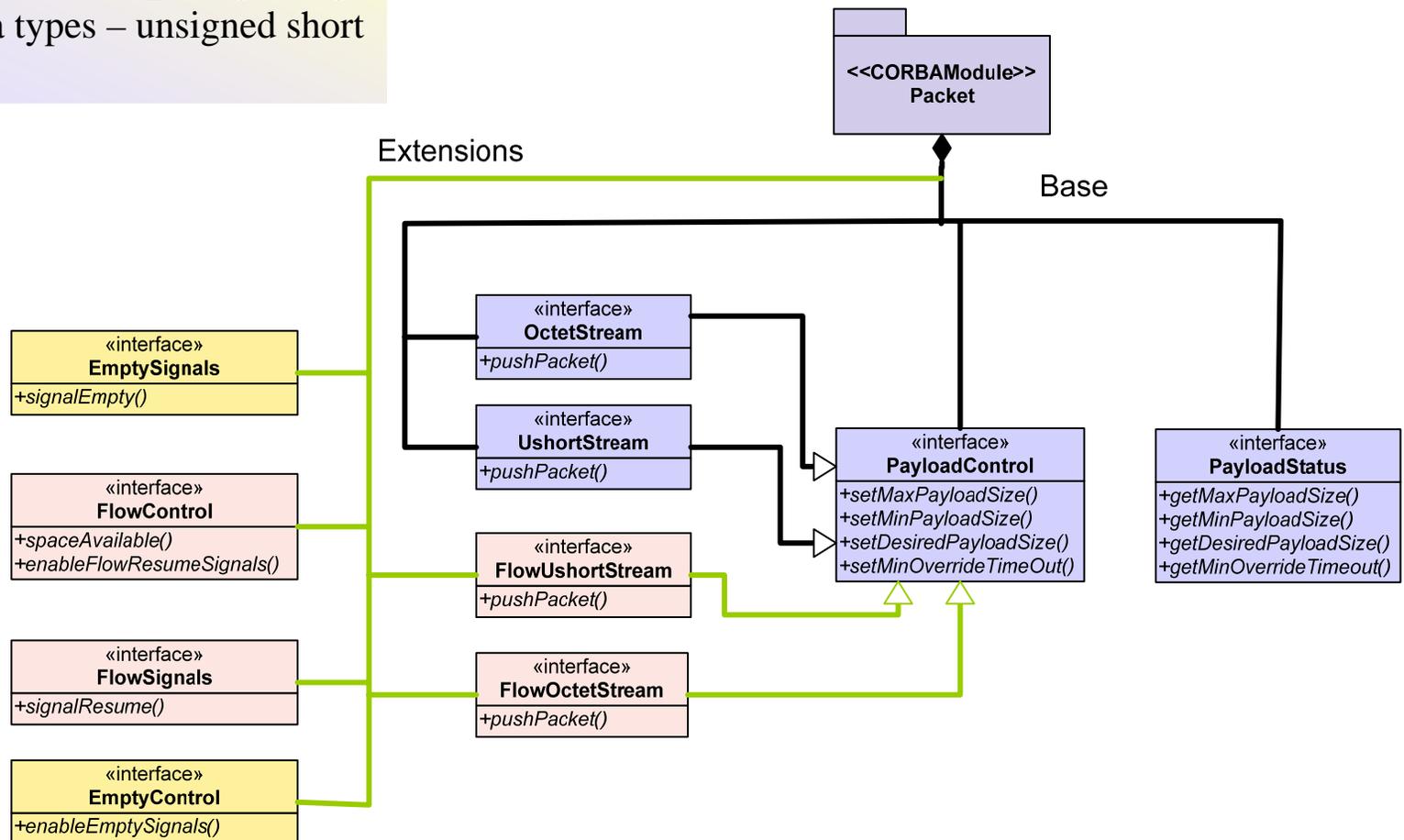
Discrete Port Example





JTRS Packet API

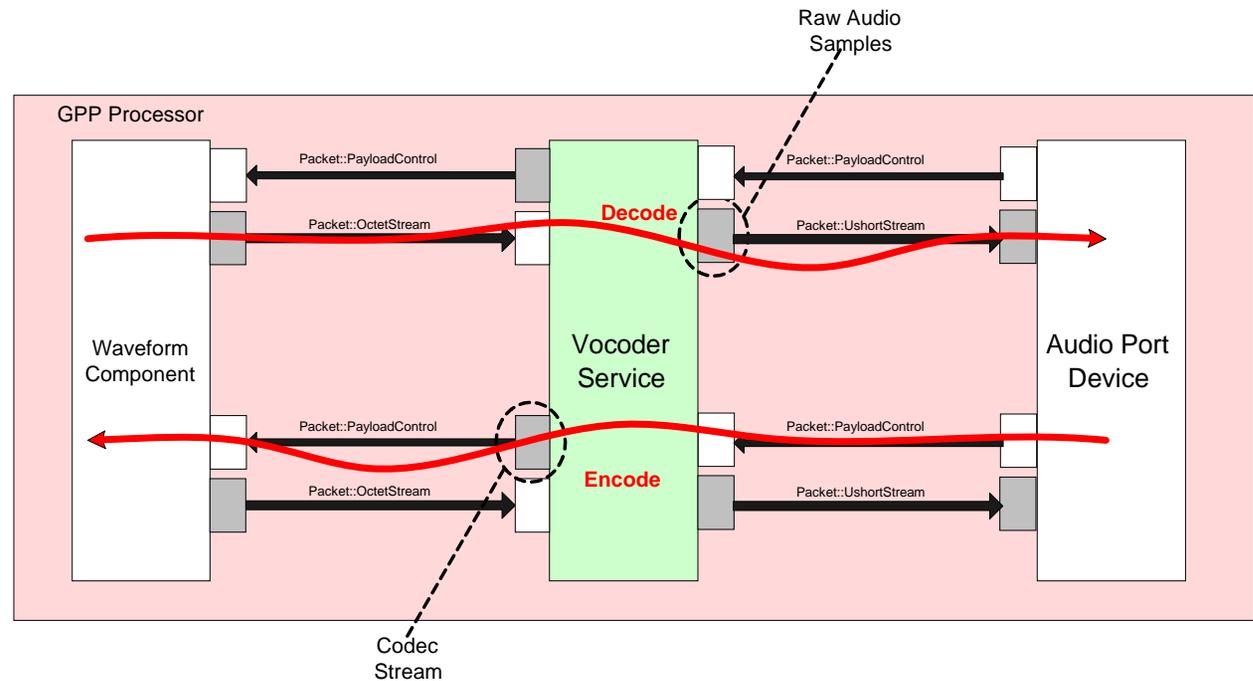
- Packet API is recommended for new API implementations.
- Consists of a base interface with two extensions for flow and empty signaling.
- Supports two data types – unsigned short and octet.





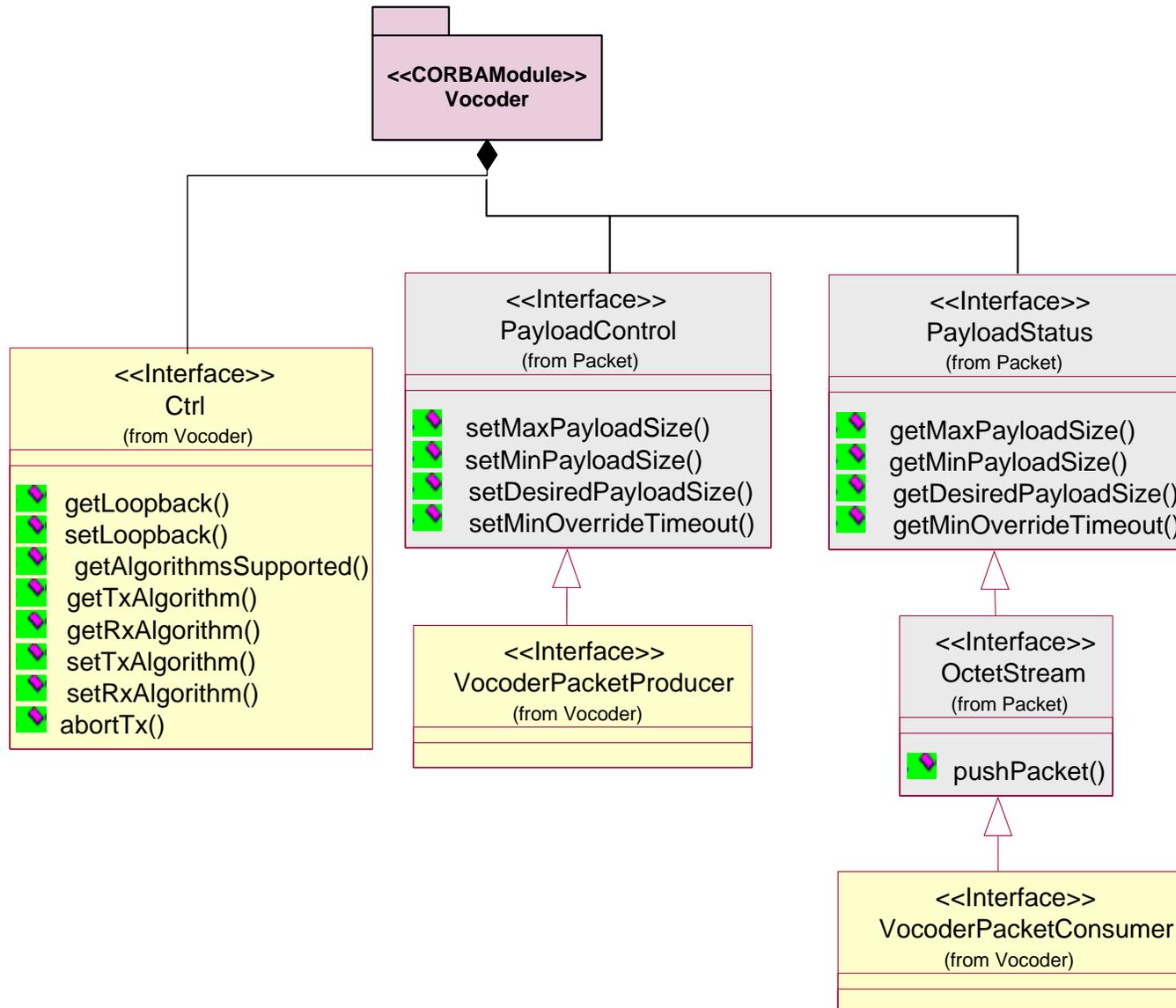
Example API – Vocoder Service

- The Vocoder Service encapsulates vocoding capabilities that are common across all waveforms and applications.
- The Vocoder Service supports loopback operations, provides for the transfer of encoded/decoded bit streams to and from the service user, and defines operations to select algorithms supplied by the vocoder.
- The base Vocoder Service and codec extensions require an implicit or explicit connection with the AudioPortDevice API.



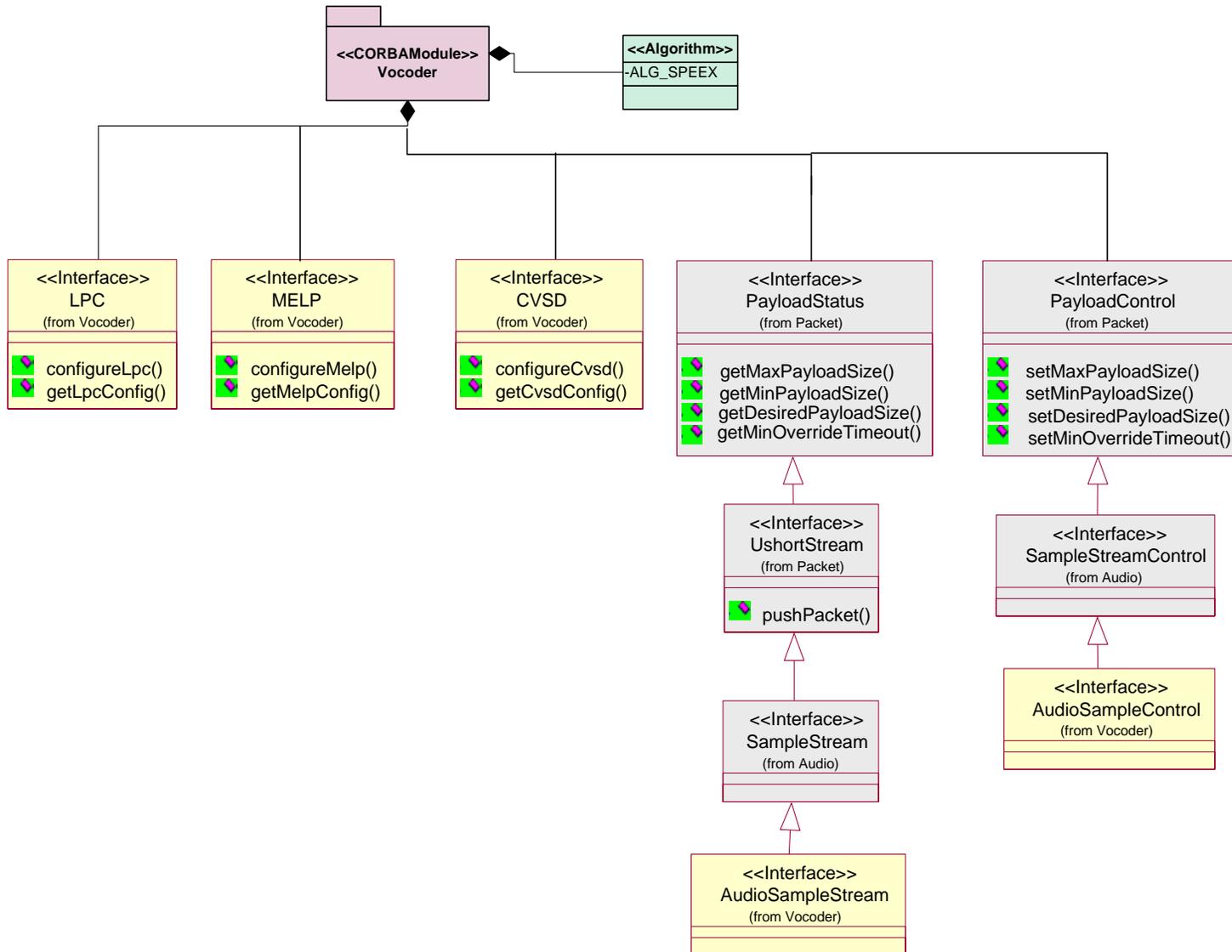


UML Depiction of the Vocoder Service API





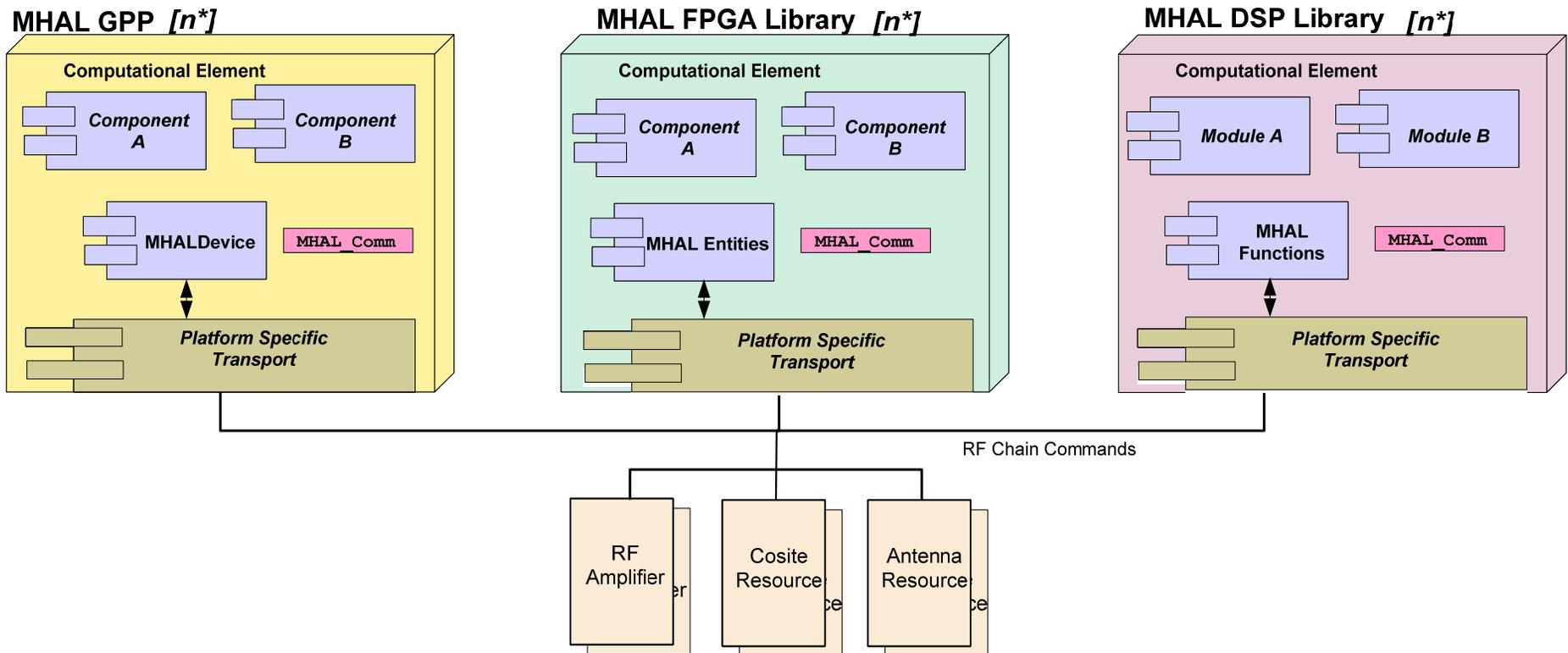
Extensions of Vocoder Service





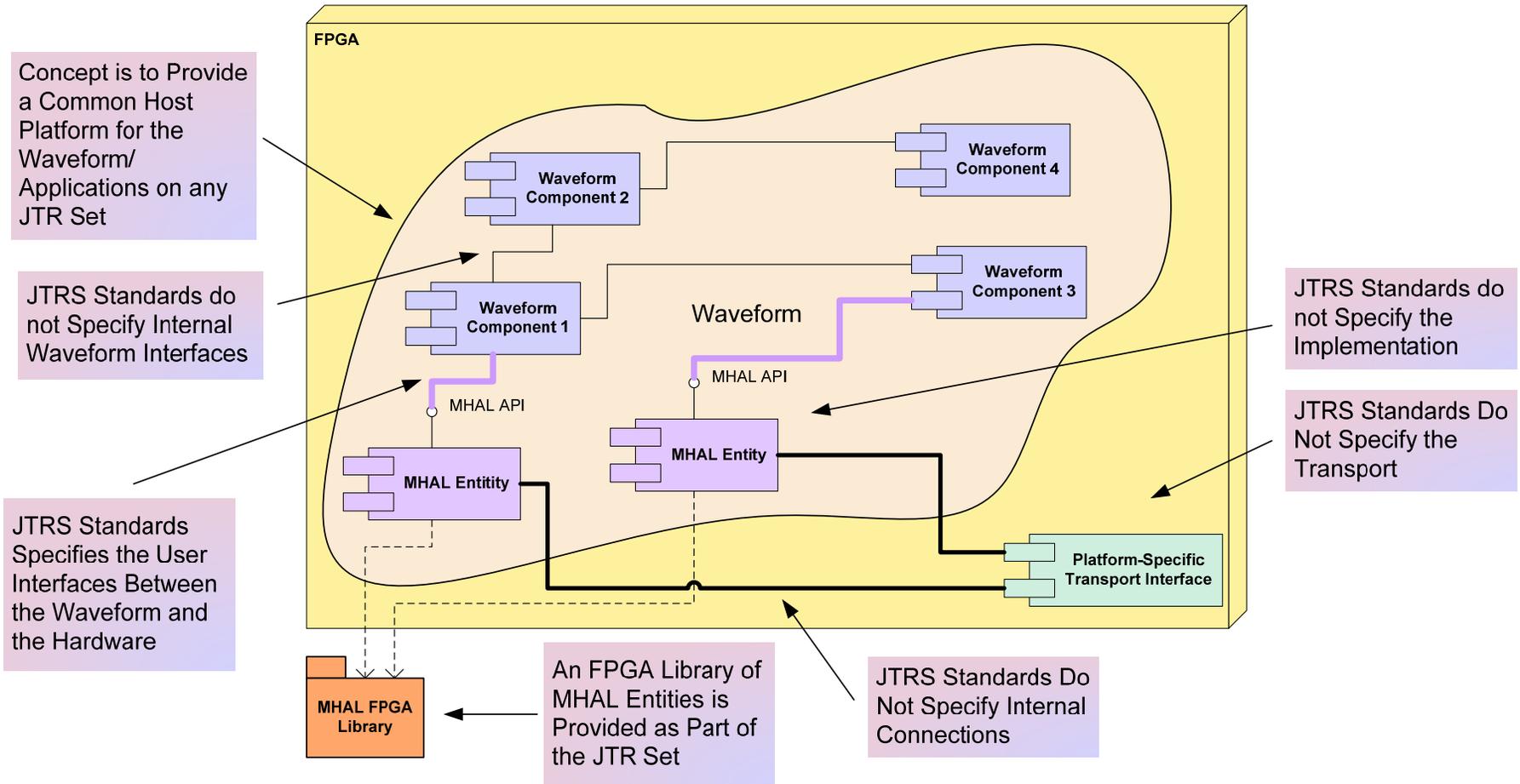
MHAL API

- The MHAL API specifies MHAL Communication Service:
 - Allows one Computational Element (CE) to access any of the other CEs
 - Provides the message transport and an abstract message routing function





MHAL for FPGAs

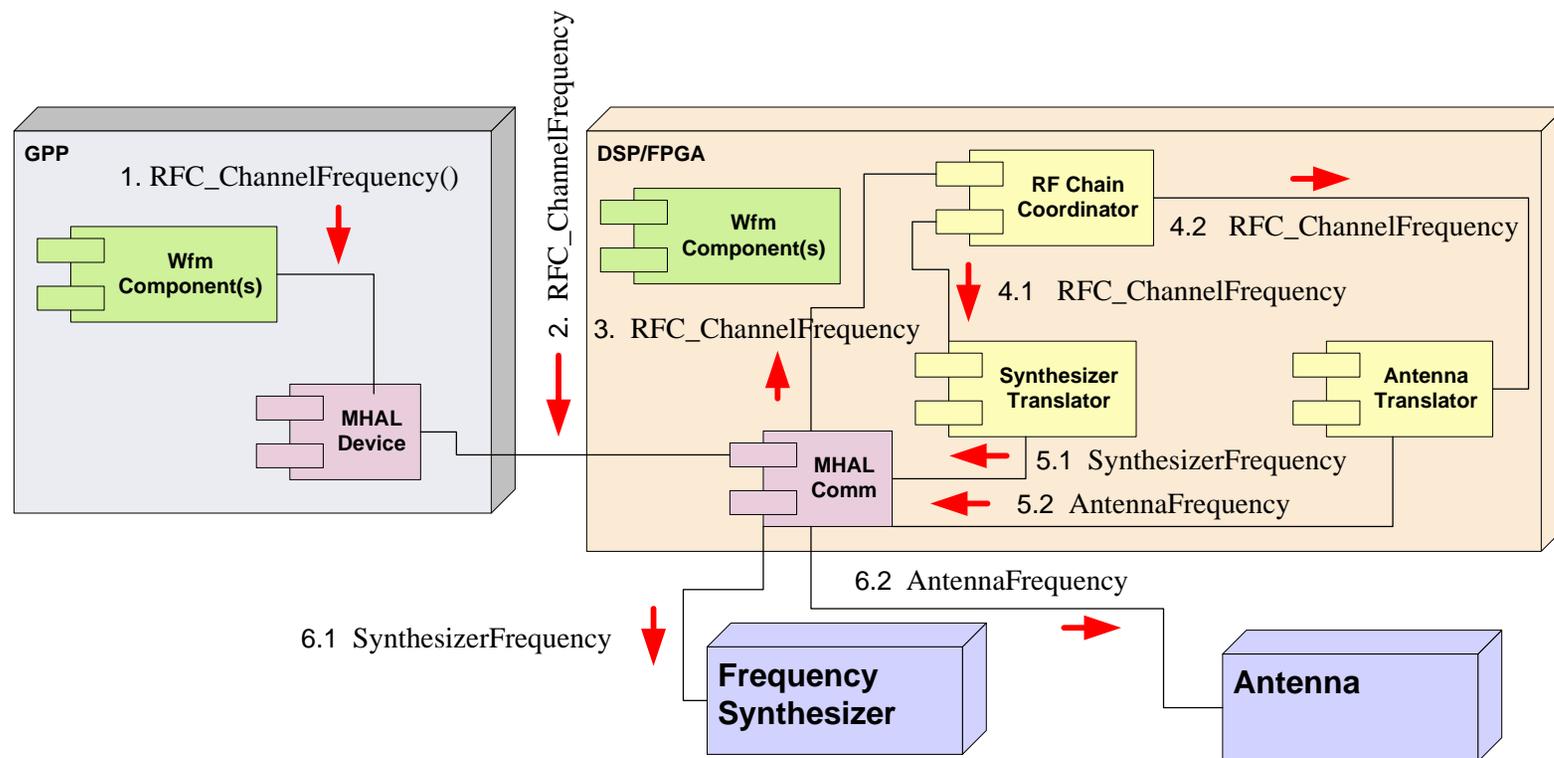


MHAL Defines Connections Between Waveform Components and External Components. Transport is Not Specified.



MHAL RF Chain

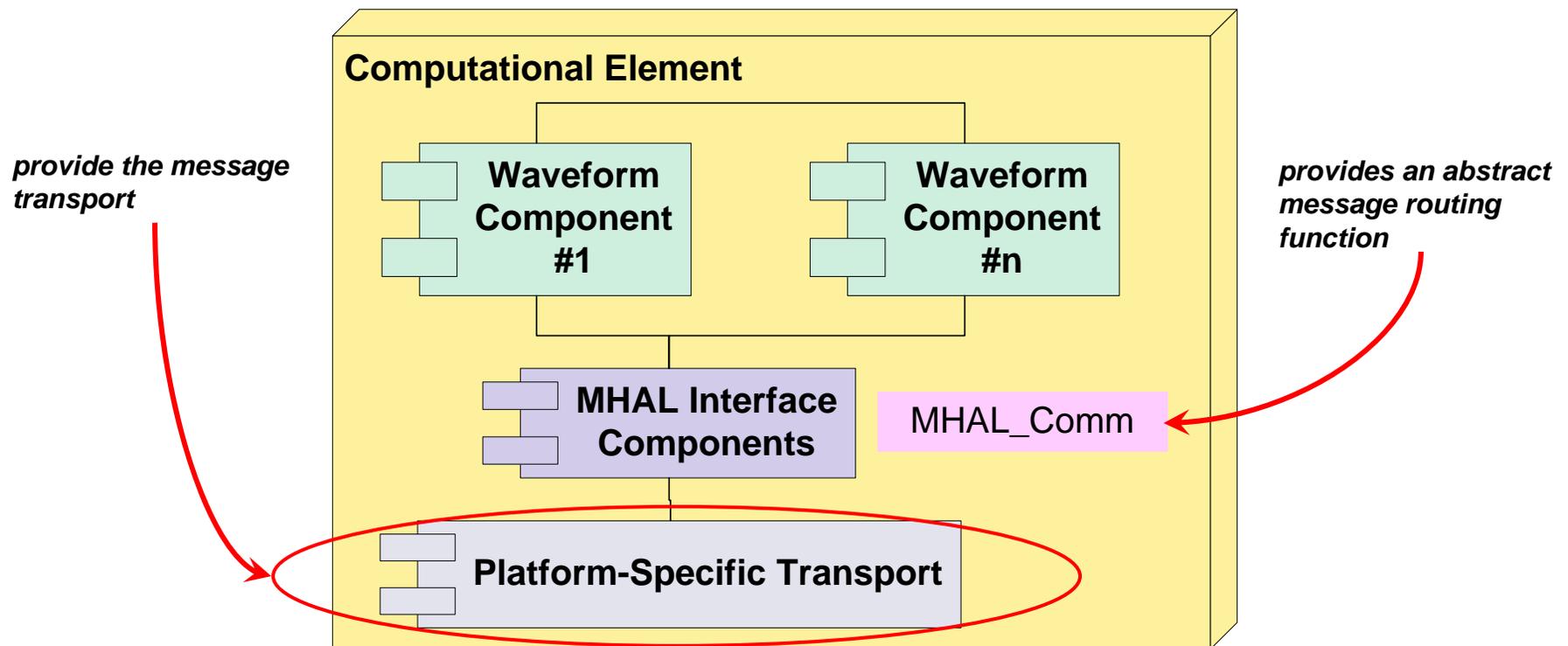
- The MHAL RF Chain Coordinator API consists of a set of sink and source functions that provide for coordinated control of a JTRS Communications Channel's RF resources.
- The Chain Coordinator also abstracts high-level commands into the primitive forms that might be required by the radio hardware.
- RF Chain Coordinator and Translators are optional.





MHAL Communication Service

The MHAL Communication Service is provided by the MHAL Communications Function (MHAL_Comm) working in conjunction with the MHAL Interface Components.



If communicating within the same MHAL CE it is not required to use the MHAL Communications Service.

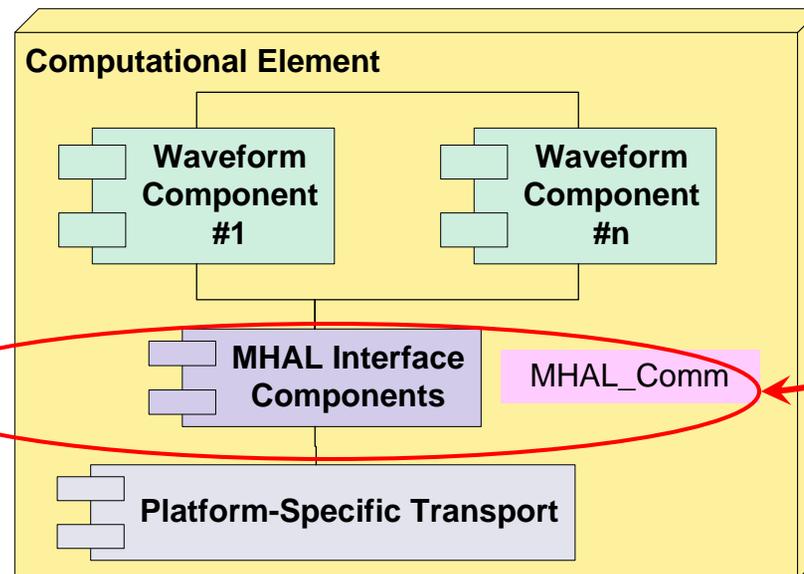


MHAL Communication Service: MHAL Interface Components

All MHAL Interface Components regardless of processor type provide the same system function.

For **GPP or DSP** Processor, these are **SW Drivers** that **manipulate interface HW** provided by the SW's host processor

For **FPGA** Processor, these are **FPGA Interfaces** that include a **mechanization of the desired physical interface**



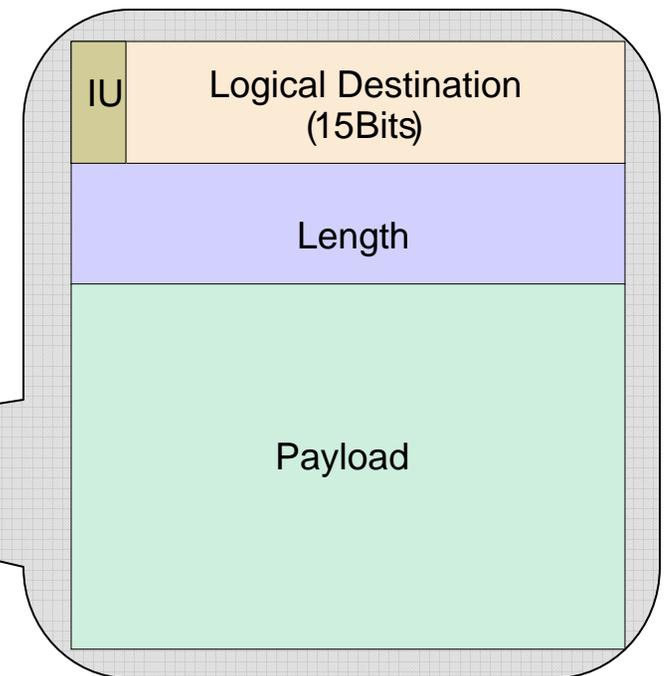
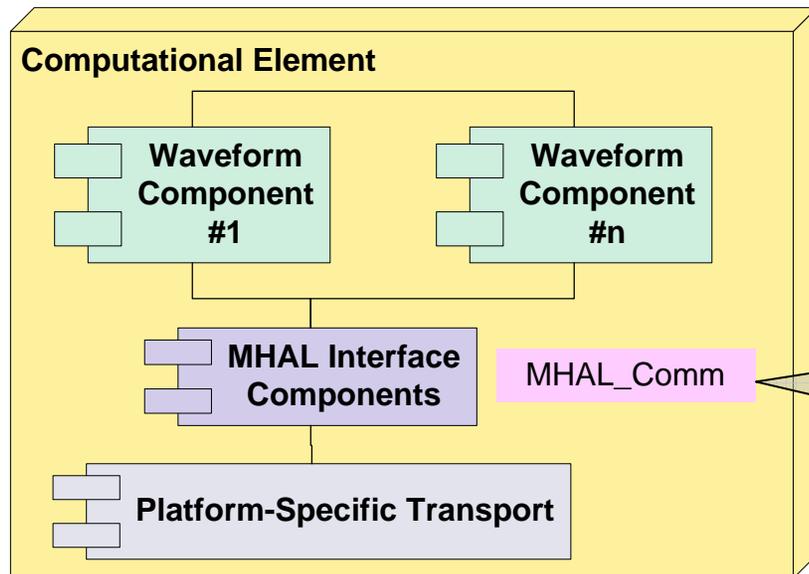


MHAL Communication Service: MHAL Message Structure

The MHAL API specifies 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 Message Structure





MHAL API Extensions

The MHAL API Extensions specify each CE's MHAL Interface Component.

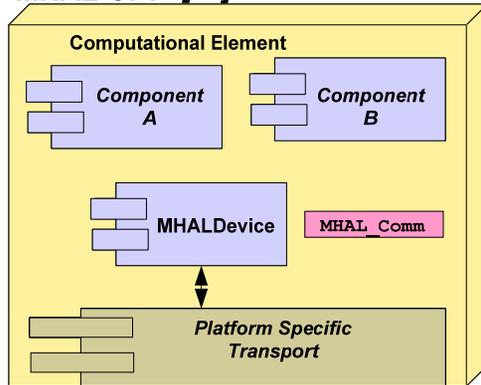
- A GPP represents a CORBA capable processor (This could be a DSP that supports CORBA.)
- A DSP represents a C capable processor, but does not provide CORBA capability.
- An FPGA represents a HDL capable processor, again without CORBA capability.

The MHAL GPP is the **CORBA-based SCA CF::Device** interface

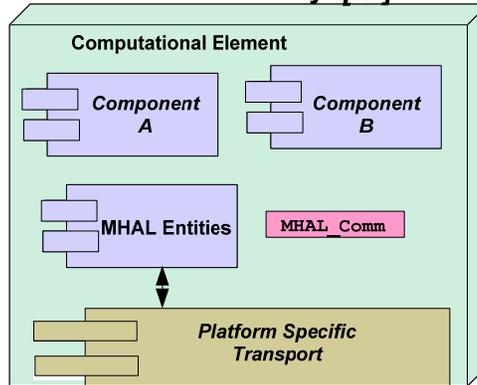
The MHAL FPGA consists of an **FPGA entity library** that is linked into a waveform build

The MHAL DSP is a **library of standardized components** that are linked into a waveform at build

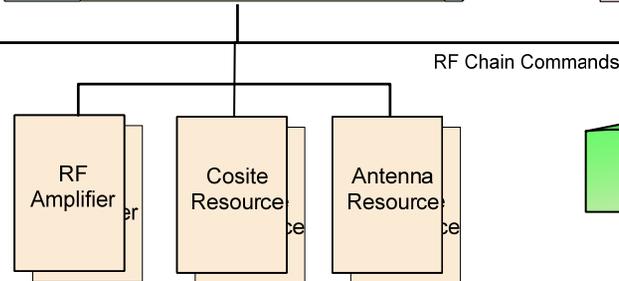
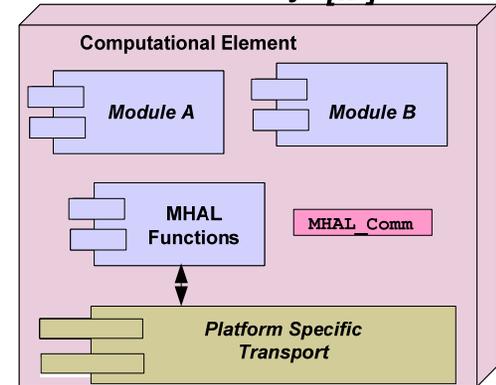
MHAL GPP [n*]



MHAL FPGA Library [n*]



MHAL DSP Library [n*]



The MHAL RF Chain is an abstraction for RF device control.