

SCA Training for Developers and Testers

Day 3: Developing SCA Compliant Waveforms / Applications and Device/DeviceManager



Copyright © 2002, Raytheon Company.
All Rights Reserved

Day 3 AGENDA

- Waveform Design
 - UML Overview
 - Service Definition
 - Waveform Design Process
- *Device* and *DeviceManager* Design
 - *Device* Implementations
 - *DeviceManager* Implementations
 - *Device* Usage and Design Examples

Waveform Design

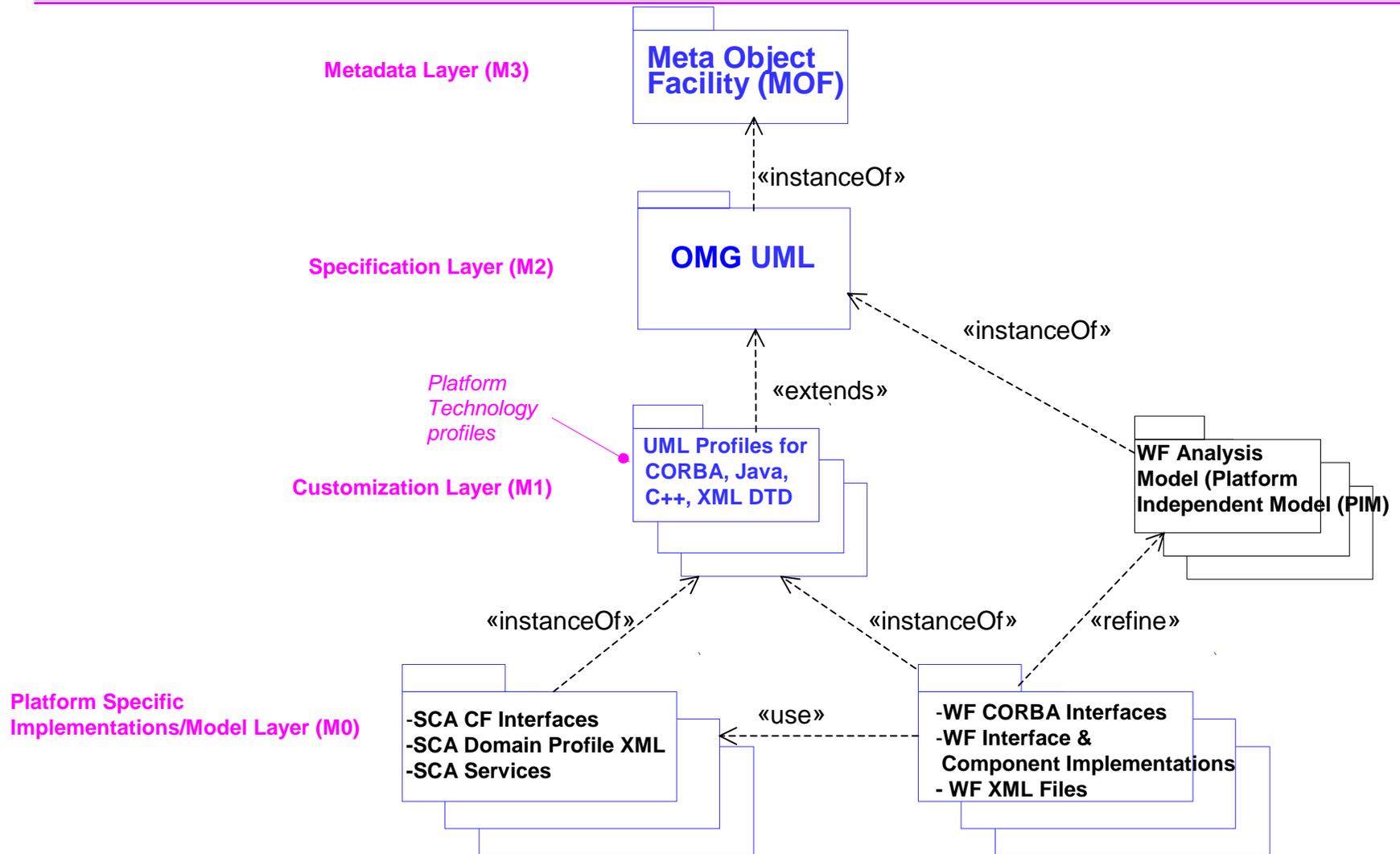
Waveform Design

- UML Overview
- Service Definition
- Waveform Design Process

Object Management Group (OMG) Unified Modeling Language (UML)

- The UML is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.
- Adopted by OMG in 1997
- Current version is UML 1.4
- UML 2.0 Revised Submissions Are Ongoing
 - Infrastructure
 - Superstructure
 - Object Constraint Language (OCL)
 - Diagram Interchange

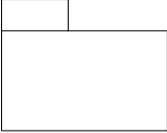
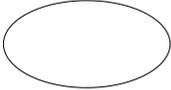
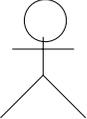
Metamodel Architecture



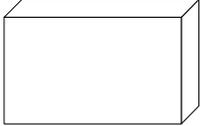
UML Building Blocks

- The basic building blocks of UML are:
 - model elements (packages, classes, interfaces, components, use cases, nodes, etc.)
 - relationships (associations, generalization, dependencies, etc.)
 - diagrams (class diagrams, use case diagrams, interaction diagrams, component diagrams, deployment diagrams, etc.)

UML Core Model Elements

Construct	Description	Syntax
class	a description of a set of objects that share the same attributes, operations, methods, relationships and semantics.	
interface	a named set of operations that characterize the behavior of an element. Is a specification without implementation.	
package	Are general-purpose hierarchical organizational units of the UML model.	
use case	Is a coherent unit of externally visible functionality provided by a system unit.	
actor	Is a type of class, stereotype "actor", that is an idealization of an external person, process, or thing interacting with a system, subsystem, or class.	
object	Is an instance of a class	

UML Core Model Elements, cont'd

Construct	Description	Syntax
constraint	A semantic condition or restriction	
component	a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces.	
node	a run-time physical object that represents a computational resource.	

UML Relationships

Construct	Description	Syntax
association	A relationship between two or more classifiers that involves connections among their instances.	
aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	
composition	A stronger form of association in which the composite has sole responsibility for managing its parts.	
generalization	Relates general descriptions of parent classifiers (superclasses) to more specialized child classifiers (subclasses), which is used for inheritance. The subclass for example adds to the superclass definition or overrides the superclass definition.	
dependency	A relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element). Types of dependencies are: trace, refinement, realization, binding, etc.	

UML Diagrams

- Class Diagram – a static graphic presentation that shows a collection of declarative model elements, such as, classes, and their contents and relationships.
- StateChart Diagram – a diagram that depicts a State Machine consisting of simple states, transitions, and nested composite states.
- Activity Diagram – a diagram that depicts an activity graph, which is a special case of a state machine, where all or most of the states are activity states or action states. Transitions are triggered by the completion of activity in the source states.
- Use Case Diagram – a diagram that depicts the relationships among actors and use cases within a system.

- Interaction Diagram – a diagram that describes sequences of messages exchanges among roles that implement behavior of a system.
 - Collaboration Diagram – a diagram that depicts interactions organized around roles
 - Sequence Diagram – a diagram that depicts object interactions arranged in time sequence.
- Component Diagram – a diagram that depicts the organization and dependencies among component types.
- Deployment Diagram – a diagram that depicts the configuration of run-time processing nodes and the software units (e.g., processes, component instances) that execute on them.

Waveform Design

- UML Overview
- Service Definition
- Waveform Design Process

Service Definition

-
- Documented By a Service Definition Description (API Supplement Appendix A) which describes the contract between the Service Provider and the Service User
 - Service States
 - Relationships to other Services
 - Services' Description
 - Synopsis
 - Parameters
 - Valid States for the Service
 - Allowable State Transitions
 - Response
 - Originator
 - Errors/Exceptions
 - Allowable Sequence of Service Primitives.
 - Precedence of Service Primitives.
 - Service User Guidelines.
 - UML
 - IDL



Service Definition & Creation Guidelines in Precedence Order

- Use existing API.
- Create a new API by inheriting an existing API and then extending its services.
- Translate an existing non-JTRS API to IDL to create a new JTRS API.
- Develop a new API based upon one or more Building Blocks. Use of Building Blocks should follow the order of using existing Building Blocks, extending existing Building Blocks, generating new Building Blocks.

Service Definition Naming Conventions

- Class, Interface, Package & Types names
 - Starts with an uppercase letter
 - Every word that composes the name starts with an uppercase letter
 - Words are run together (e.g., *DomainManager*)
- Methods and attributes
 - Starts with a lowercase letter
 - Every word that composes the name starts with an uppercase letter (except the first one)
 - Words are run together (e.g., *getApplications*)
- Constants & Enumeration Literals
 - Every letter of a word is an uppercase letter.
 - Multiple words are separated by an underline

Waveform Design

- UML Overview
- Service Definition
- Waveform Design Process



Waveform Design Process

- Build Waveform Analysis Model
- Build Waveform Language Interfaces
- Build Waveform Component Implementations
- Integrate Waveform Components

Waveform Design Process

- Build Waveform Analysis Model
 - Build Waveform UML Analysis Model
 - Build Waveform Simulation Model
- Build Waveform Language Interfaces
- Build Waveform Component Implementations
- Integrate Waveform Components

Build Waveform Analysis Model

- Waveform UML Analysis Model – a service definition of the waveform APIs in UML
 - Captures the waveform APIs definition
 - Captures Waveform components definitions
- Simulation Model – a simulation of the execution of an application(s) against the real-world physical constraints of the system.



Build Waveform UML Analysis Model

- Build Waveform UML Analysis Model Activities
- Waveform Analysis UML Model Elements
- Waveform Analysis UML Model Examples



Build Waveform UML Analysis Model Activities

- Identify functionality to be provided by the waveform software { SCA Developer's Guide section 6.1.2 }
- Determine which API Service Groups are needed {SCA Developer's Guide section 6.1.3 }
- Determine what services are needed beyond the API Service Groups {SCA Developer's Guide section 6.1.4 }

Identify Waveform Functionality

- Legacy Waveform Elements
- Use Cases
- Data, Control, and Events to and from a component
- HCI



Determine SCA API Service Groups Needed

- Packet Services
- Networking Services
 - MAC
 - Logical Link Control
- I/O Services
- Physical Services
 - Non-Real Time
 - Real Time

Determine New API Service Groups

- Extend Existing Service Group
 - Adding functionality to existing Service Group
 - Creates a new service within the same Service Group.
 - Example – Cosite Service to Physical Services
- Define a New Service Group
 - Examples – Audio, Cross-banding, Networking, Filters, etc.
- Extensions and New Service Groups can be specific to waveform definition or submitted as a change to the API Supplement.



Build Waveform UML Analysis Model

- Build Waveform UML Analysis Model Activities
- Waveform UML Analysis Model Elements
- Waveform UML Analysis Model Examples

Waveform UML Analysis Model Elements

- UML Concepts Utilized
 - Model Elements
 - Class – Service Component Definition & realization of services, Types, Exceptions
 - Interface – Service Definition
 - Use Case – to define service functionality
 - Actor – identifies the roles who are using the services
 - Constraint – constraints denoted by the service
 - Component – defines the provides and uses ports for a waveform component.
 - Node – where a waveform component can be or is deployed within a system.

Waveform UML Analysis Model Elements, cont'd

- UML Concepts Utilized
 - Diagrams
 - Class Diagram – Service Definition, Waveform Component Class definition
 - Use Case – Captures the requirements for a service definition
 - Activity – Graphical captures the behavior requirements of an operation or use case.
 - StateChart – Captures a service's states and state transitions
 - Interaction - Allowable sequence of service primitives and realization of a use case.
 - Component – a components uses and provides port definitions
 - Relationships
 - Association, Aggregation, Composition, Generalization, and Dependency are use to depict the definition of a service and component.



Build Waveform UML Analysis Model

- Build Waveform UML Analysis Model Activities
- Build Waveform UML Analysis Model Elements
- Build Waveform UML Analysis Model Examples



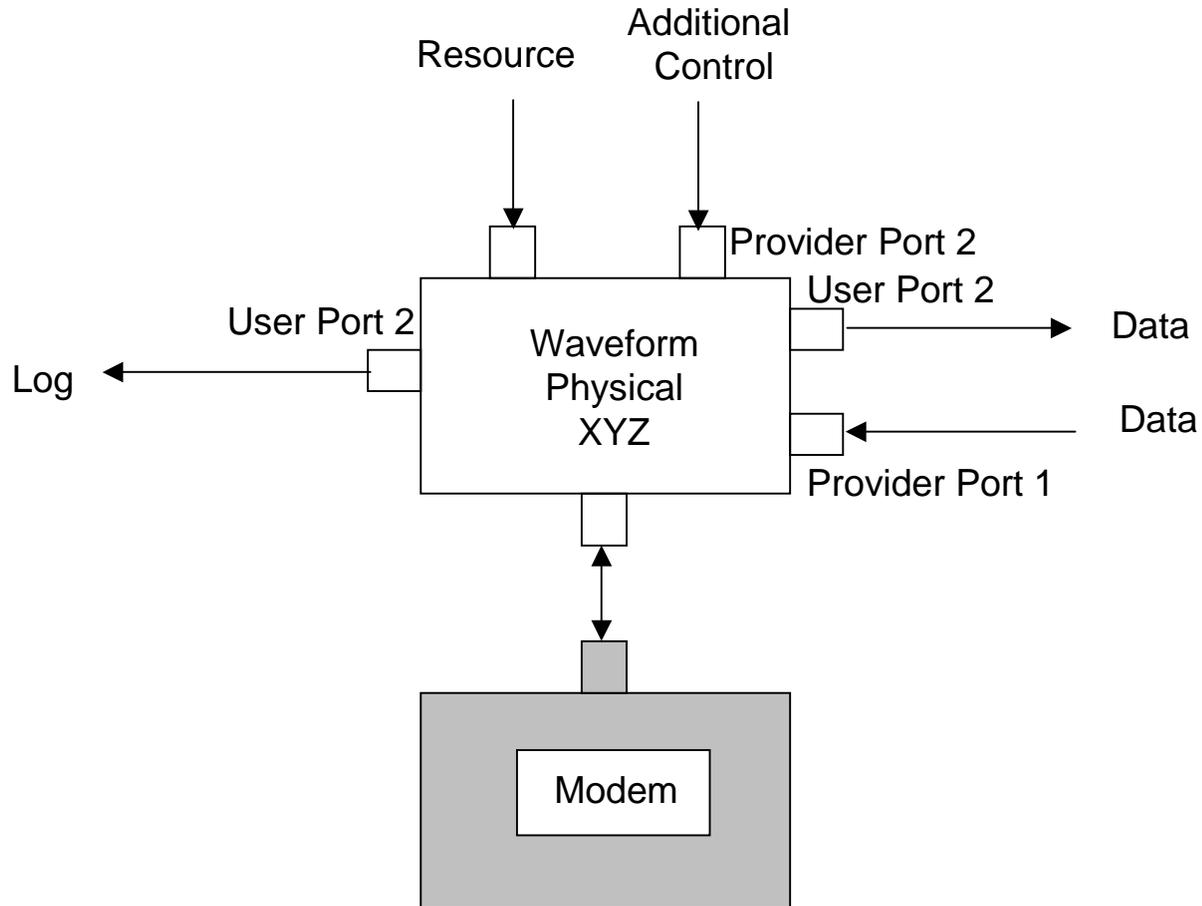
Build Waveform UML Analysis Model Examples

- One-Layer Waveform - Physical
- Two-Layer Waveform - Physical & Media Access Control (MAC)
- Three-Layer Waveform - Physical, MAC & Logical Link Control (LLC)
- XYZ Assembly Controller

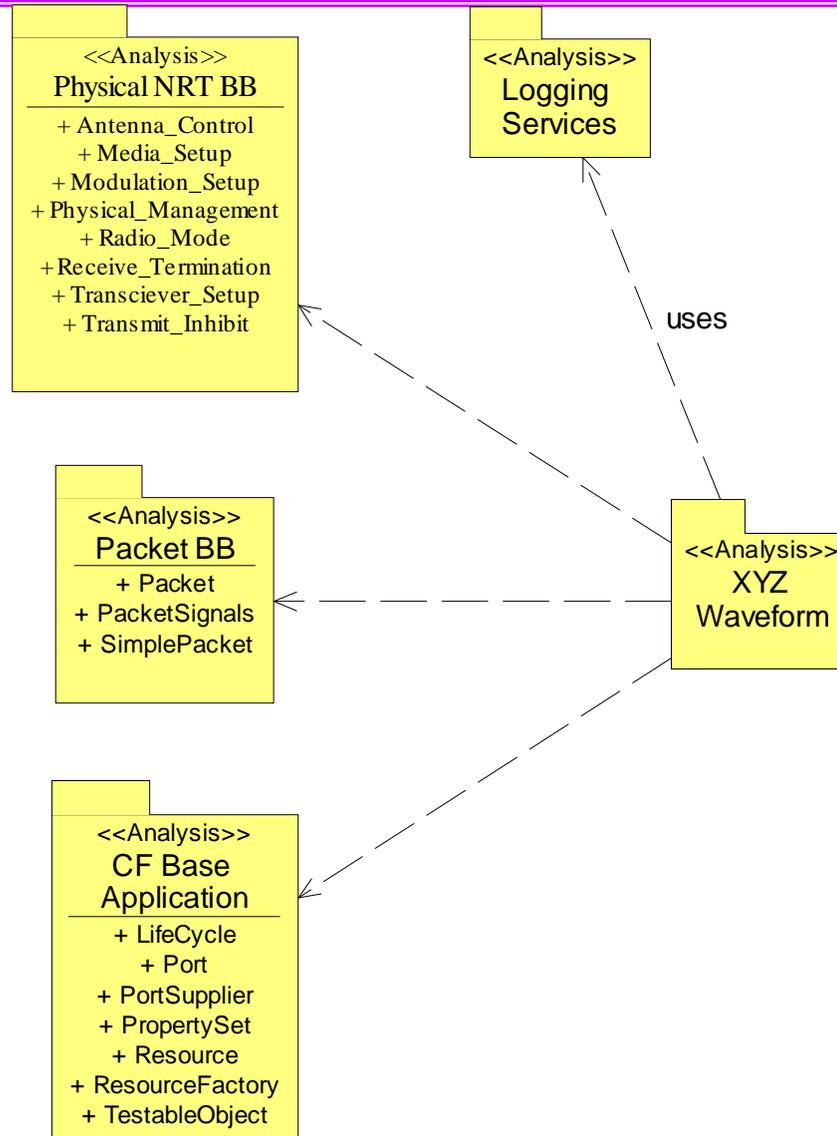
XYZ One-Layer UML Service Examples

- One-Layer Waveform
 - Analysis Packages
 - XYZ Packet Interfaces
 - XYZ Control Physical Interface
 - XYZ Class
 - XYZ Physical Component

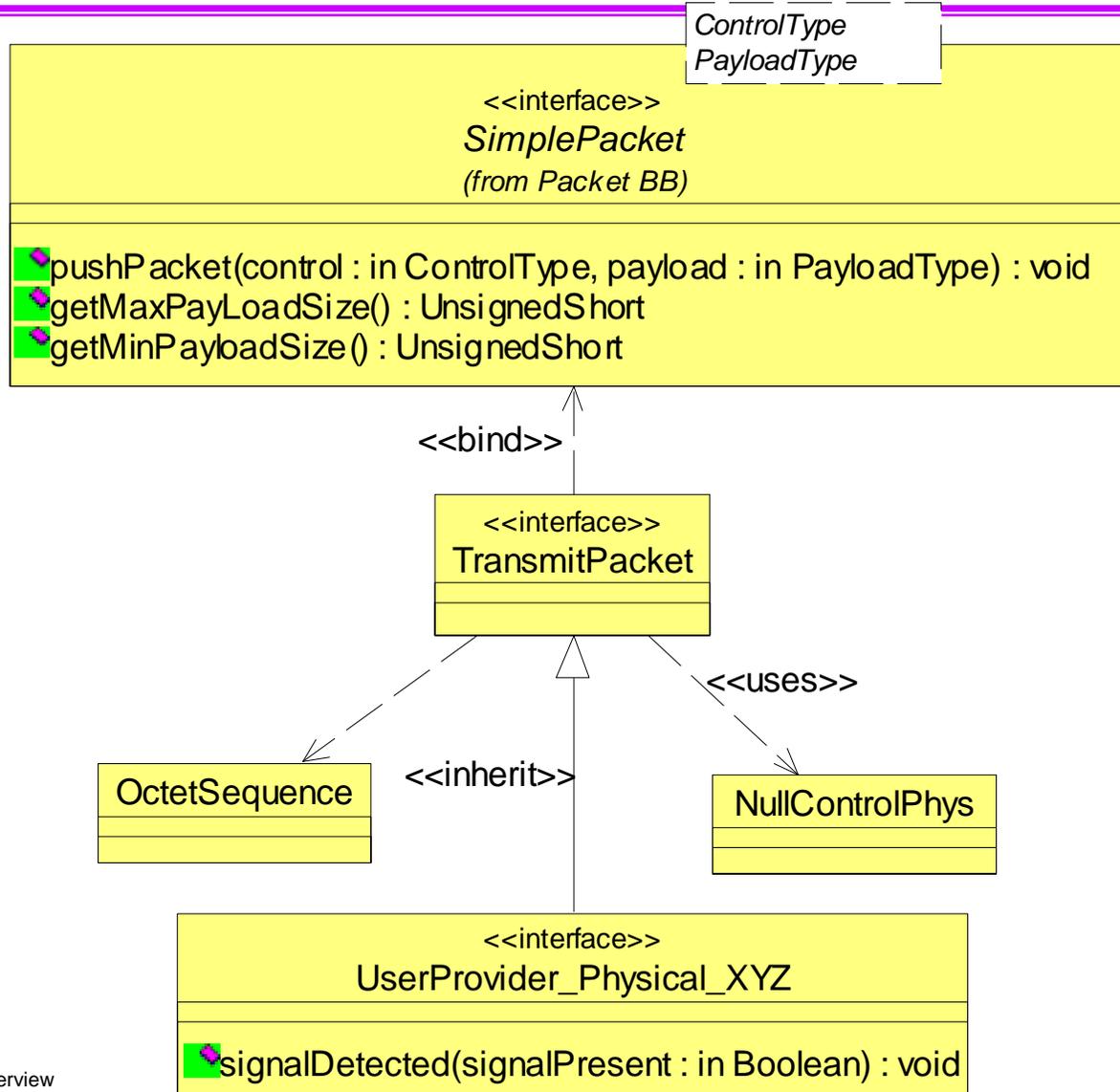
One-Layer Waveform Overview



XYZ One-Layer Waveform Analysis Package Relationships Example

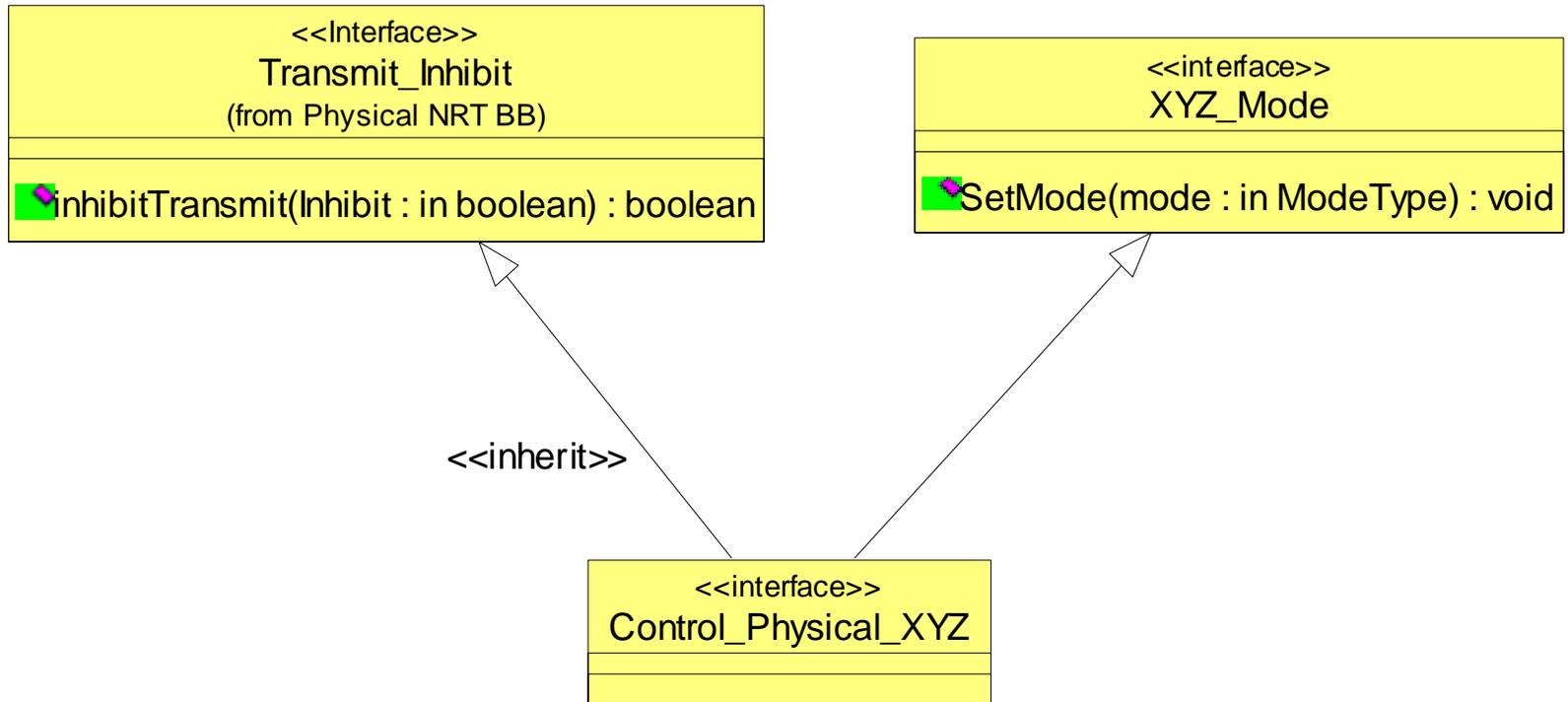


Extending a Service Group – XYZ Packet Interfaces Example

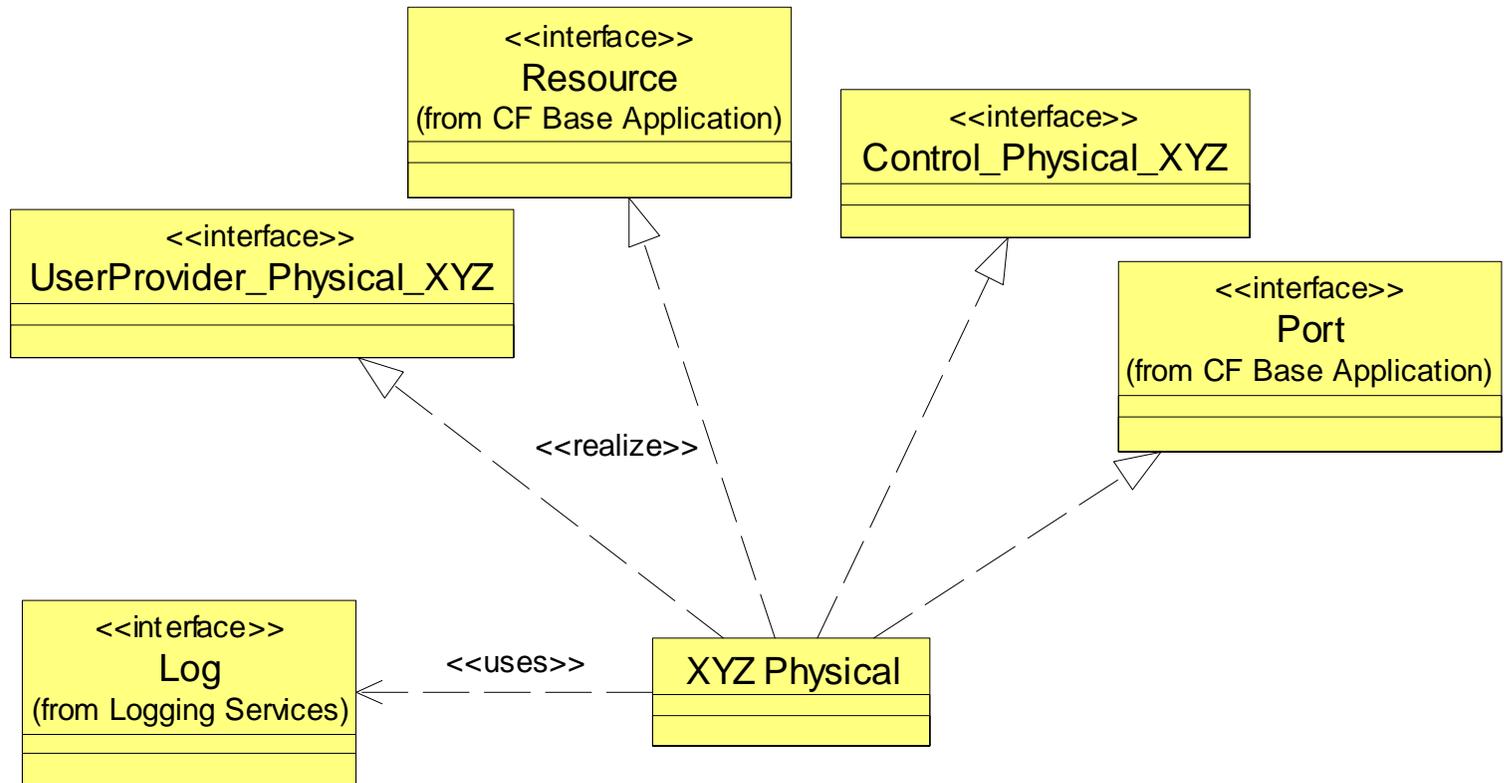


XYZ Control Interface Example

Constructing New & Combining Service Groups



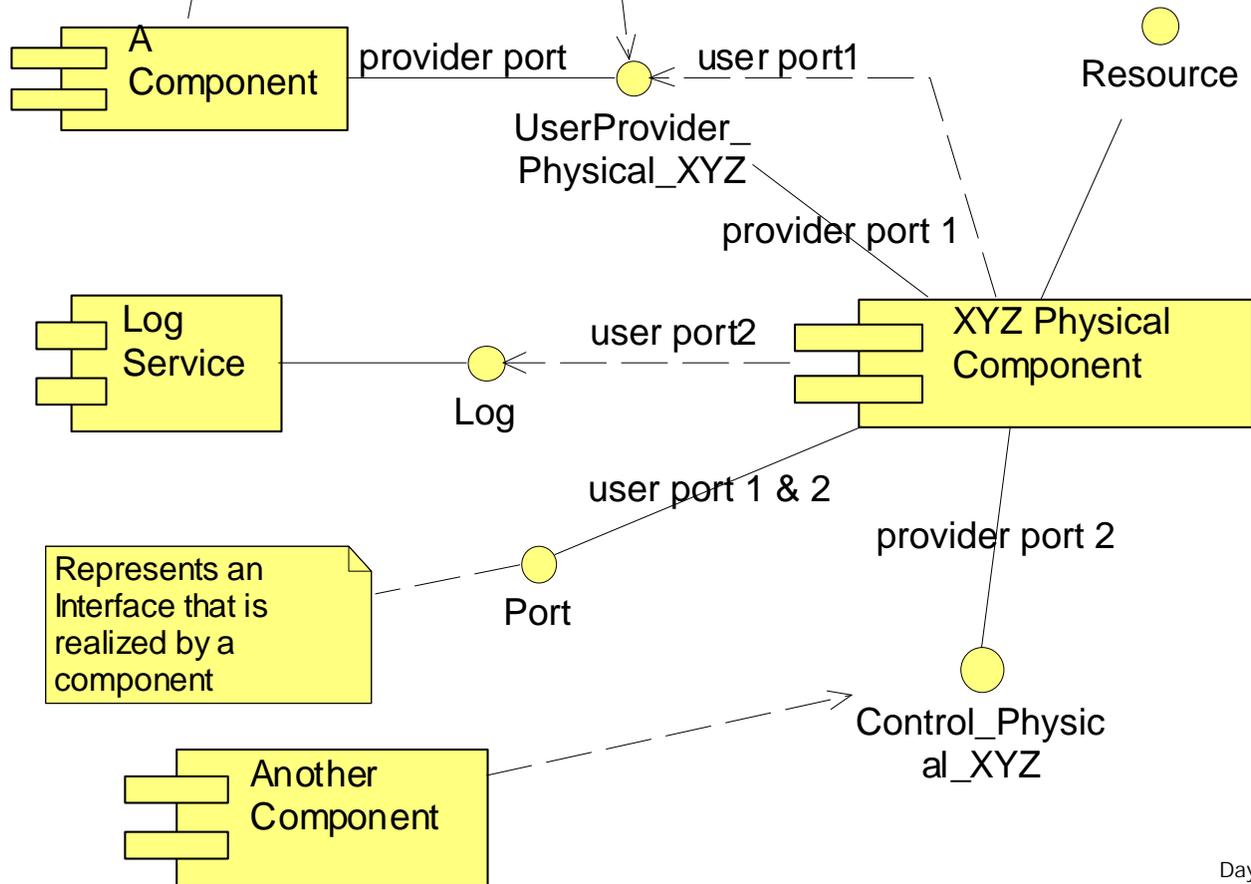
XYZ Physical Class



XYZ Physical Component Ports

User Ports require Provides Port(s) and implement the CF Port Interface

Represents a dependency to an Interface that is realized by another component

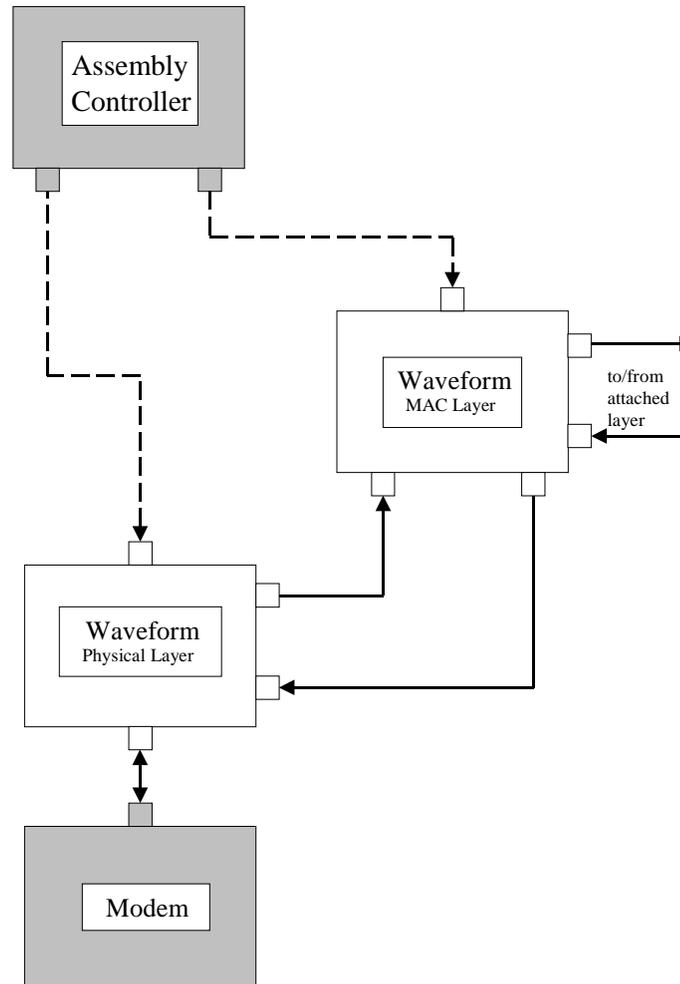




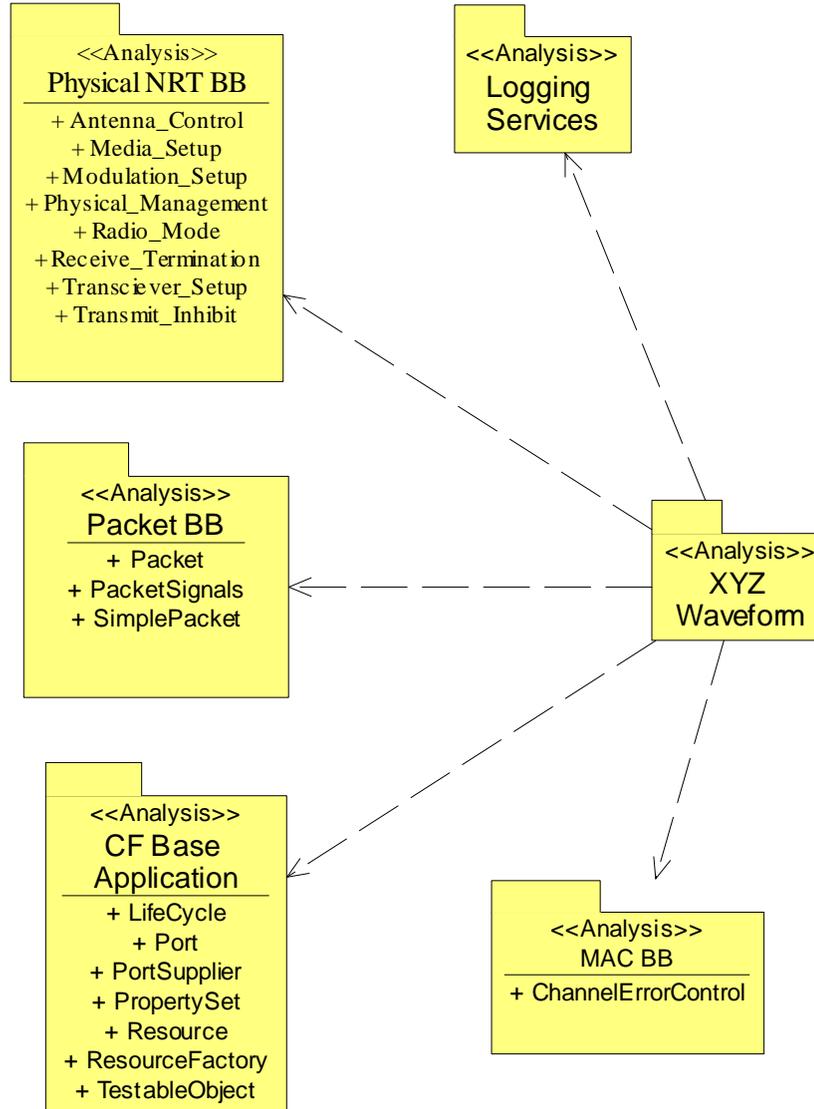
XYZ Two-Layer Waveform UML Service Examples

- Two-Layer Waveform
 - Two-Layer Analysis Packages
 - XYZ MAC Interface
 - XYZ Mac Class
 - XYZ MAC Component

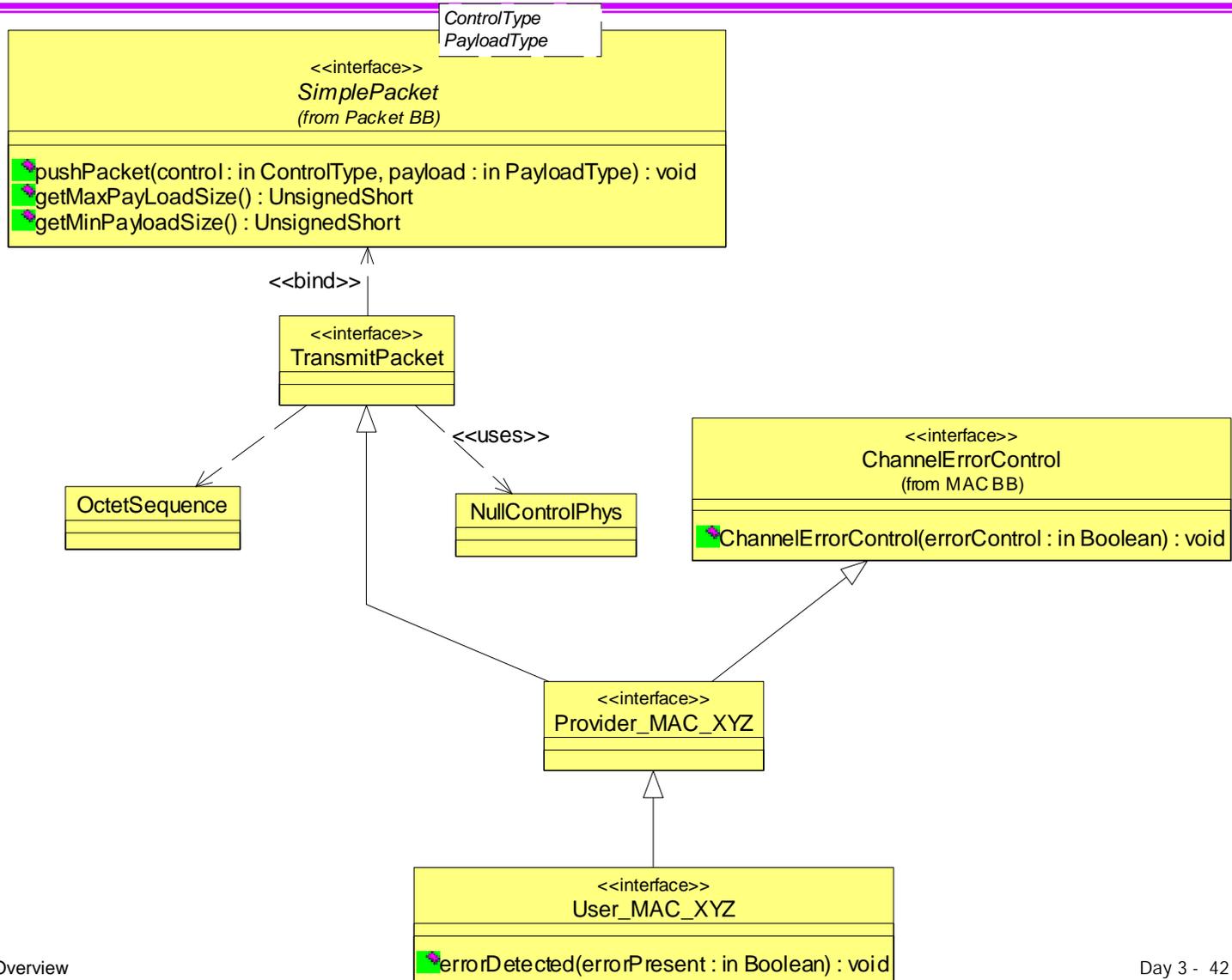
Two-Layer Waveform Example



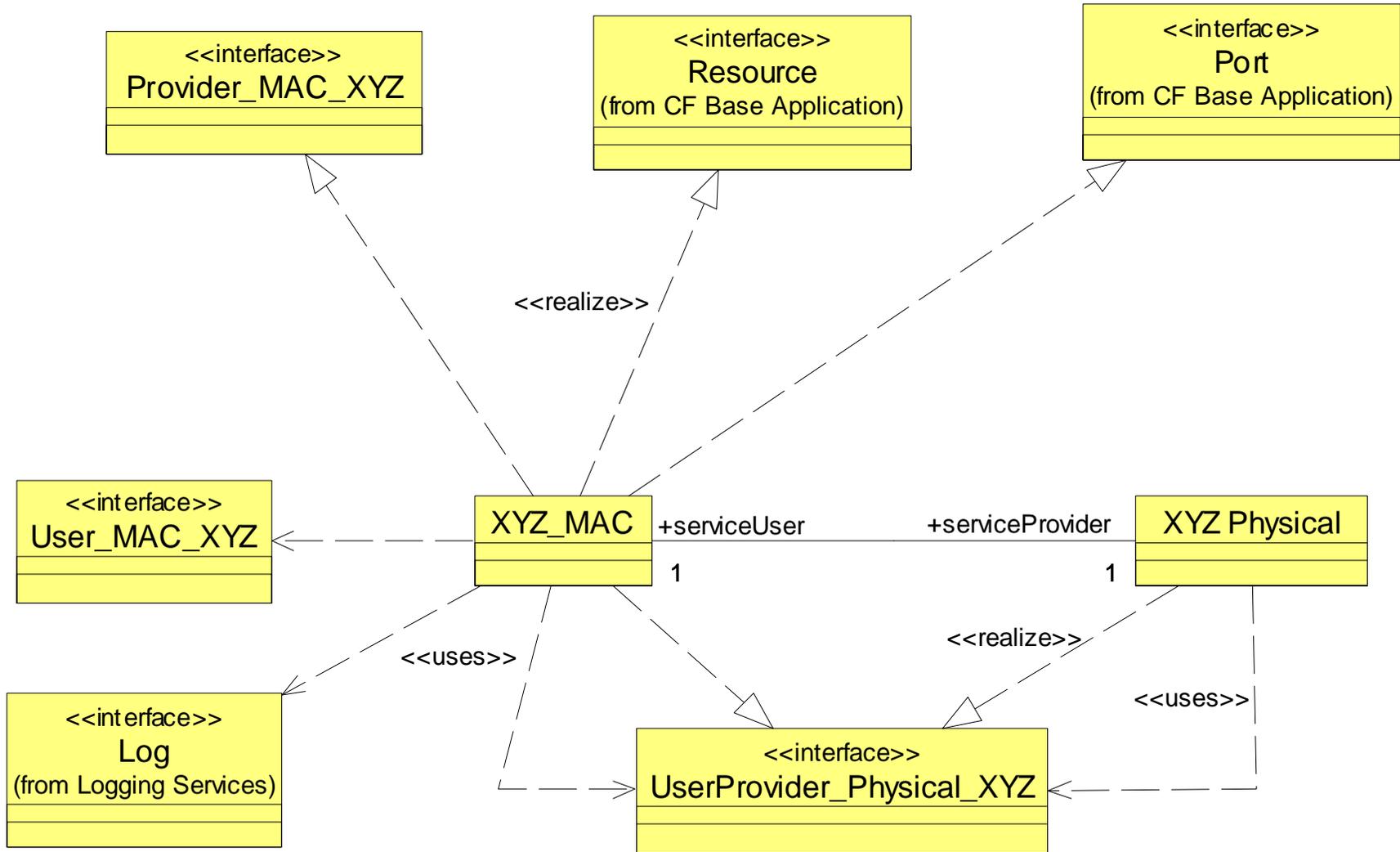
XYZ Two-Layer Waveform Analysis Package Relationships Example



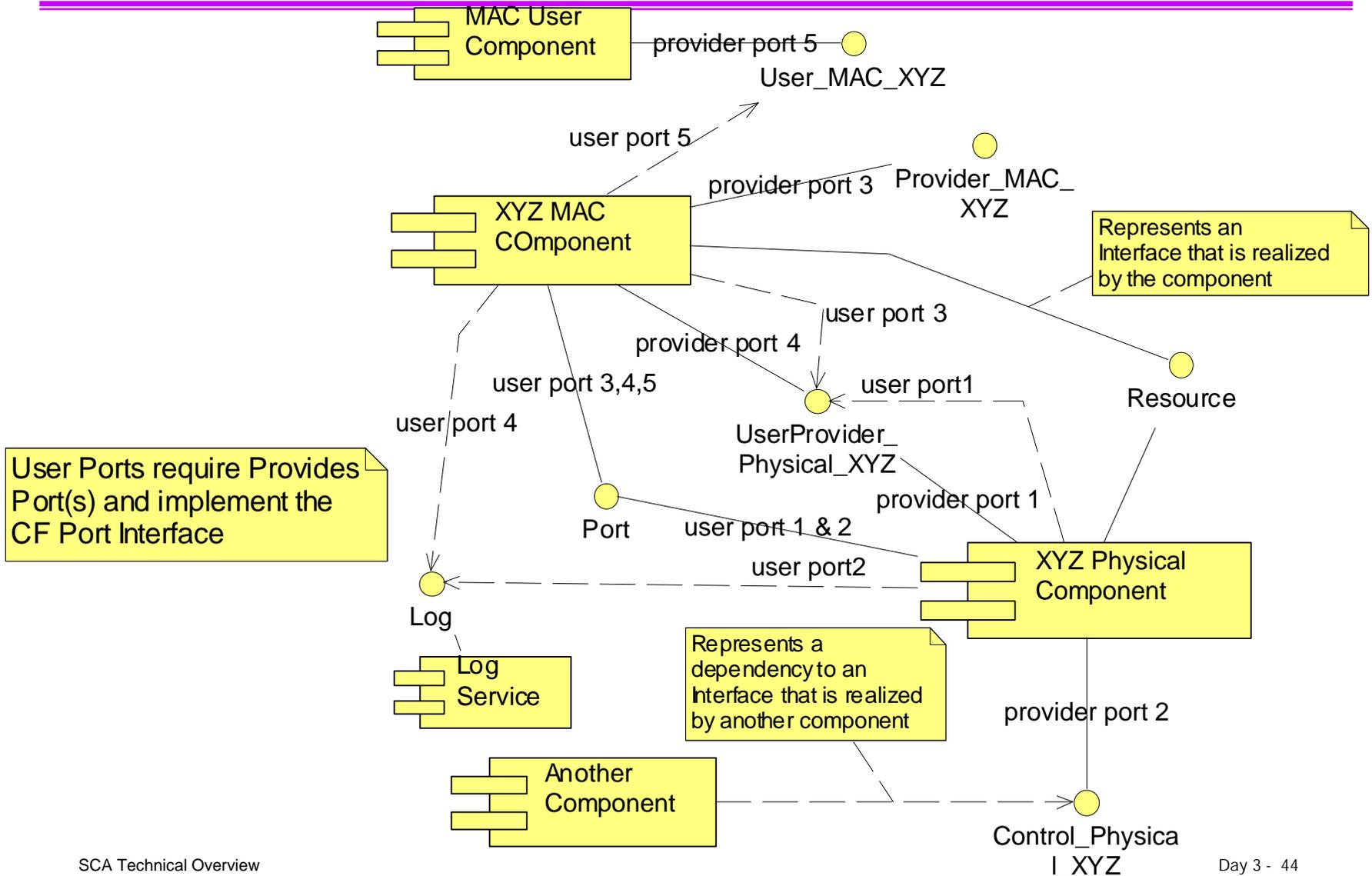
XYZ MAC Interface Example



XYZ MAC Class



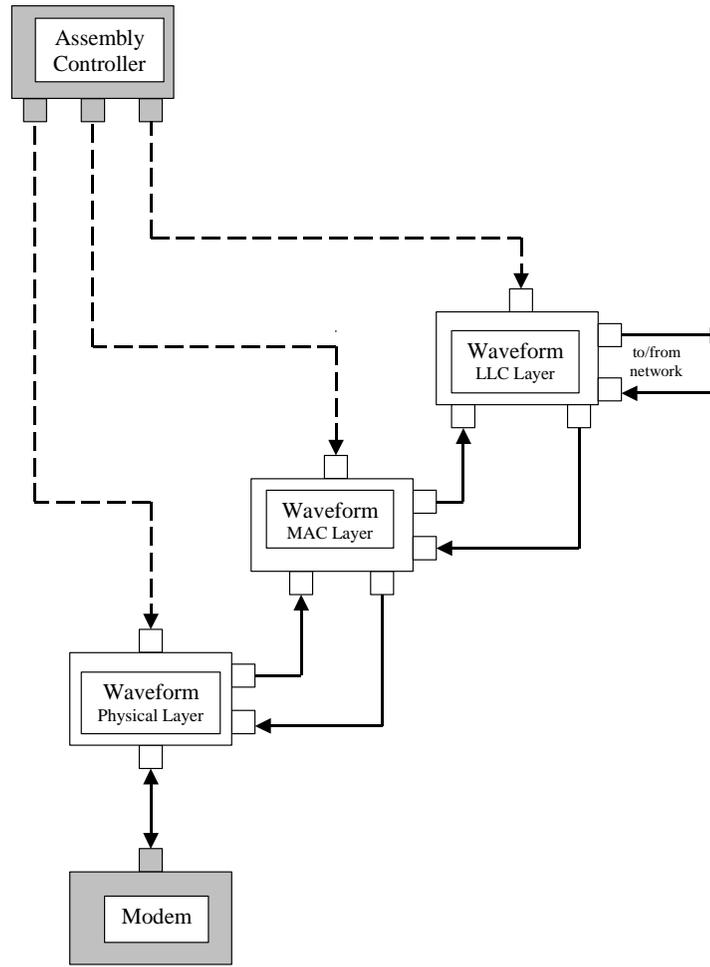
XYZ MAC Component Ports



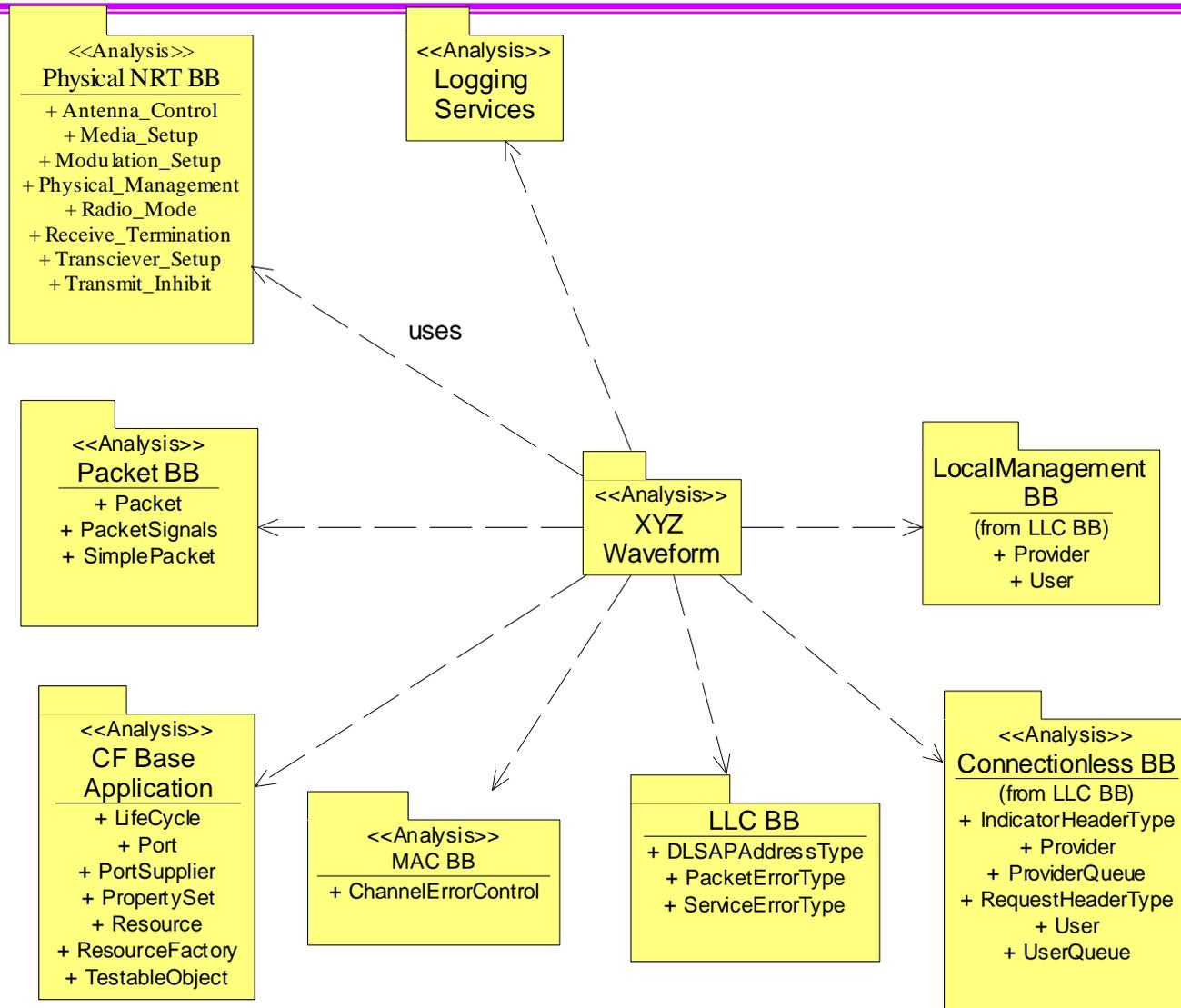
XYZ Three-Layer Waveform UML Service Examples

- Three-Layer Waveform
 - Packages
 - XYZ Logical Link Control Queue Interfaces
 - XYZ Logical Link Control Provider Interface
 - XYZ Logical Link Control User Interface
 - XYZ Logical Link Control Local Management Interfaces
 - XYZ Logical Link Control Class
 - XYZ Logical Link Component

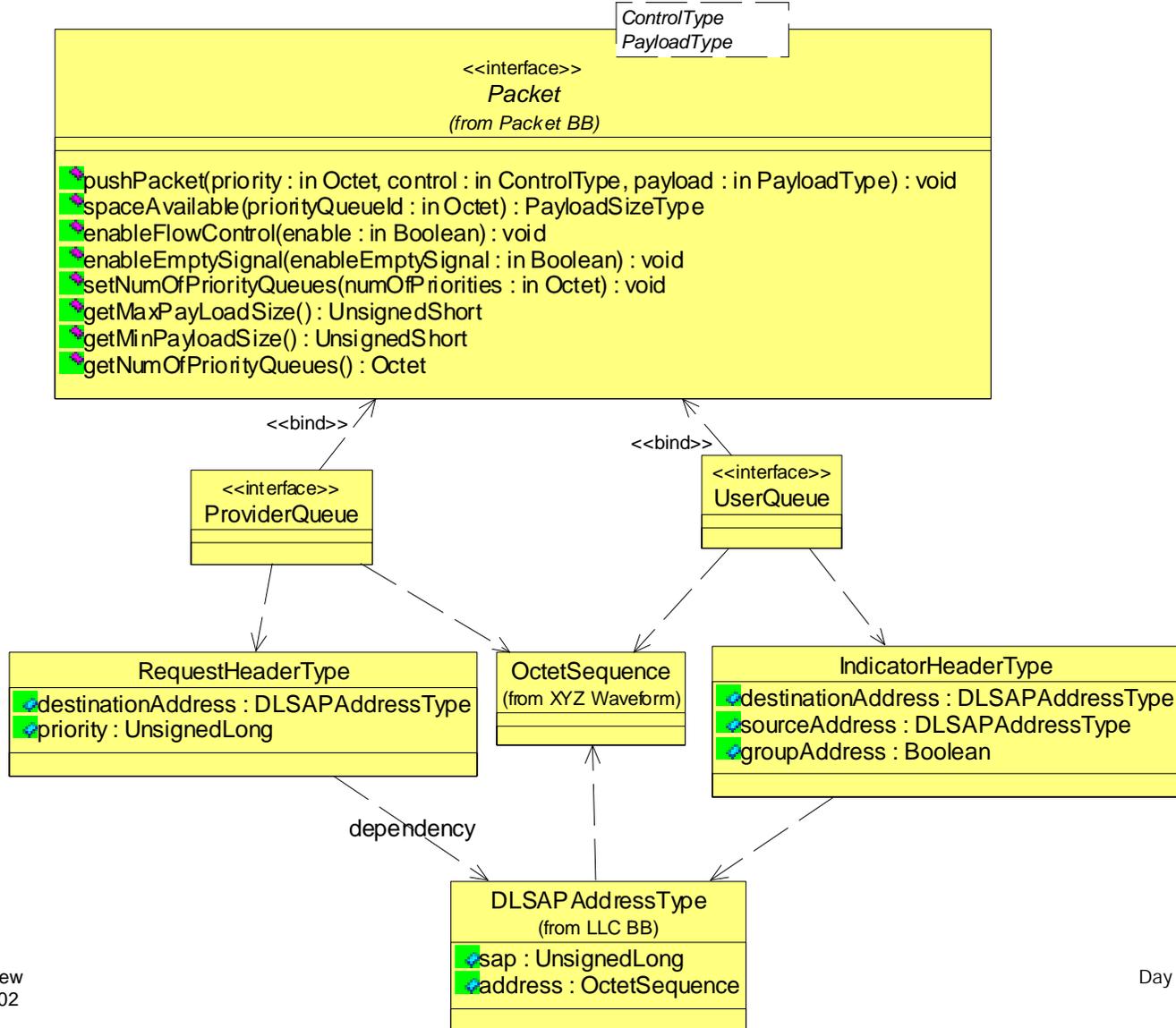
Three Layer Waveform Example



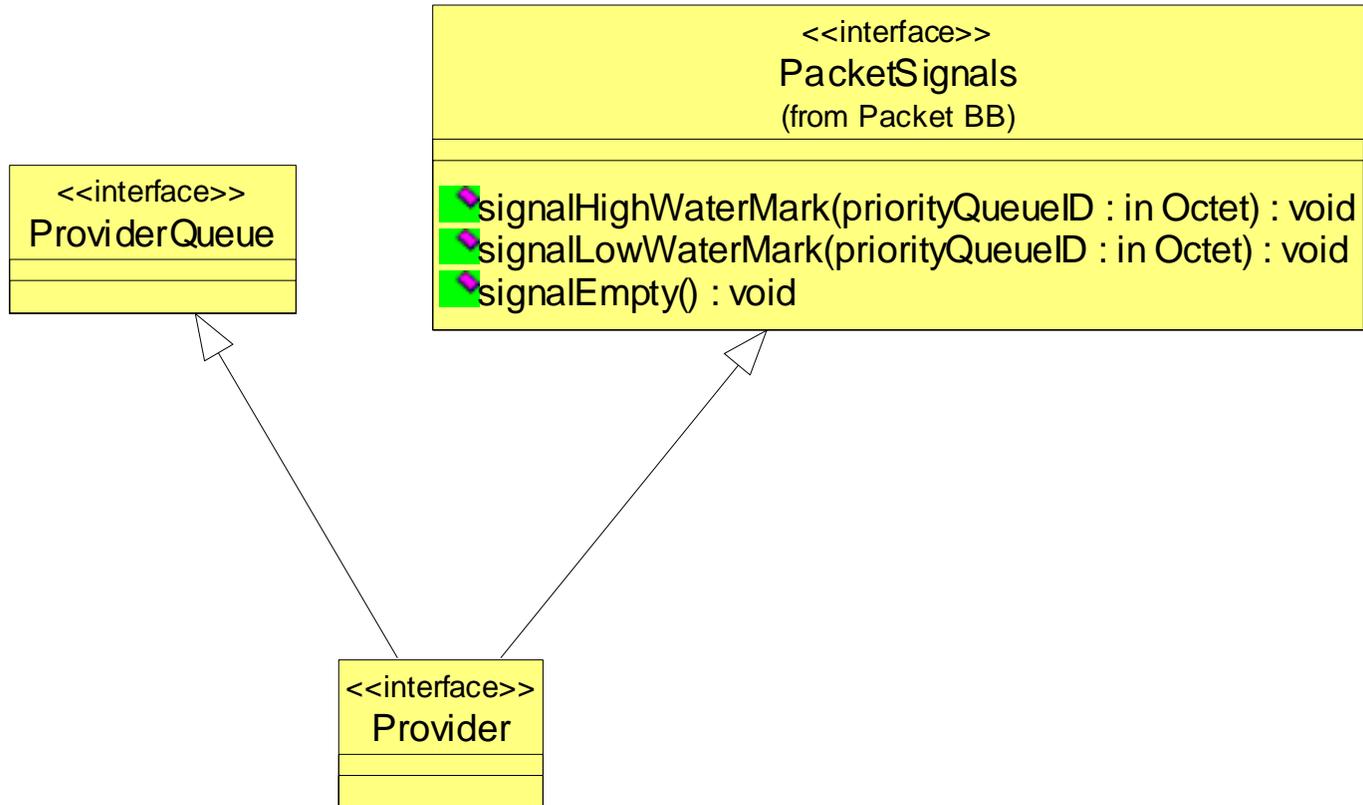
XYZ Three-Layer Waveform Analysis Package Relationships Example



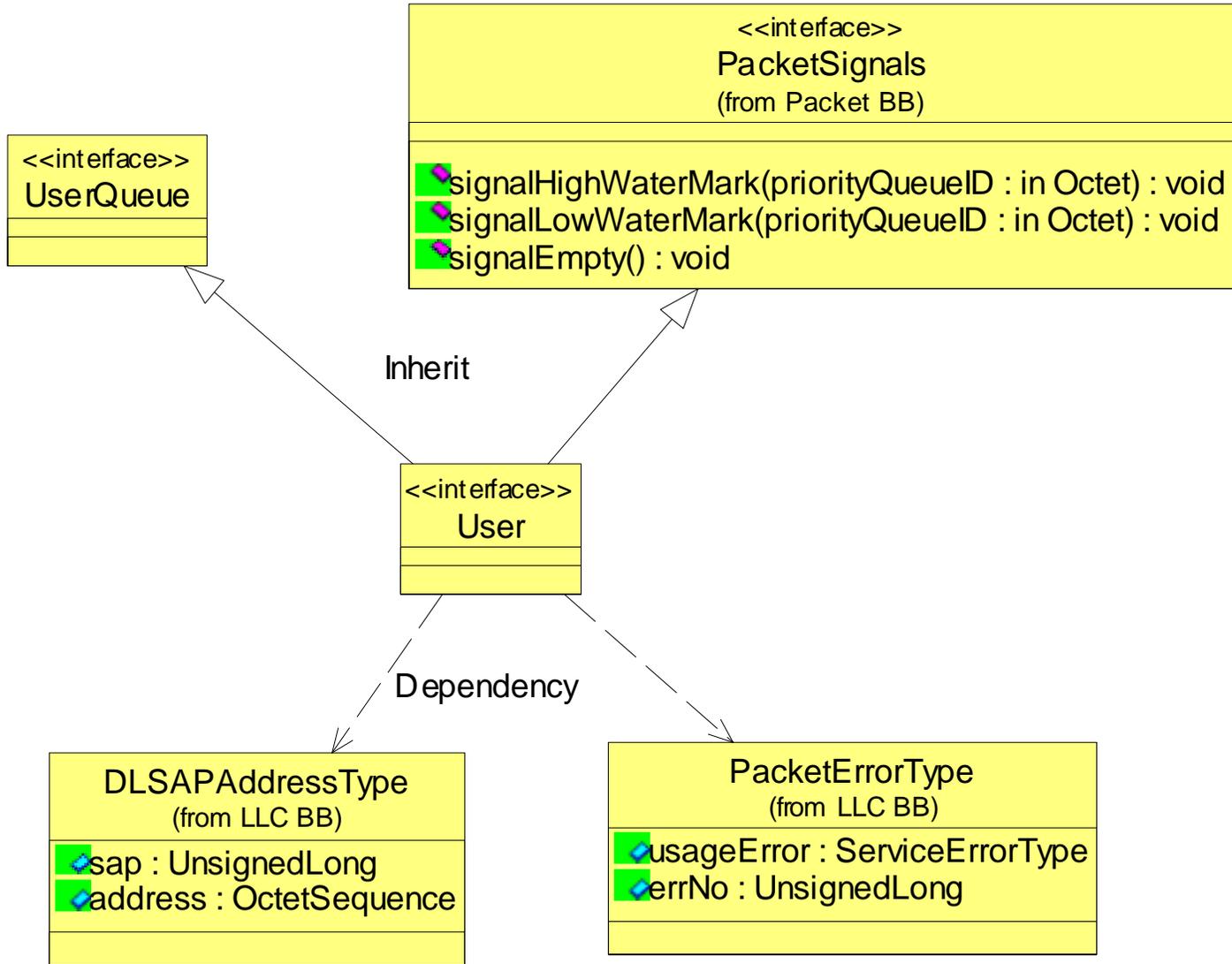
XYZ LLC Connectionless Queue Interfaces Example



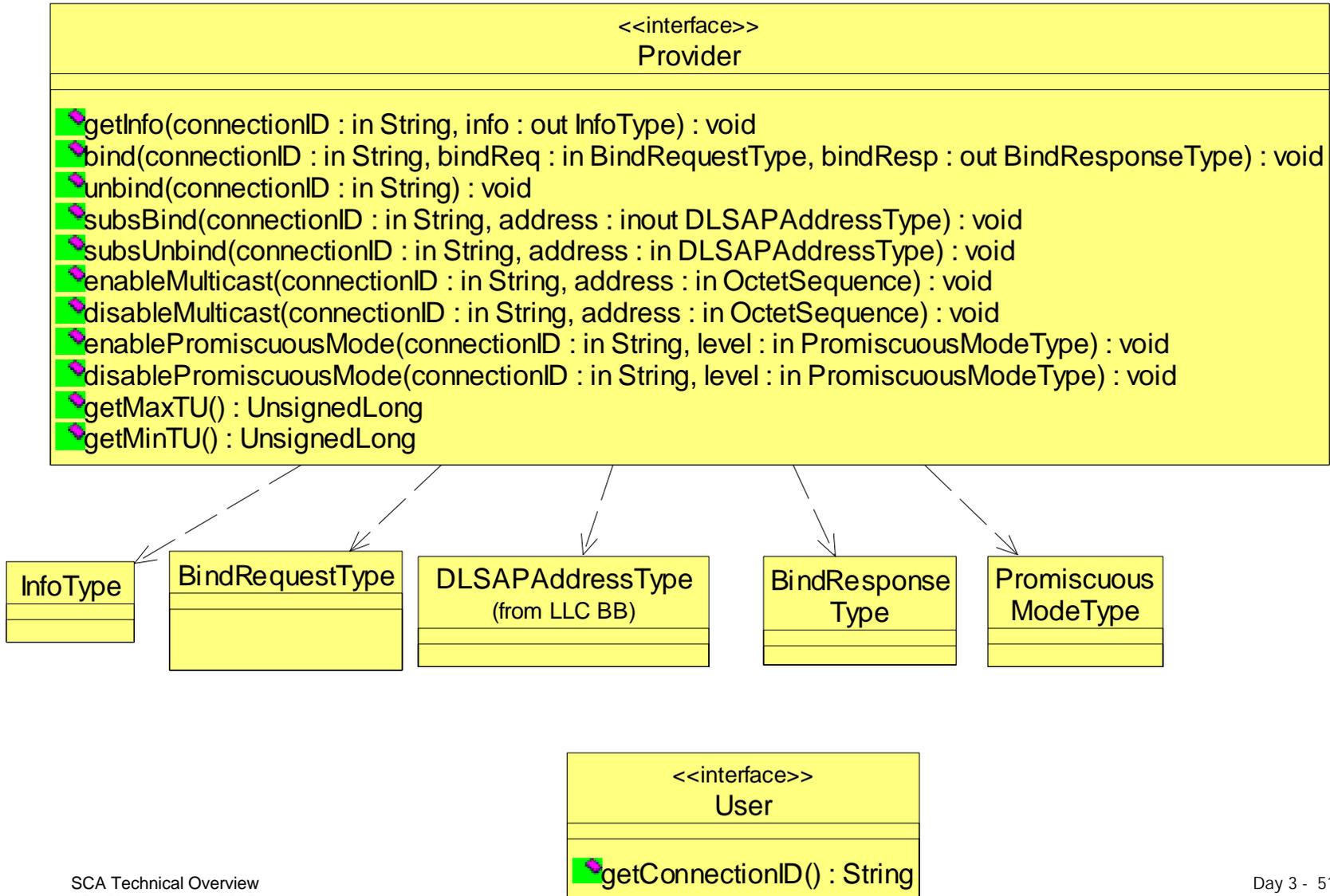
XYZ LLC Connectionless Provider Interface



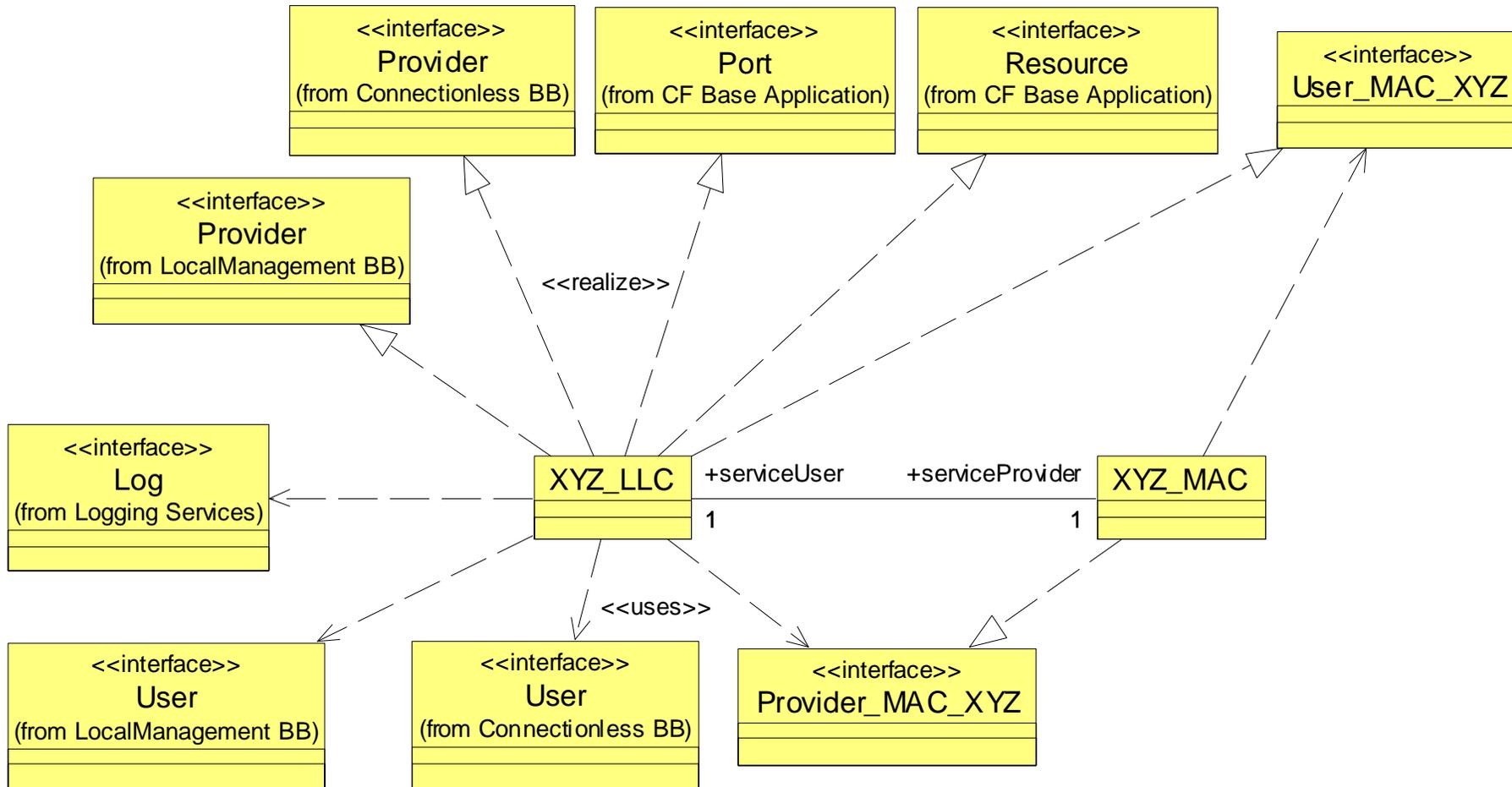
XYZ LLC Connectionless User Interface Example



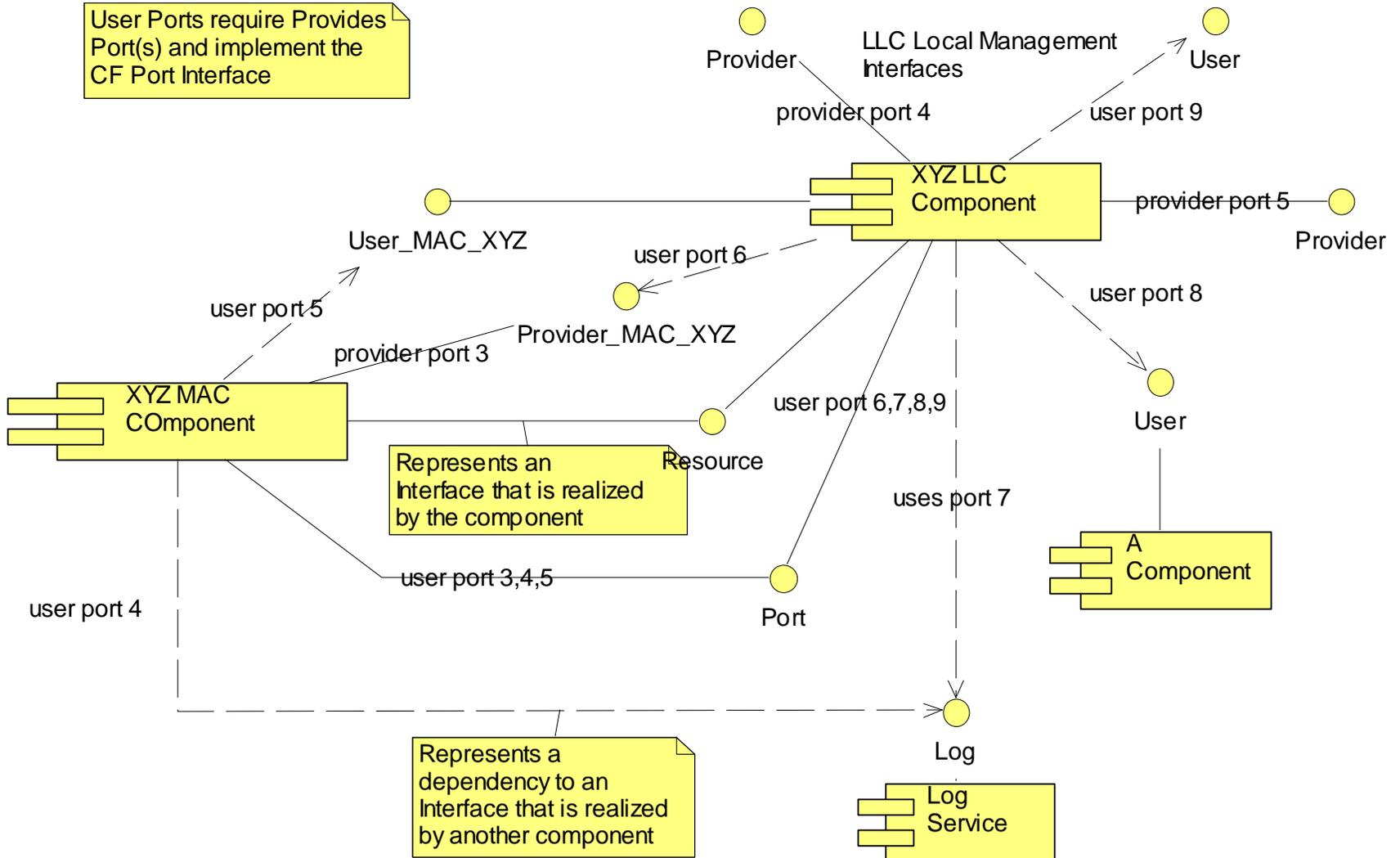
XYZ LLC Local Management Interfaces Example



XYZ LLC Class Example



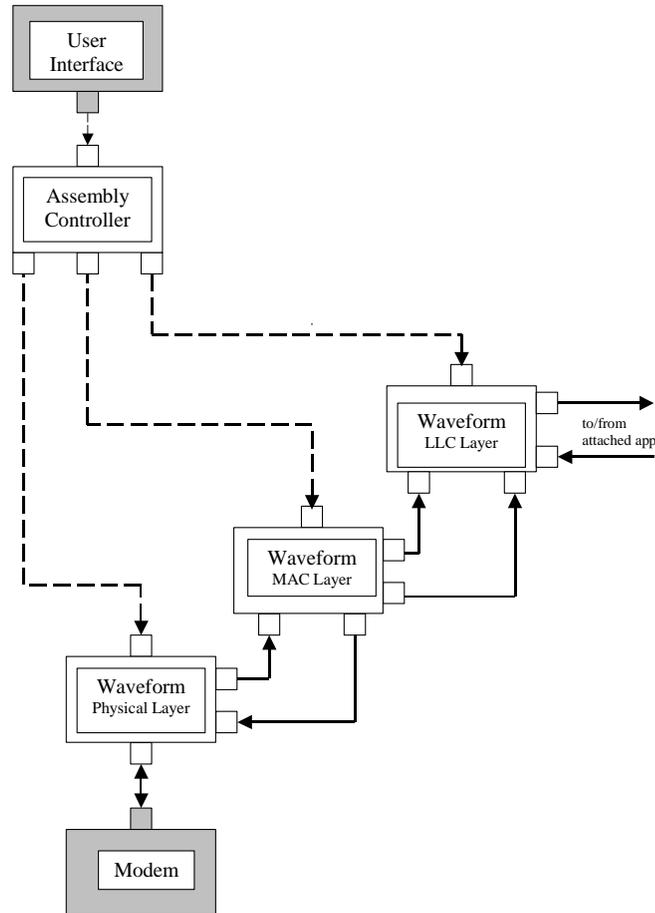
XYZ LLC Component Ports



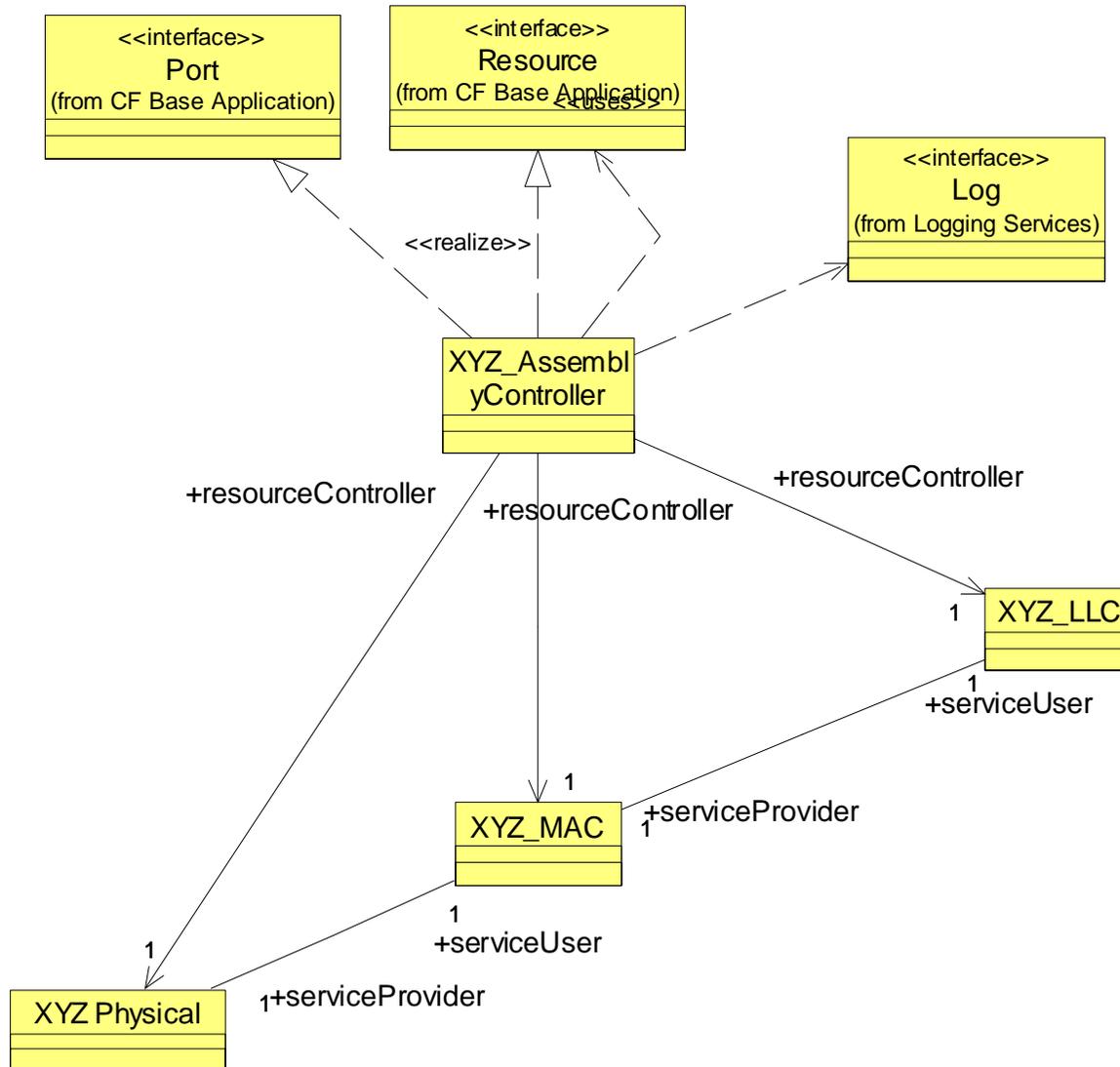
XYZ Waveform Assembly Controller

- Three-Layer Waveform
 - XYZ Assembly Controller Class
 - XYZ Assembly Controller Component

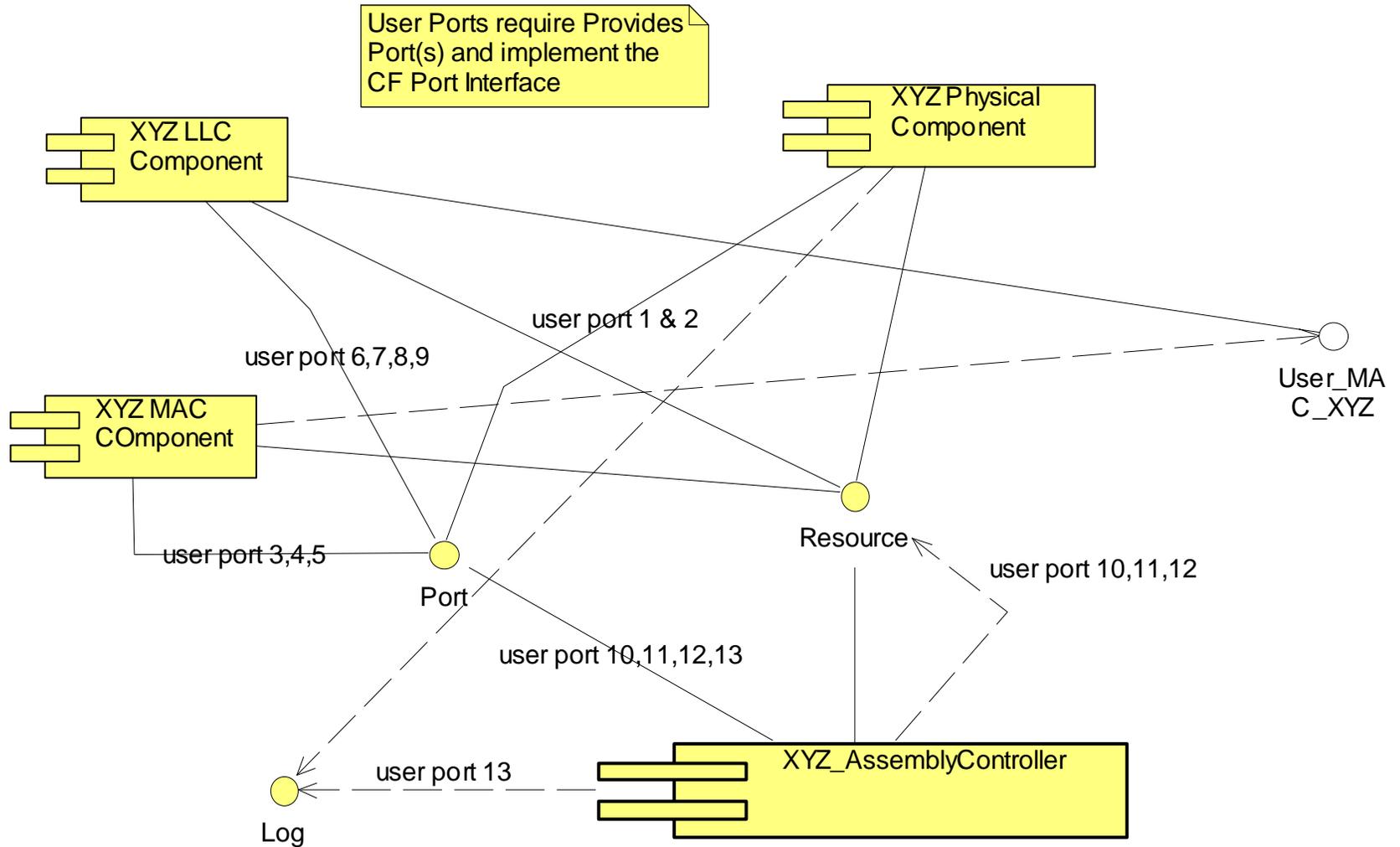
XYZ Assembly Controller Relationships



XYZ Assembly Controller Class



Assembly Controller Component Ports



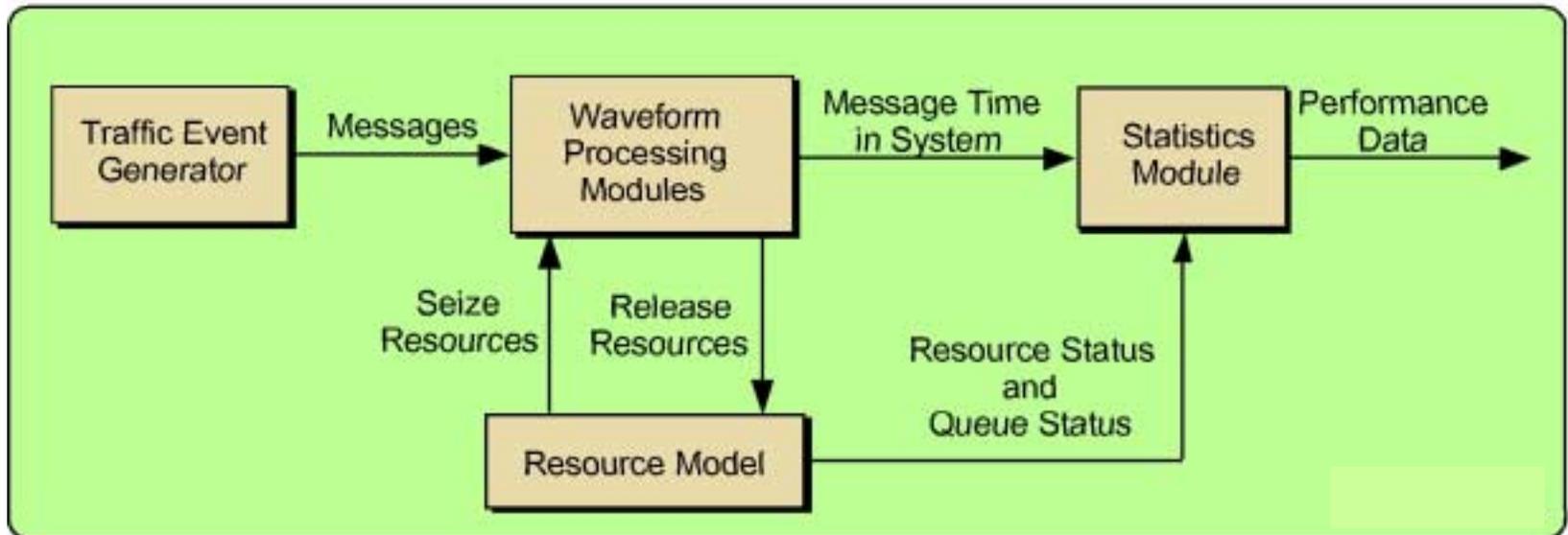
Build Waveform Analysis Model Activities

- Waveform UML Analysis Model – a service definition of the waveform APIs in UML
 - Captures the waveform APIs definition
 - Captures Waveform components definitions
- Simulation Model – a simulation of the execution of an application(s) against the real-world physical constraints of the system.

Waveform Simulation Model

- Simulation Model Elements
 - Trigger Generator
 - Waveform Processing Elements
 - Resource Model
 - Statistic Gathering
- Simulation Modeling Activities
- Simulation Model Summary

Simulation Model Elements Example



Simulation Traffic Generators

- Traffic Generators
 - Traffic Generators provide stimulus for waveform models in the form of transmit and receive events.
 - Attributes of traffic events are message type (such as voice, data or packet data), message size and arrival rate.
 - Message size and arrival rate often have statistical distributions derived from field test data, or from higher-level simulations of battlefield scenarios.
 - Traffic Generators can also be used to create inputs for software testing.

Simulation Waveform Processing Modules

- Maps to the Waveform classes and components in the Waveform UML Analysis Model
 - Contains the APIs and Message definitions for the waveform

Simulation Resource Model

- Resource(Capacity) Models
 - Resource models represent discrete processing resources in the system.
 - The resources may be processors, interfaces, or hardware devices.
 - Processing events seize resources on execution.
 - The allocation of resources to waveform processing objects defines the physical processing architecture of the waveform operating in the real system.

Simulation Statistic Gathering

- **Statistic Gathering**
 - Performance analysis focuses on requirements and design verification, resource utilization and timing budget allocation. Several waveforms are run concurrently to show simultaneous waveform operation.
 - Gathered queue statistics determine optimum data storage and buffer sizes.
 - Risks in achieving continuous data streams to user outputs are identified. Test cases to be analyzed are based on concept of operations.
 - Provides feedback to the Waveform UML Analysis Model
 - Deployment constraints and capacity needed

Simulation Modeling Activities

Early simulation of WF and CF operations provide insight into hardware requirements.

Simulation Goals. Simulation of waveforms results in early performance estimates.

<i>Goal</i>	<i>Approach</i>	<i>Modeling Outputs</i>
Verify processor utilization	Simulate waveform processing against a model of the physical resources	CPU Utilization for each processor
Verify timing	Model delays through the hardware and software based on physical architecture and processing work load.	Estimates of event latencies such as Attack and Release times, and end to end performance
Analyze a variety of waveforms running concurrently	Create models of each waveform that compete for discrete resources	CPU Utilization for each resource and latencies for each waveform
Investigate queuing	Model queues of resource grabs and queues of data and messaging services	Queue statistics such as maximum size, empty times (important for continuous data requirements), overrun.
Determine timing budgets of physical components	Model all components in a resource model that has timing budgets. Enter budget values and run waveforms to make sure budgets support waveform timing requirements	Detailed delay analysis of waveform events propagating through the system.

Simulation Model Summary

- Modeling significantly reduces the risks in porting. It is easy to modify resource models and resource allocations for new JTR Set LRUs.
- If the models use the XML descriptor files, then the task is simplified.
- To get an early indication of performance issues in porting to a new system or architecture, a new resource model is generated that matches the new target. Then the waveform simulations are run with the new physical architecture model to evaluate performance.

Waveform Design Process

- Build Waveform Analysis Model
- Build Waveform Language Interfaces
- Build Waveform Component Implementations
- Integrate Waveform Components

Build Waveform Language Interfaces

- A refinement of the interfaces defined in the WF Analysis Model.
- Create Platform or Technology-specific interfaces
 - CORBA
 - Java
 - C APIs

Build WF CORBA Interfaces

- Build UML model of interface { SCA Developer's Guide section 6.2 }
- Generate IDL from UML model of interface {SCA Developer's Guide section 5.2 }
- Translate IDL into language-appropriate implementation files {SCA Developer's Guide section 5.3 }
- Compile code generated {SCA Developer's Guide section 5.4 }
- Reverse engineer UML model from language-specific implementation files {SCA Developer's Guide section 5.5 }
(optional)

Waveform CORBA Interface Model Elements

- UML Concepts Utilized
 - Model Elements
 - Class (with CORBA Stereotypes, UML profile for CORBA)
 - Package (CORBAModule stereotype)
 - Component – CORBA Module
 - Diagrams
 - Class Diagram – captures CORBA types, exceptions and interface definitions.
 - Component Diagram – CORBA Module component and the interfaces realized within the CORBA Module
 - Relationships
 - Composition, Generalization, Dependency used to define the CORBA interfaces and types.

Waveform CORBA Naming Conventions

- APIs and building blocks should use scoped names. For example, the API set for LOS may have the following module hierarchy:

```
module LOS {  
    module Physical {  
    }  
    module Mac {  
    }  
}
```

Waveform CORBA Naming Conventions, cont'd

- Class, Interface, Package & Types names
 - Starts with an uppercase letter
 - Every word that composes the name starts with an uppercase letter
 - Words are run together (e.g., DomainManager)
- Methods and attributes
 - Starts with a lowercase letter
 - Every word that composes the name starts with an uppercase letter (except the first one)
 - Words are run together (e.g., getApplications)
- Constants & Enumeration Literals
 - Every letter of a word is an uppercase letter.
 - Multiple words are separated by an underline

Create Waveform CORBA UML Model

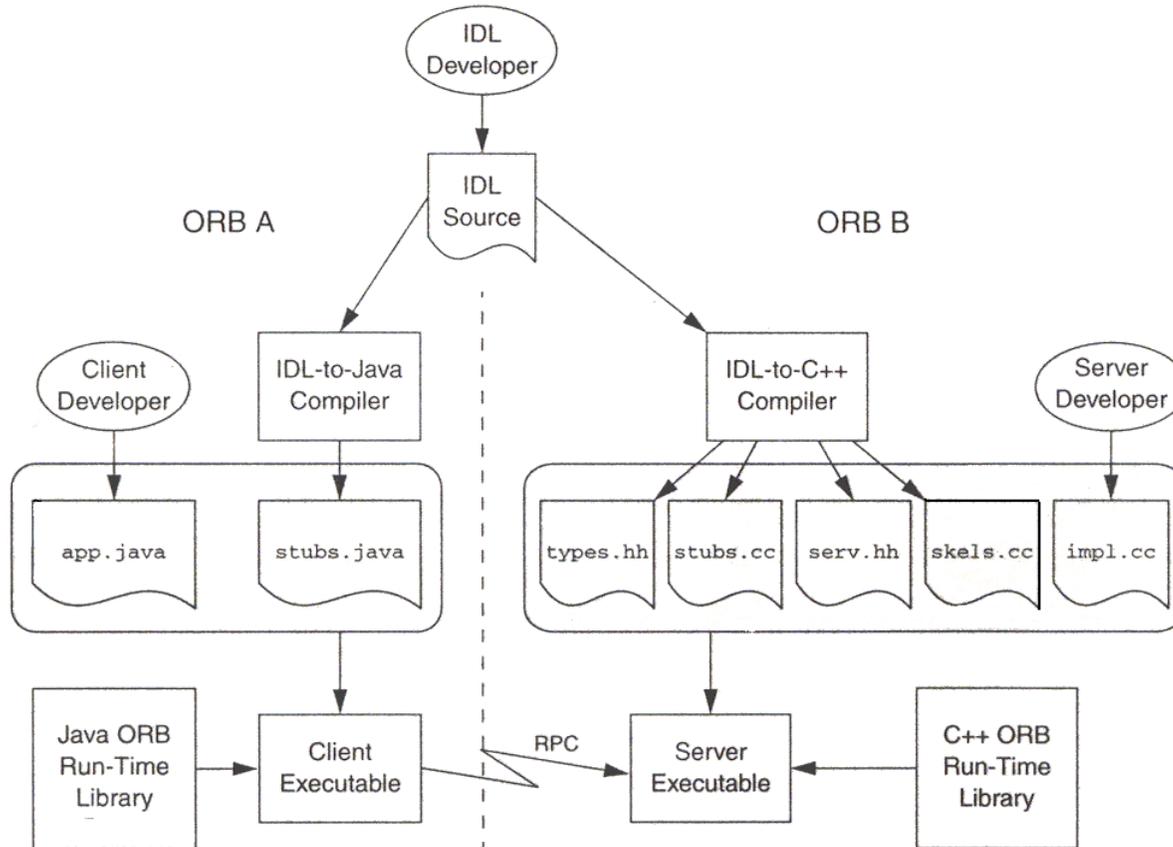
- Create a new Model file using UML Profile for CORBA
- A Refinement of the Waveform Analysis UML model
 - UML Interface map to CORBAInterface
 - Additional grouping of interfaces to form a unified interface
 - Show mapping to interfaces in the Waveform Analysis model using a refine relationship
 - Copy and paste types and interfaces from analysis model to CORBA model
 - Convert to CORBA stereotypes. Note in Rational Rose 2000-2002 version CORBAInterface is Interface, no change necessary.
 - Translate UML syntax into CORBA syntax
 - Stereotype
 - Types
 - Create CORBA Module packages and components
 - Module component realizes the CORBA interfaces.
- Uses Services' CORBA interfaces
 - As separate UML packages that can be loaded/Imported into a model

Generate IDL Files

- Create the IDL Files from the UML model.
 - IDL generation can happen from the component view or logical view

- UML Tool Additional Features
 - Browse the IDL
 - Reverse Engineer IDL
 - Syntax check

Translate IDL Into Language Implementation files & Compile



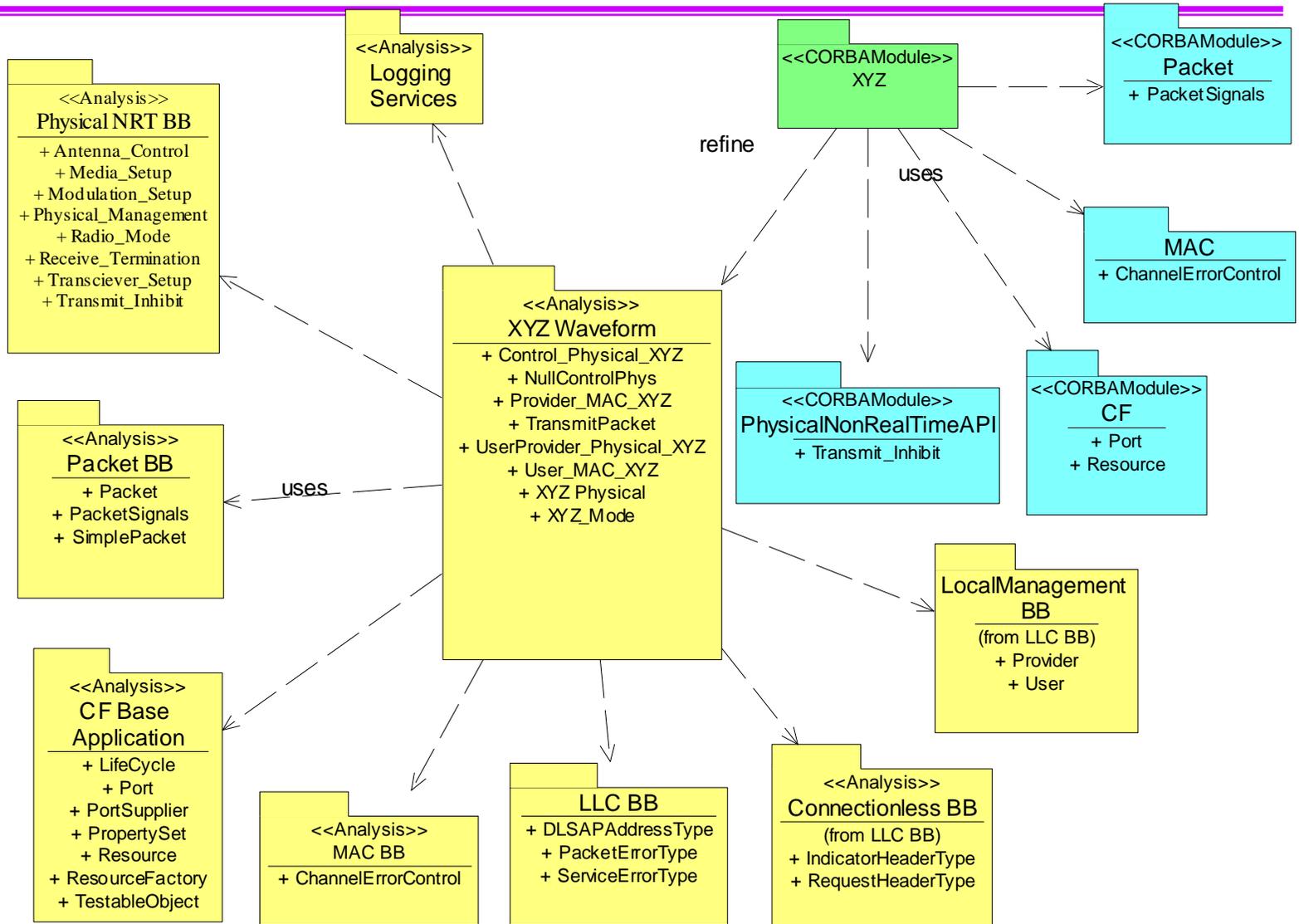
Reverse Engineering Generated Code

- Create UML packages from generated source files (e.g., C++ h & cpp) from IDL compiler.
 - CF
 - Log Service
 - Event Service
 - Waveform
 - Services
- Generated UML Classes
 - Client-Side Class Names are the same as the CORBA module and CORBA interface names.
 - Var type classes (e.g., XYZ_var) are managers of object pointers
 - Server-Side Class names are POA type classes (e.g., POA_XYZ), which are used by server-side implementation classes (servants).

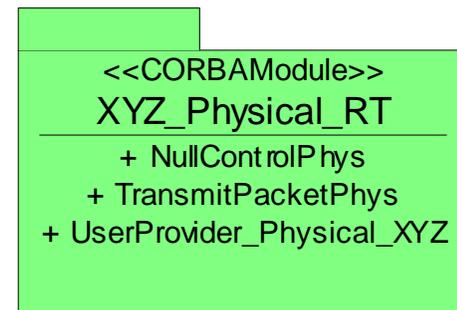
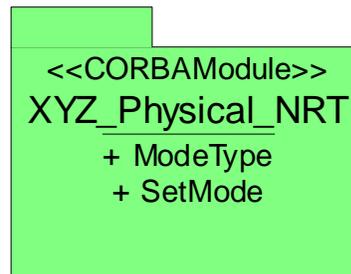
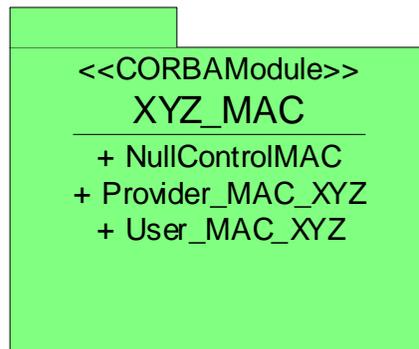
XYZ UML CORBA Interface Examples

- Package Relationships
- XYZ CORBA Modules
- XYZ Packet Interfaces
- XYZ Control Physical Interface
- Unified XYZ Physical Interface
- Partitioned XYZ Physical Interface
- XYZ MAC Interface
- Logical Link Control ProviderQueue Interface
- Logical Link Control UserQueue Interface
- Logical Link Control Provider Interface
- Logical Link Control User Interface
- LLC Local Management Interface

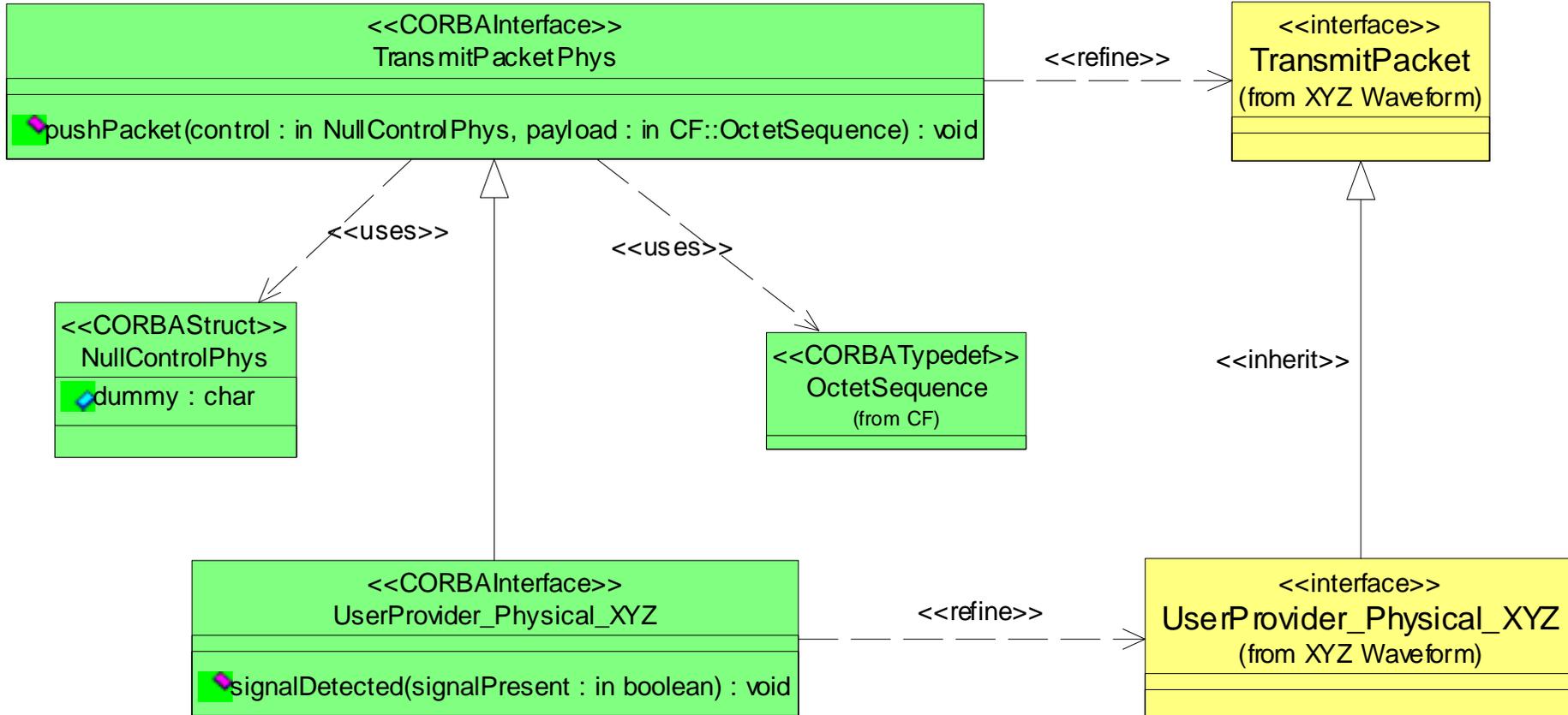
XYZ CORBA Module Package Relationships



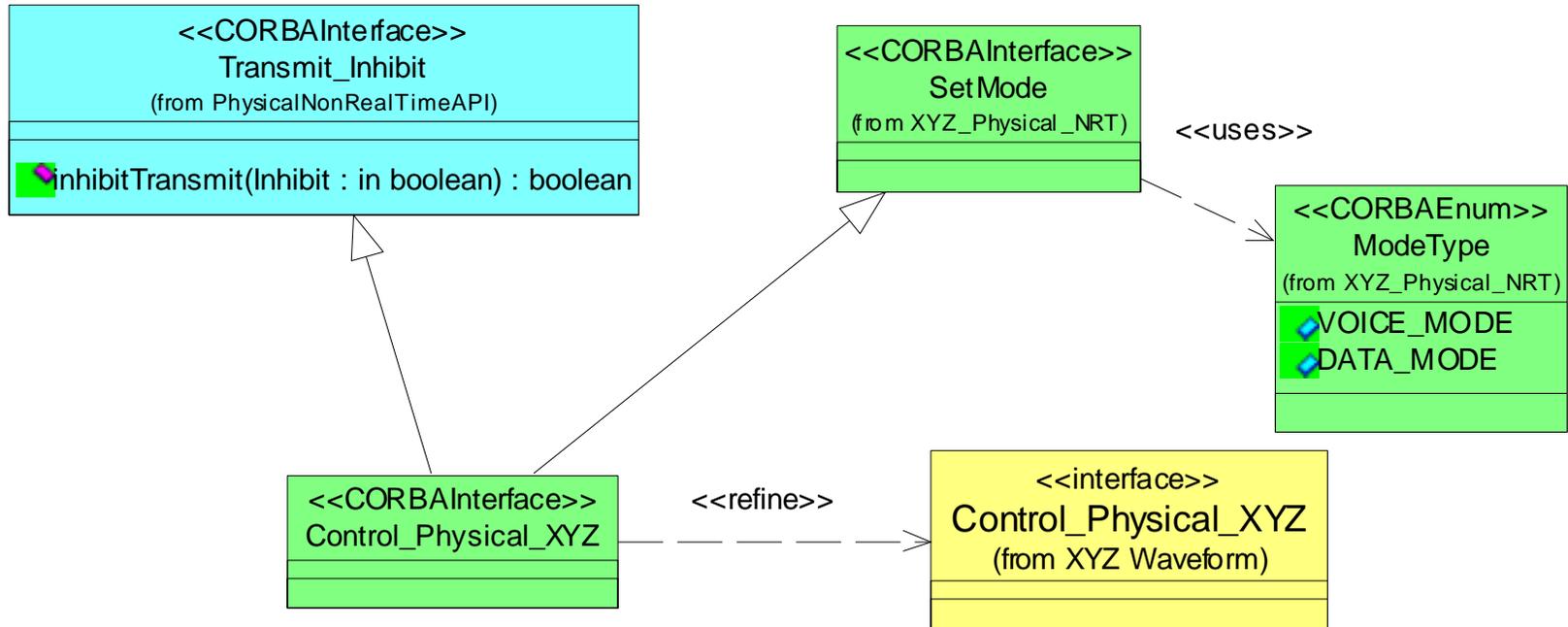
XYZ's sub CORBA Modules



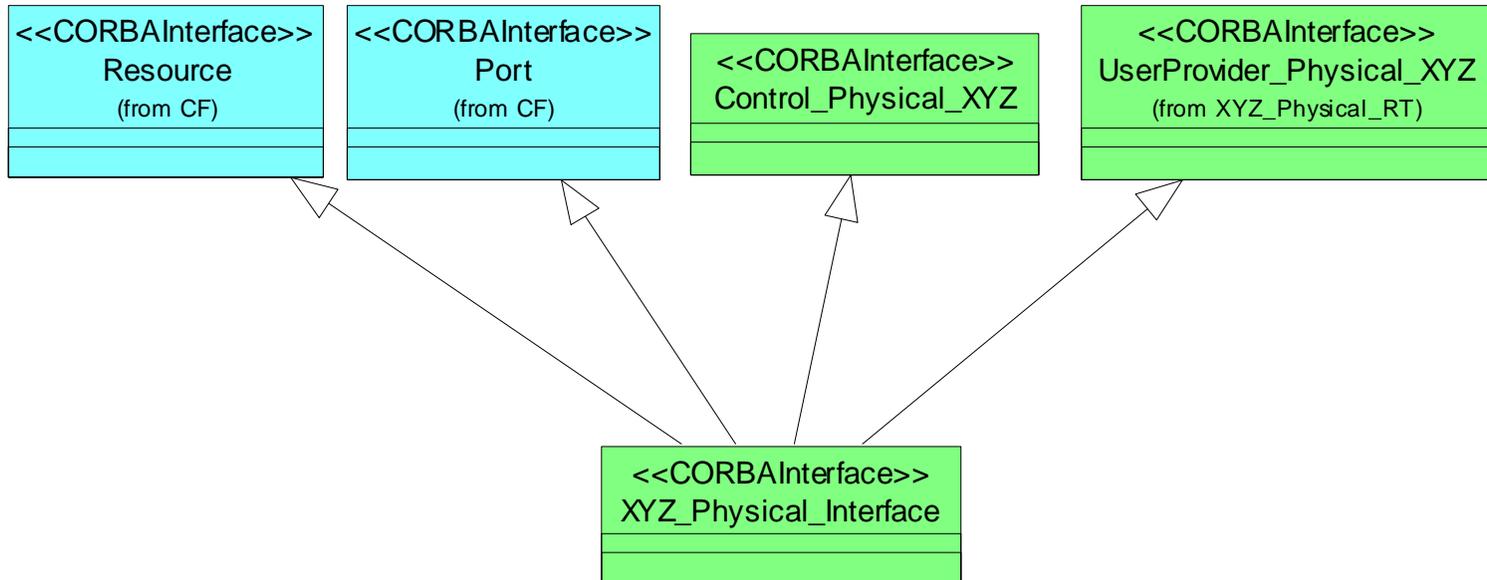
XYZ Packet CORBA Interfaces



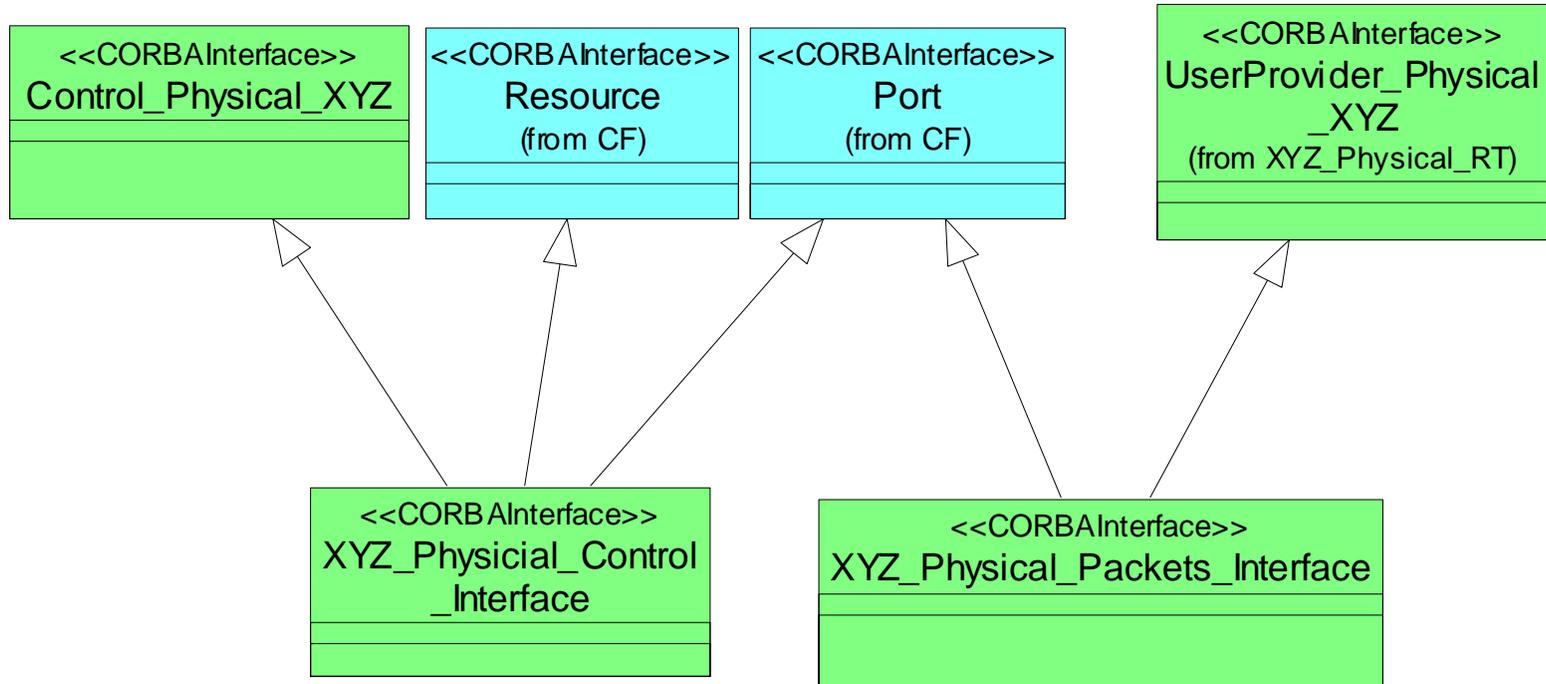
XYZ Control Physical CORBA Interface



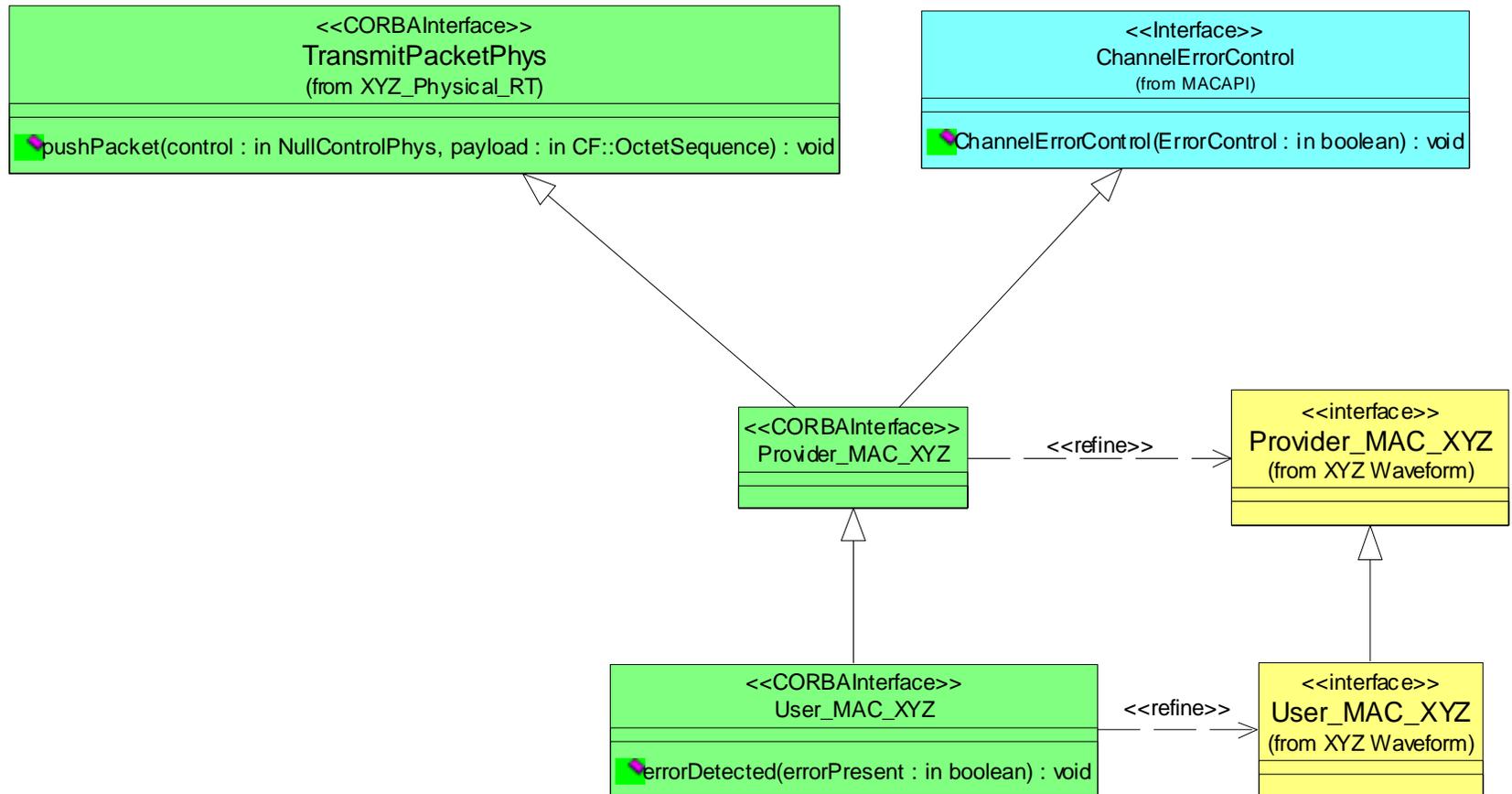
Unified IDL Design for XYZ Physical Layer



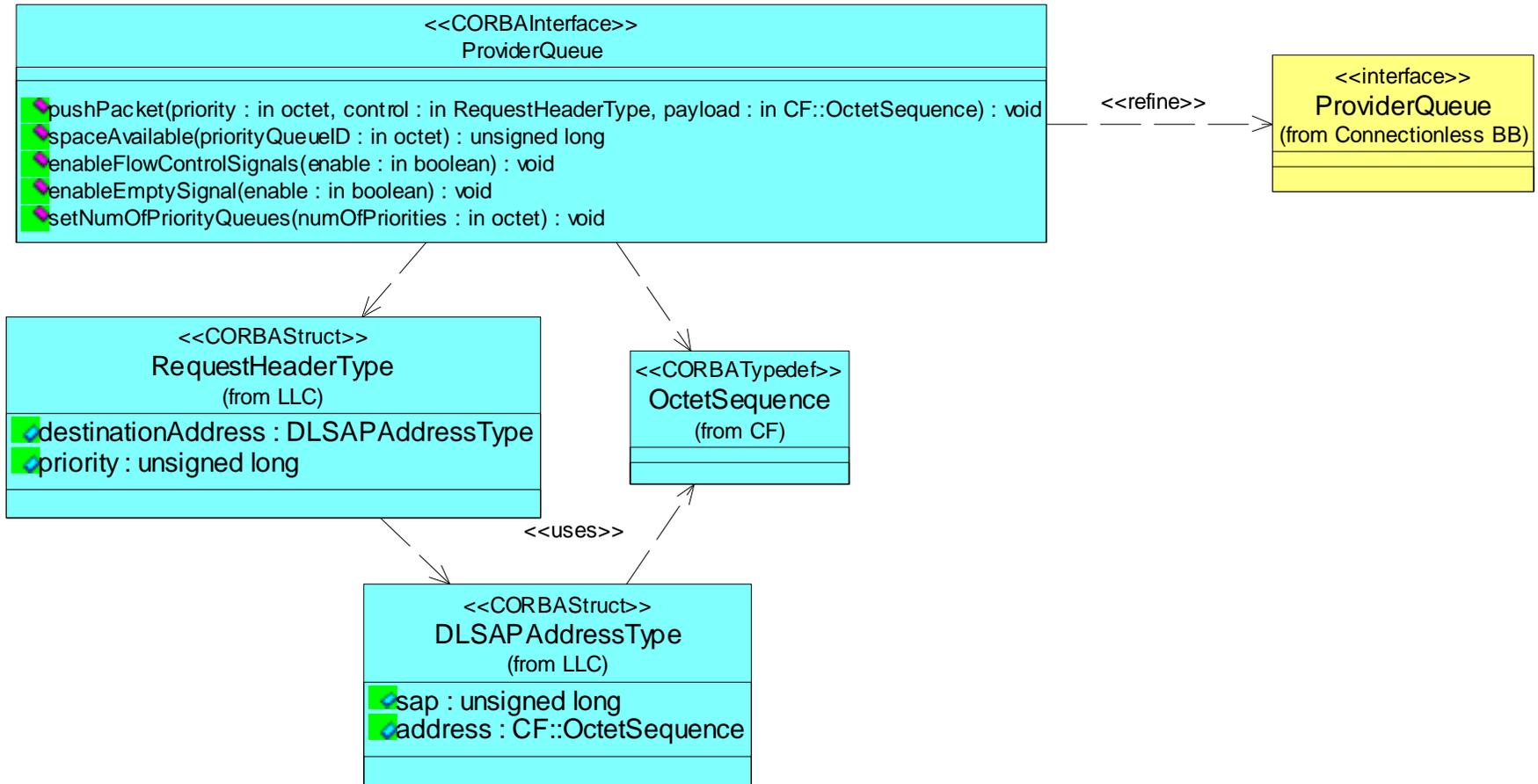
Partitioned IDL Design for XYZ Physical Layer



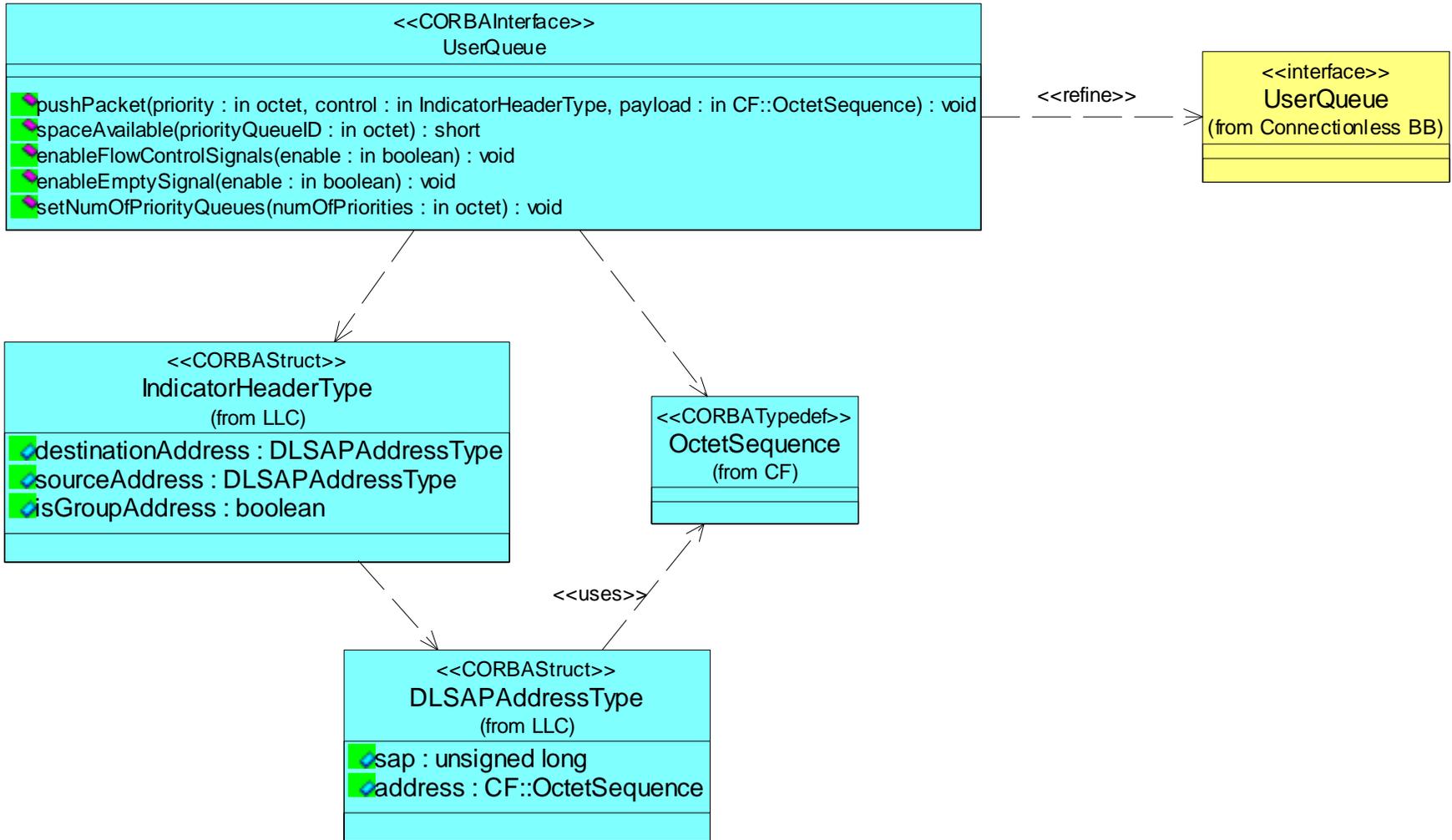
XYZ MAC CORBA Interfaces



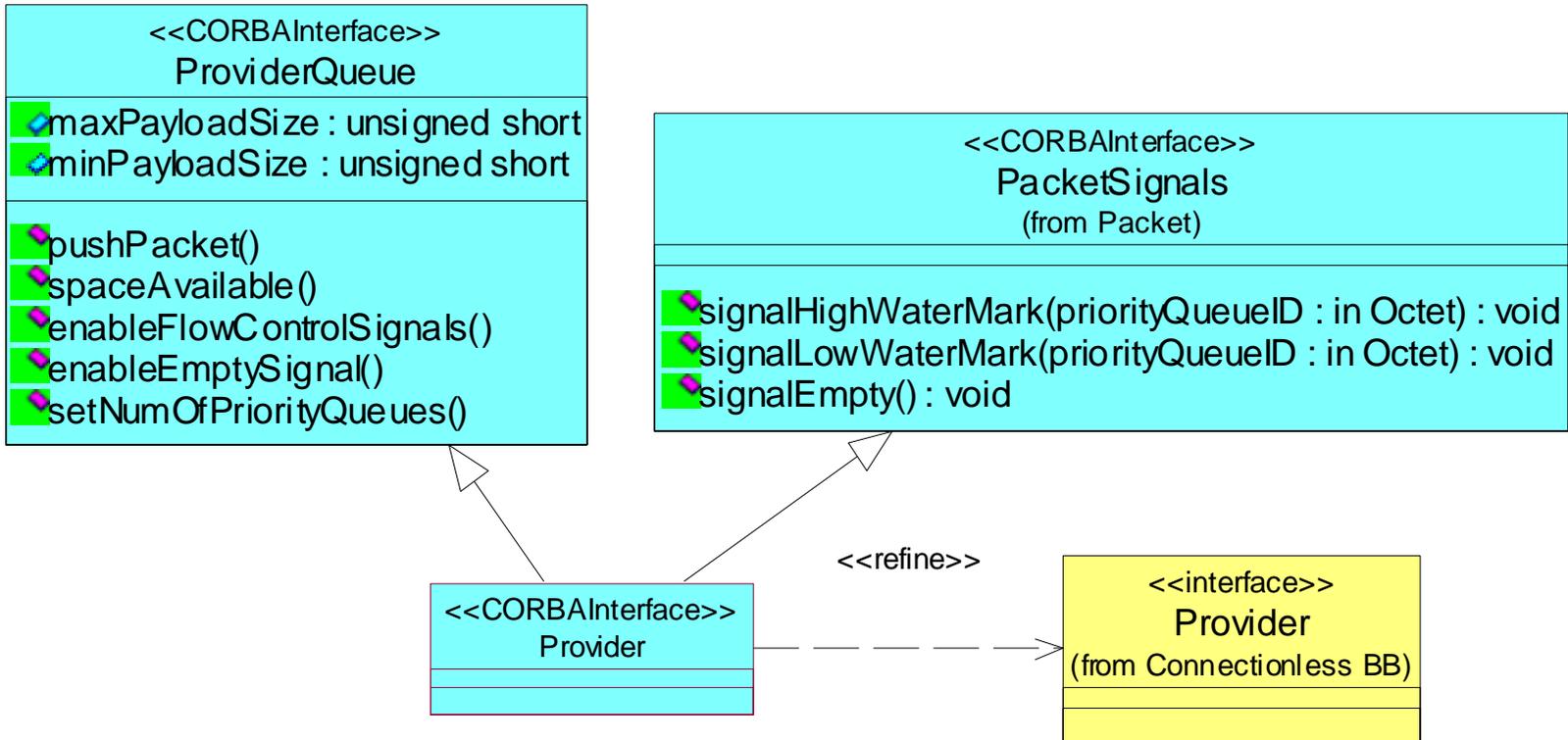
Logical Link Control Connectionless ProviderQueue CORBA Interface



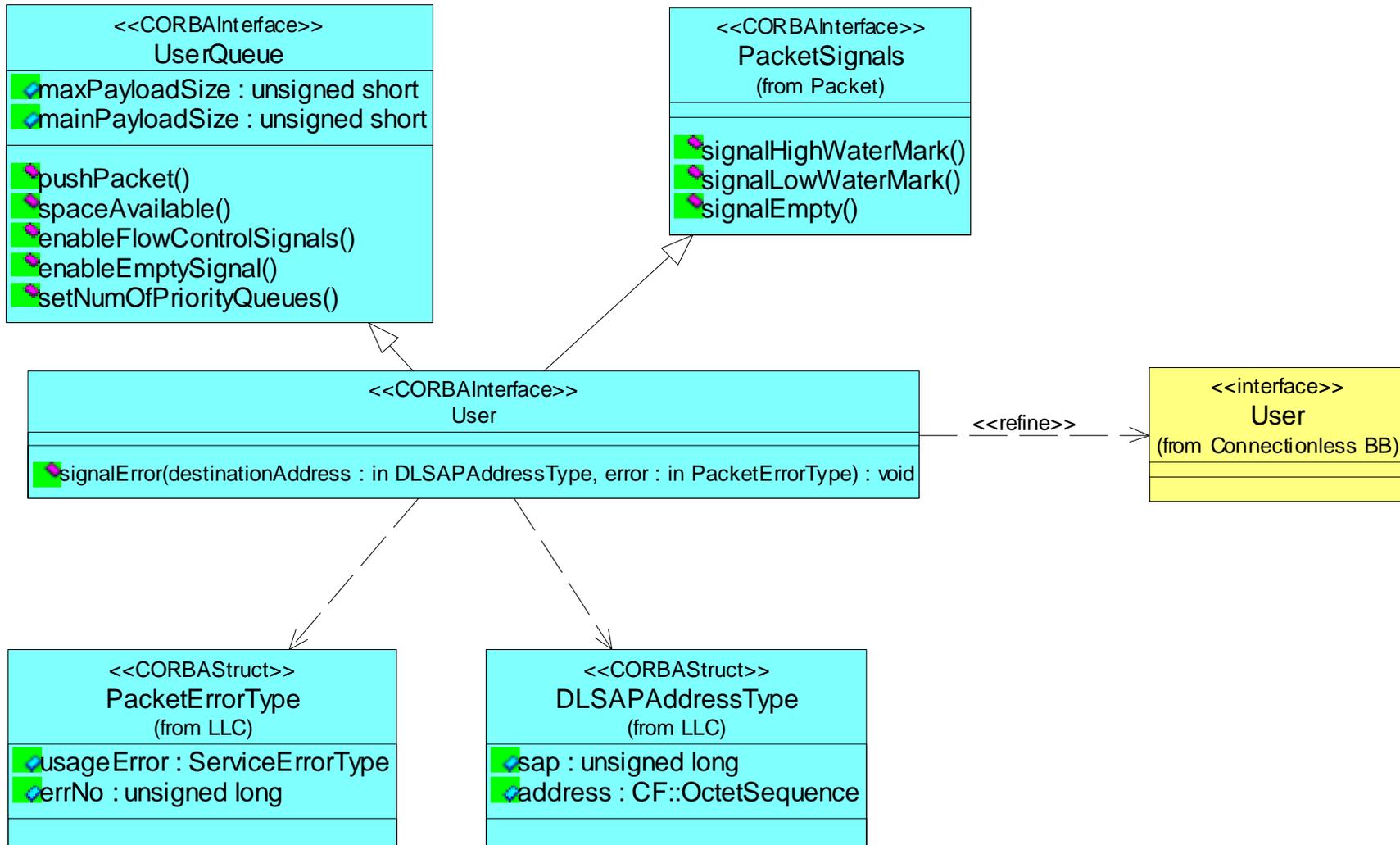
Logical Link Control Connectionless UserQueue CORBA Interface



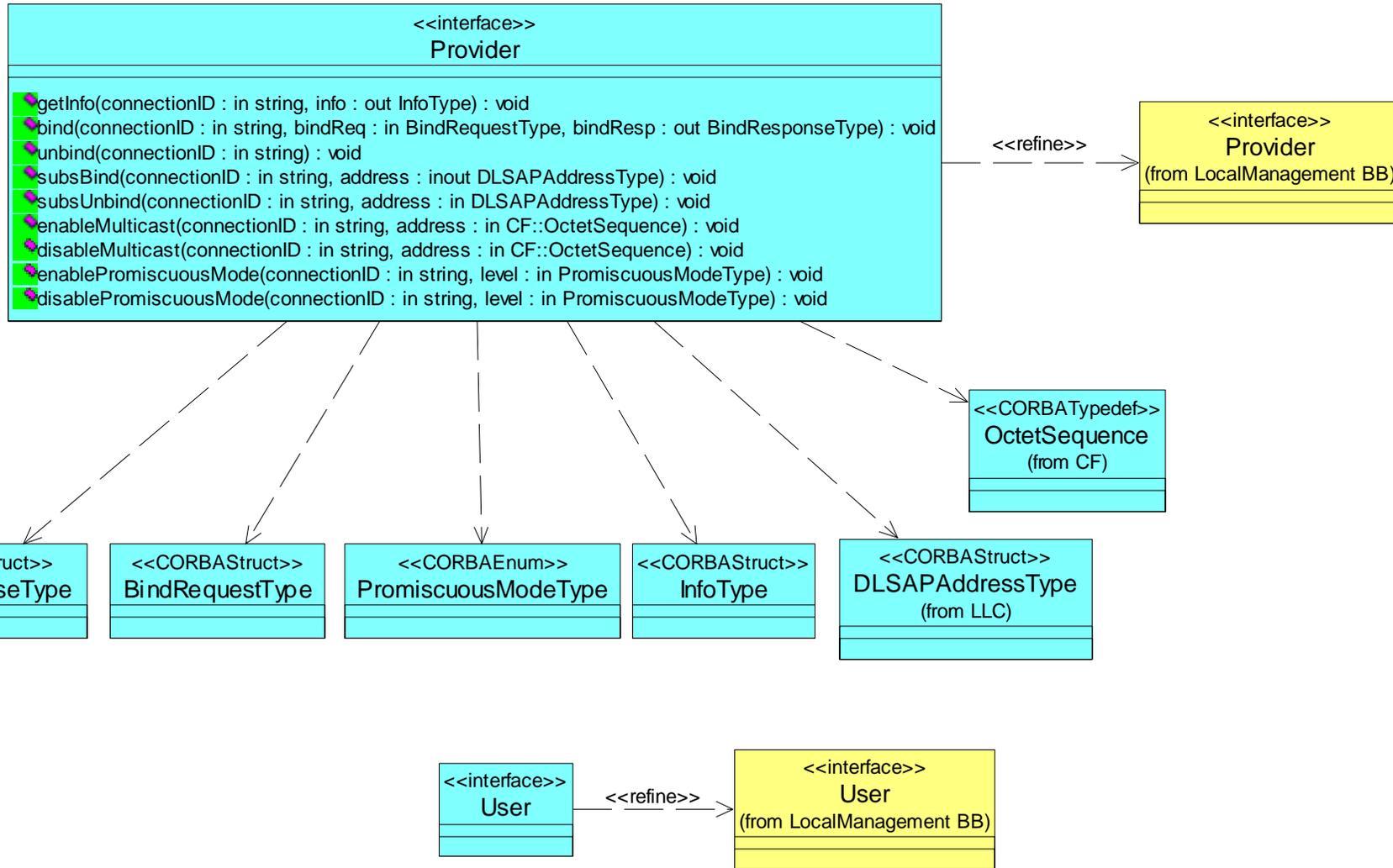
Logical Link Control Provider CORBA Interface



Logical Link Control Connectionless User CORBA Interface



Logical Link Control Local Management CORBA Interfaces



Waveform Design Process

- Build Waveform Analysis Model
- Build Waveform Language Interfaces
- Build Waveform Component Implementations
- Integrate Waveform Components

Build Waveform Component Implementations

- Build Waveform UML Implementation model of waveform software { SCA Developer's Guide section 6.3 }
- Generate language-appropriate source files for servant and user software {SCA Developer's Guide section 5.7 }
- Complete Language Source Files Coding
- Create XML for each component {SCA Developer's Guide section 5.8 }
- Build User Interface {SCA Developer's Guide section 8 } (optional)

Build Waveform Implementation Model

- Create Waveform Implementation Model
- Waveform UML Implementation Model Elements
- Waveform Resource Component Implementations
- XYZ Component Design Implementation Examples

Create Waveform Implementation Model

- Create a new Model file using UML Language Profile(s) (C++, Java, etc.) for Waveform Implementation
 - UML Tool can act as File Browser with configuration management for all the waveform artifacts.
 - Most UML tools supports add-in capability for a configuration tool
 - Component View can contain all the waveform artifacts that make up the waveform.
 - Each different artifact category (e.g., type of software) could map to a different component package in the component view
 - Types of waveform artifacts that are deployed on the various re-programmable devices
 - General-Purpose Processor (GPP) that runs CORBA ORBS – software usually written in High-Order language (e.g., ANSI C++ language)
 - Digital Signal Processors (DSP) – software usually written in ANSI C language
 - FPGAs – software usually written in VHDL
 - Deployment View can capture the waveform artifacts on the re-programmable devices and the devices that are used by the artifacts.

Create Waveform Implementation Model, cont'd

- A Refinement of the Waveform UML Analysis model and realization of the Language interface models (e.g., CORBA)
 - Logical View
 - Create Waveform Functionality Packages
 - One package for the Waveform Implementation with nested sub-packages based upon functionality and/or language of artifact (e.g., physical, MAC, LLC, etc.).
 - Create Servant classes that refine the corresponding analysis class and implements the corresponding language interfaces
 - Create classes (stereotyped for C language) for DSP functionality
- Uses the Reversed Engineered packages associated with the Language Generated Source Code from CORBA IDL Compilers
 - As separate UML packages that can be loaded/Imported into a model
 - Provides CORBA interface definition in the native implementation language



Build Waveform Implementation Model

- Create Waveform Implementation Model
- Waveform UML Implementation Model Elements
- Waveform Resource Component Implementations
- XYZ Component Design Implementation Examples

Waveform Implementation Model Elements

- UML Concepts Utilized
 - Model Elements
 - Class – Waveform Artifact design implementation definition for waveform interfaces. Artifacts are implemented in High-Order Languages (e.g. C++, Java, C. etc.).
 - Package – contains the Waveform artifact design implementations
 - Use Case – realization of the of the use cases using the waveform servant classes.
 - Component – Waveform Artifacts (e.g., main server programs, header and body source files (e.g., h, cpp,c), XML files, etc.)
 - Node – where a waveform artifact can be or is deployed within a system.

Waveform Implementation

Model Elements, cont'd

- UML Concepts Utilized
 - Diagrams
 - Class Diagram – Waveform Artifact (e.g., classes, etc.) Definitions
 - Use Case – Captures the design realization of the use case
 - Interaction - design realization of the use case.
 - Component – Waveform artifacts – component and main program source files, XML files, etc.
 - Relationships
 - Aggregation, Composition, Generalization, and Dependency are use to depict the definition of waveform artifacts and components.



Build Waveform Implementation Model

- Create Waveform Implementation Model
- Waveform UML Implementation Model Elements
- **Waveform Resource Component Implementations**
- XYZ Component Design Implementation Examples

Waveform Resource Component Implementations

- Gives out provides and uses port in the form of CORBA Interoperable Object References (IORs).
 - Provides Port is a CORBA interface implemented by a servant on the server side.
 - Uses Port requires a specific CORBA interface(s) to use.
- The CF Resource getPort operation returns a CORBA Object reference, which all CORBA objects are derived from.
 - The getPort operation can delegate this behavior to other operations or simply does a lookup to return a port object reference.
- Note: All CORBA objects can be widened to a CORBA Object type.

Waveform Resource Component Provides Port Implementations

- Provides port can be implemented using TIE or non-TIE approach, provided the compiler generated the corresponding code.
 - TIE
 - In C++, the servant software must instantiate the IDL-generated template skeleton class.
 - Non-TIE
 - In C++, the servant software must inherit from the IDL-generated skeleton class.
 - The logical Provides port name is associated with the servant software (class).
 - Note: Servant software can support multiple interface implementations.

Waveform Resource Component Uses Port Implementations

- Uses port implement the CF Port interface.
- Uses port software maintains one or more pointers (of the correct type) to the servant.
- In C++, recommendation is to use the CORBA “XYZ”_var type, because this "intelligent pointer" relieves the programmer of most responsibilities relating to memory management.
- When the Uses Port is associated with multiple provides port the code has more checking to do.
 - Ensures the right interface type is being specified and places the Provides port object reference with the appropriate code.
 - For multiple providers with the same interface type the connection identifier has be used as the discriminator.

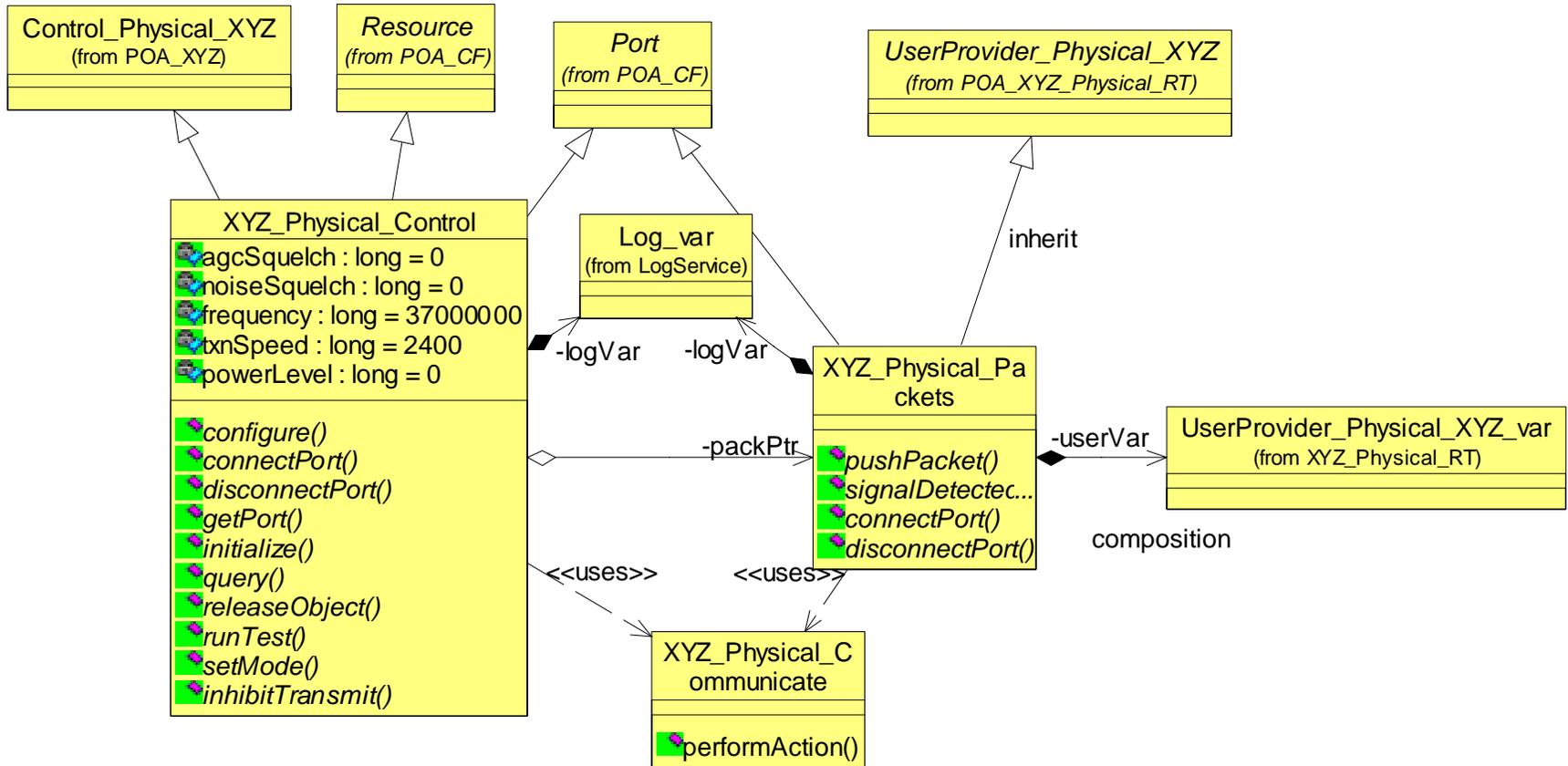
Build Waveform Implementation Model

- Create Waveform Implementation Model
- Waveform UML Implementation Model Elements
- Waveform Resource Component Implementations
- XYZ Component Design Implementation Examples

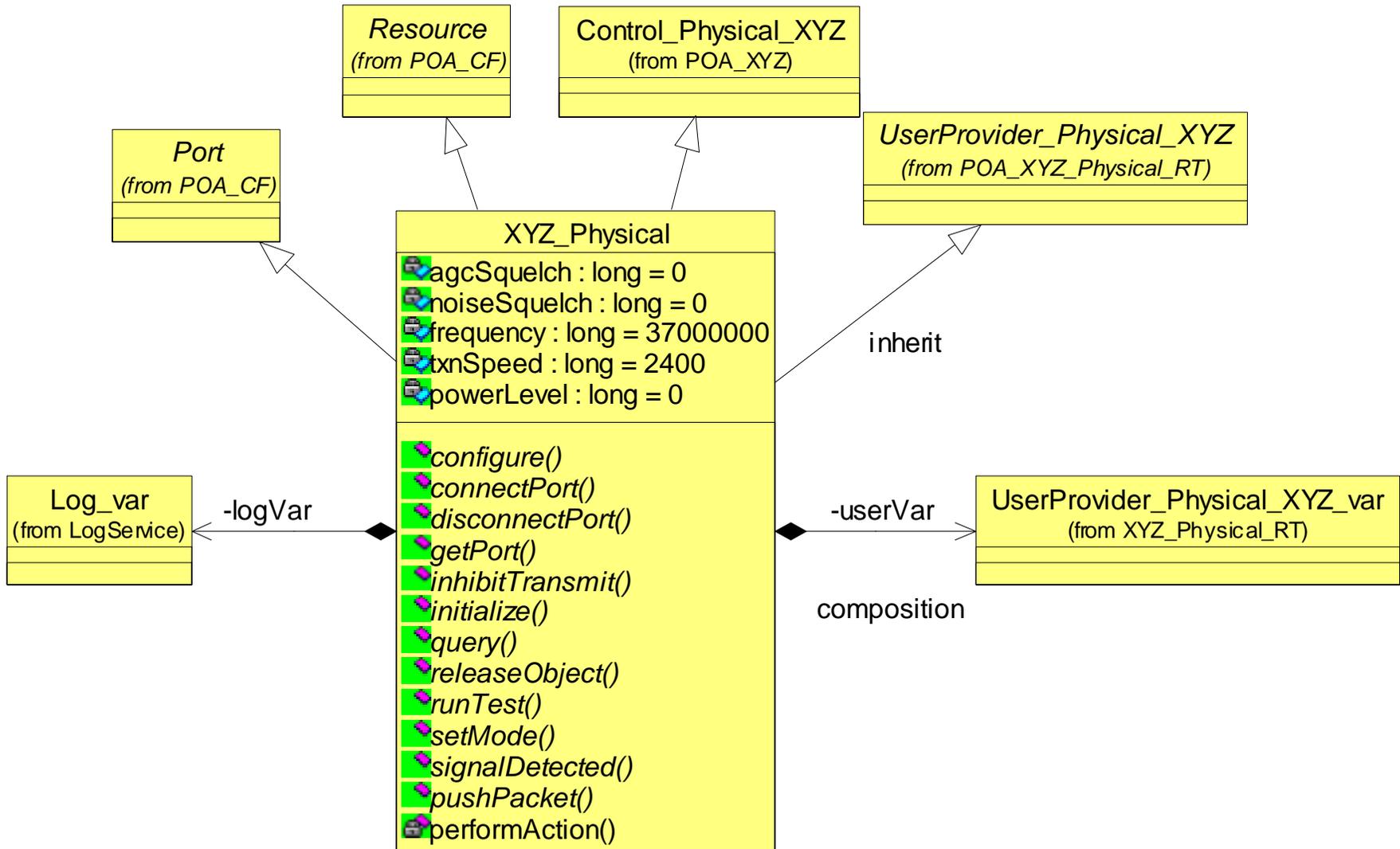
XYZ Component Design Implementation Examples

- Partitioned Design for XYZ Physical
- Unified Design for XYZ Physical Interface
- XYZ MAC Design Implementation
- XYZ Logical Link Design Implementation
- XYZ Assembly Controller Design Implementation

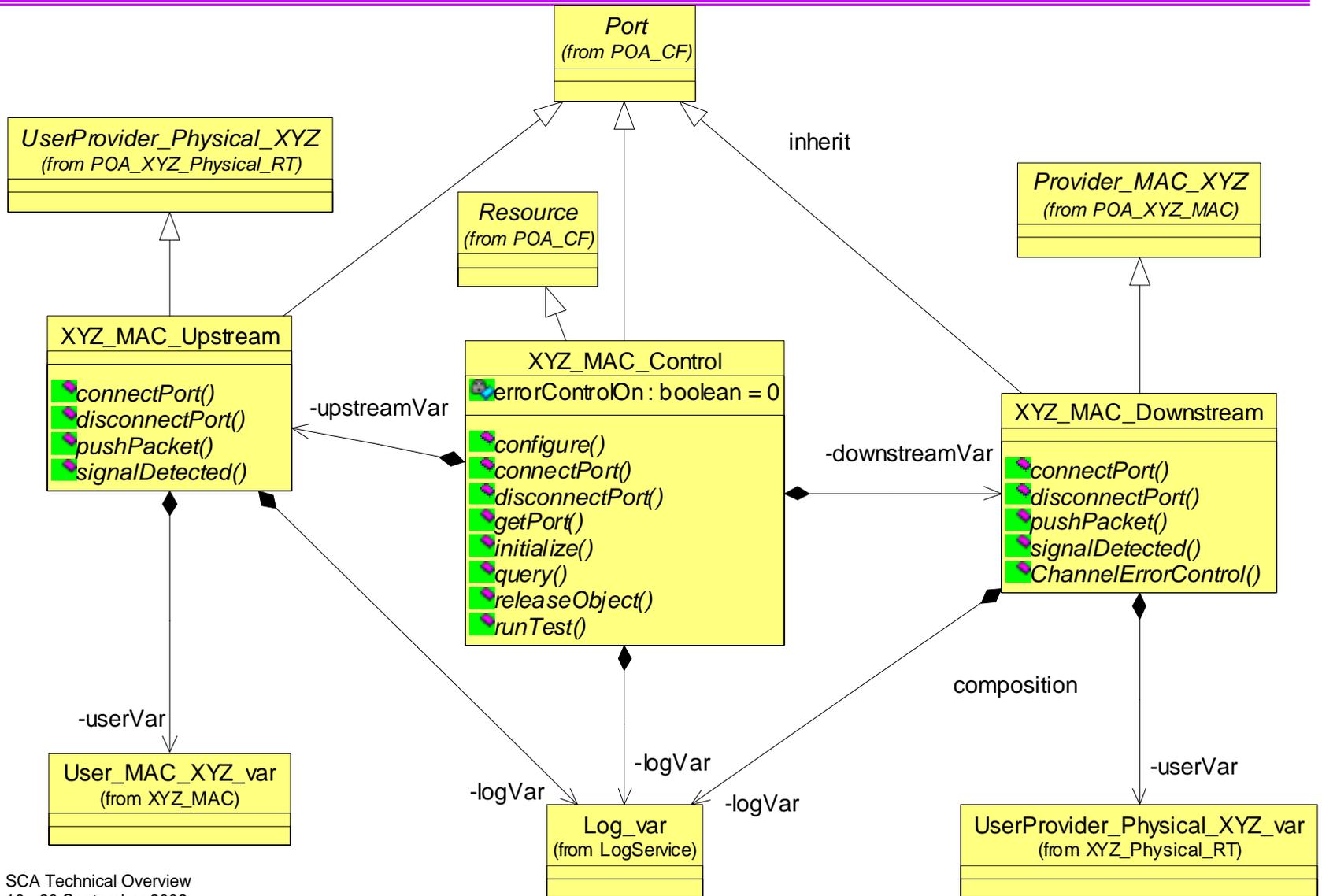
Partitioned Design for XYZ's Physical Layer



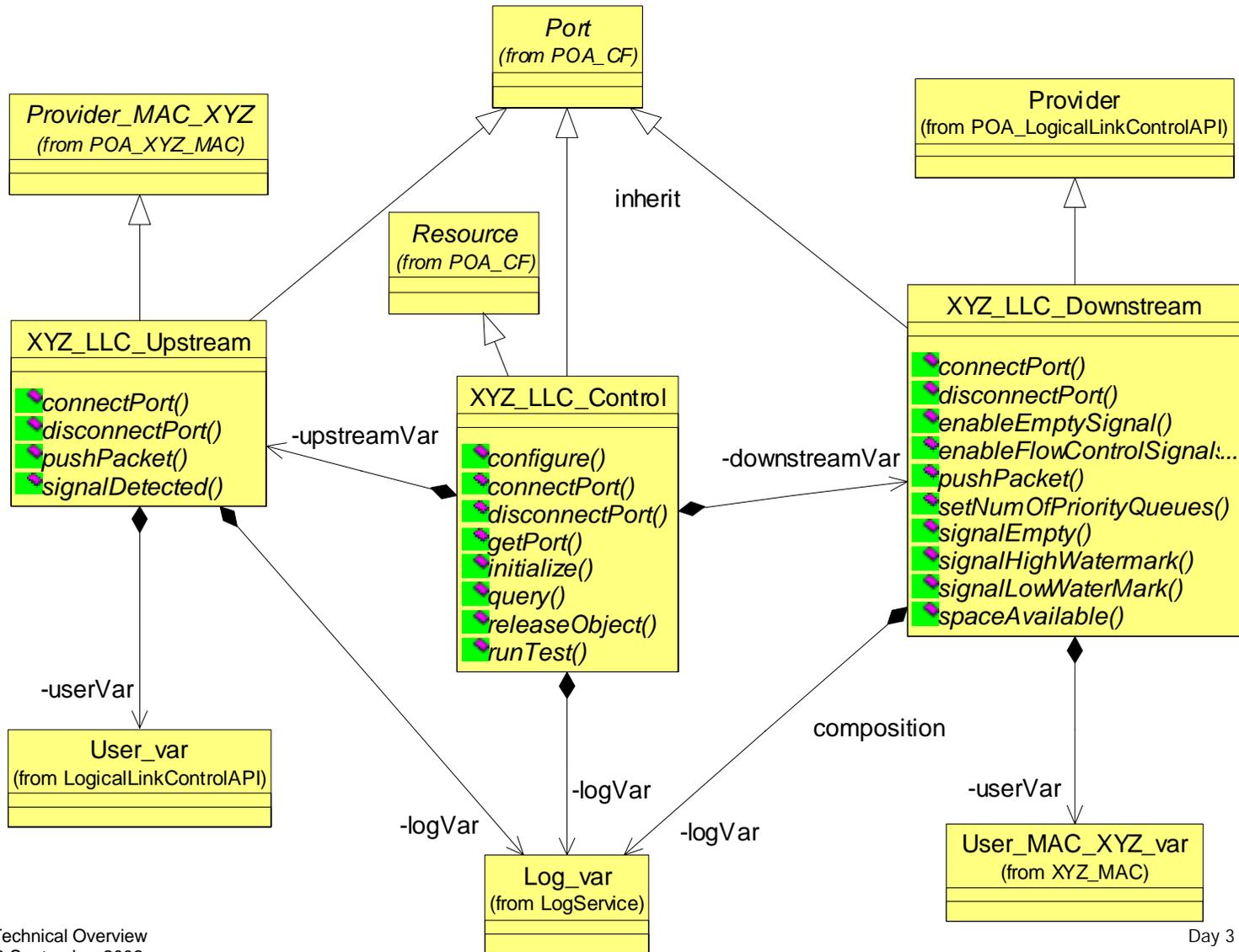
Unified Design for XYZ's Physical Layer



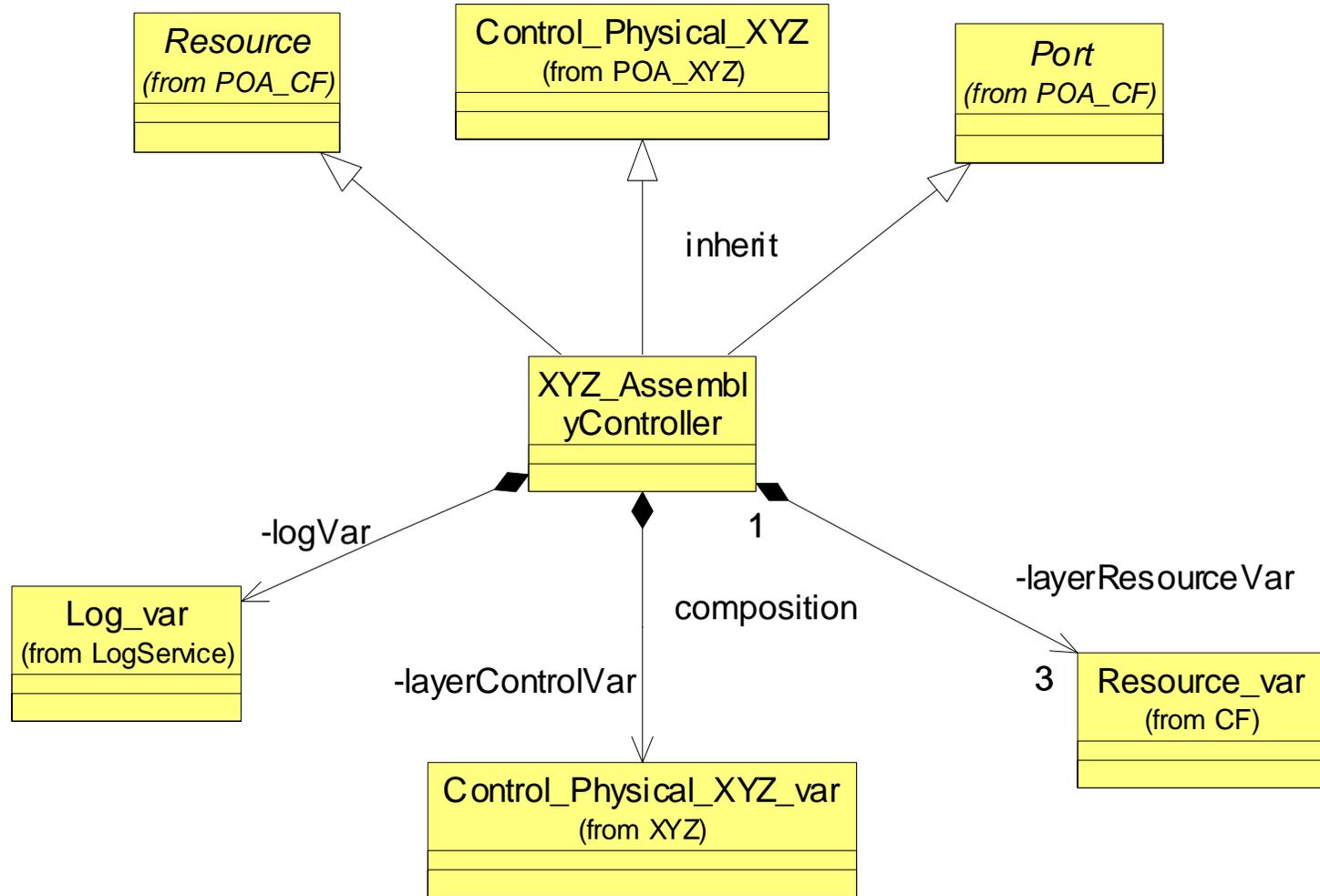
Partitioned Design for XYZ's MAC Layer



Partitioned Designed for XYZ's LLC Layer



Design for XYZ's Assembly Controller





Build Waveform Component Implementations

- Build Waveform UML Implementation model of waveform software { SCA Developer's Guide section 6.3 }
- Generate language-appropriate source files for servant and user software {SCA Developer's Guide section 5.7 }
- Complete Language Source Files Coding
- Create XML for each component {SCA Developer's Guide section 5.8 }
- Build User Interface {SCA Developer's Guide section 8 } (optional)

Generate Language Source Files

- Create the Source Files from the UML model.
 - source generation can happen from the component view or logical view
- UML Tool Additional Features
 - Browse the source code (header or body)
 - Reverse Engineer source code
 - Syntax check
 - Model Analysis
 - Configuration Management

Complete Language Source Files Coding

- Complete the source coding from the generated source stub files from UML Implementation Model.
 - User
 - Servant
- Compile and Unit Test.

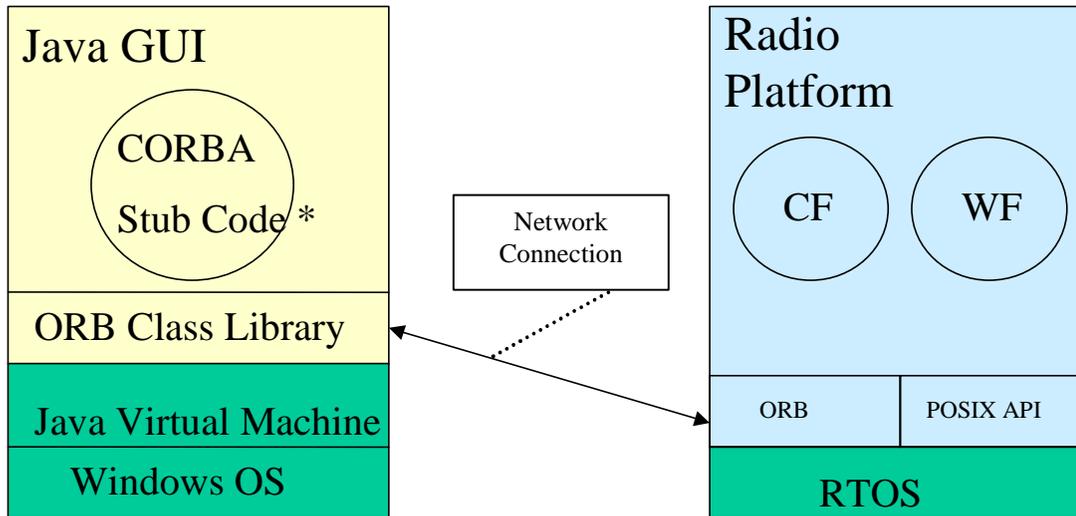
Create Waveform Component XML

- Software Package Descriptor (SPD)
 - Main server programs, shared libraries, or dynamic linkable code have an associated Software Package Descriptor (SPD).
 - Implementation Device Capacities/Resources needed based upon Simulation and testing results.
- Software Component Descriptor (SCD)
 - All components that uses and/or provides CORBA interfaces have an associated Software Component Descriptor.
- Properties Descriptor
 - Is optional for all software elements.
 - Can be defined at three different levels.
 - SCD – Component Definition Level
 - SPD – Additional implementation properties for all implementations.
 - SPD Implementation – Implementation properties specific to one implementation.

Build User Interface

- Direct GUI CORBA Link Interface
- Non-Direct GUI CORBA Link Interface

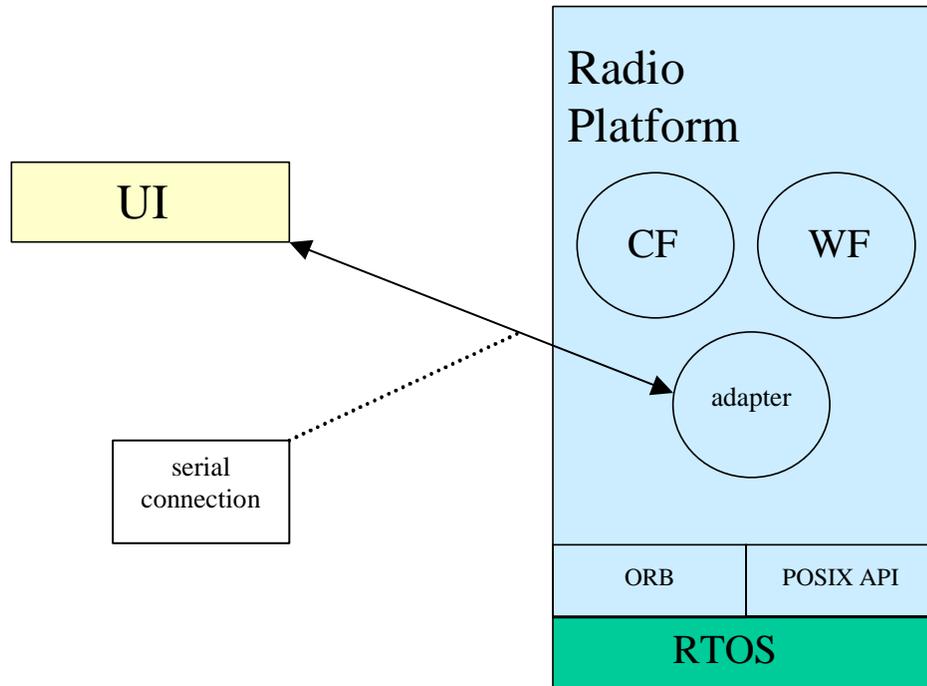
Direct GUI CORBA Link Example



* generated with idltojava compiler

- UI is on a CORBA Platform in this case Java
- Uses CF interfaces
- WF Specific Interfaces
- GUI performs CORBA calls using the appropriate APIs for the CF components and WF Resource components
- Network Connection supports IIOP type

Non-Direct GUI CORBA Link Example



- "adapter" software in the radio that translates between Non-CORBA messages to CORBA messages (CF or Waveform interfaces)
- Non-CORBA Messages can be over a serial connection, 1553 or UDP connection like SNMP.

Waveform Design Process

- Build Waveform Analysis Model
- Build Waveform Language Interfaces
- Build Waveform Component Implementations
- Integrate Waveform Components

Integrate Waveform Components

- Integrate software and hardware
 - Create Software Assembly Descriptor (SAD)
 - Incrementally integrate the waveform components
 - Integration
 - Emulation of Target Hardware – by Developers
 - Deployment of real Target Environment – by System Integrators or Hardware Manufactures.
 - Note: If Service Definitions are not adequately stated then this will lead to longer integration times.
- Test resultant application
 - Make Device Capacity/Resource Adjustments
 - Update SPD files as appropriately
 - Update Waveform Simulation Model as appropriately
 - Compliant testing will be discuss on Day 5.

Waveform Design Summary

- Waveform Development Process
 - A Refinement of Models
 - Service Definitions
 - Waveform Definition
 - Language Specific Interfaces
 - Implementation
 - Reuse of Standardize Service Definitions at all Process Activities
 - Many Different types of Waveform Artifacts
 - Simulation Model
 - UML Models
 - Implementation Files
 - SCA XML Files
 - VHDL
 - DSP
 - CORBA Interfaces
 - Waveform Component Implementations
 - UML Tool can be used create and manage the waveform artifacts

Device and DeviceManager Design



Device and DeviceManager Design

- *Device* Implementations
- *DeviceManager* Implementations
- *Device* Usage and Design Examples

Device Implementations

- Has associated XML files
 - Software Package Descriptor
 - Software Component Descriptor
 - Properties (mandatory)
 - At a minimal device's capacity/resource artifacts.
- States
 - Usage State is implementation dependent and tied to its capacity model
 - Operational State is implementation dependent
 - Administrative State behaves the same for all implementations
- A Logical Device can be implemented as single Device or as a composite Device.
- Devices are usually associated with a device driver.
- Can make any operating system call, therefore code may not be portable.
- Manufacturer or System Integrator supplies Device Package Descriptor associated with Logical Device.

Device Implementations, cont'd

- Capacities Artifacts
 - Visible Types
 - These artifacts are used to describe the characteristics of the device in order to find the right type of device to use or to be deployed on.
 - These capacity/resource artifacts are like constant characteristics of the device and do not affect the states of the Device.
 - The predefined ones in the SCA are:
 - OS Name
 - Processor Name
 - Allocation Types
 - These capacity types are managed by the Device and are used by the Device's Capacity Model, which effects the usage state for a Device that indicates its availability.
 - The capacities are also used to find the right type of device that has this set of capacities.

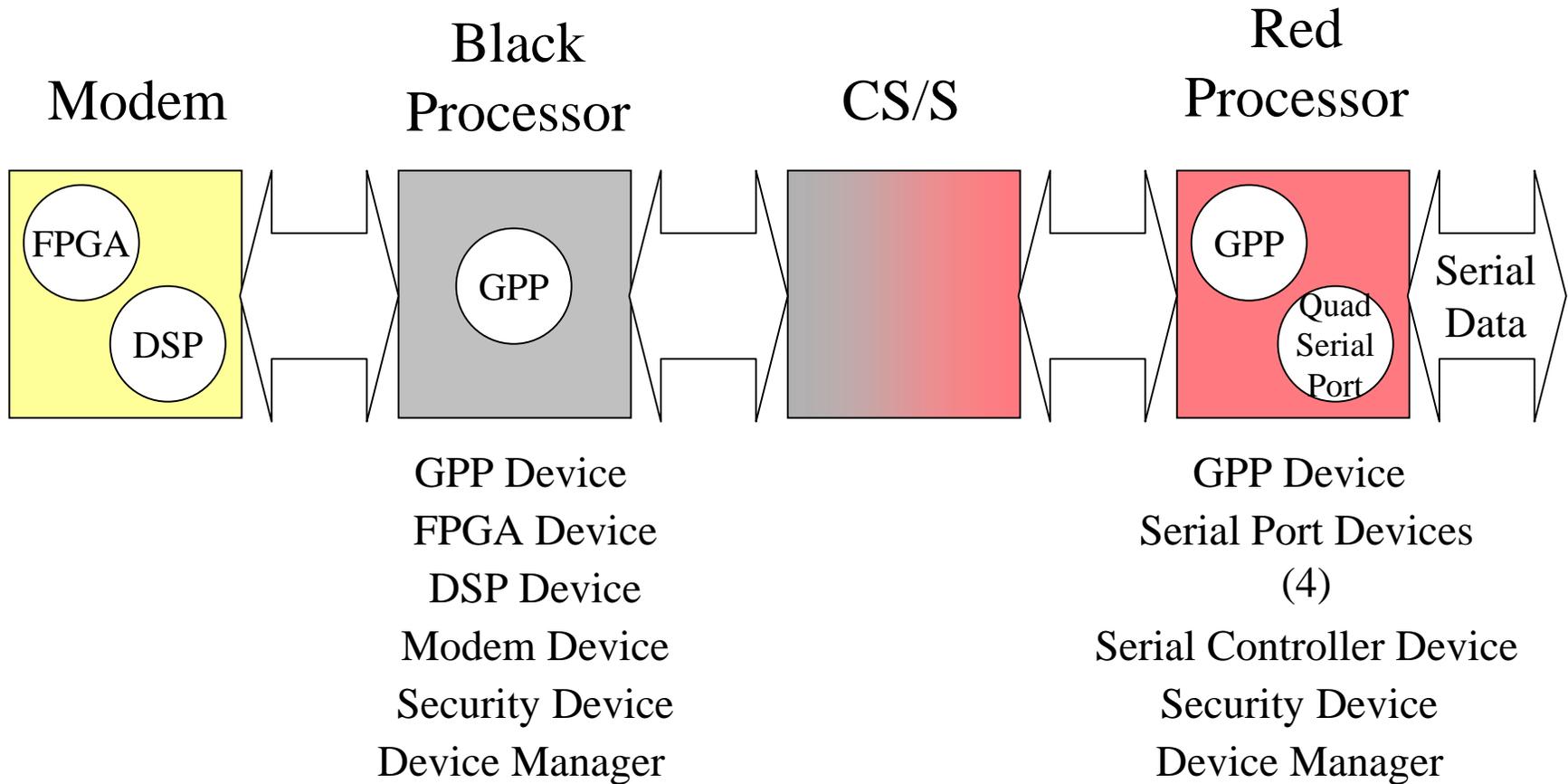
DeviceManager Implementation

- Device Configuration Descriptor
 - Provides the mechanism to startup 3rd party logical Devices on a node.
 - Provides the mechanism to startup services on a node.
 - Provides the mechanism to create *CF::FileSystem(s)*
 - Associations between logical Devices and Device Package Descriptors.
- Services started up on a node depends on the System Design and hardware platform

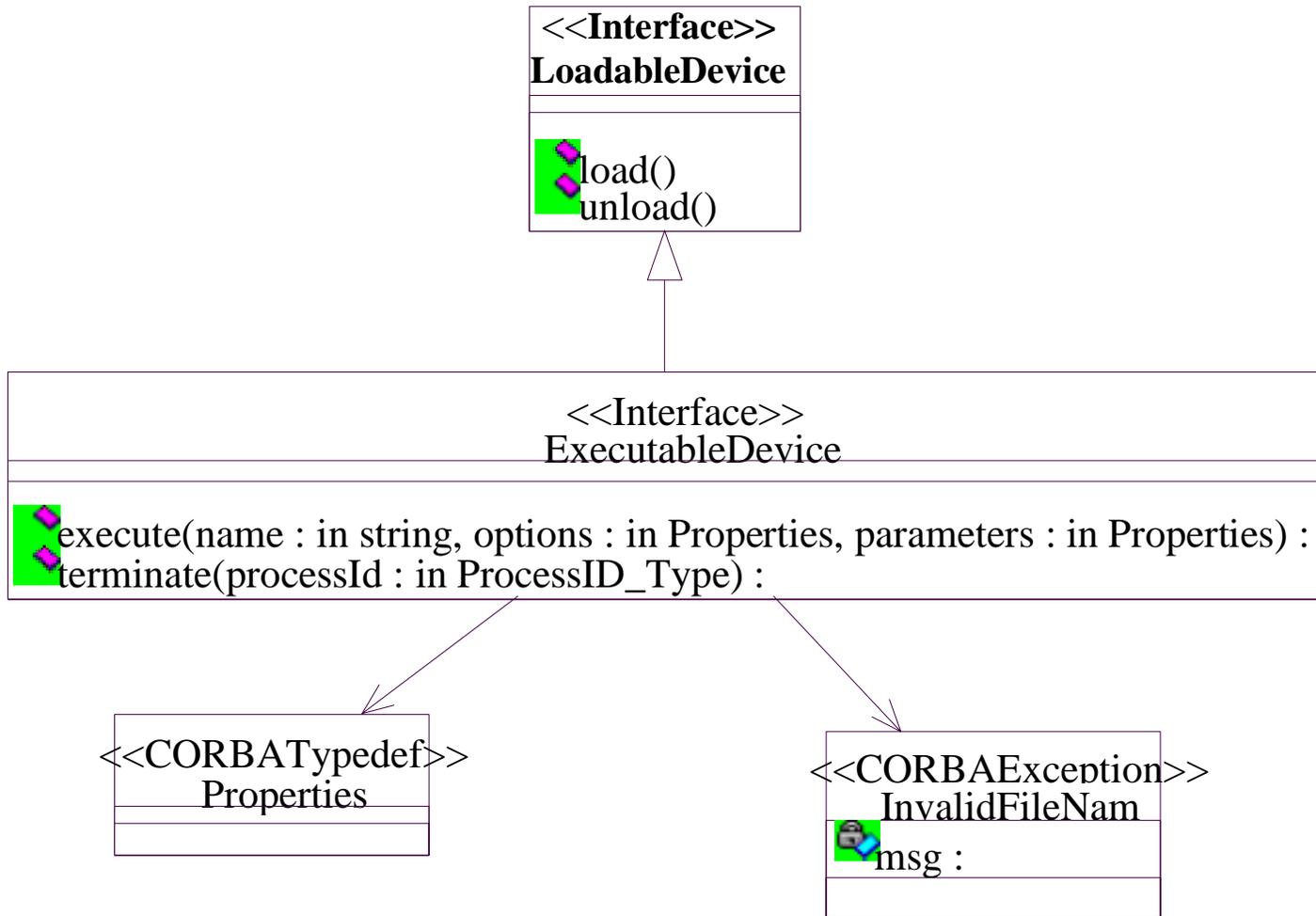
Device Usage and Design Examples

- Base *Device*
 - Audio
 - Serial Data
 - Security
- *LoadableDevice*
 - FPGA, Modem
- *ExecutableDevice*
 - GPP
 - DSP
- *AggregateDevice*
 - Multi-channel
 - Multi-function

Simplified Example



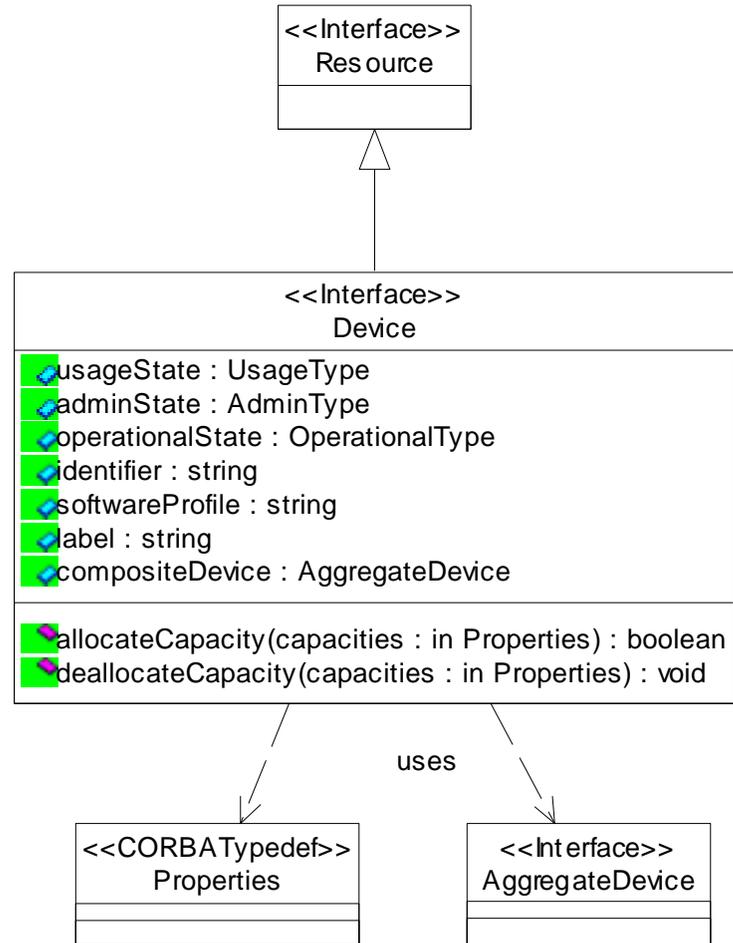
GPP Device (Red and Black)



GPP Device (Red and Black)

- Executable Device Interface used to manage state, capacity and loading and executing of software on GPP
- Load and Execute Implementations can be different based on OS
 - POSIX multi-process (e.g. Linux, LynxOS, QNX)
 - Single Process (e.g. VxWorks 5.4, DSP BIOS II)
 - Protection Domains (VxWorks AE)

Serial Port Device



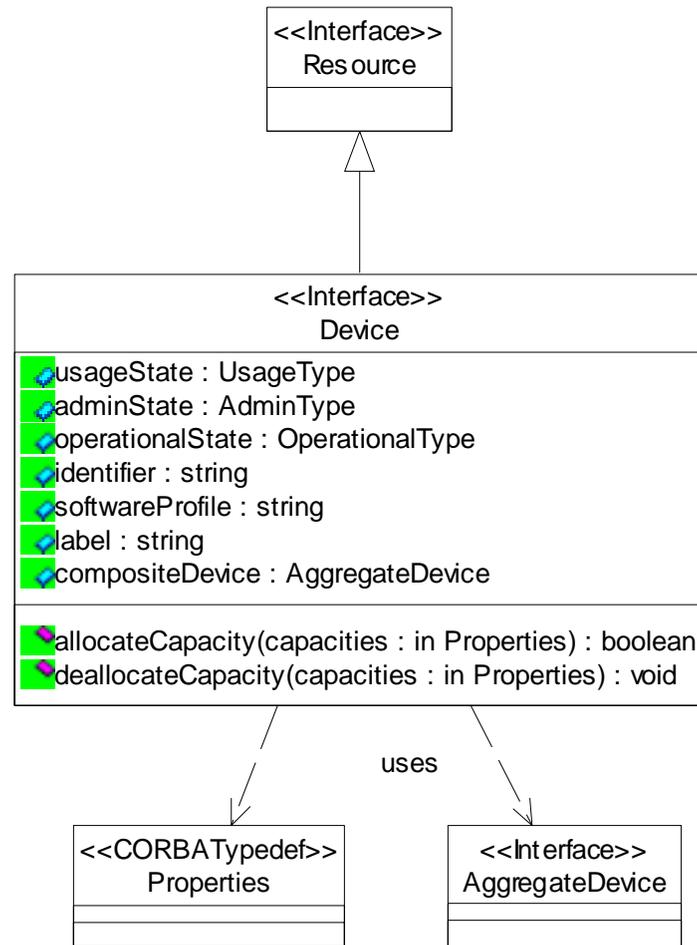
Serial Port Device

- Device interface used to manage state and device configuration
 - Configuration Properties
 - Asynchronous Mode
 - # Bits/character
 - # Stop bits
 - Parity
 - Flow Control (HW, XON/XOFF, None)
 - Baud Rate
 - Synchronous Mode
 - Rx Clock Source
 - Tx Clock Source
 - Baud Rate

Serial Port Device

- Ports used for operational device controls, data flow
 - PTT
 - RTS
 - CTS
- Use Building Blocks to define APIs at ports
 - Use same process as Waveform API development process
 - Define UML, IDL
 - Create Service Definitions
 - I/O Building Blocks
 - Packet Building Blocks

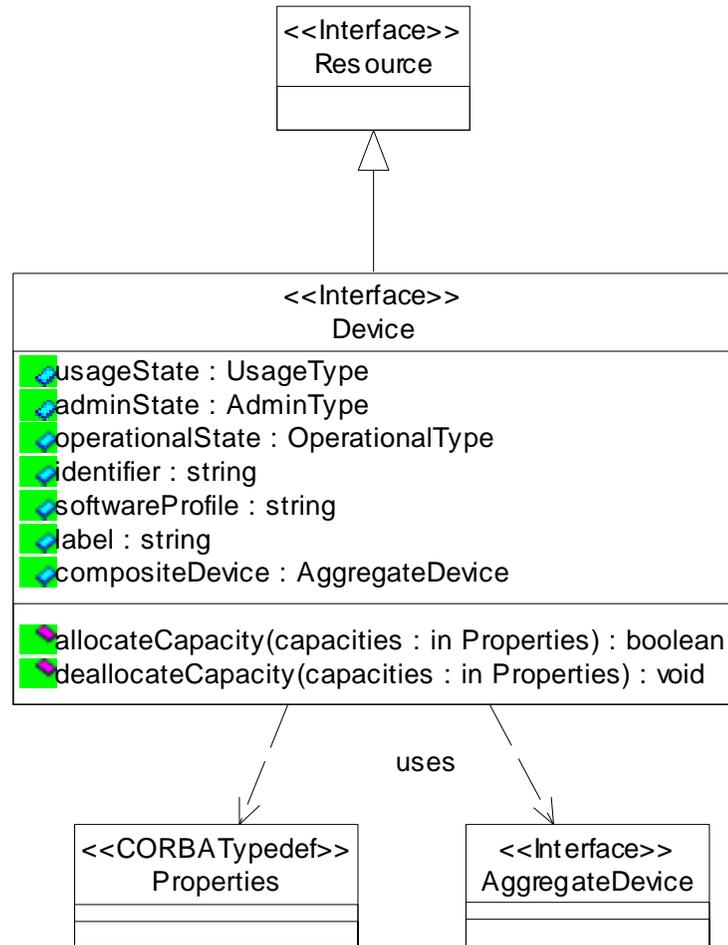
Serial Communications Controller Device



Serial Communications Controller Device

- Aggregate Device
 - Aggregates 4 Serial Port Devices under one umbrella
 - Common management functions handled here
 - Manage Capacity

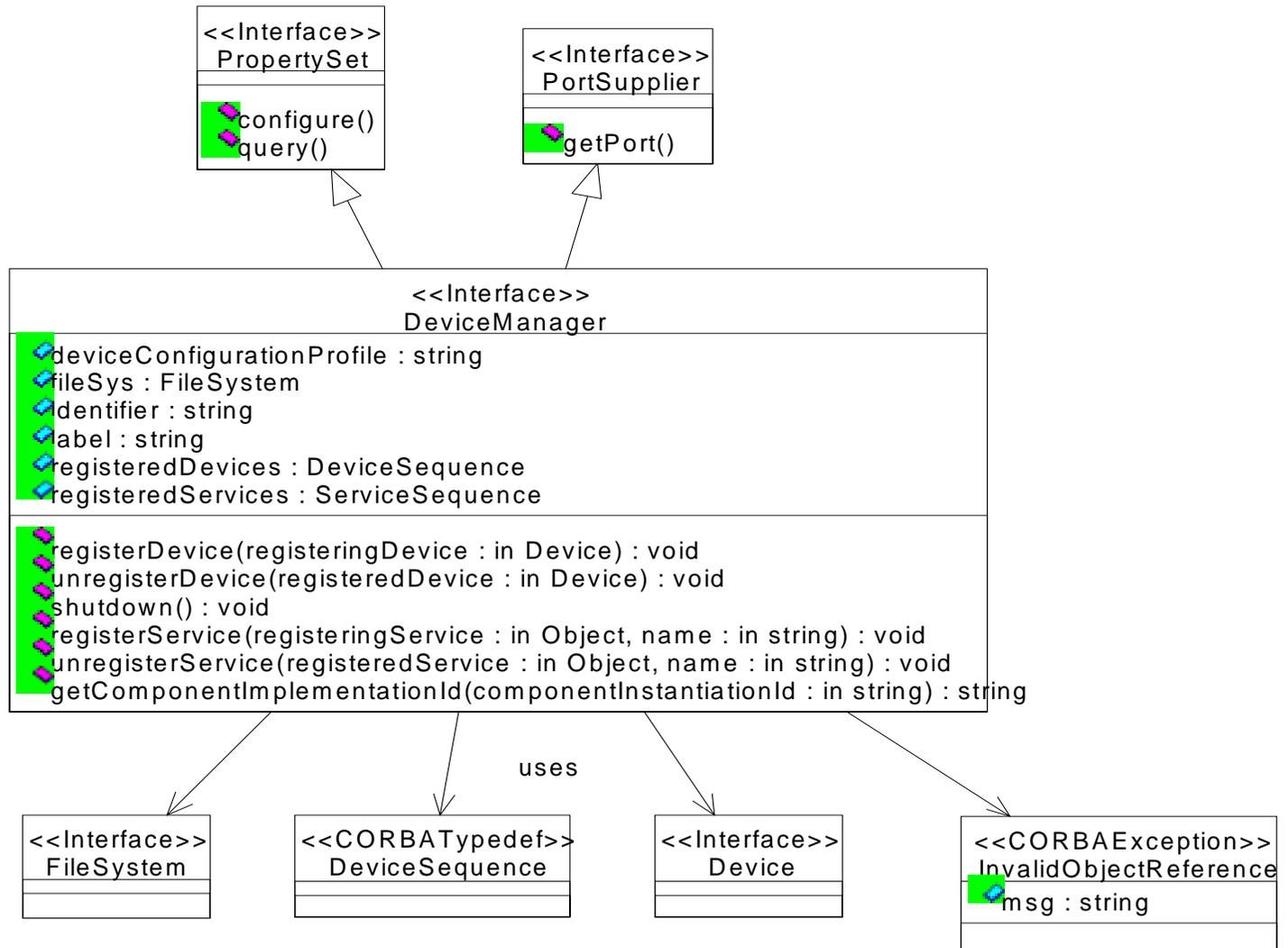
Security Device (Red)



Security Device (Red)

- Assuming Black Side Control
 - Uses ports and provides ports implement Encrypt/Decrypt Interface from Security API
 - Performs no CS/S management function
 - Likely to implement test functionality

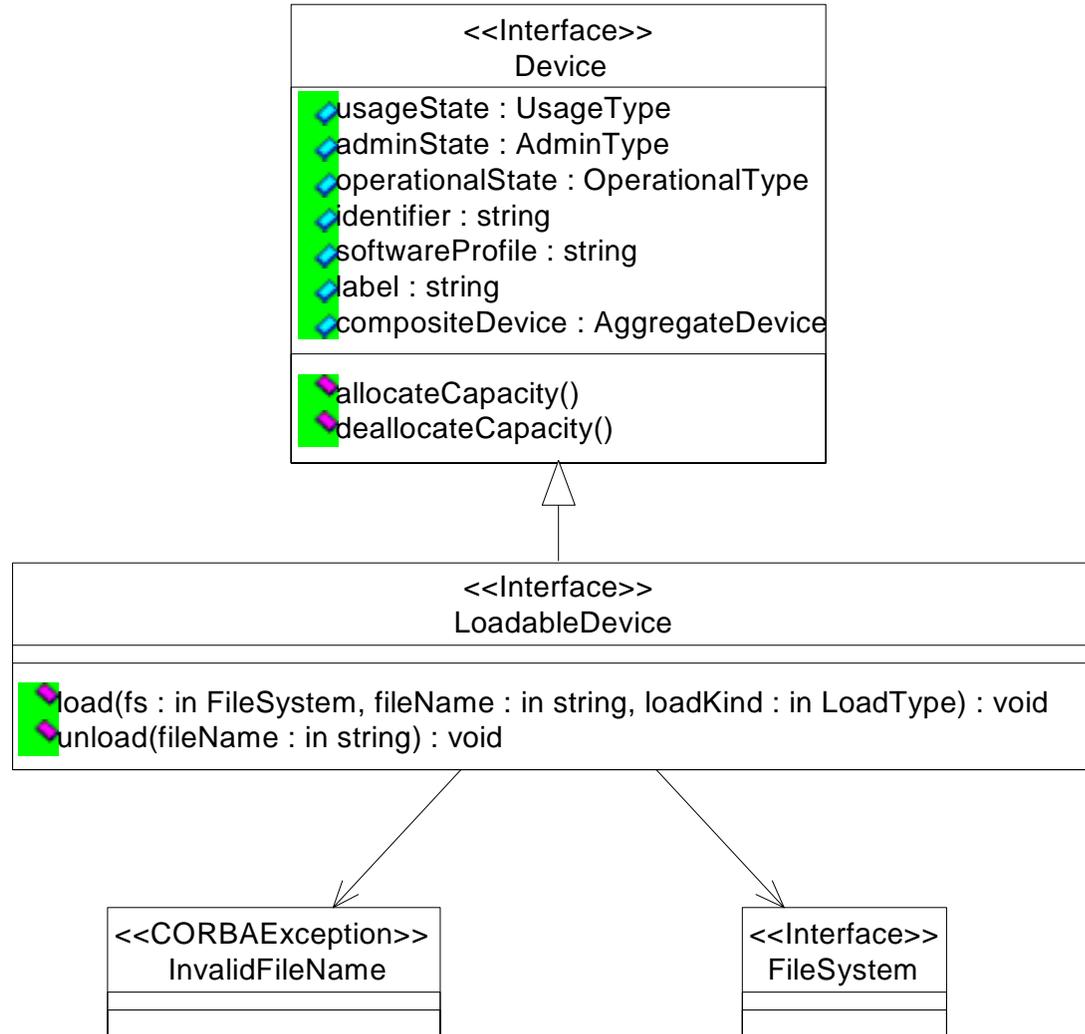
DeviceManager (Red)



DeviceManager (Red)

- Registered Devices
 - 1 Red GPP Device
 - 1 Serial Controller Device (communications controller chip)
 - 4 Serial Port Devices
 - 1 Security Device
- Registered Services
 - Event Service
 - Log Service

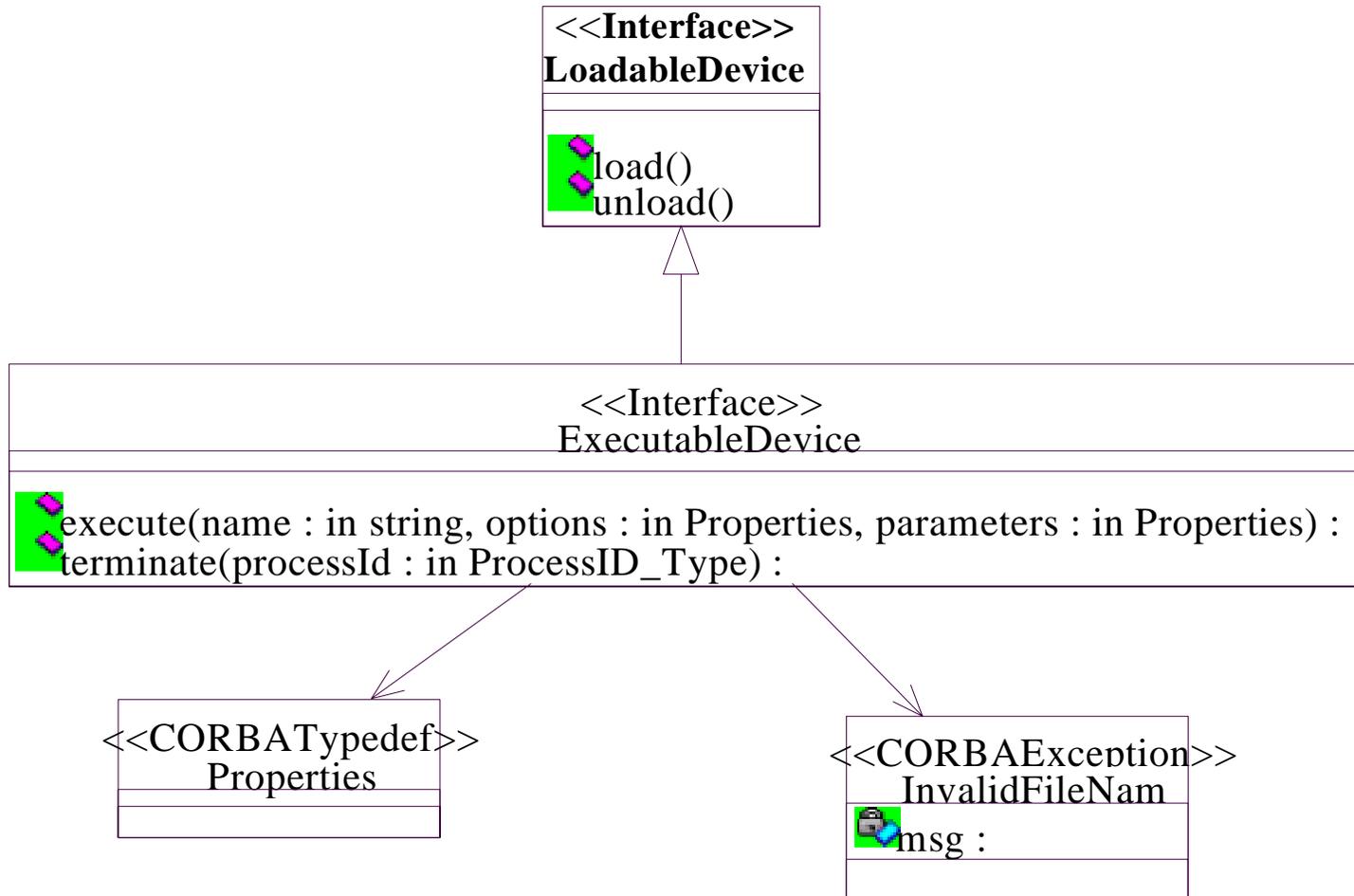
FPGA Device



FPGA Device

- Proxy for FPGA
- Resident on Black GPP
- Communicates with FPGA directly or via DSP
- Implements load protocol
- May or may not implement ports
- If ports implemented:
 - Data transfer API built using Packet Building Blocks

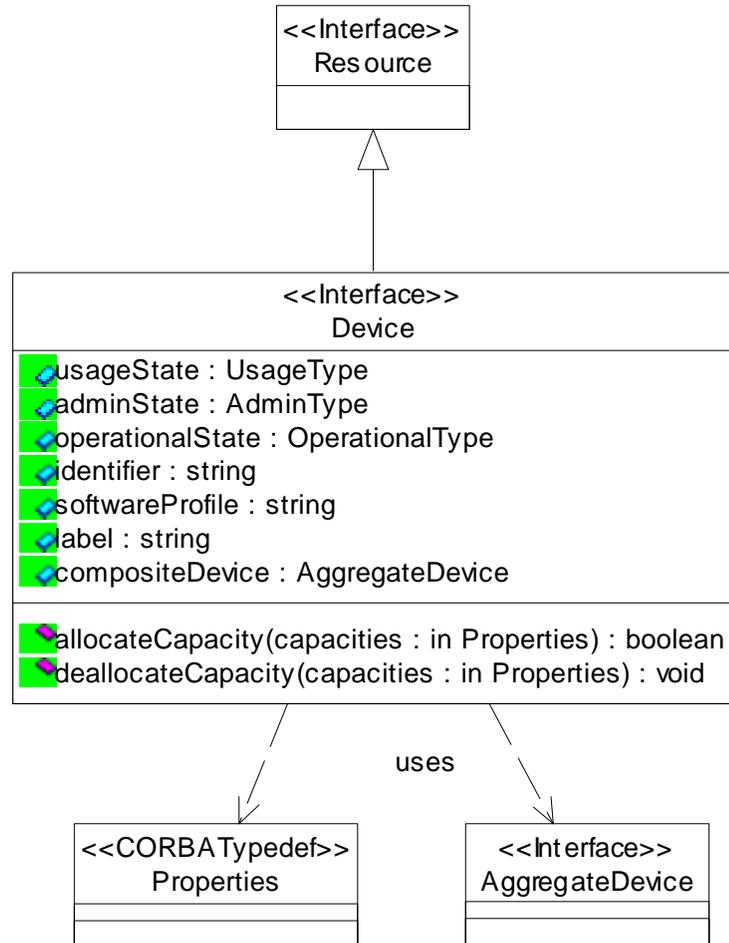
DSP Device



DSP Device

- Proxy for DSP
- Resident on Black GPP
- Communicates directly with DSP
- Implements load protocol
- Ports implemented:
 - Data transfer API built using Packet Building Blocks

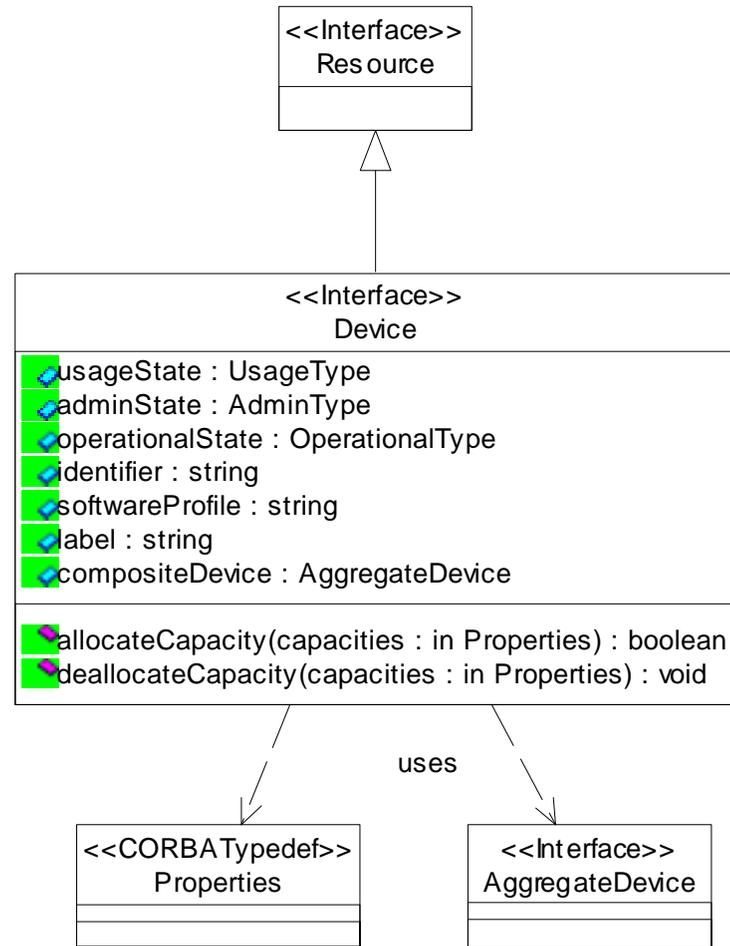
Modem Device



Modem Device

- Aggregate Device
 - Aggregates FPGA and DSP Devices under one umbrella
 - Common management functions handled here
 - Manage Channel Capacity

Security Device (Black)



Security Device (Black)

- Assuming Black Side Control
 - Manages state and capacity via Device Interface
 - Control Ports implement Security APIs for:
 - Fill
 - Key Mgmt
 - Policy Mgmt
 - TRANSEC Mgmt
 - Certificate Mgmt
 - Crypto Control (Channel Instantiation and Teardown)
 - I&A
 - Data ports implement Security APIs for:
 - Encrypt/Decrypt
 - Keystream Generation

DeviceManager (Black)

- Registered Devices
 - 1 Black GPP Device
 - 1 Modem Device
 - 1 FPGA Device
 - 1 DSP Device
 - 1 Security Device
- Registered Services
 - Event Service
 - Log Service

Device Design Summary

- Device Development Process
 - Similar Process as Waveform Development
 - Service Definitions
 - Device Definition
 - Language Specific Interfaces
 - Implementation
 - Reuse of CF Device, I/O and Physical Service Definitions at all Process Activities
 - Many Different types of Device Artifacts
 - Simulation Model
 - UML Models
 - Implementation Files
 - SCA XML Files
 - Device Specific Driver
 - CORBA Interfaces
 - Device Component Implementations
 - UML Tool can be used create and manage the Device artifacts

Open Discussion