

SCA Training for Developers and Testers



Day 2: Security and Waveform API Supplements

Copyright © 2002, Raytheon Company.
All Rights Reserved

Day 2 AGENDA

- Security Supplement
 - Security Supplement Overview
 - Security APIs
- API Supplement
 - API Overview
 - SCA APIs
 - Building Blocks
 - API Creation
 - API Summary



Security Supplement Overview



SCA Security Supplement

- Defines security requirements for JTRS
- Identifies behavioral impacts of these requirements
- Defines an API set to support the requirements
- Uses Common Criteria Evaluation Assurance Levels (EAL) used as a basis for requirements definition
- Written for SECRET System-High (EAL-3) operation but does not prohibit higher levels of assurance

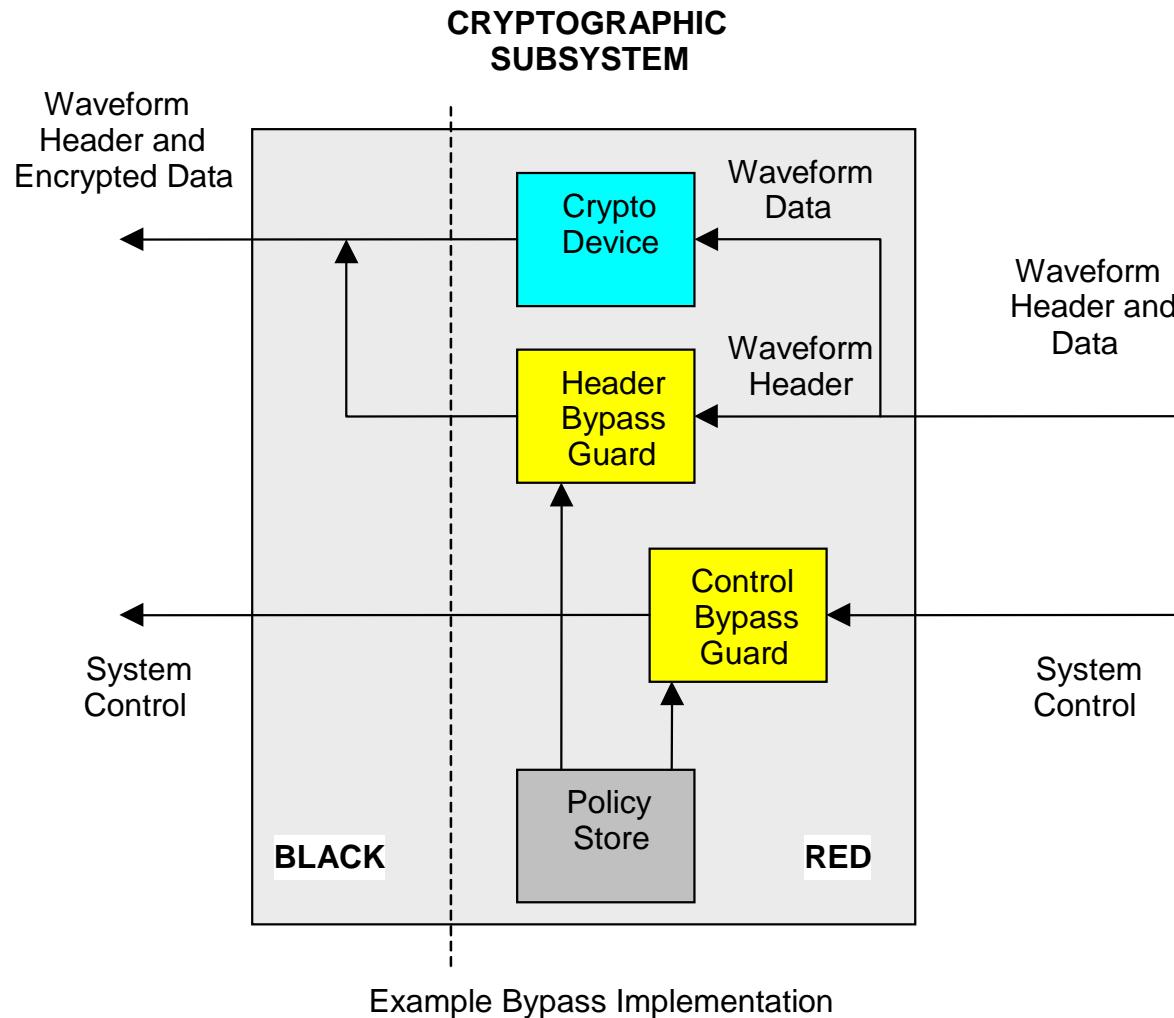
Security Requirements

- Requirements are broken down into the following elements
 - Cryptographic Subsystem (CS/S)
 - INFOSEC Boundary
 - Equipment Level Boundary
 - JTRS Security Policy

CS/S Requirements Breakdown

- General Requirements
- Initialization, Operation and Termination
- Keystream Functions
- Identification and Authentication Functions
- Integrity Function
- Security Management Function
- Cryptographic Bypass
 - Gets special attention in this presentation
- Cryptographic Status and Control Functions

Bypass



SCA Impacts on Bypass

- The distributed component nature of the SCA, using CORBA, requires Core Framework control / status messaging between red and black sides
- CORBA messages consist of "marshaled" data embedded in a transport protocol (i.e. not in a readable structure when not in application space), making security checking difficult
- SCA-compliant Embedded INFOSEC solutions are achievable but will require new approaches and innovative systems engineering to solve

Bypass

- Bypass enforcement mechanisms (guards) are parameterized by security policies
 - Type
 - Size
 - Range of values
 - Frequency
- Control/Status Bypass
 - Control/Status Bypass has no specific API
 - Platform and Waveform specific security policies define the parameters of the control/status bypass guard
- Header Bypass
 - Encrypt / Decrypt APIs for Red-Black / Black-Red support bypass of a header
 - Waveform-specific security policies define the parameters of the header bypass guard

INFOSEC Boundary Requirements Breakdown

- Process Separation
- Discretionary Access Control
- Identification and Authentication
- Object Reuse
- System Integrity
- Core Systems Applications
 - Security API
 - HMI Security Policy Enforcement
 - Installer
 - Audit
 - Crossbanding

Equipment Level Boundary Requirements Breakdown

- TEMPEST
- Tamper
- Electrical Protection



Behavioral Impacts Breakdown

- Key Selection
- Software Installation
- Power-up Sequence
- Instantiation
- Operating System

Security APIs

Purpose of Security APIs

- Provide interfaces which support the security architecture and requirements defined in the Security Supplement
 - Interfaces to the cryptographic subsystem
 - Support security policy enforcement
- Portability of Applications
 - Security Aware application portability

Security API Organization

- JTRS Security Services are partitioned into groups
- Each group corresponds to a CORBA Module (IDL)
- Each interface with a CORBA module represents a service
- Each operation within an interface represents a service primitive
- This convention provides consistent naming and avoids global name space pollution

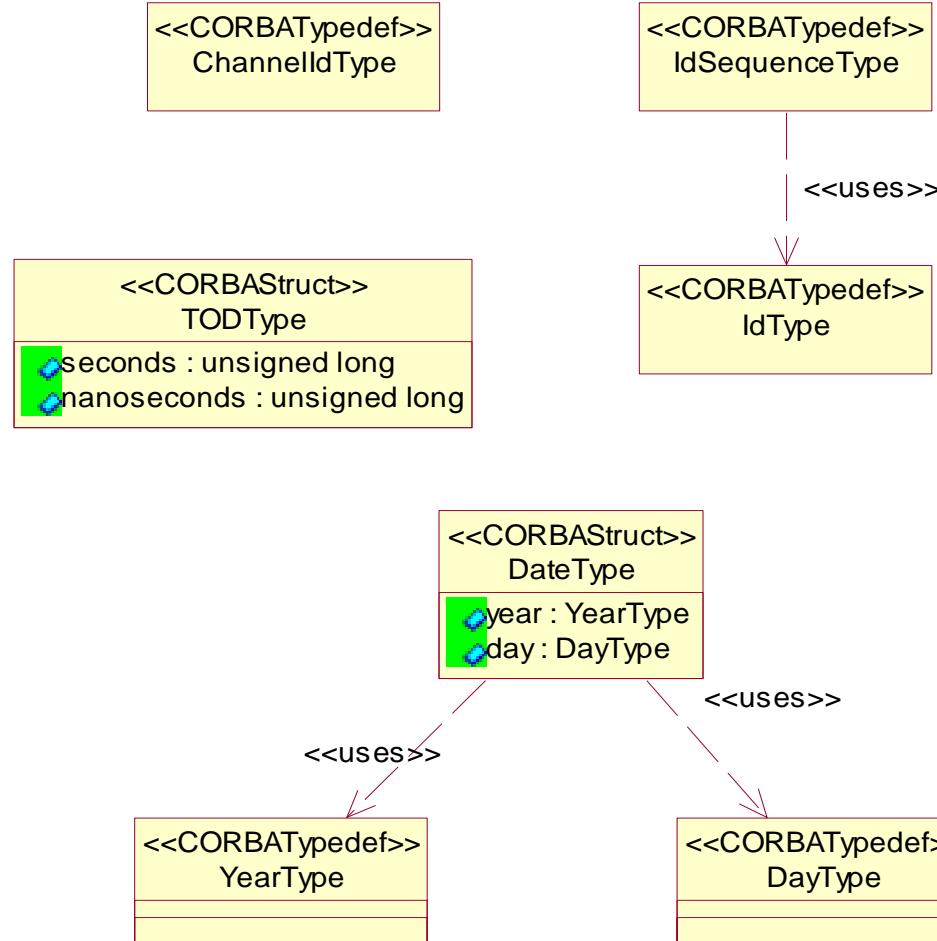
JTRS Security API Organization



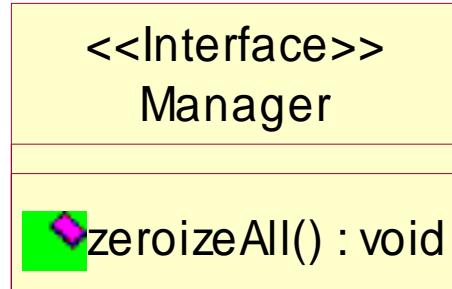
Security APIs and the CF

- Security APIs exist at CF ports
 - Static elements such as managers and controllers are implemented within a *CF::Device* responsible for security management
 - Dynamically instantiated elements such as those implementing the Encrypt/Decrypt interfaces may be implemented within *CF::Devices* responsible for data transfer
- Security components are connected with waveform components by the *ApplicationFactory*
 - The components and connections are described in the SCD and SAD
- Security components are connected with CF and other platform services via the *DomainManager*

Basic Types



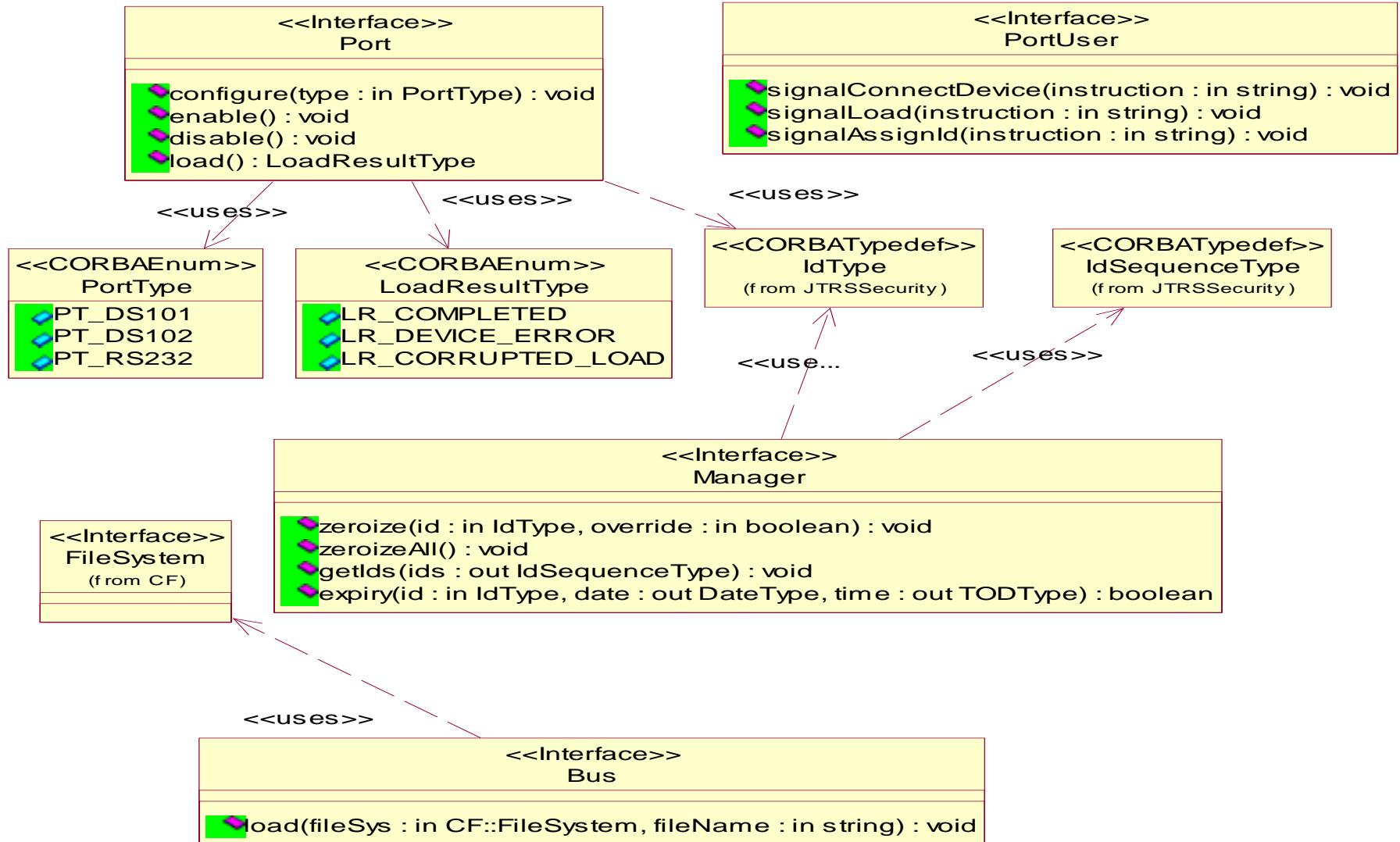
JTRSSecurity (Management Service)



- Provides global zeroization

- Port Service
 - Provides control over fill port
 - Supports DS-101, DS-102 and serial port fills
- Port User Service
 - User (HMI) must implement this interface
 - Primarily to support DS-102 which requires manual intervention to enter DS-100-1 tag information.
- Bus Service
 - Supports fill over a network (Black fill)
- Management Service
 - Interface is not implemented itself. It is inherited by other services
 - Defines generic operations to manage fill data

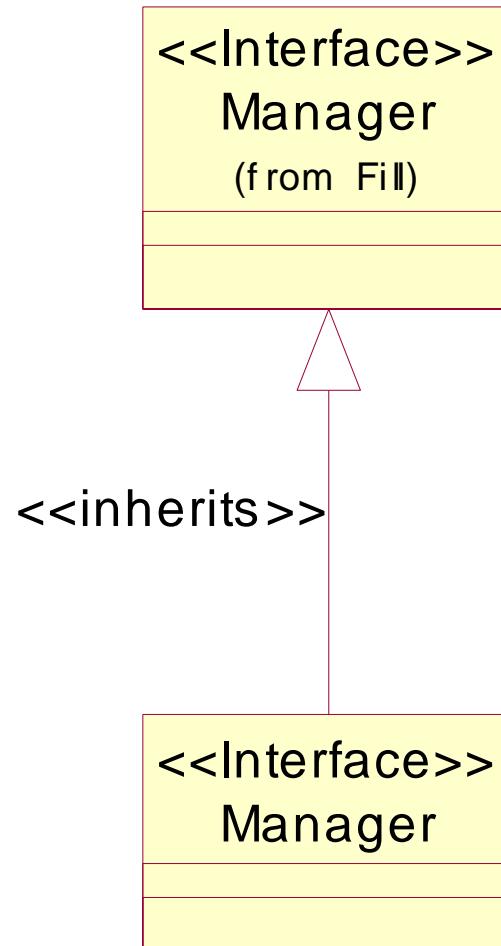
Fill (cont'd)



Algorithm

- Management Service
 - Algorithms include both COMSEC and TRANSEC and are identified as such by associated tags
 - Algorithm identifiers used by the management are the ones supplied with the algorithm at fill time
 - Algorithms only require the basic fill management service

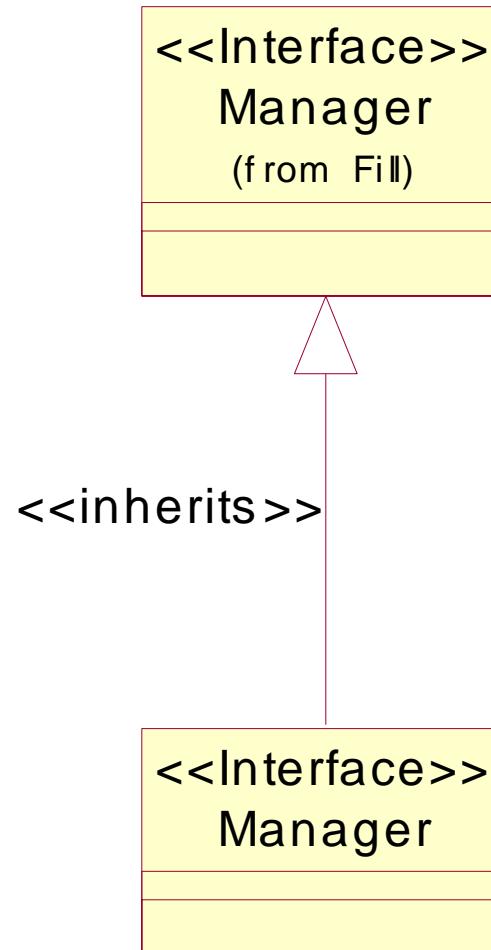
Algorithm (cont'd)



Certificate

- Management Service
 - Certificates are used for Integrity and Authentication of files and blocks of data
 - Certificate identifiers used by the management service are the ones supplied with the algorithm at fill time
 - Format of the certificates has not been defined (X.509 is a candidate)
 - Certificates only require the basic fill management service

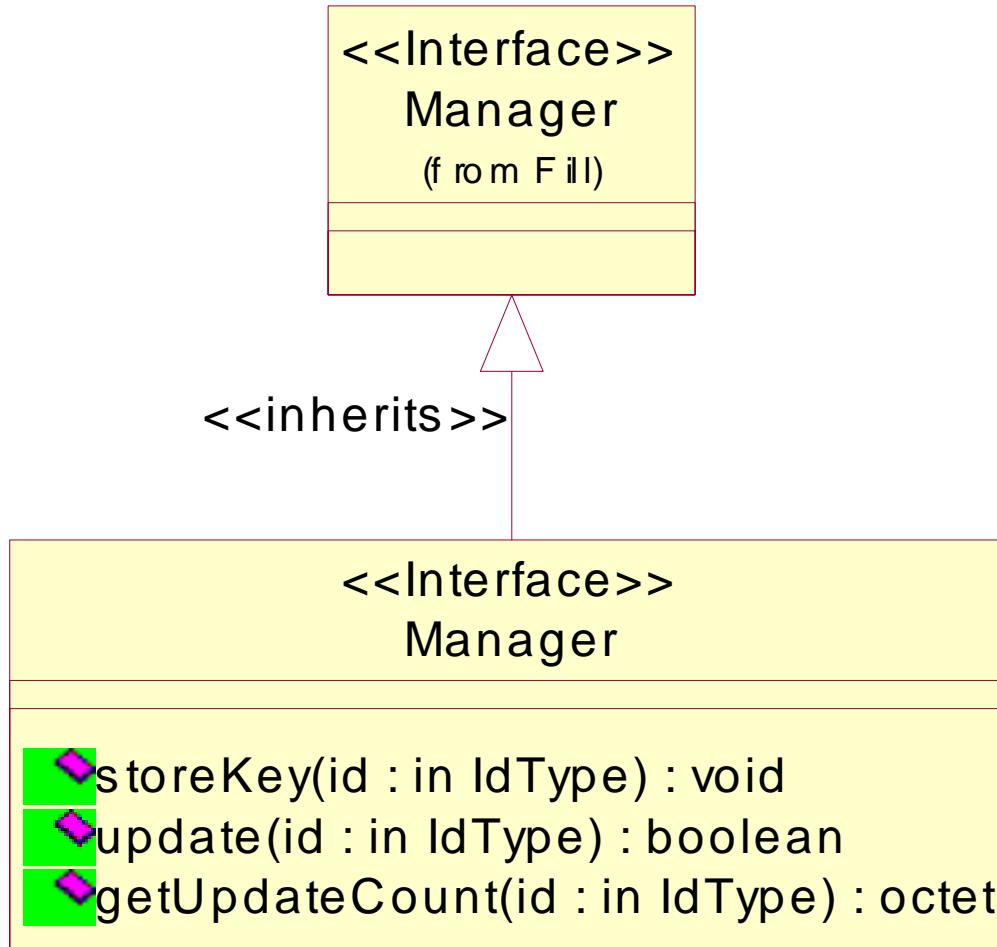
Certificate (cont'd)



Key

-
- Management Service
 - Keys include both COMSEC and TRANSEC and are identified as such by associated tags
 - Key identifiers used by the management are the ones supplied with the key at fill time
 - Supports both Red and Black keys
 - Supports key updates
 - Keys require an extension of the basic fill management service

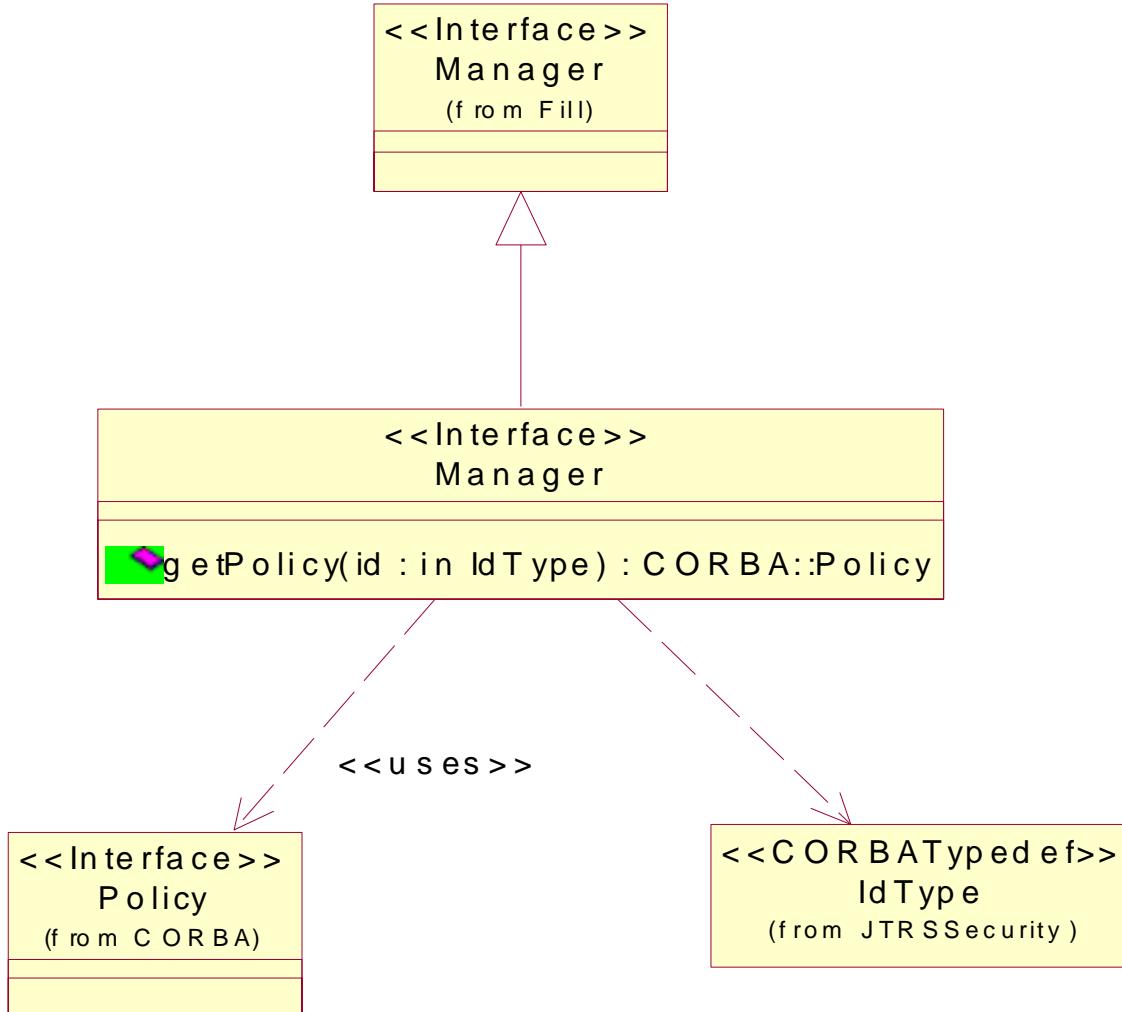
Key (cont'd)



Policy

- Security Policies are used to identify what needs to be protected.
- Information contained in the security policies is used to parameterize policy enforcement mechanisms
- Policies are distributed in XML format
 - The detailed format remains undefined at this time
- Management Service
 - The Policy Management Service requires an extension of the basic fill management service
 - Policy identifiers used by the management service are the ones supplied with the policy at fill time

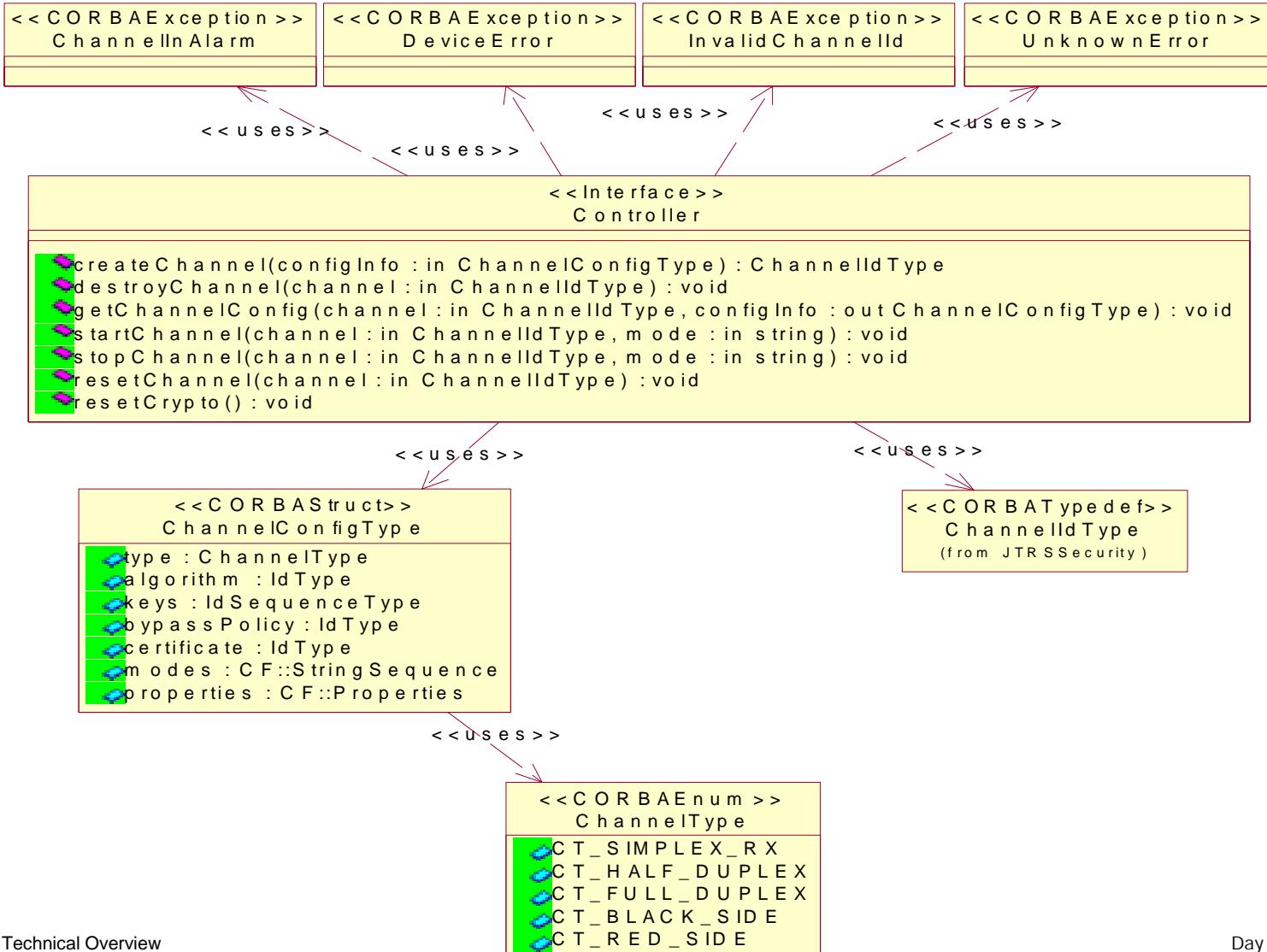
Policy (cont'd)



Crypto - Control

- Control Service
 - Provides interface for creating, destroying and controlling COMSEC channels
 - Channel creation operation provides support for modes and straps
 - Firefly exchange supported
 - Supports exceptions for algorithm - key mismatches, assurance level violations, key expiration, etc.

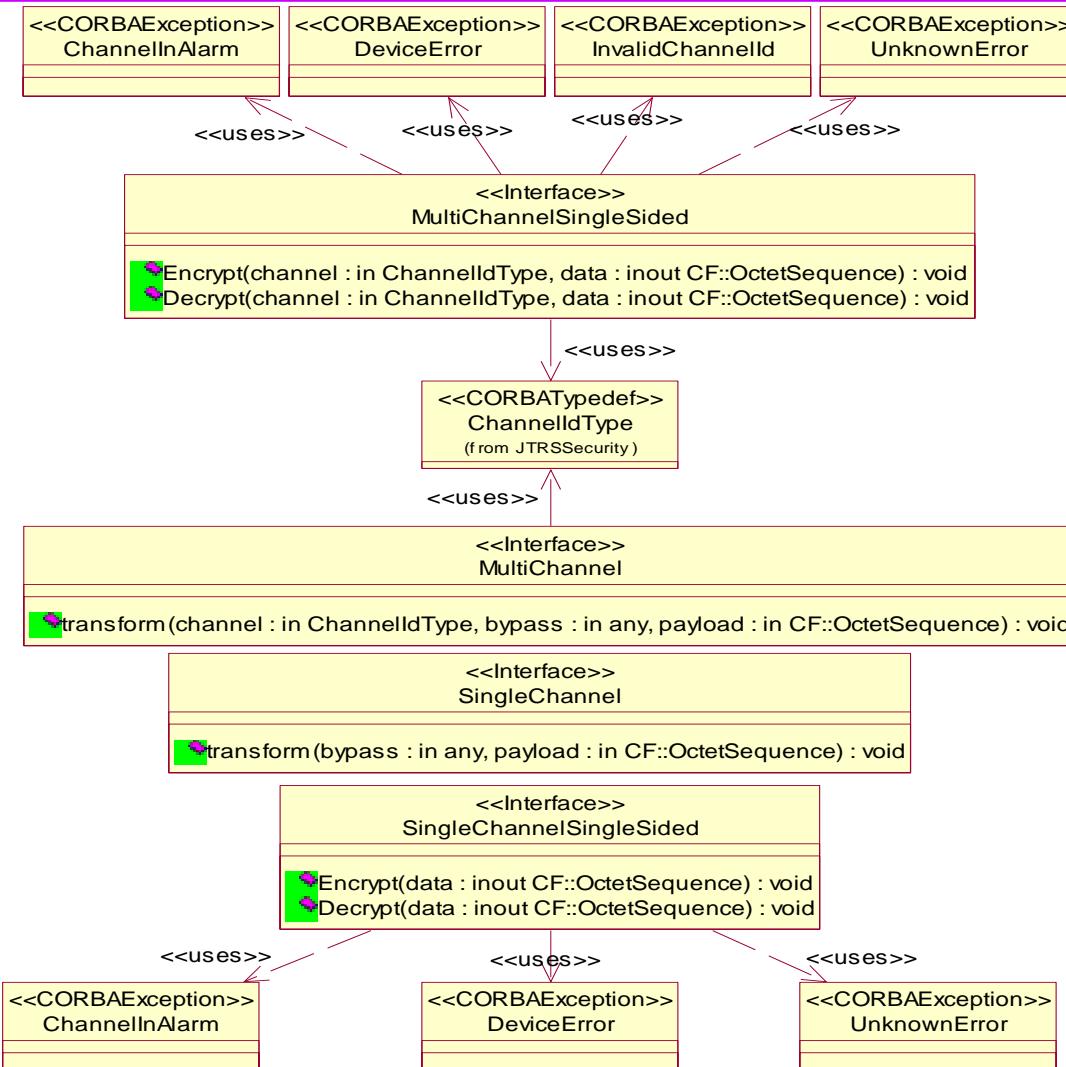
Crypto - Control (cont'd)



Crypto - Encrypt/Decrypt

- Encrypt/Decrypt Service
 - Provides the interfaces to the CS/S for encryption and decryption
 - Supports red-red, black-black, red-black, and black-red
 - Supports multi-channel and single channel interfaces

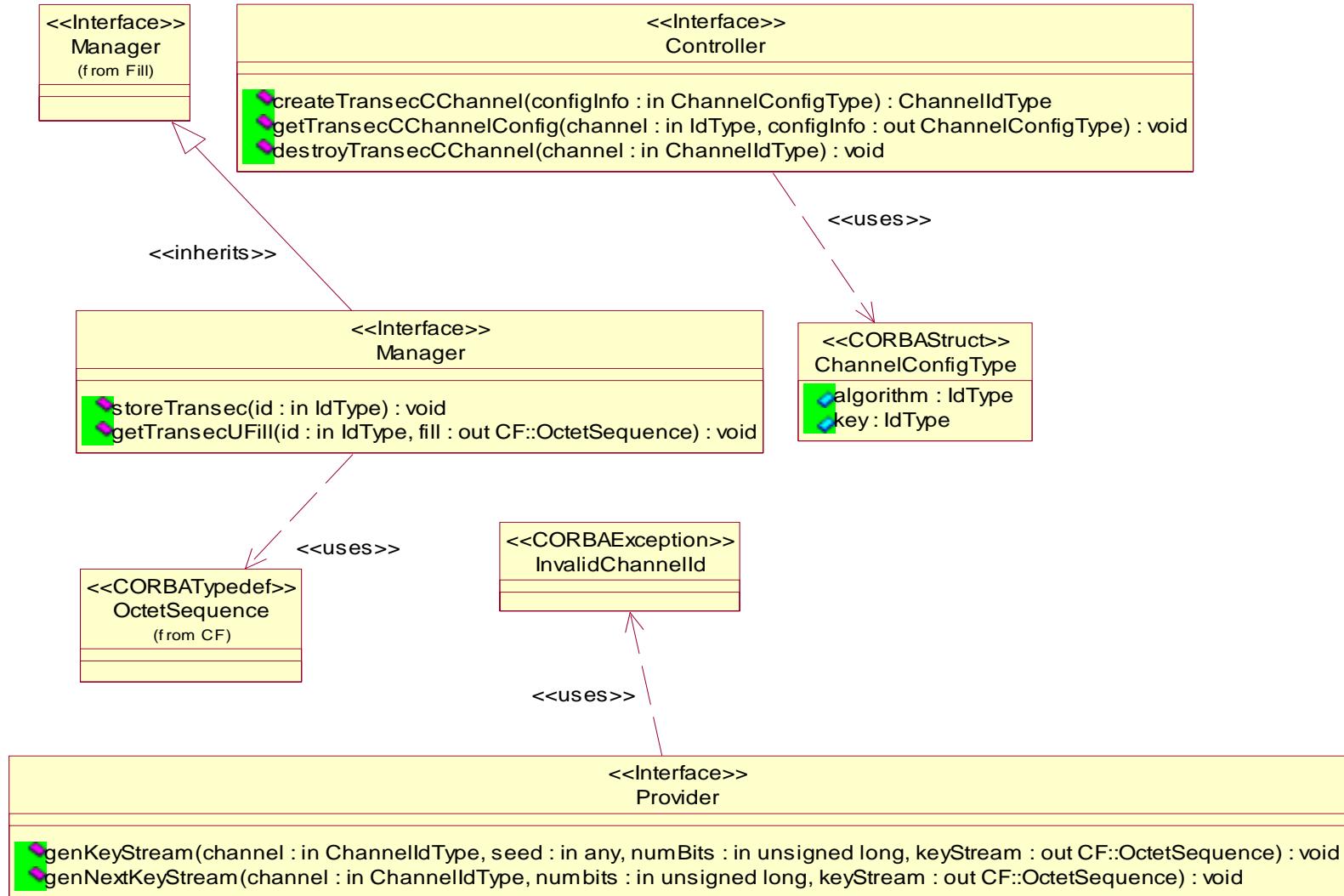
Crypto - Encrypt/Decrypt (cont'd)



TRANSEC

-
- Management Service
 - Supports unclassified TRANSEC
 - Extends basic fill management service
 - Control Service
 - Creates and destroys classified TRANSEC channels
 - Key Stream Service
 - Is the interface waveform applications use to request classified key streams generated within the CS/S

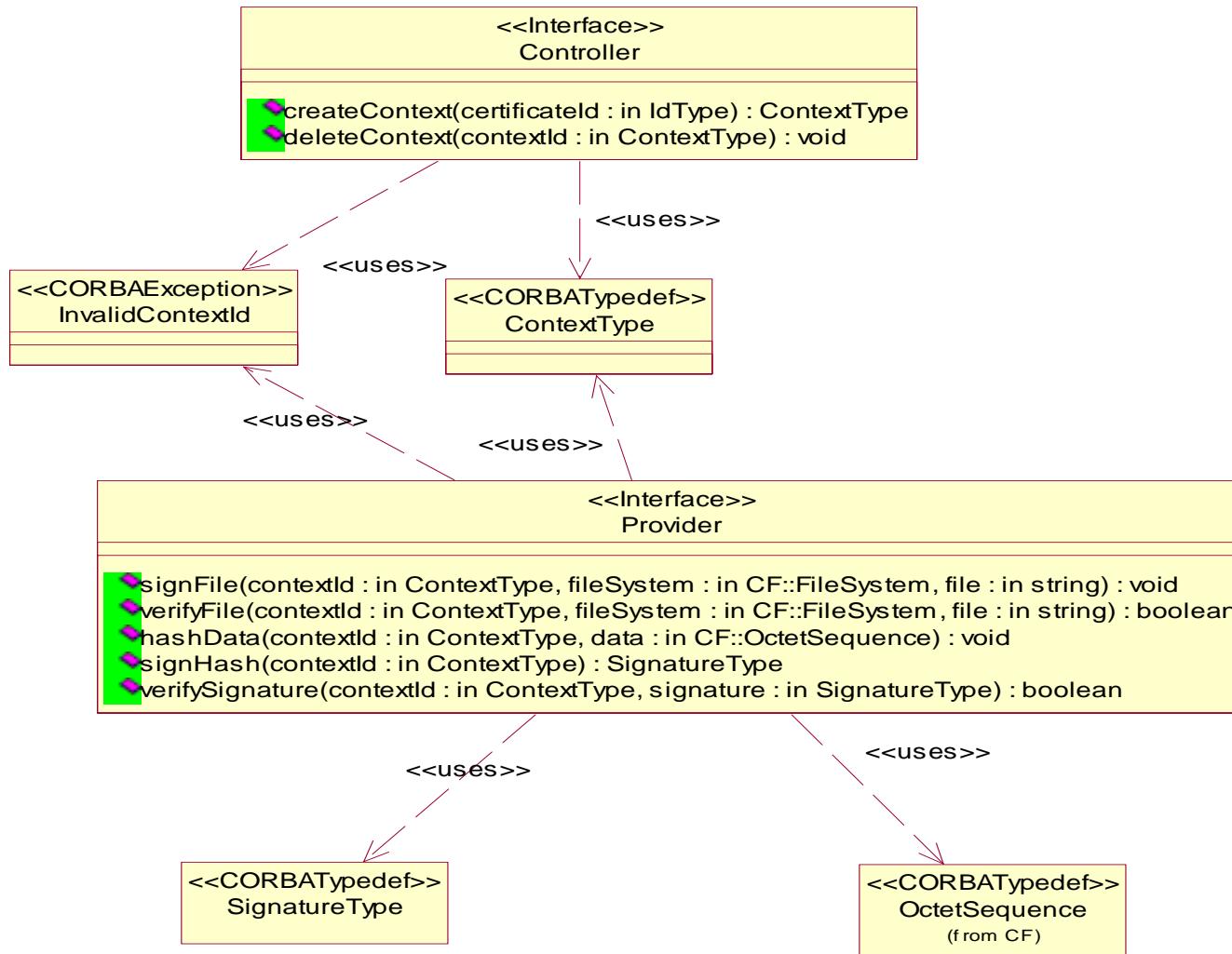
TRANSEC (cont'd)



Integrity and Authentication

- Used for verifying software downloads, HMI commands, etc.
- Control Service
 - Creates and Destroys I & A contexts based on certificates
- Digital Signatures Service
 - Provides interface for hashing, signing and verifying of files and data

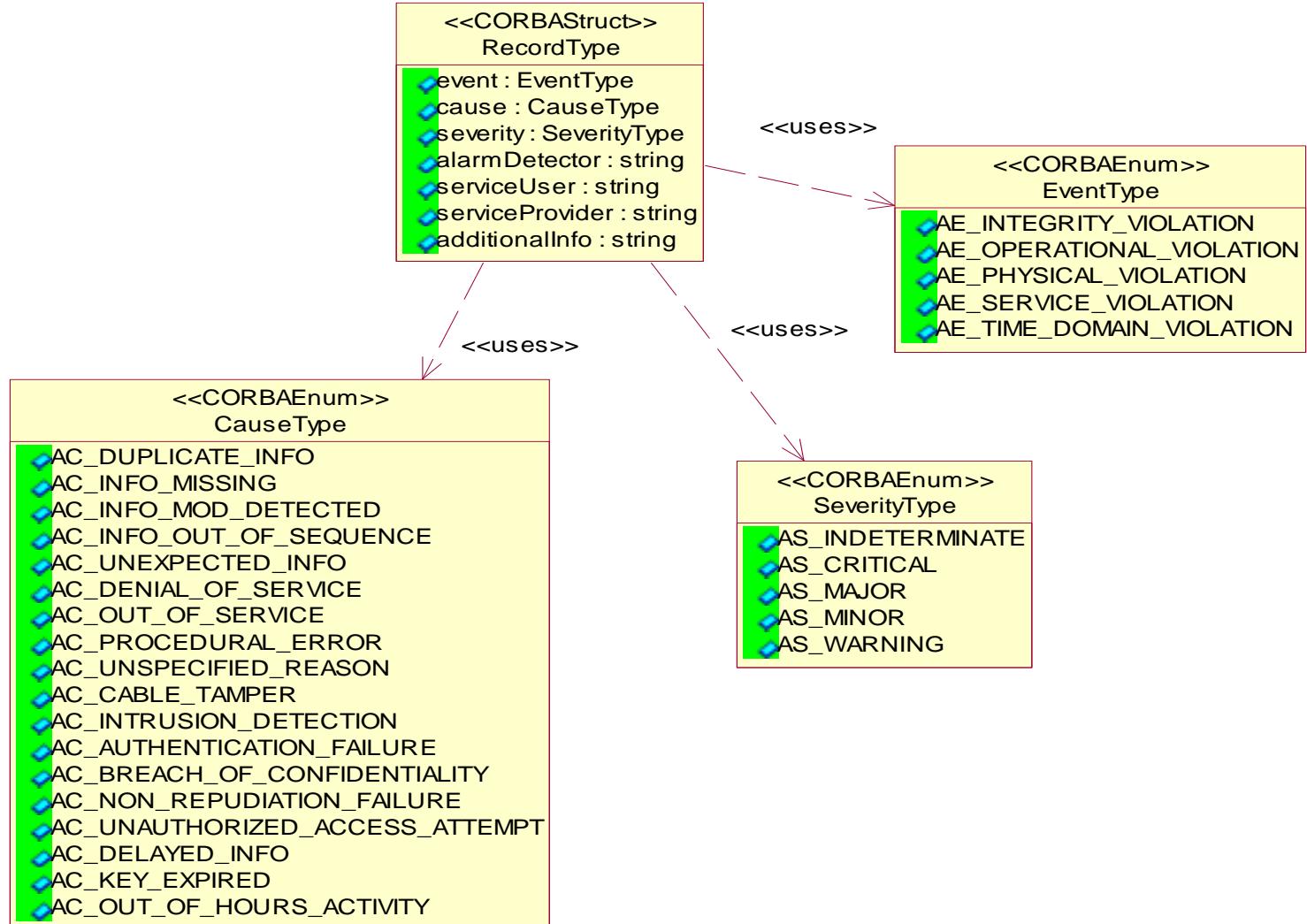
Integrity and Authentication (cont'd)



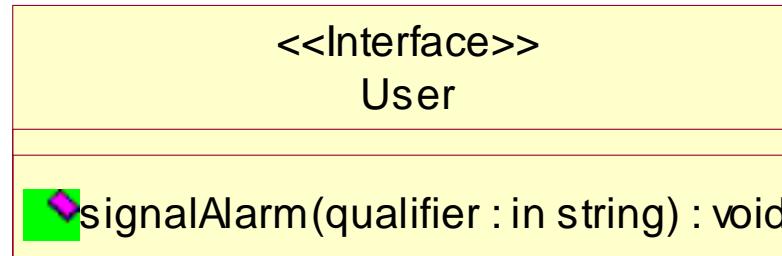
Alarm – Audit Types

- Audit Types
 - Defines Alarm information to be provided to audit function
 - Based on X.736 standard

Alarm – Audit Types (cont'd)



Alarm – User Service

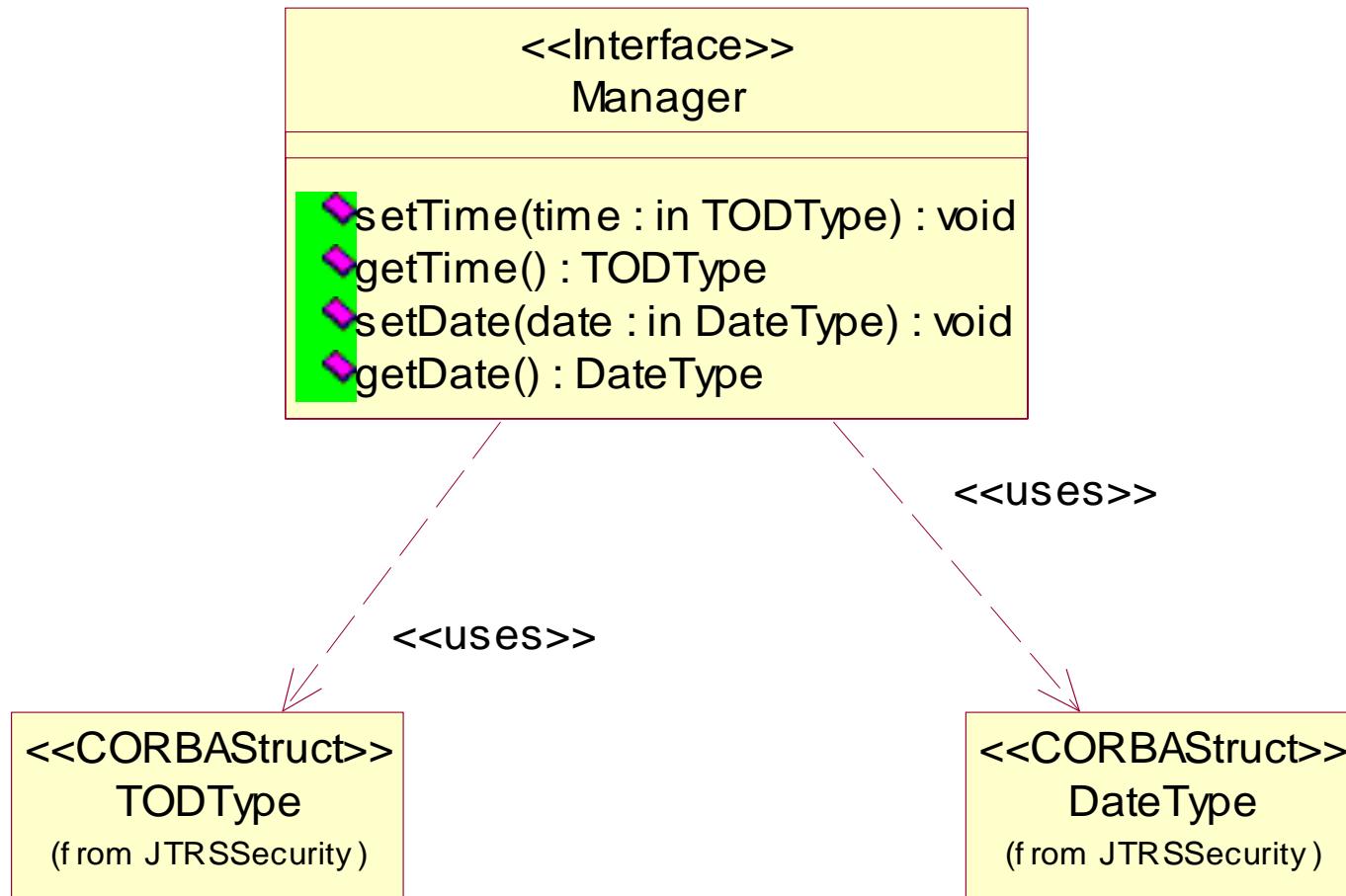


- User Service
 - Defines interface to be provided by user for alarm indication
 - Registered with CS/S

Time

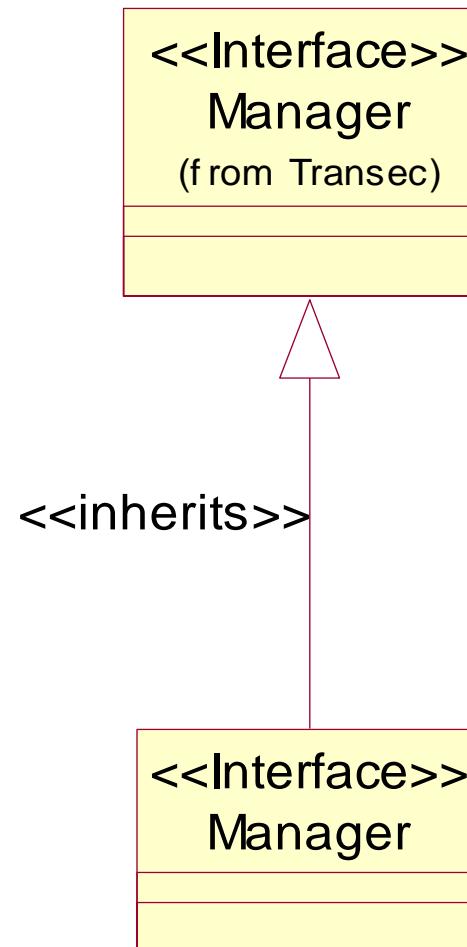
-
- Management Service
 - Provides time interface to CS/S
 - Required for TOD seeding
 - May be integrated with OE/CF Time Services at a later time

Time (cont'd)



-
- Management Service
 - GPS only requires the basic TRANSEC management service

GPS (cont'd)



Security Supplement Summary

- Requirements defined in the supplement are confined to the JTRS platform
- Impacts to entities outside the JTRS platform have been identified
 - Infrastructure requirements to support JTRS will need to be defined in the future which are beyond the scope of the supplement
- APIs have been defined to support implementation of requirements defined in the supplement
- Some requirements are still to be defined
 - DTD for policies
 - Format of certificates



API Supplement Overview

API Supplement

- API Overview
- SCA APIs
- Building Blocks
- API Creation
- API Summary

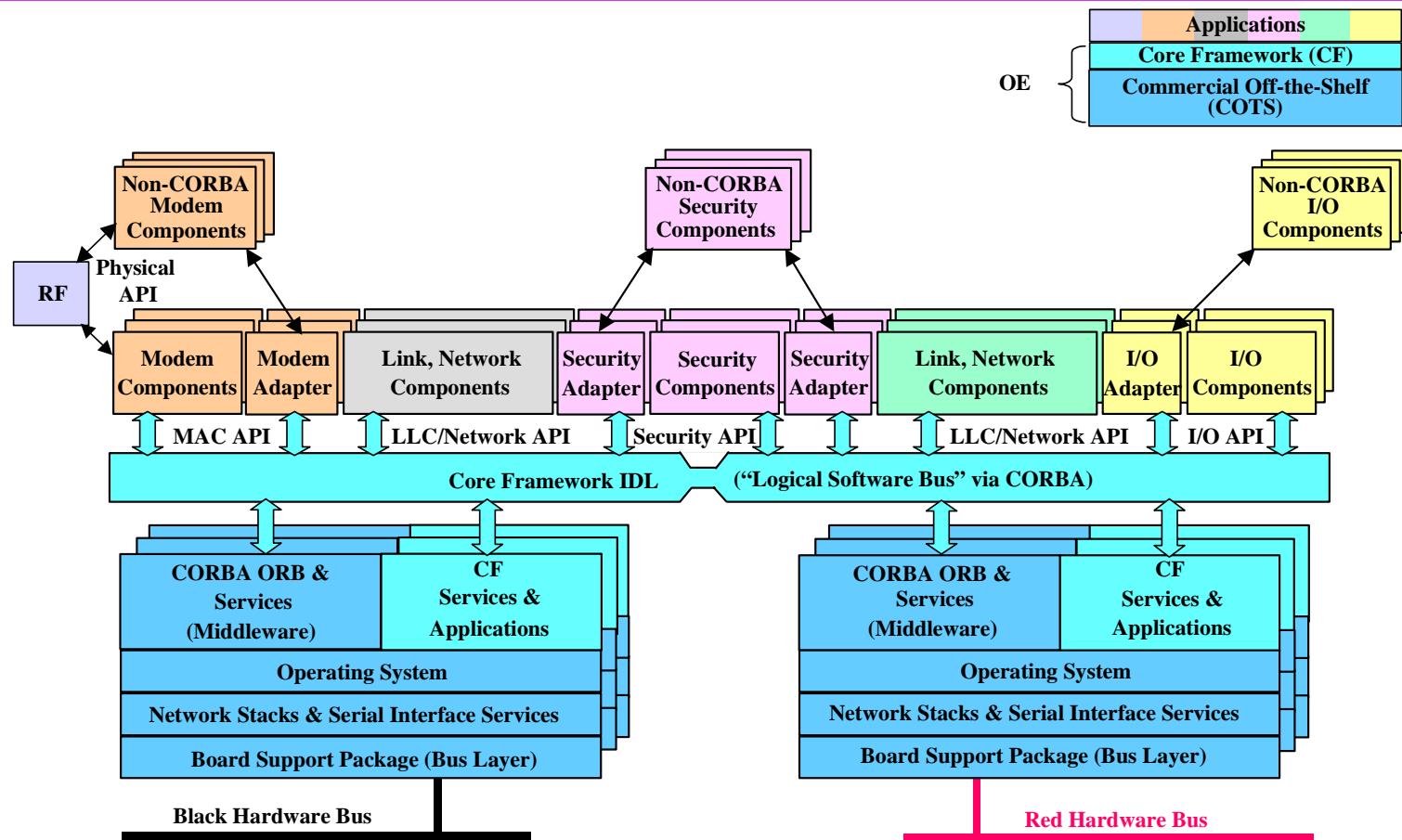
API Overview

- An API forms an agreement between two components on the
 - language and semantics used to communicate
 - services provided (Service Definitions)
 - behavior resulting from invocation of operations
- An SCA-compliant API is described using IDL and is formed by inheriting Interfaces derived from previously defined Building Blocks.

Why APIs are Defined in SCA

- Standardized APIs are essential for portability of applications and interchangeability of devices.
- APIs guarantee Service Provider and User can communicate regardless of OE or programming language.

Application Program Interfaces



*APIs are Located at Logical Boundaries
Common to Waveforms*

SCA API Decisions

- Why waveform-specific?
 - desire is single API at an interface
 - the range and variety of services at the various interfaces, most notably the MAC and Physical, make a common API for all waveform applications large and burdensome for resource constrained implementations
 - Building Blocks have been defined to provide as much commonality across waveform APIs as possible
- Goal is a standard API set for each waveform

SCA API Decisions (cont'd)

- Level of granularity
 - Initially, JTRS APIs are defined at interfaces where, in today's environment, waveform applications can readily be partitioned for the most porting utility
 - starting first with applications, API definitions can grow over time to encompass reusable components at any level where reuse is programmatically feasible

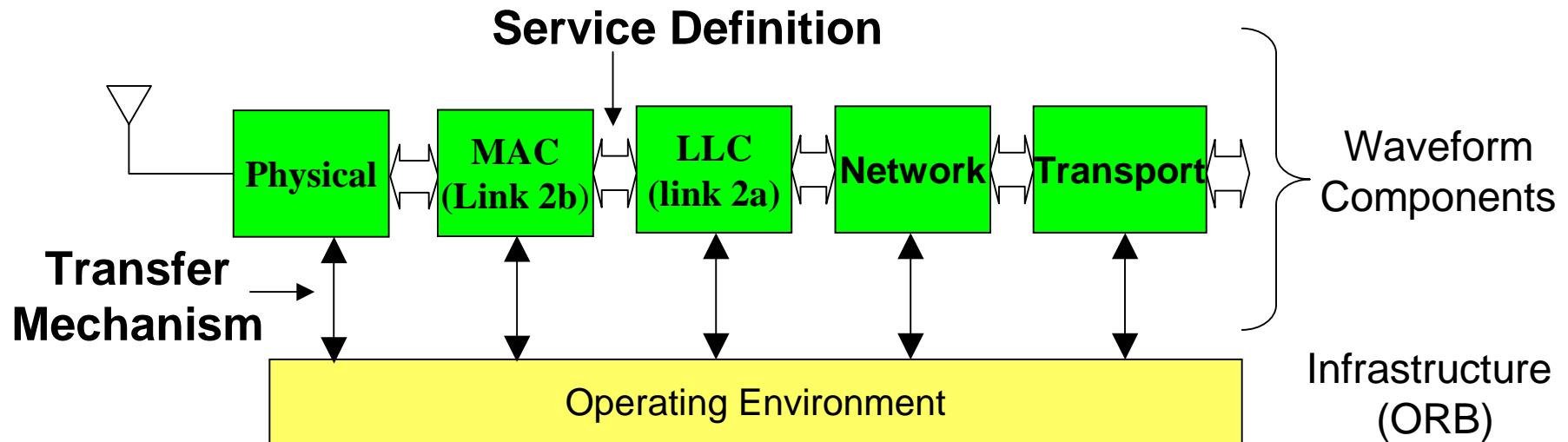
SCA APIs

- I/O
 - provides a common audio/data interface at a domain component containing voice and/or data processing
- Security (in Security Supplement)
- Network
 - provides a component level interface used for waveform network behavior
- Logical Link Control
 - component level interface used by waveform applications requiring link layer behavior (Data Link Service conforming to the Open Systems Interconnect (OSI) model for networking systems)

SCA APIs (cont'd)

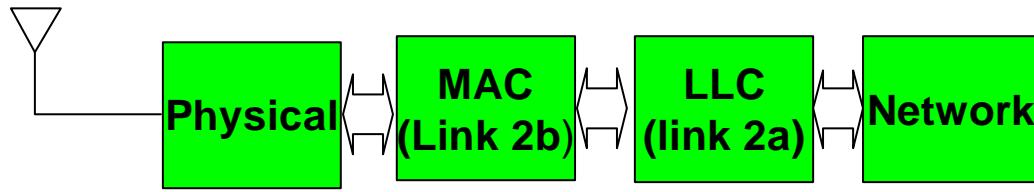
- Medium Access Control (MAC)
 - analogous to and fully supports services of the Medium Access Control sub-layer of the OSI Link Layer model
 - provided for waveform applications that have medium access control behavior (e.g. transmit/receive time slot control in TDMA, error correction coding control, etc.)
- Physical (Real-time & non-Real-time)
 - Real-time provides for translation from bits/symbols to RF and RF to bits/symbols for waveform transmission and reception.
 - non-Real-time provides for initialization and configuration

API = Service Definition + Transfer Mechanism



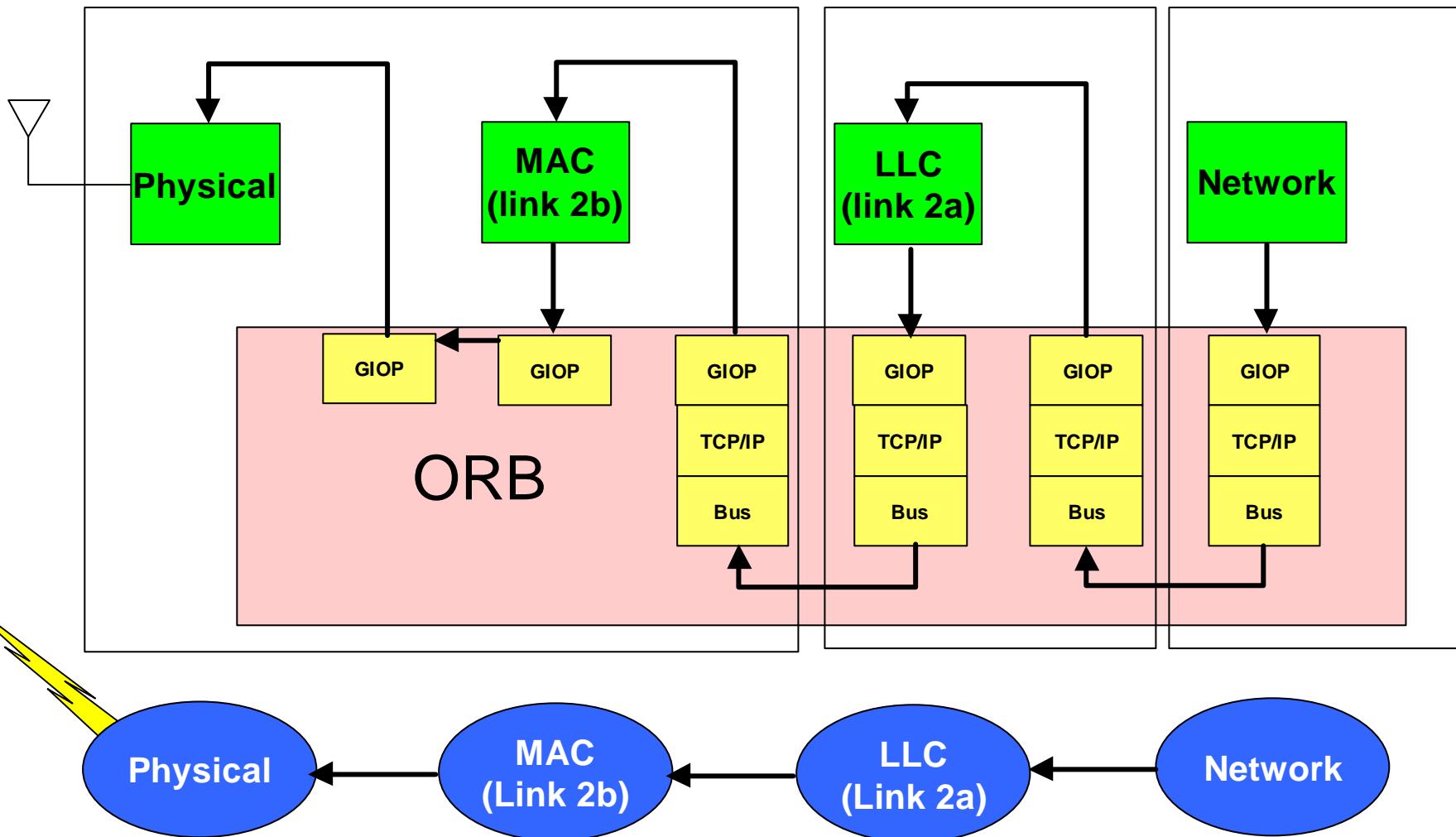
Waveform is an application running on the OE infrastructure

What Is The Service Definition?



- The service definition is the documentation of IDL, state, and behavior of the components at a particular interface.
- The service definition defines:
 - what components do (behavior, state information, and exceptions),
 - how components are accessed (methods),
 - what the message parameters are (arguments)
 - what the message parameters representations are (structures, types, formats).

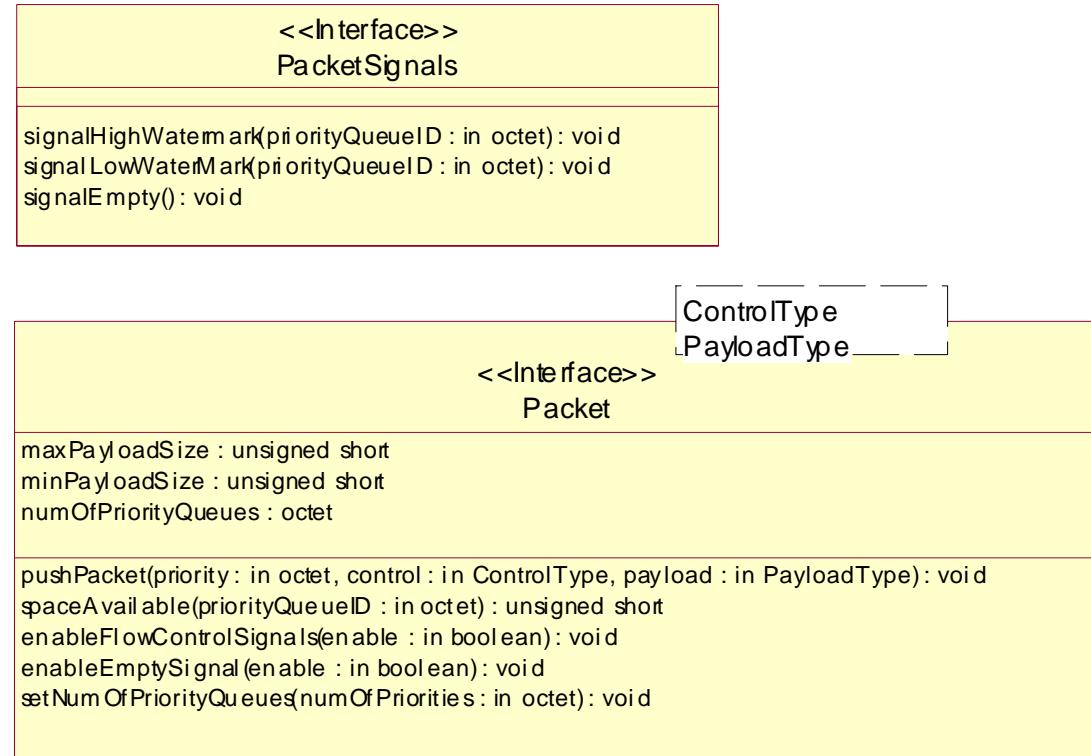
CORBA Transfer Mechanism Example



What is a Building Block?

- A BB is an abstract element used to provide the approach and interface definitions for services common to many APIs.
- BBs are conceptual elements that are used to form basic elements of APIs.
- BBs are converted into API elements by taking the abstract aspects of the BB and forming concrete IDL interfaces.
- An API interface is formed by inheriting these concrete IDL interfaces.
- BBs are used to foster reuse and commonality of accepted interface designs.

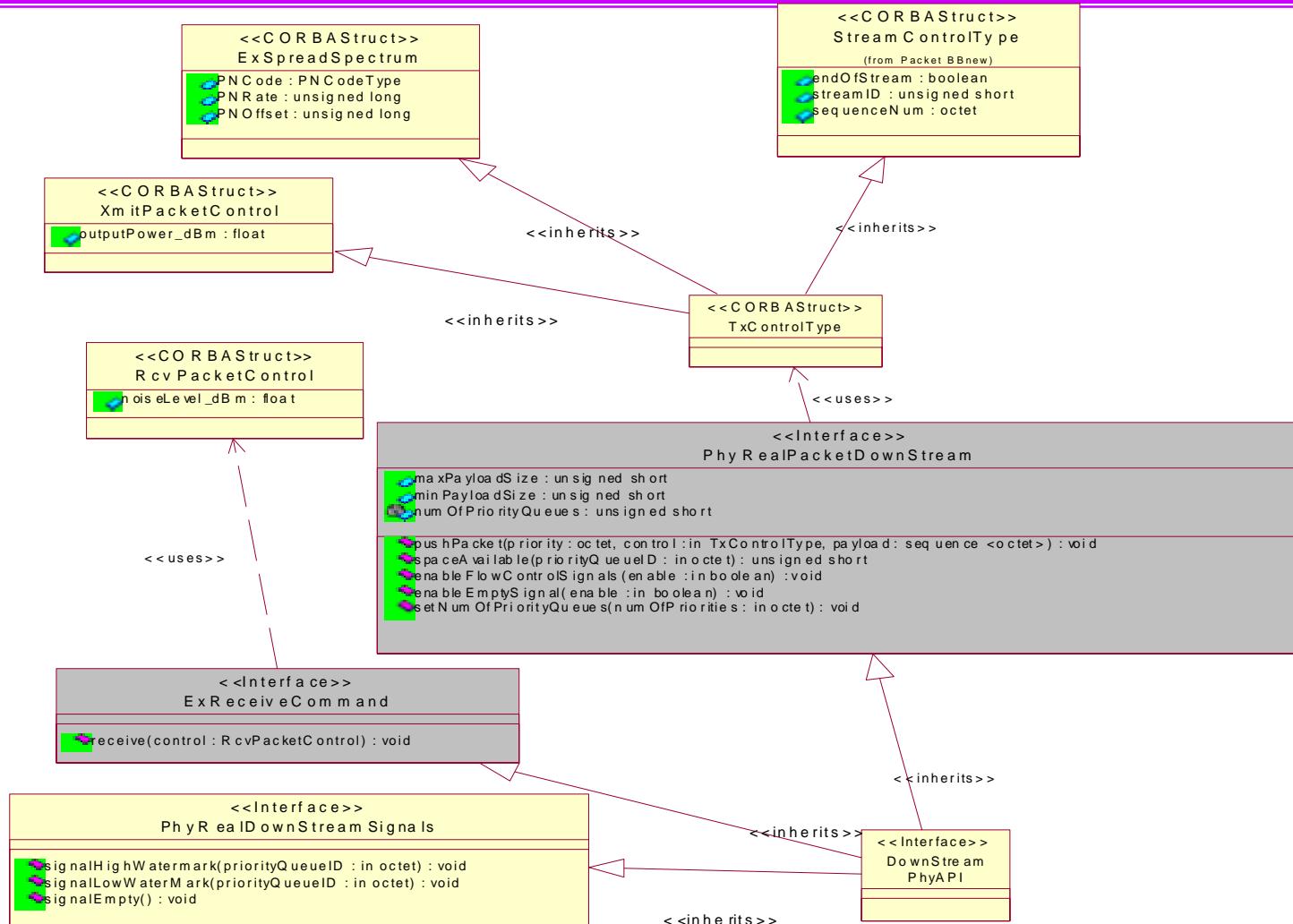
What is a Building Block?



Steps to Create an API

- Reuse an existing API (e.g. all SINCGARS should use the same API)
- If an API meets some of the requirements of an interface but not all, inherit the existing API and extend.
- If an API does not exist that closely matches the requirements, create a new API from the SCA building blocks.
 - APIs and their unique identifier (UUID) registered with a Registration Body
 - Registered APIs may be used by other contractors

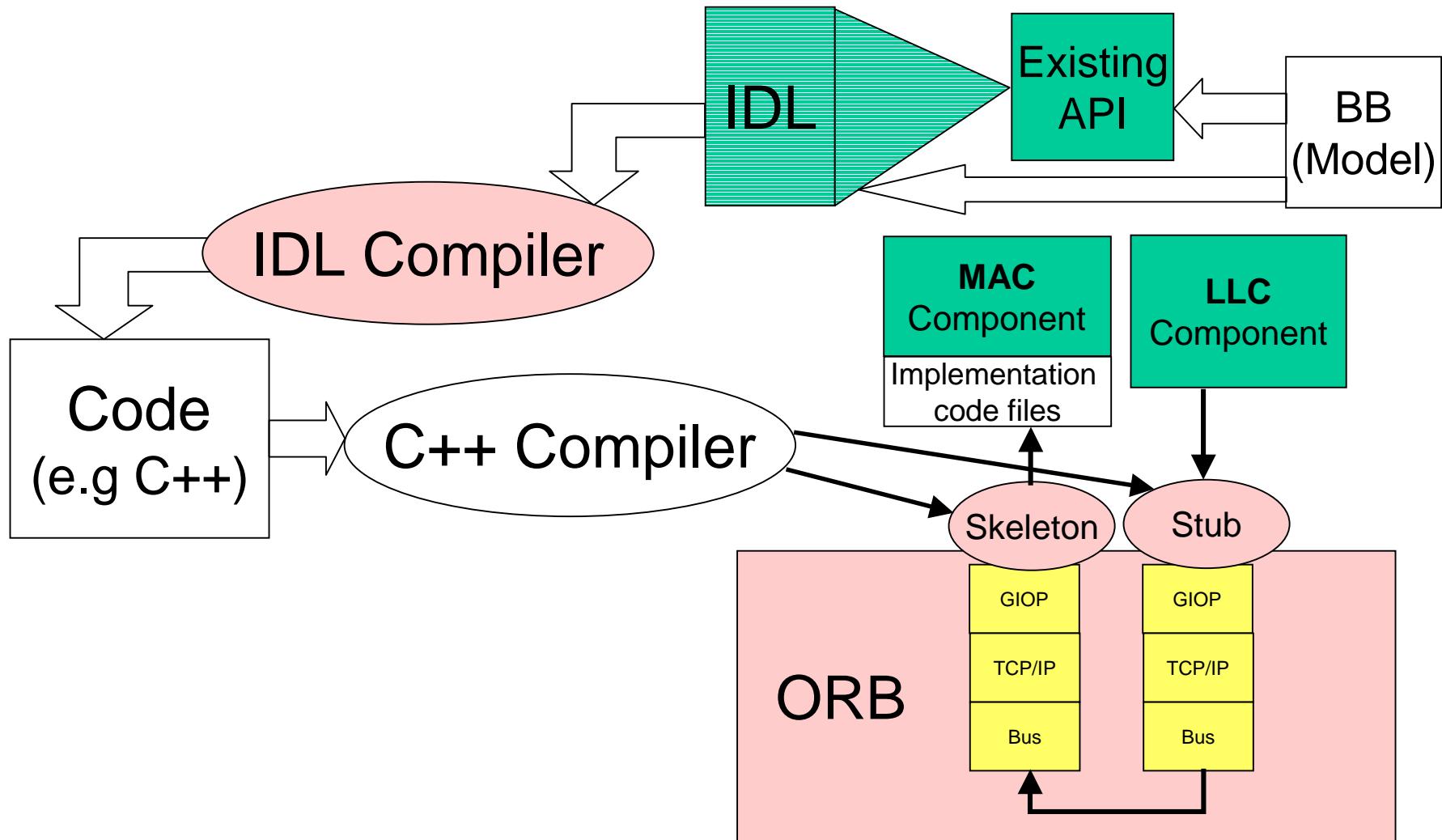
How to Build an API from Building Blocks.



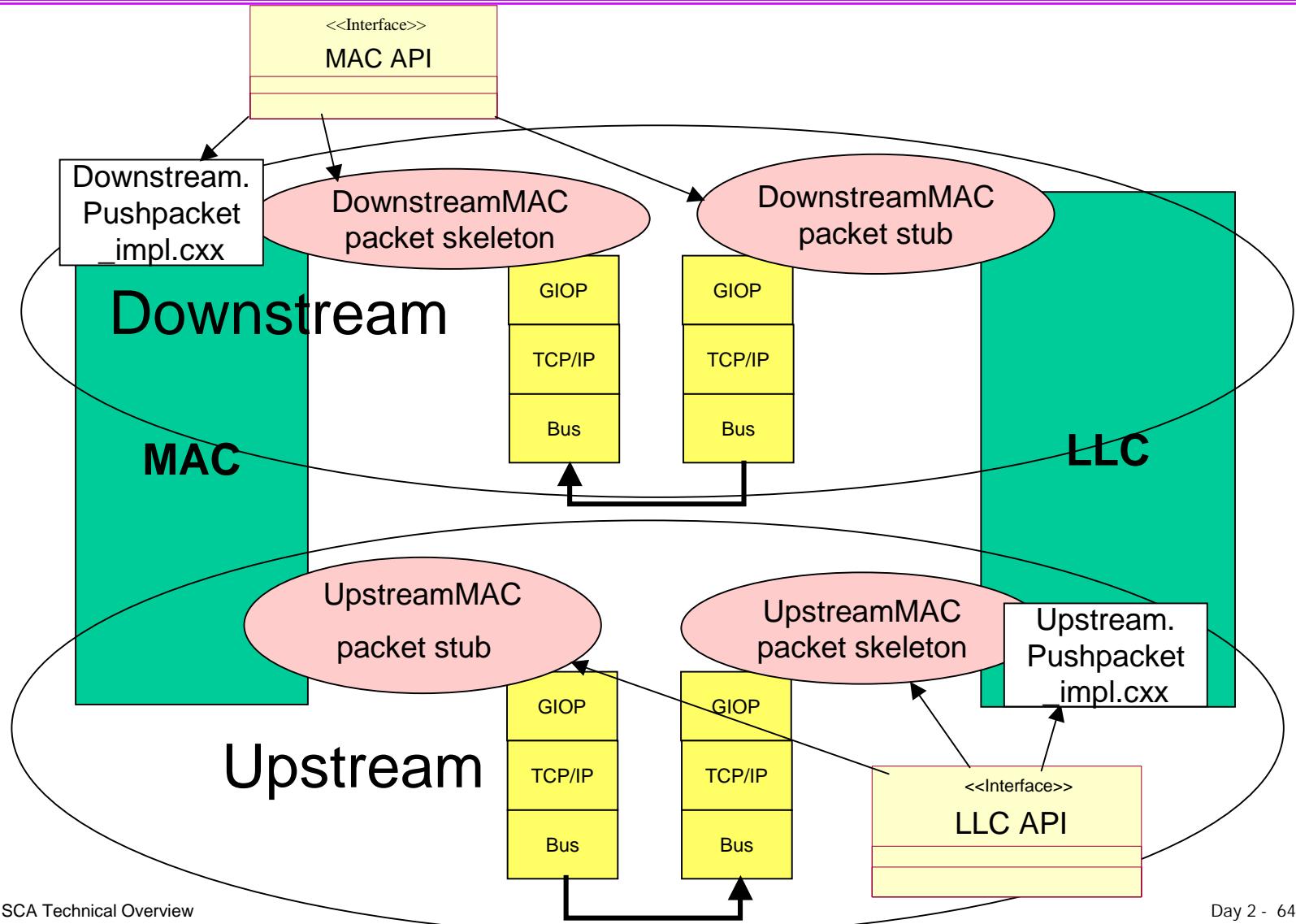
API Implementation and Use

- Generate Interface Definition Language (IDL); Can be written in text editor or auto-generated from a model (e.g. Rose)
- Document and register API.
- Compile IDL with the selected compiler; each ORB vendor supplies a compiler to convert the IDL for the selected ORB in a specific language.
- Compile to object file.

MAC Implementation Example



How the API is Used



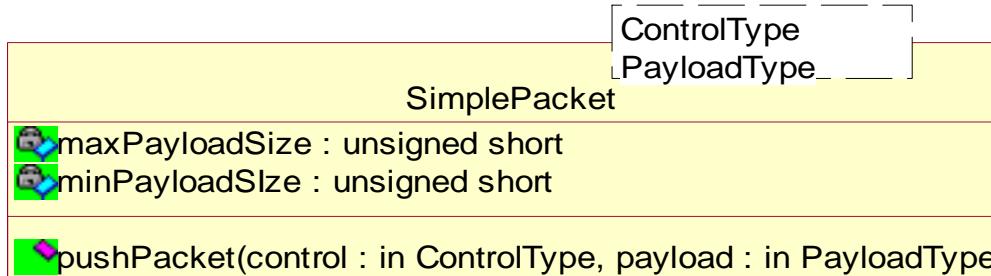
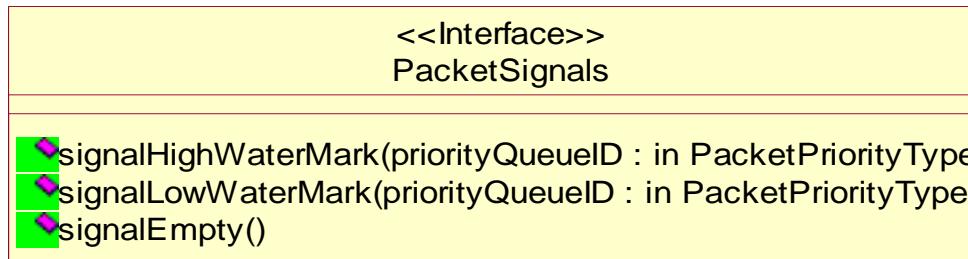
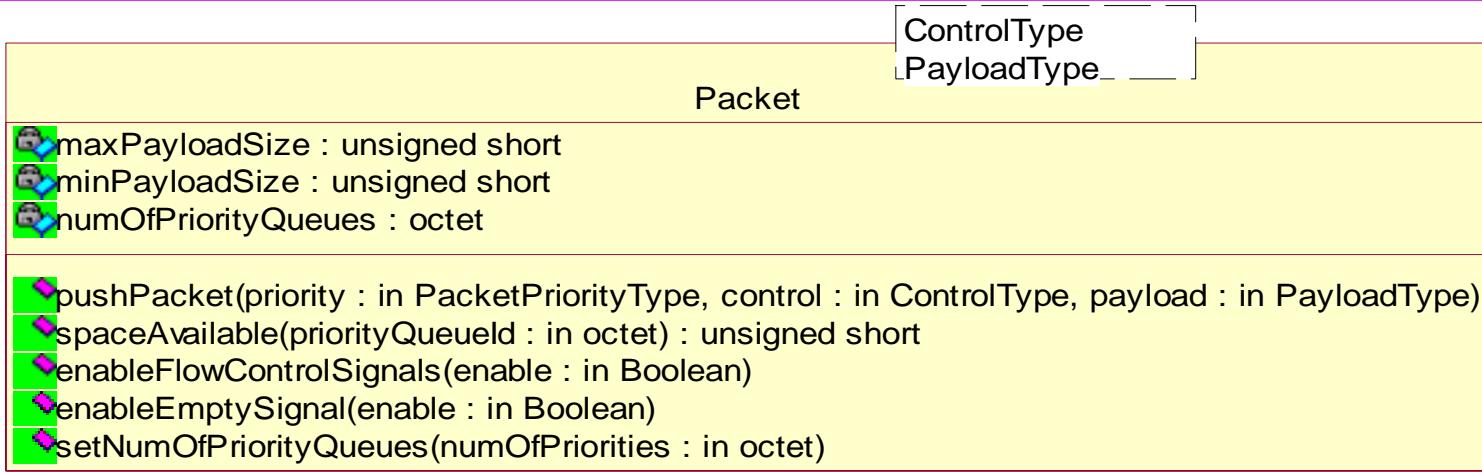
Currently Defined Building Blocks/APIs

- There building blocks and APIs currently defined as attachments to the API Supplement are:
 - Packet
 - Physical Real Time
 - Physical Non-Real Time
 - MAC
 - LLC
 - I/O

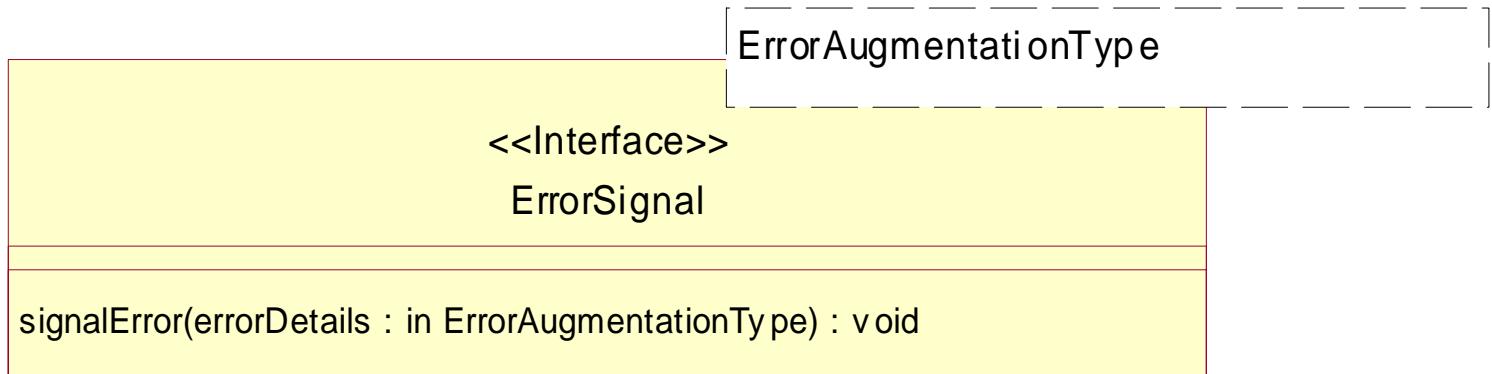
Packet

- Building Blocks that are used for data transfer between components
- Flow control and priority queueing available in Packet interface where needed, otherwise the SimplePacket interface is enough.
- Error signaling, Stream Control and Time are also defined

Packet (cont'd)



Packet (cont'd)



- This interface is used to asynchronously inform a connected component of an error

Packet (cont'd)

<<CORBAStruct>>
TimeType

seconds : unsigned long
nanoSec : unsigned long

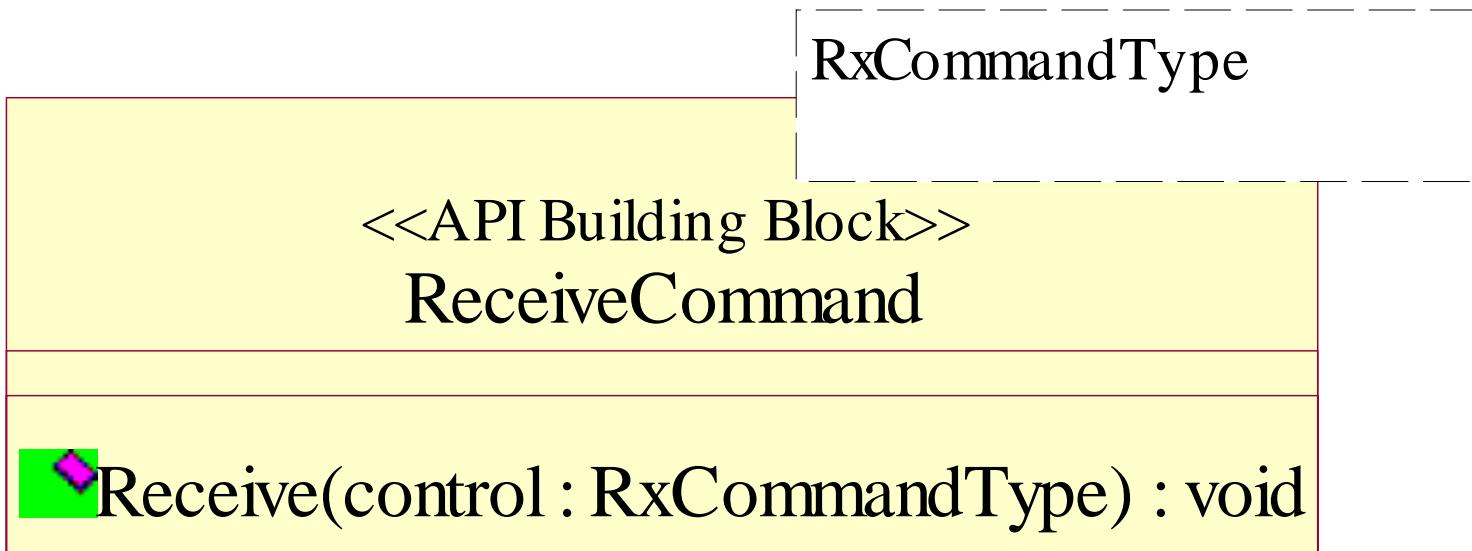
<<CORBAStruct>>
StreamControlType

endOfStream : boolean
streamID : unsigned short
sequenceNum : octet

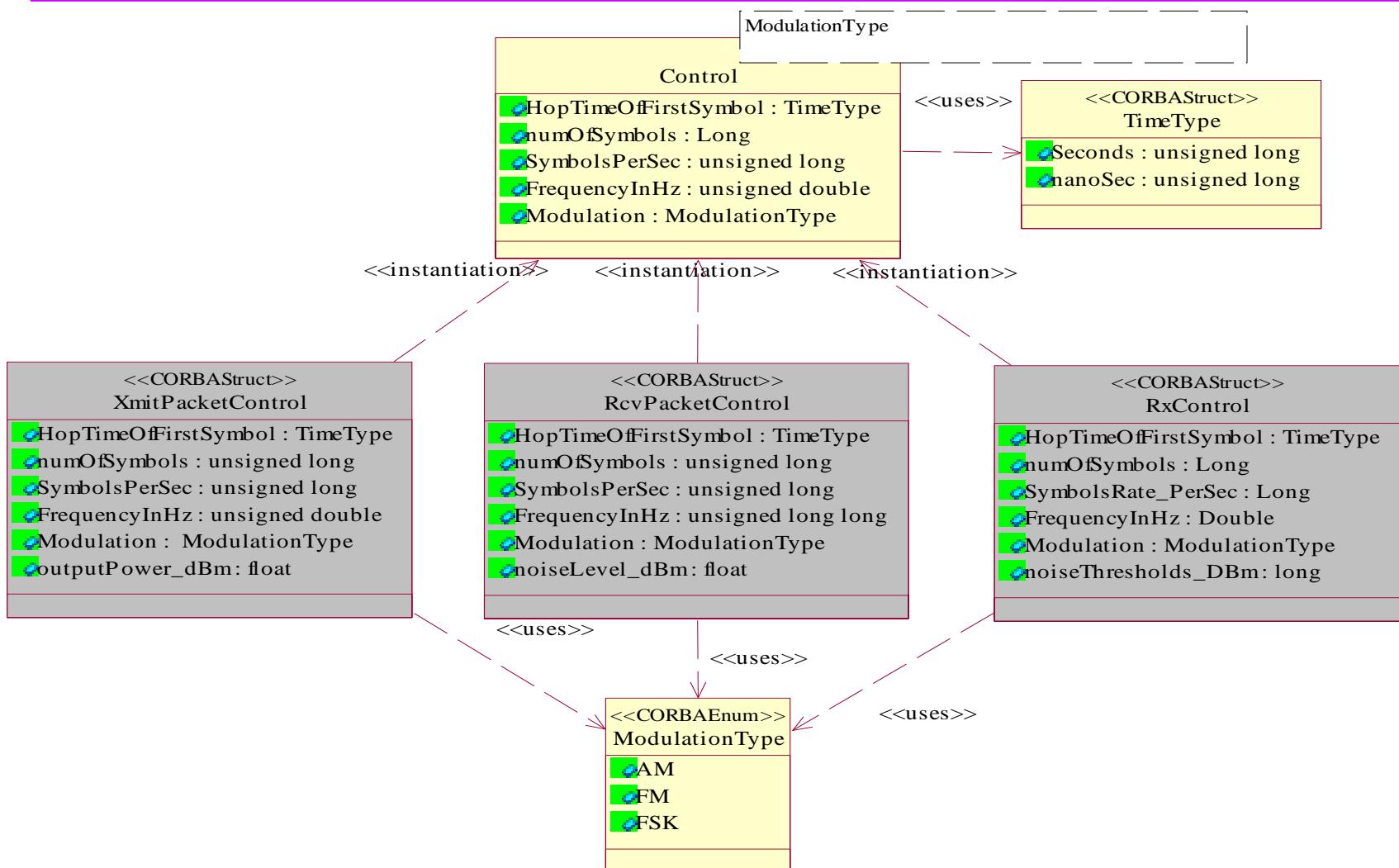
Physical Real Time

- Defines an interface for commanding a receiver to receive
 - Parameter is defined by the waveform
 - May contain time, frequency, noise thresholds, modulation type, symbol rate, etc.
- Receive data interface is defined using the Packet BBs
- Transmit command and data are defined using the Packet BBs

Physical Real Time (cont'd)



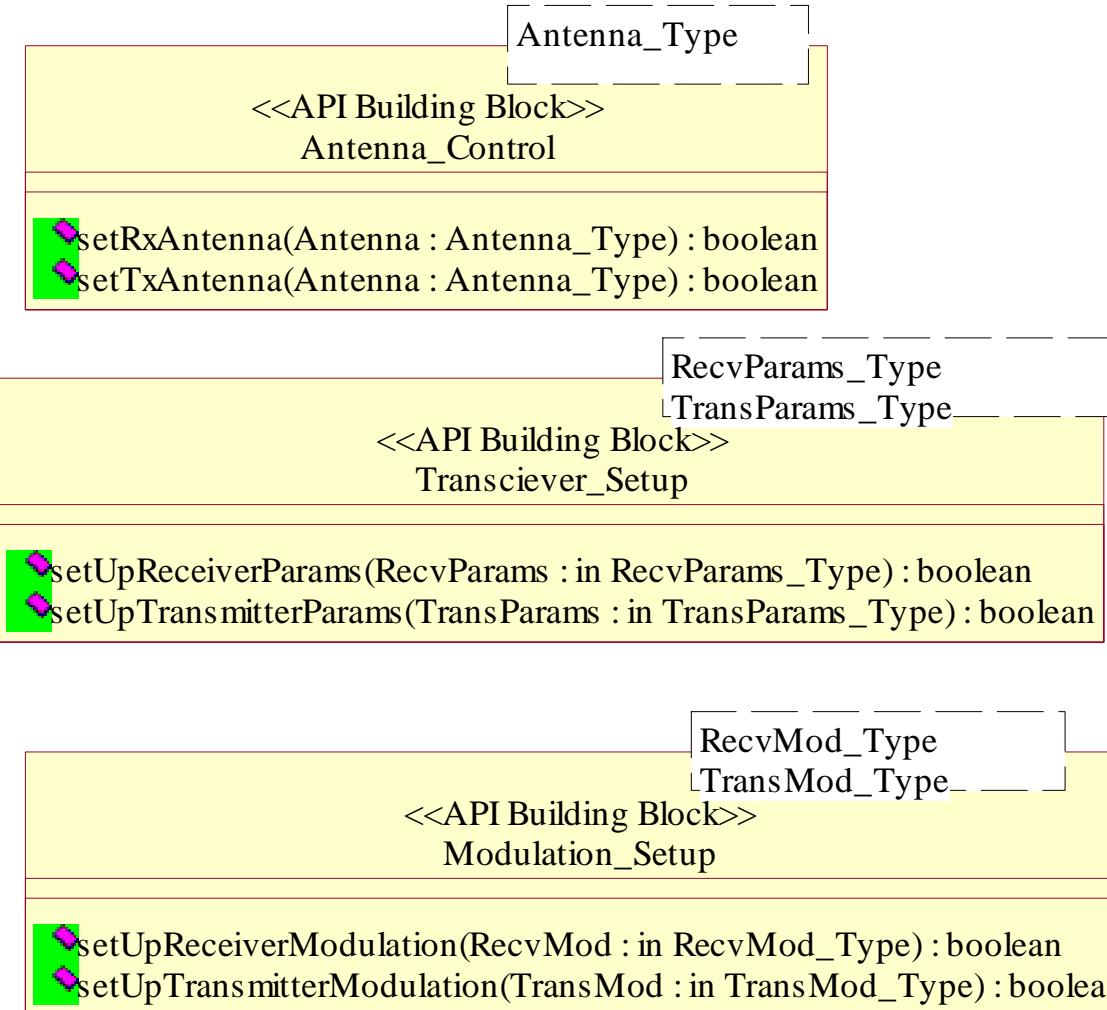
Physical Real Time – Example of Control



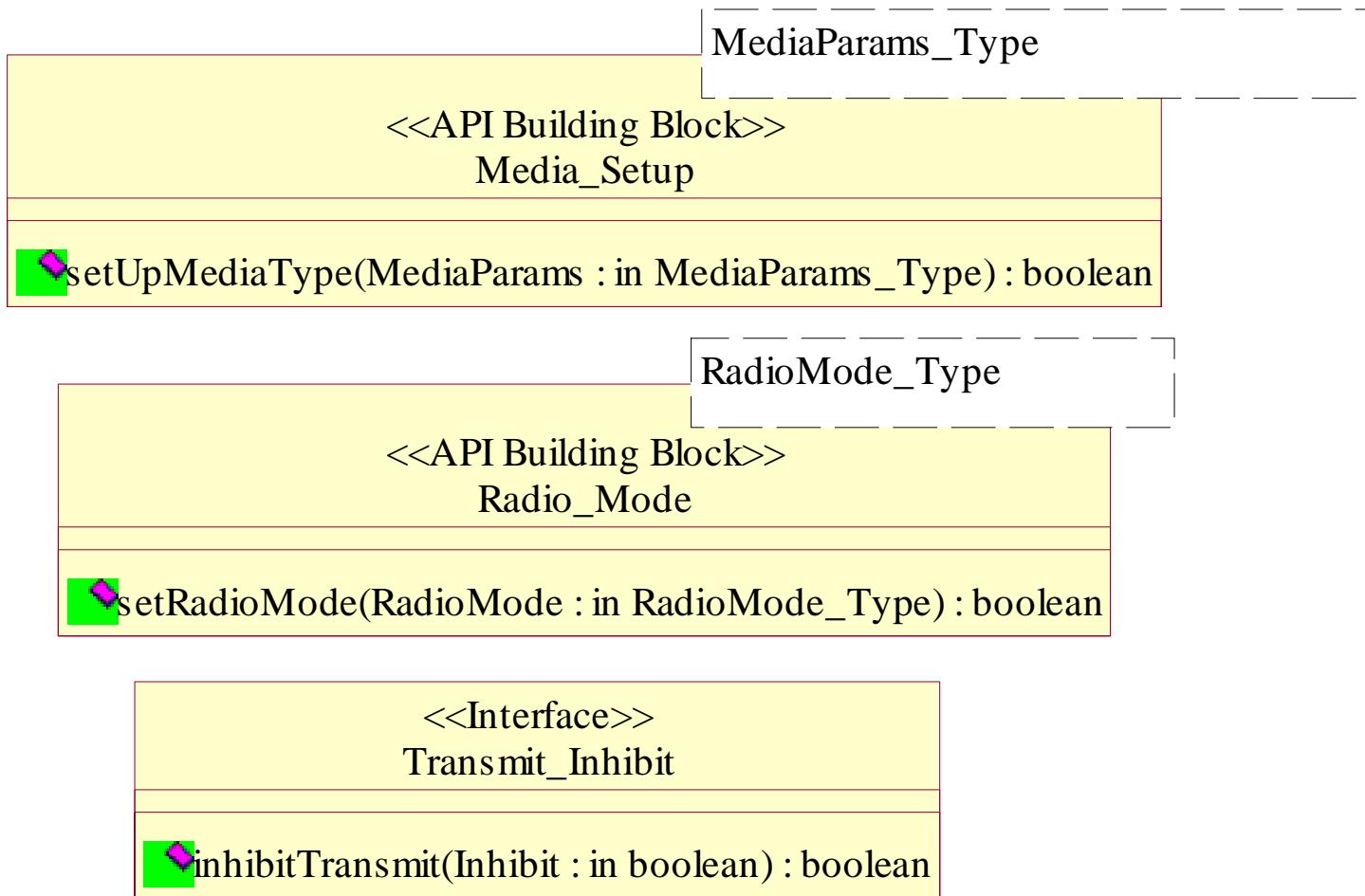
Physical Non-Real Time

- Non-real-time physical layer services provide Service Users with methods to send non-real-time (i.e. independent of user data transfer) configuration and control data to the physical layer.
 - Antenna Control (switching)
 - Transceiver Setup (set receive/transmit parameters that are independent of modulation type (e.g. output power, ramp time))
 - Modulation Setup (set parameters for a particular modulation type)
 - Media Setup (e.g. squelch)
 - Radio Mode (Off, Standby, Operational, Test)
 - Transmit Inhibit (Radio Silence)
 - Receive Termination (terminate reception altogether or attempt to reacquire)
 - Physical Management (Set min and max transmission unit size)

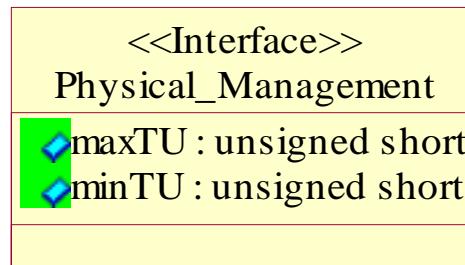
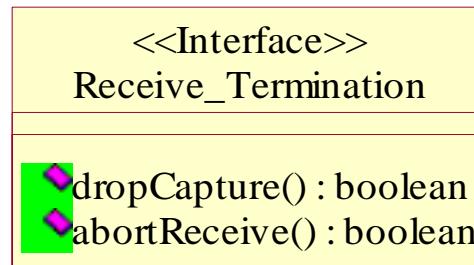
Physical Non-Real Time (cont'd)



Physical Non-Real Time (cont'd)



Physical Non-Real Time (cont'd)

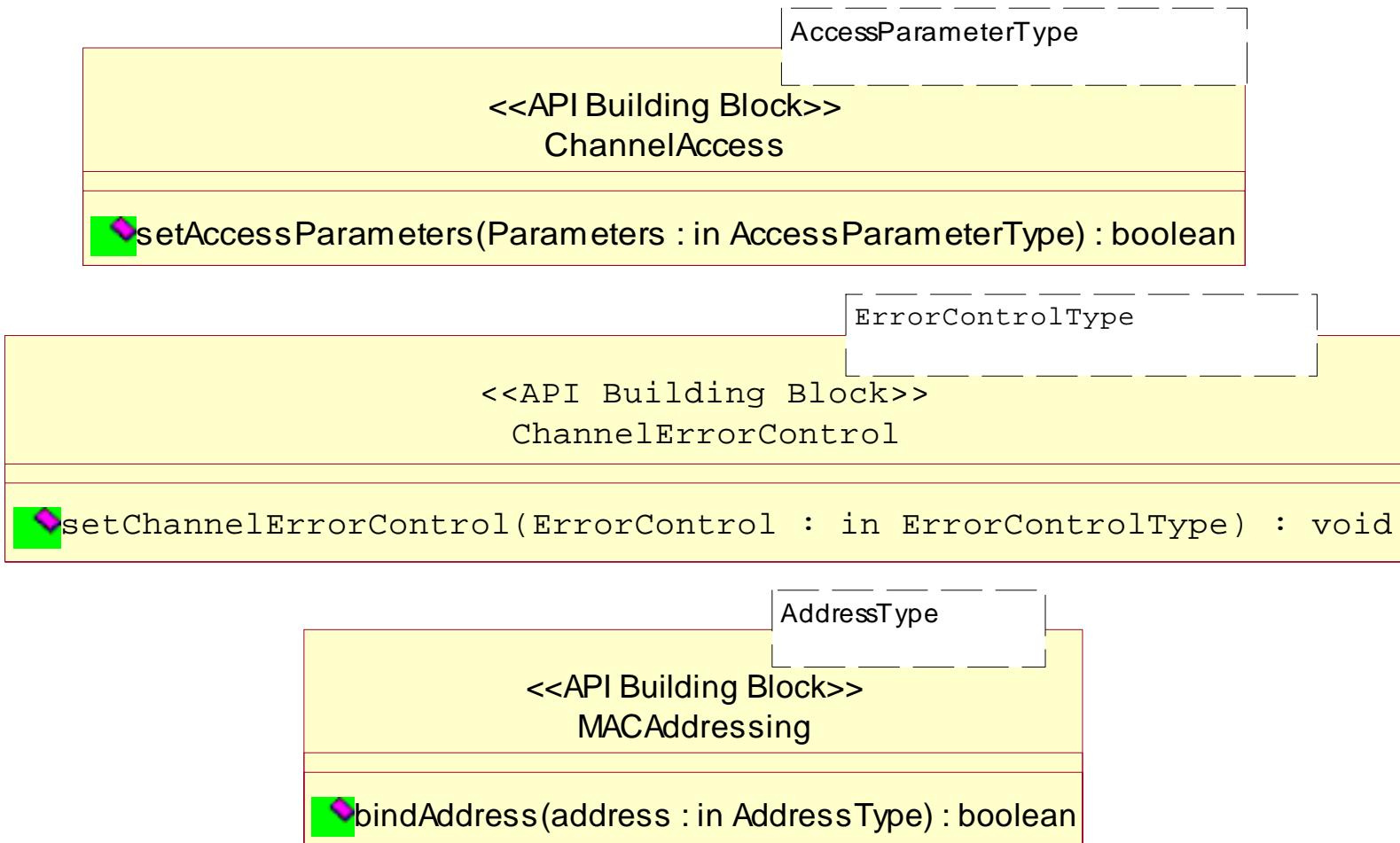


MAC

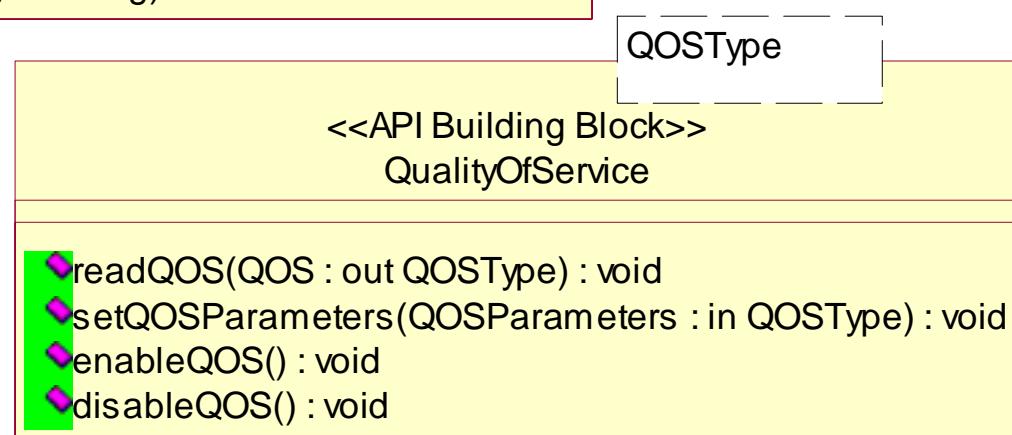
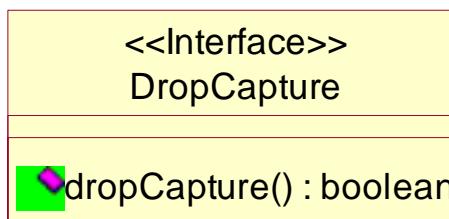
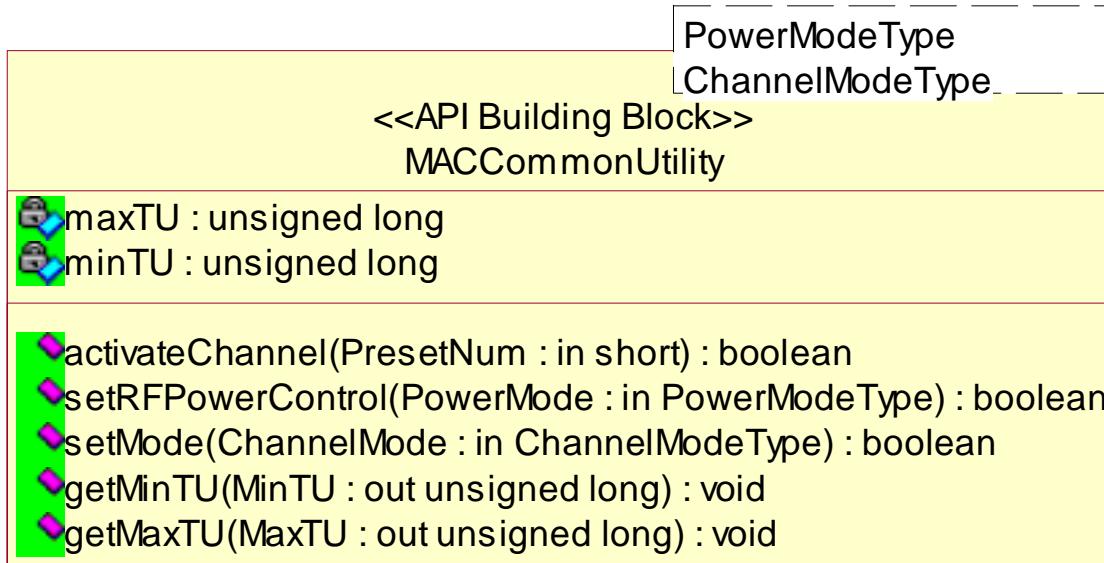
- MAC Services provide Service Users with methods to send non-real-time control and data between software resources and methods to signal the Service User that an event has occurred.
 - Channel Access (configure a net access algorithm)
 - Channel Error Control (enable, disable)
 - MAC Address (set the MAC address)
 - Drop Capture (terminate and reacquire)
 - MAC Common Utility (HMI parameters)
 - QoS (e.g BER)
 - TRANSEC (unclassified)

- Real-time control and signals are communicated via the packet interface.
- The MAC Building Blocks are intended to support at least the following Waveform APIs:
 - Demand Assigned Single Access/Demand Assigned Multiple Access (DASA/DAMA)
 - Have Quick
 - HF Automate Link Establishment (ALE)
 - Line Of Sight (LOS)
 - SINCGARS
 - VRC-99
 - Wideband Networking Waveform

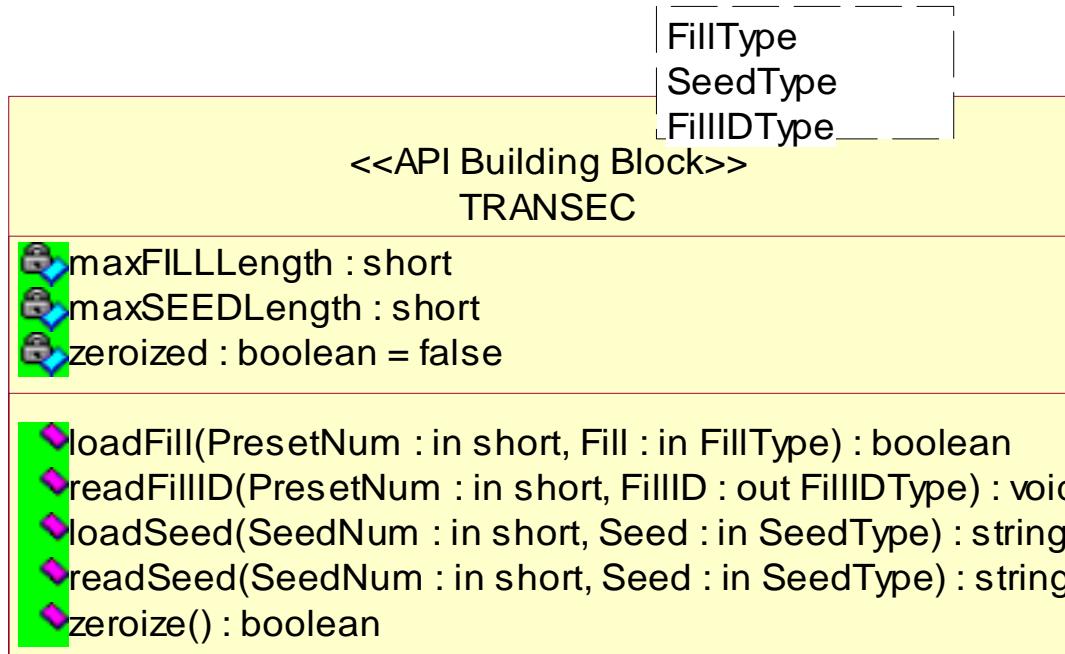
MAC (cont'd)



MAC (cont'd)

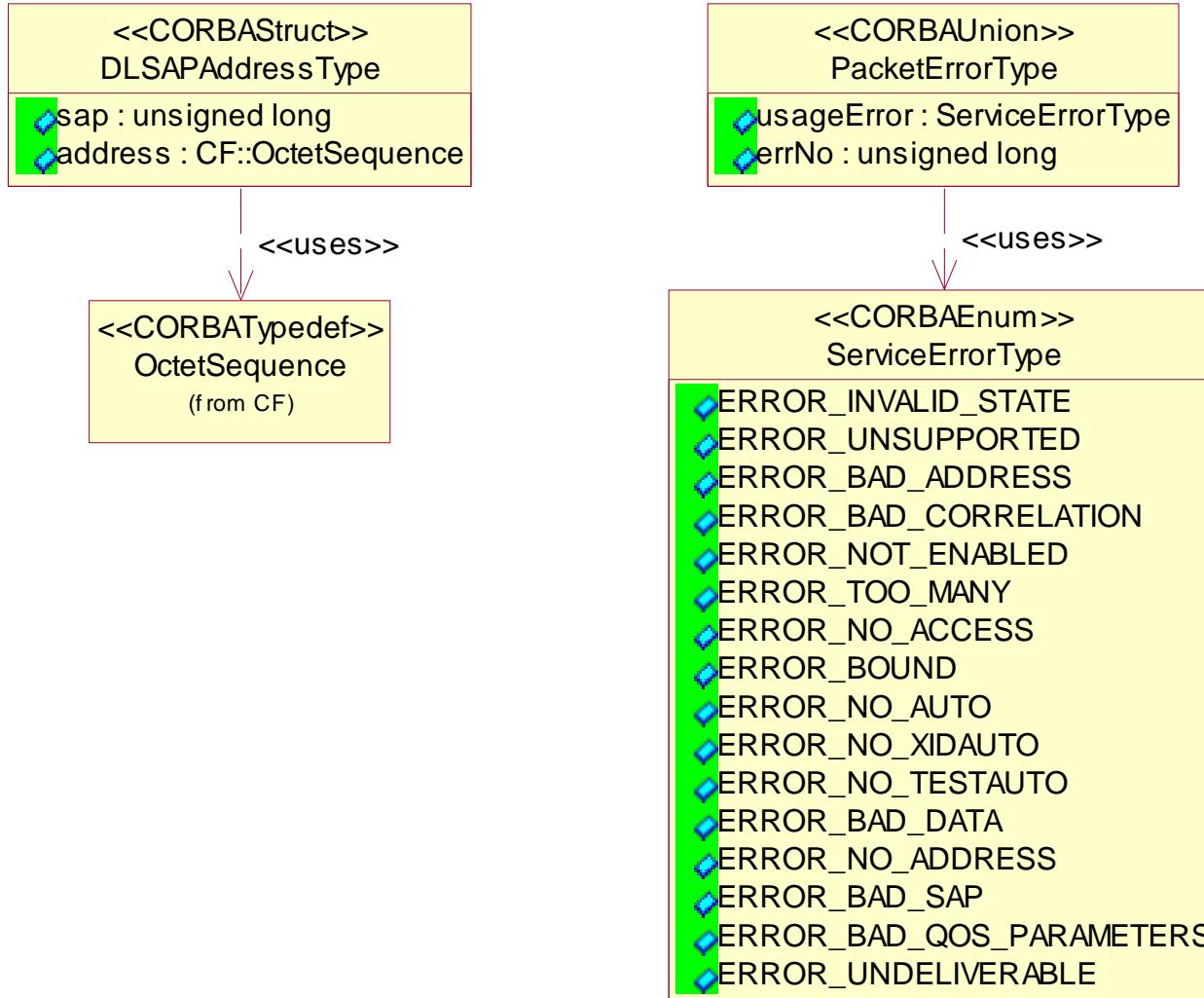


MAC (cont'd)



-
- The LLC interfaces are based on the Data Link Provider Interface Standard (DLPI) standard
 - The Connectionless and Acknowledged Connectionless Service Interfaces are currently defined
 - Services currently undefined
 - Connection Oriented
 - The Local Management Interfaces support a connection oriented service, only the service itself remains undefined
 - QoS

LLC – Types



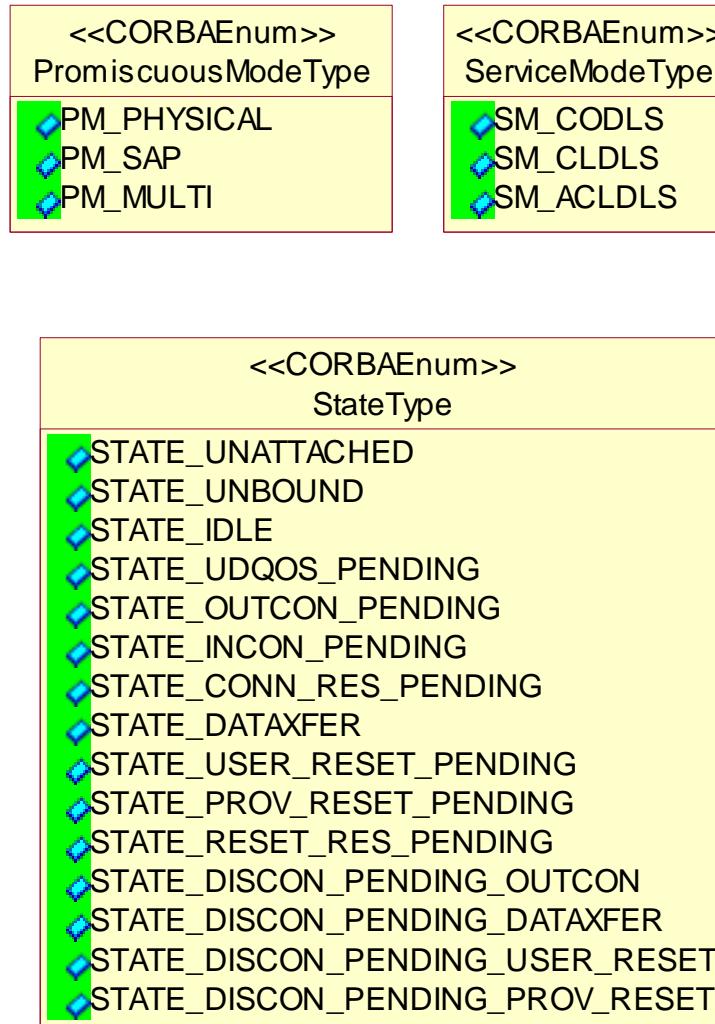
LLC – Types (cont'd)

- SAP = Service Access Point
 - Used for mux/demux between LLC and Service Users (e.g. IP and ARP)

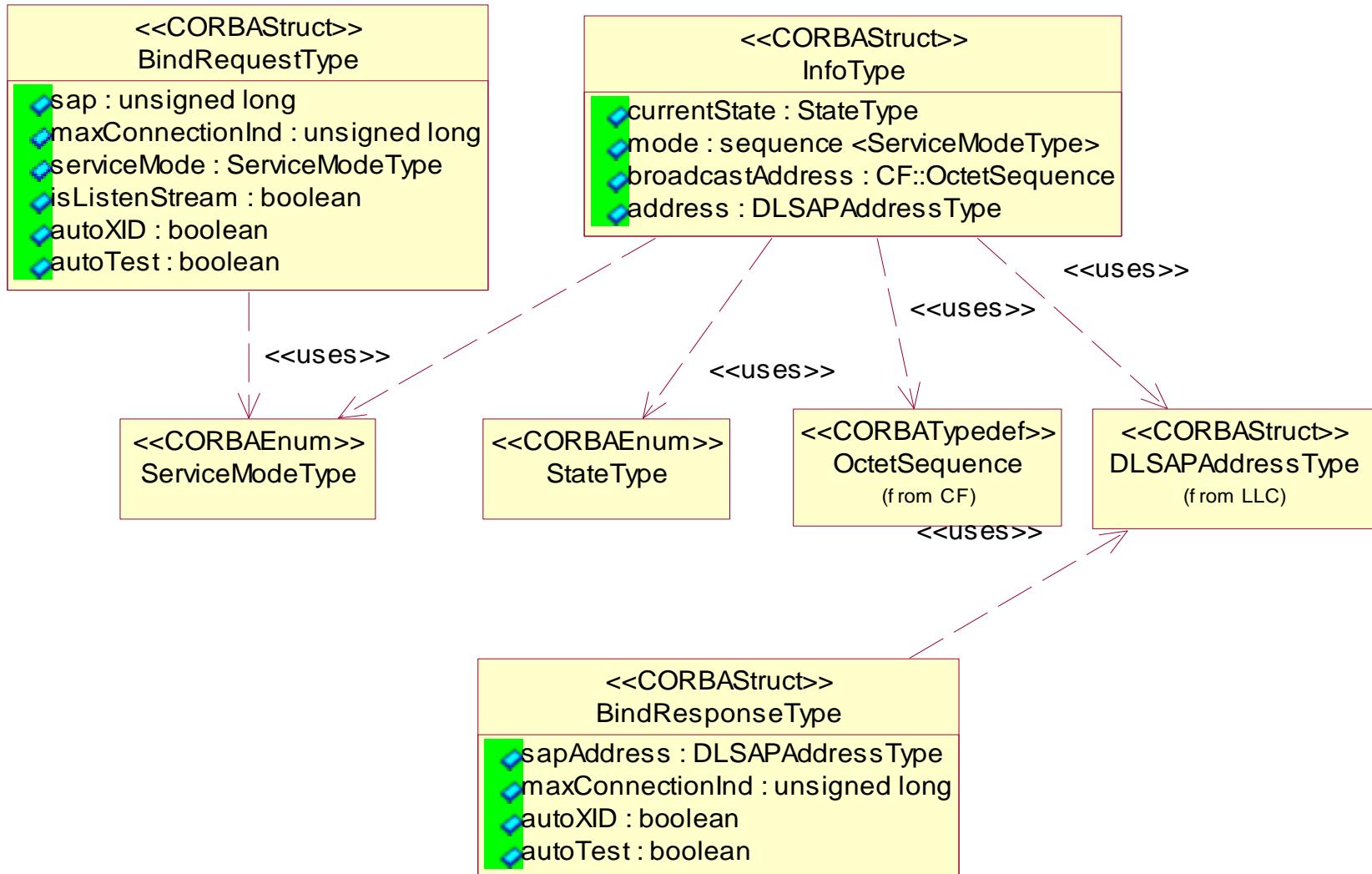
LLC – Local Management

- Supports Information reporting, binding, enabling/disabling of multicast addresses and turning on/off promiscuous mode.
- Once a connection has been established between the Service provider and the Service User, these primitives initialize the layer, preparing it for use.

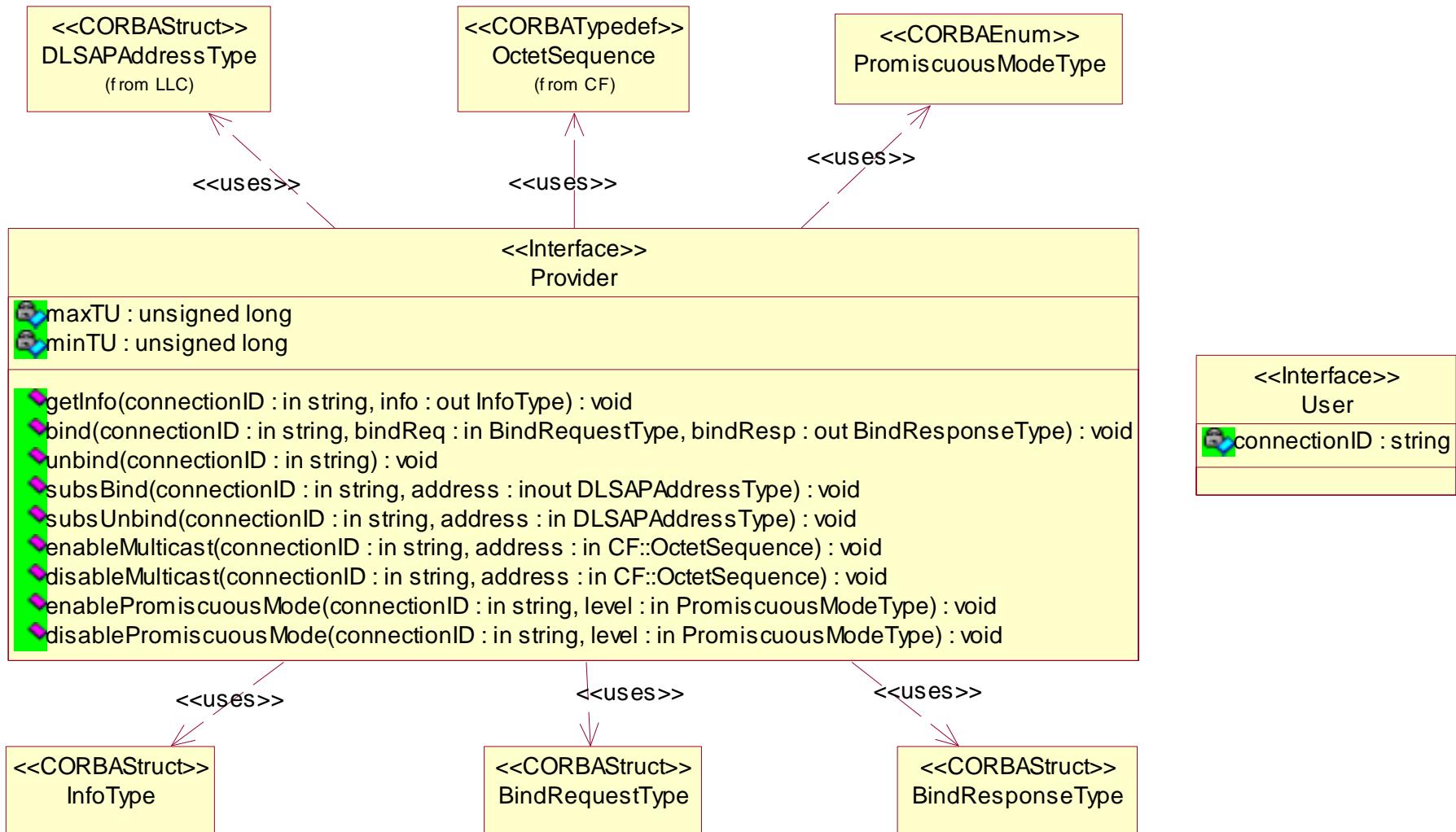
LLC – Local Management (cont'd)



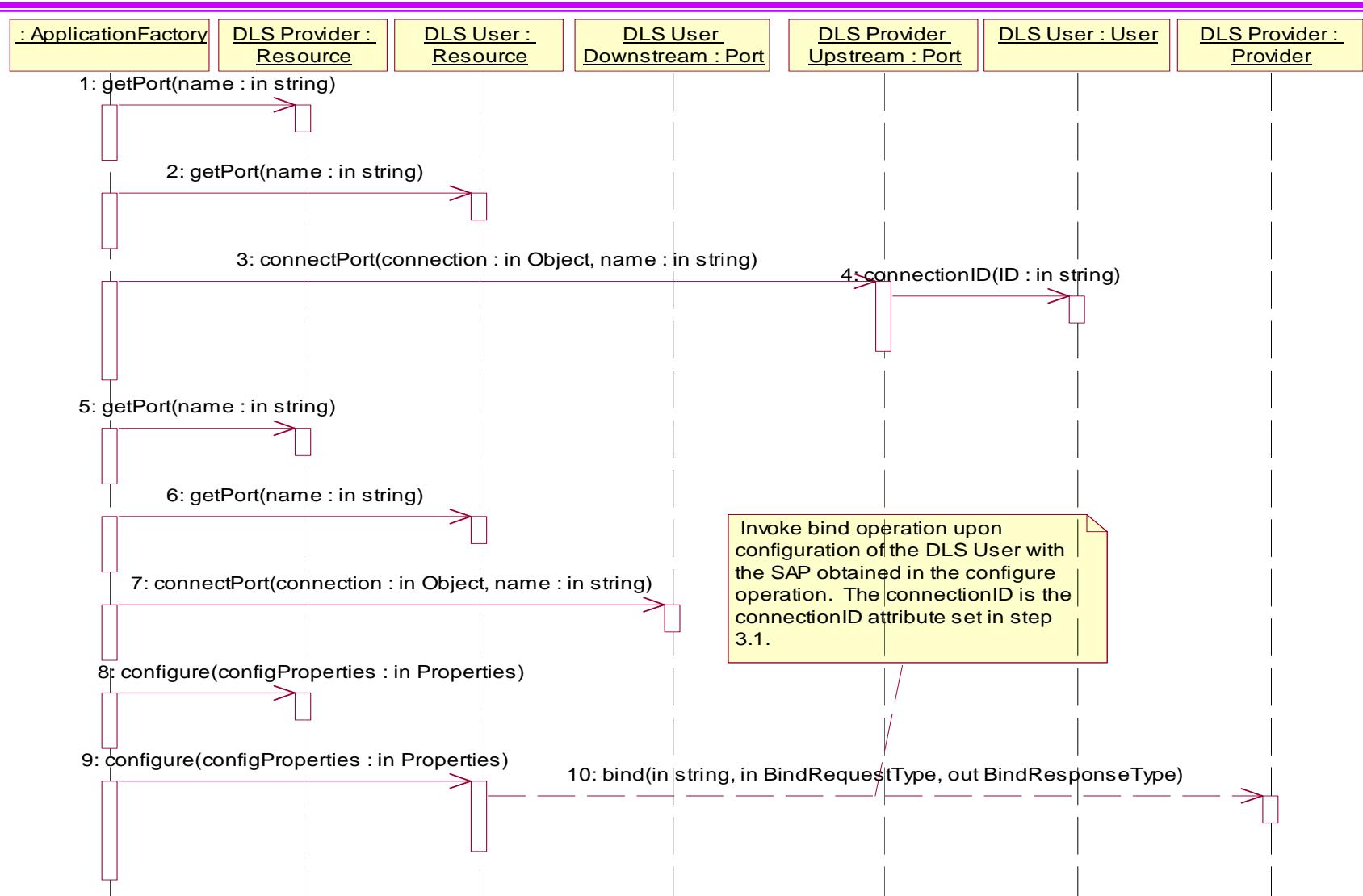
LLC – Local Management (cont'd)



LLC – Local Management (cont'd)



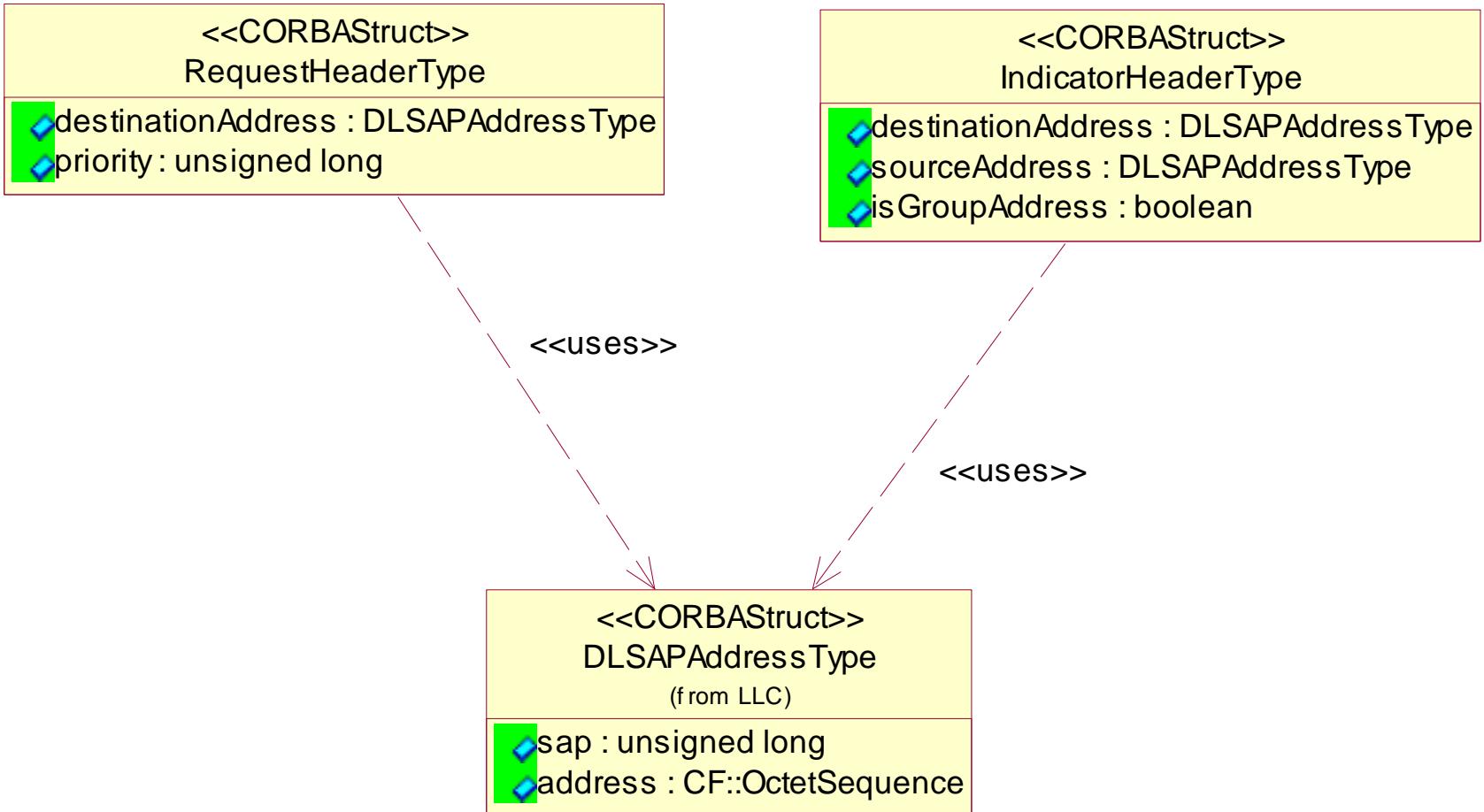
LLC – Local Management (cont'd)



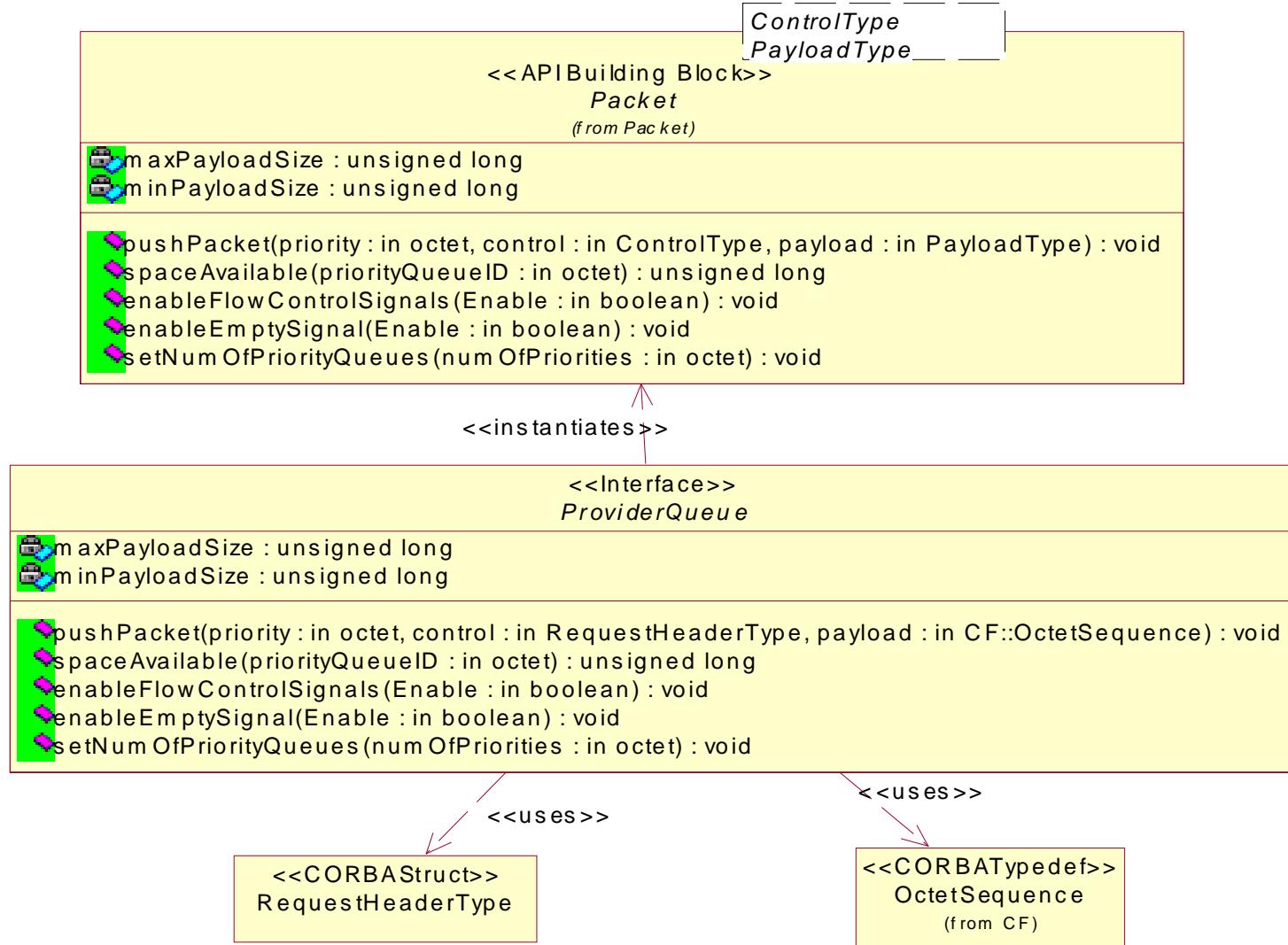
LLC – Connectionless

- Defines an API for a data transfer service without guarantee of delivery.
- Typical use would be as the interface between a waveform link layer or Subnetwork Dependent Convergence Function (SNDCF) and IP.

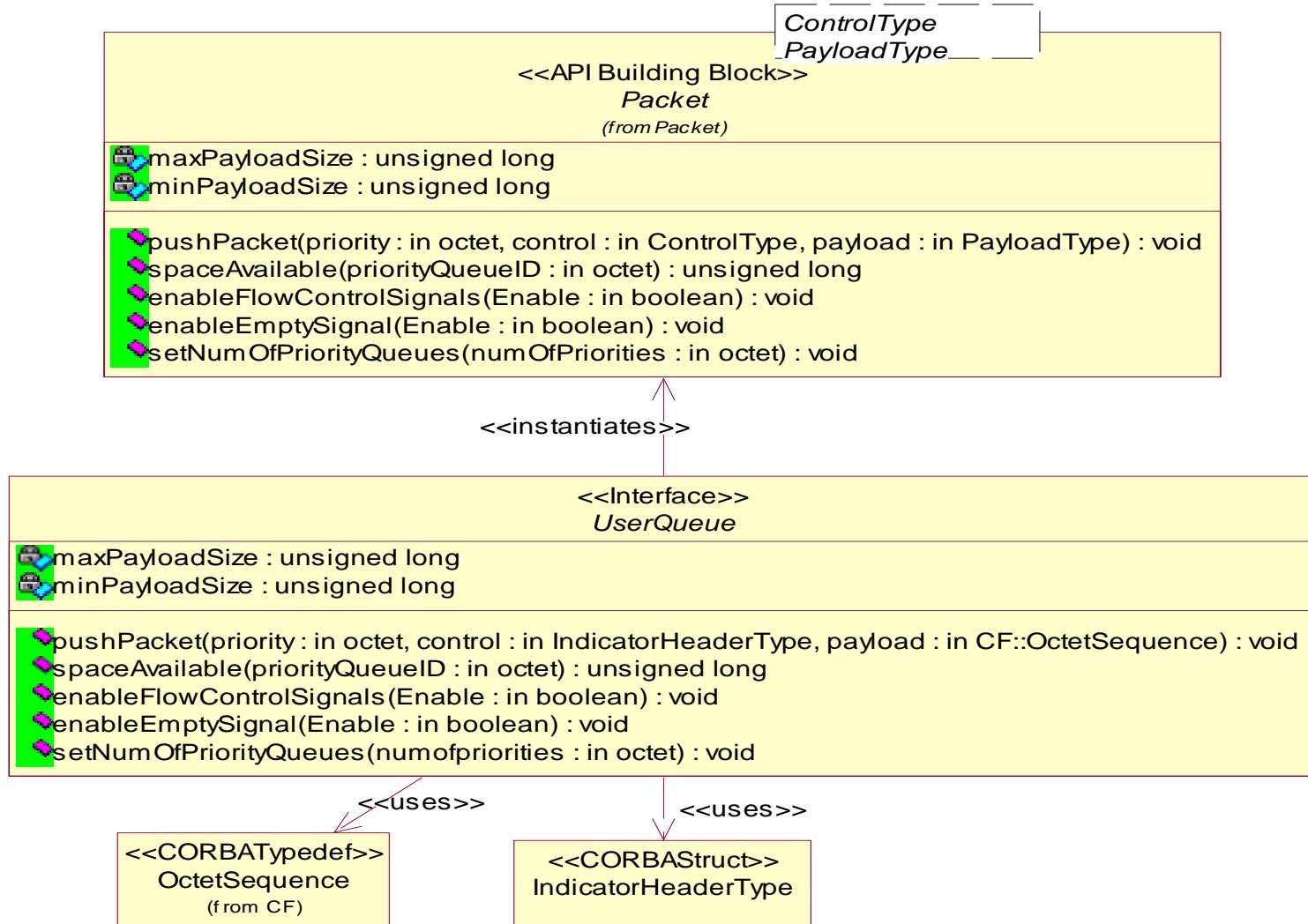
LLC – Connectionless (cont'd)



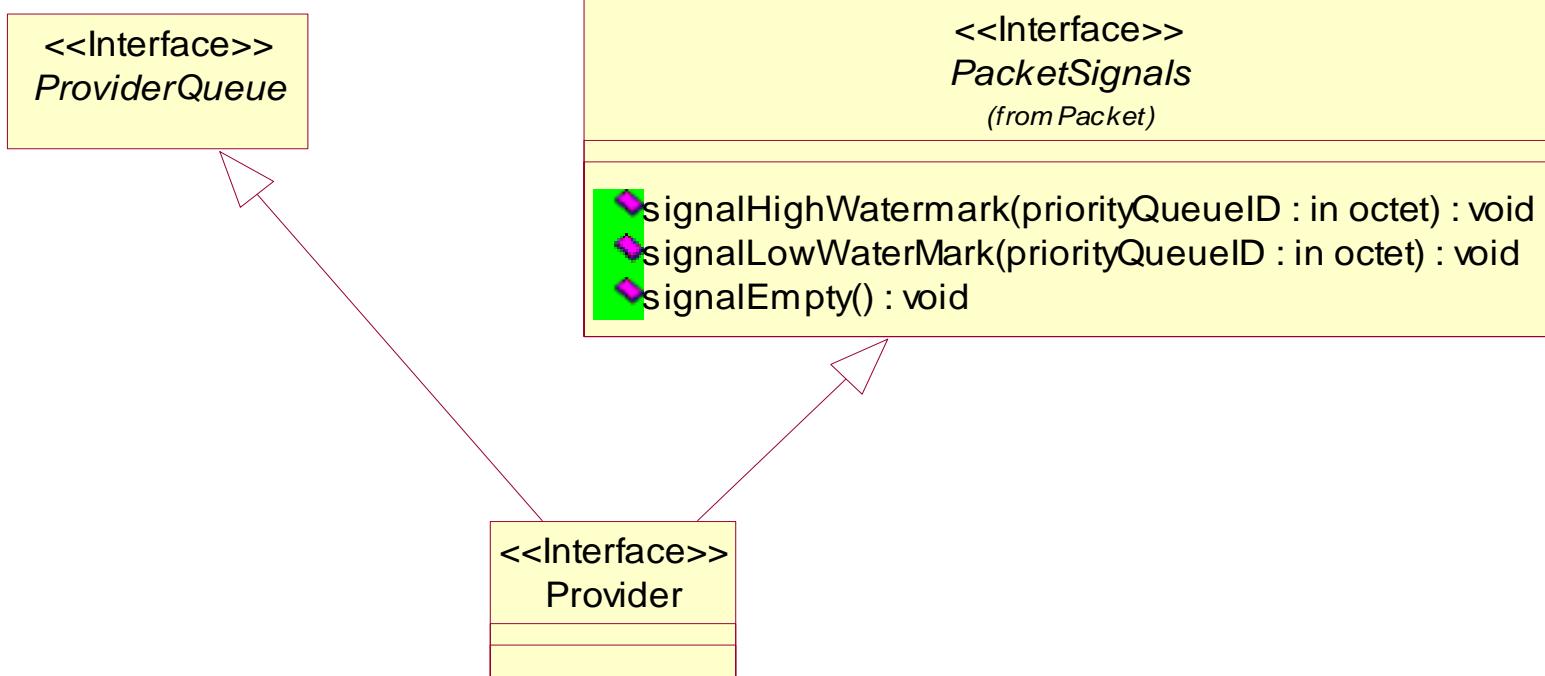
LLC – Connectionless (cont'd)



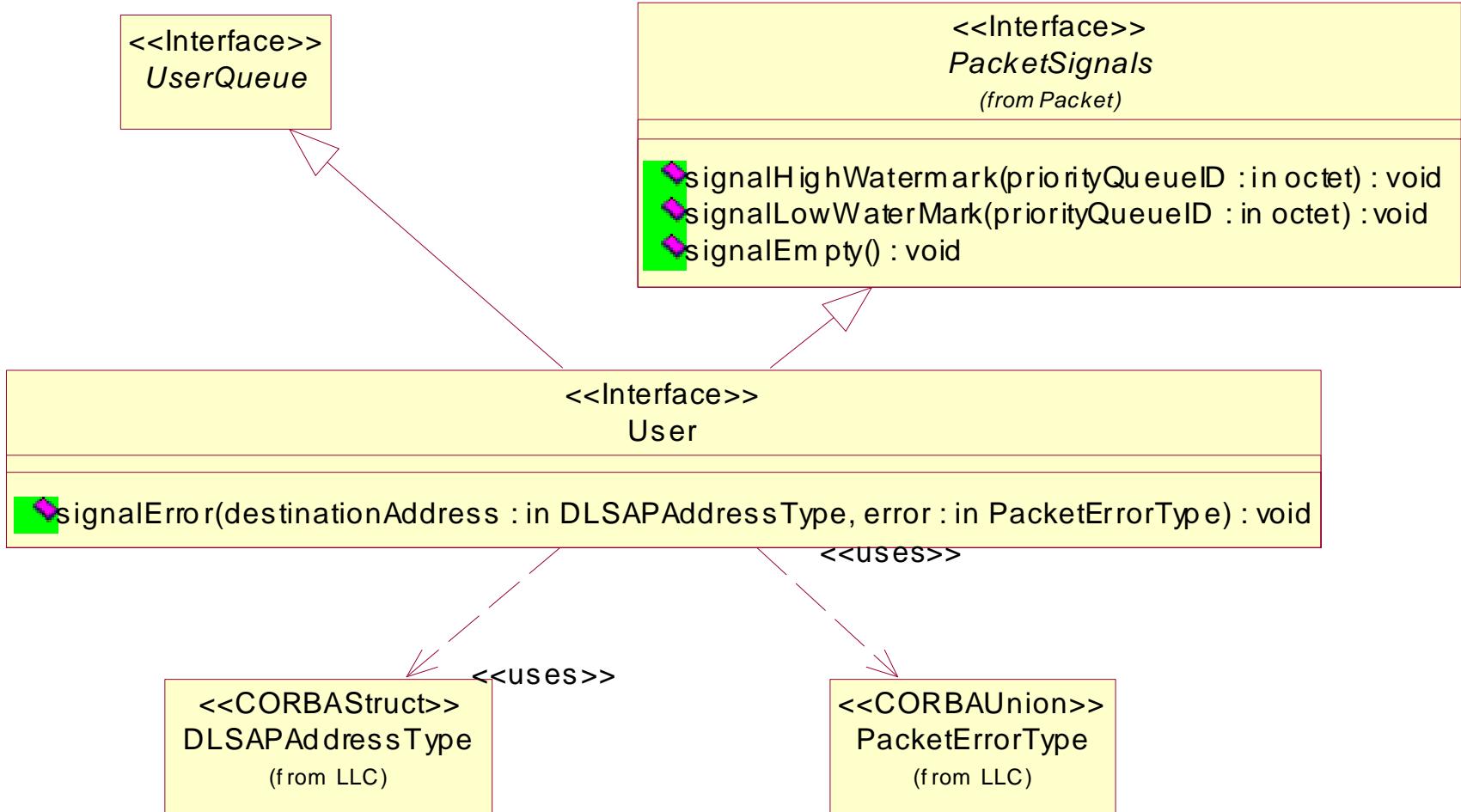
LLC – Connectionless (cont'd)



LLC – Connectionless (cont'd)



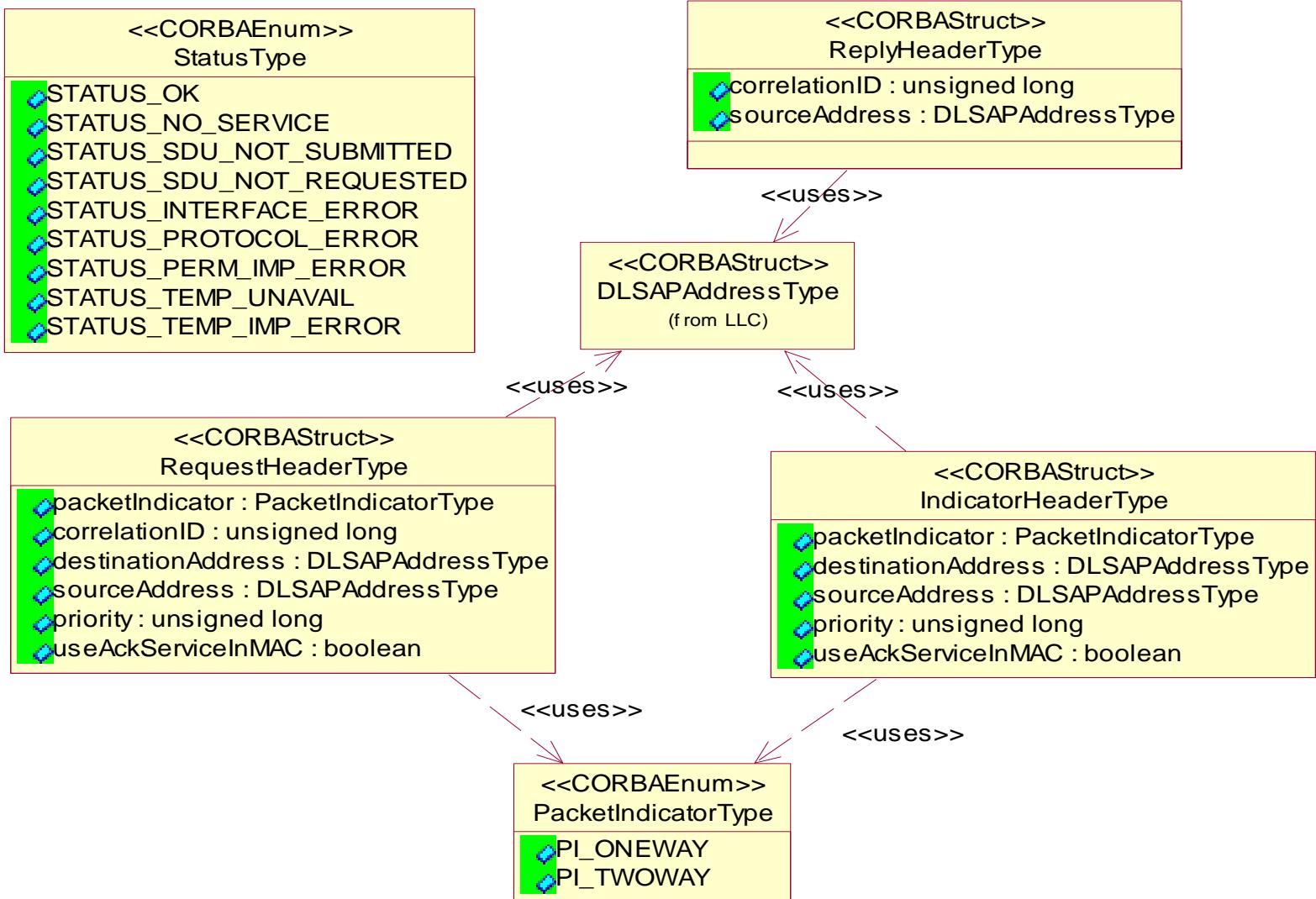
LLC – Connectionless (cont'd)



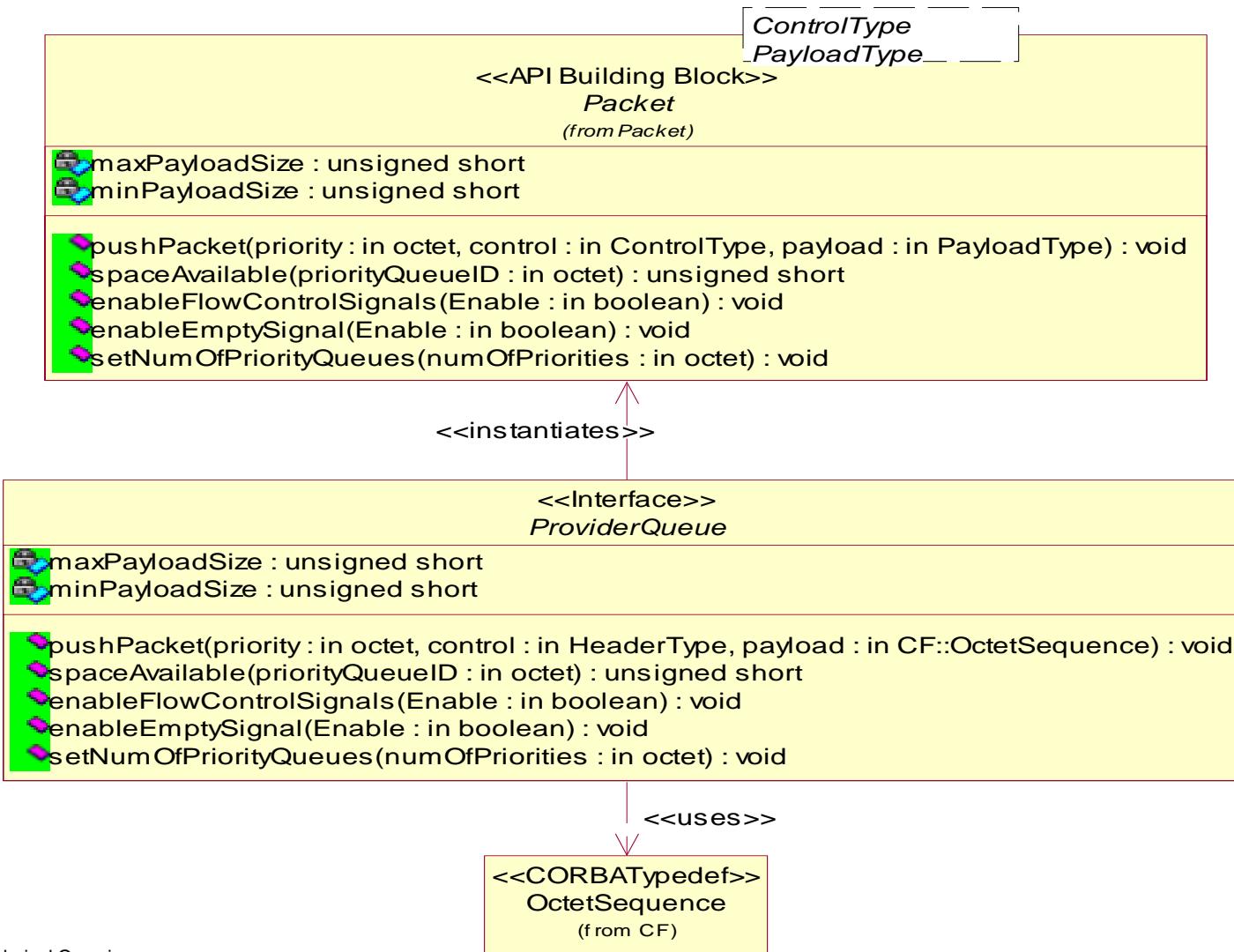
LLC – Acknowledged Connectionless

- Defines an API for a data transfer service between peer Service Users with in-sequence guarantee of delivery.

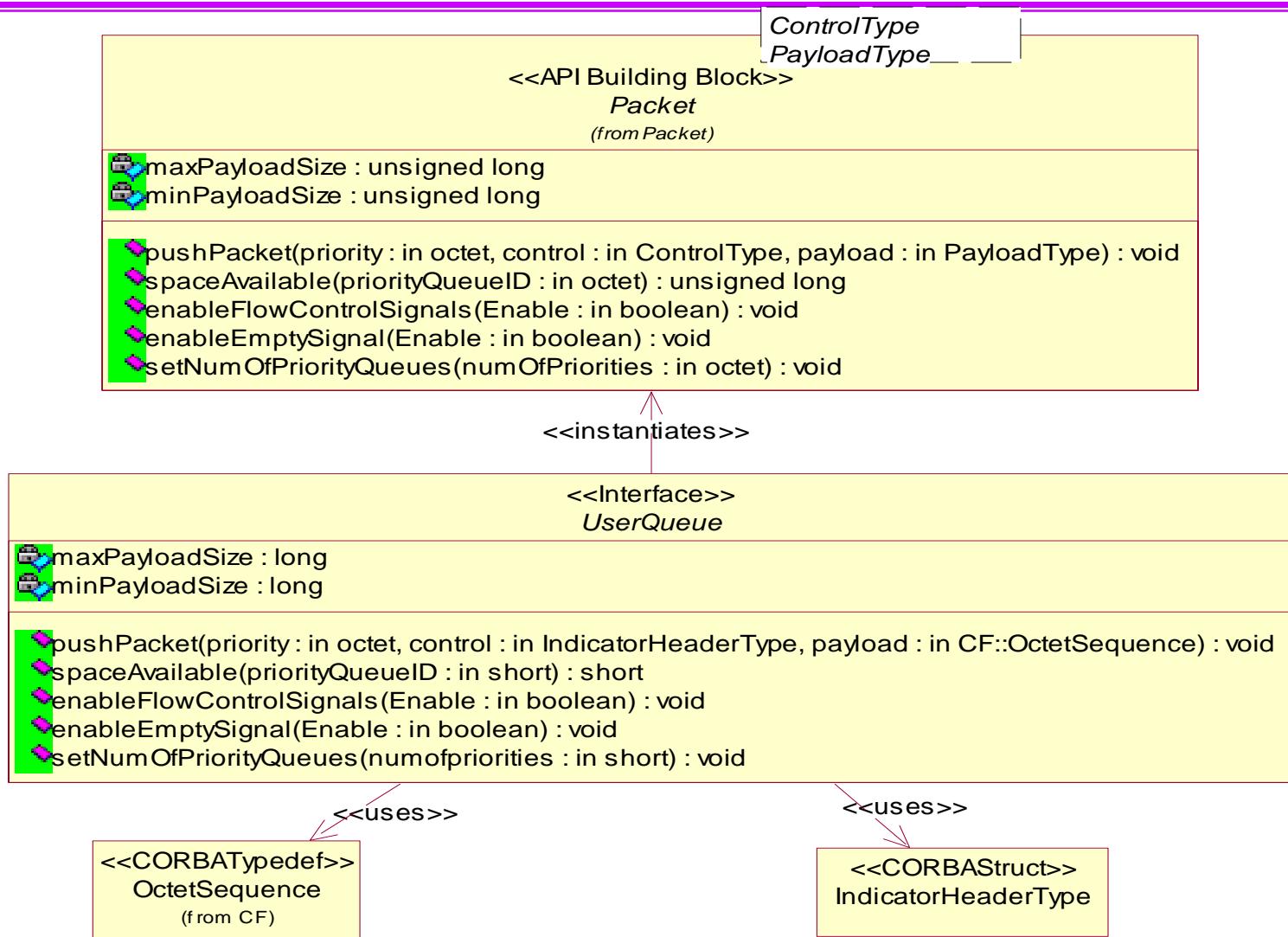
LLC – Acknowledged Connectionless (cont'd)



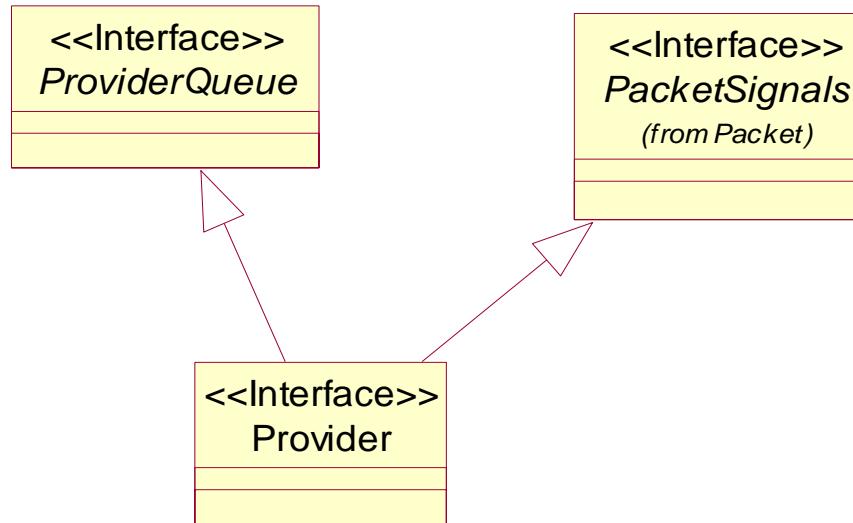
LLC – Acknowledged Connectionless (cont'd)



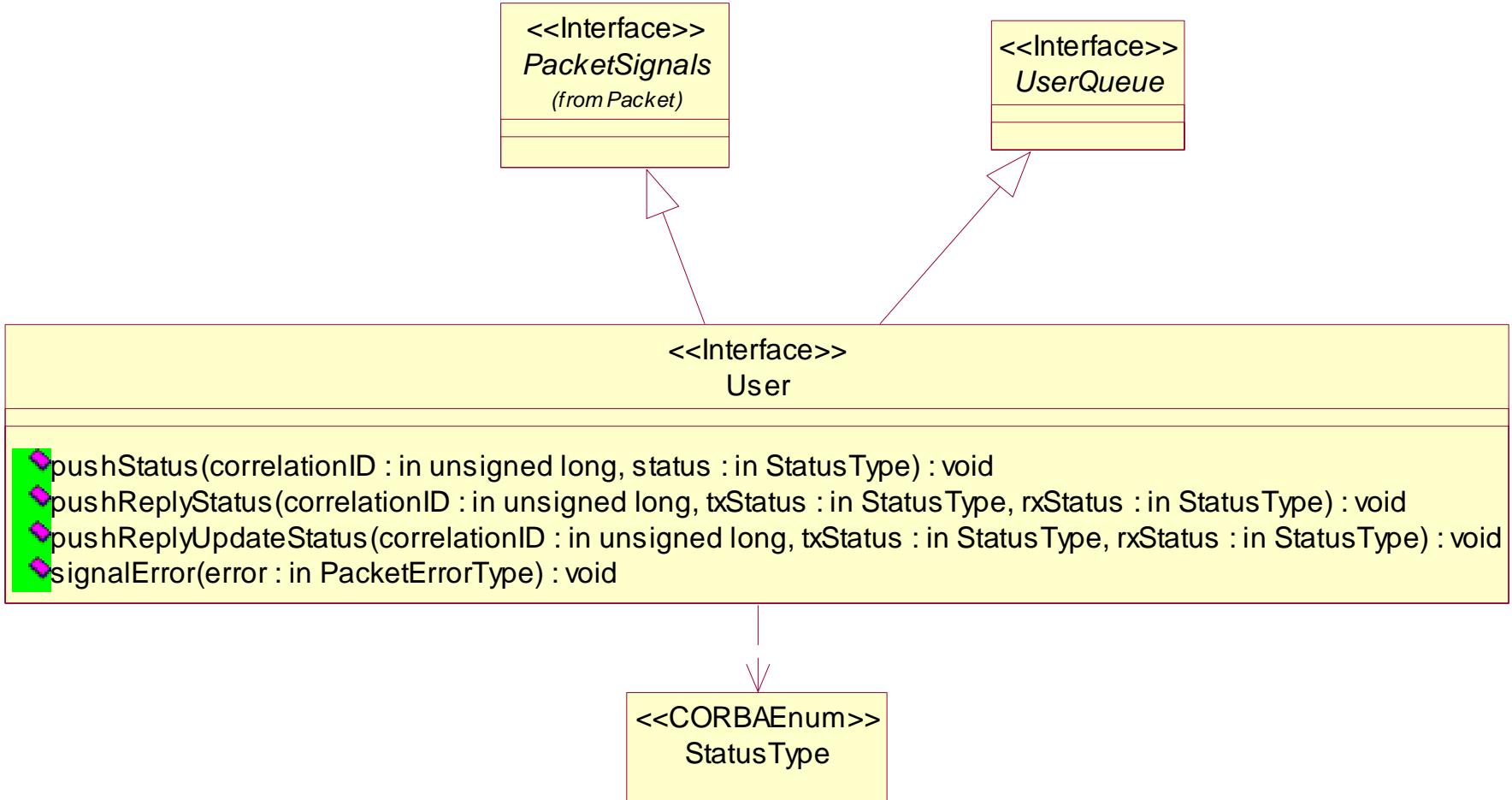
LLC – Acknowledged Connectionless (cont'd)



LLC – Acknowledged Connectionless (cont'd)

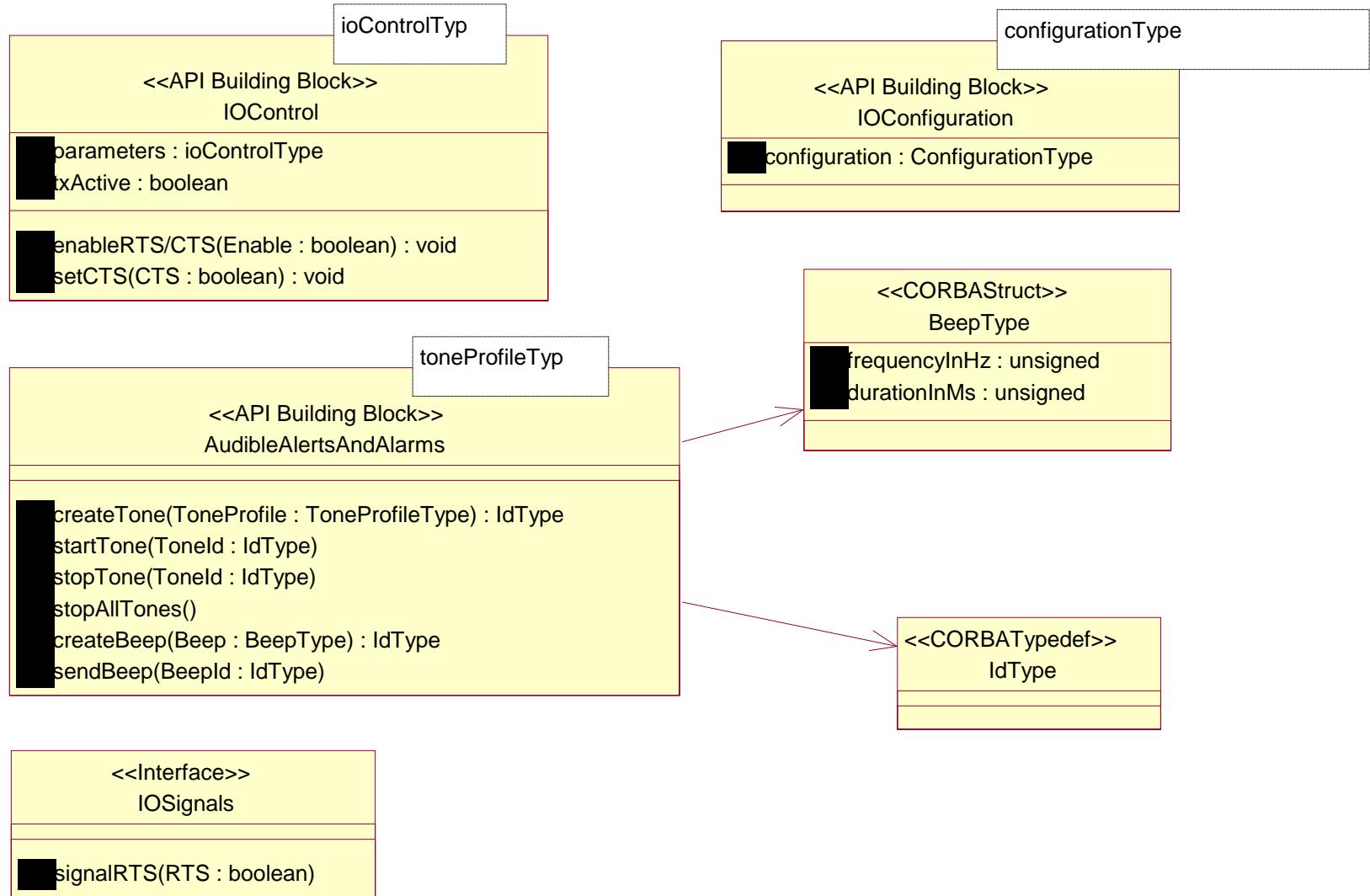


LLC – Acknowledged Connectionless (cont'd)



-
- The I/O building blocks provide the following Services:
 - I/O Control: This building block provides the mechanism for the service user to control the operation of the I/O Device.
 - Audible Alerts and Alarms: This building block provides to the service user the ability to define and generate audible alerts and alarms to the operator.
 - I/O Configuration: This building block provides startup and runtime configuration of the device as necessary.
 - I/O Signals: This building block defines the signals that the I/O device will generate.

I/O (cont'd)



Waveform APIs Defined during JTRS Step 2A

- The following waveform APIs were defined during JTRS Step 2A using the API building blocks
 - Have Quick
 - SINCGARS
 - HF ALE
 - LOS (FM, AM)

API Summary

- An API is a definition and standardization of common interfaces between functional partitions of an SCA Application
- APIs guarantee Service Provider and User can communicate regardless of OE or programming language
- Standardized APIs are essential for portability of applications and interchangeability of devices

Open Discussion