

Application Program Interface Supplement  
to the  
Software Communications Architecture Specification

**APPENDIX D**

**Physical Real-Time  
Building Block Service Definition**

Revision Summary

1.0	Initial Release
-----	-----------------

## Table of Contents

<b>D.1</b>	<b>INTRODUCTION.....</b>	<b>D-1</b>
D.1.1	OVERVIEW .....	D-1
D.1.2	SERVICE LAYER DESCRIPTION. ....	D-1
D.1.3	MODES OF SERVICE.....	D-2
D.1.4	SERVICE STATES. ....	D-2
D.1.5	REFERENCED DOCUMENTS.....	D-2
<b>D.2</b>	<b>UUID.....</b>	<b>D-2</b>
<b>D.3</b>	<b>SERVICES.....</b>	<b>D-3</b>
D.3.1	RECEIVE COMMAND BUILDING BLOCK.....	D-5
D.3.2	CONTROL TYPE/STRUCTURE DEFINITIONS.....	D-6
D.3.2.1	Spread Spectrum Structure.....	D-6
D.3.2.2	Transmit Packet Control Structure.....	D-7
D.3.2.3	Receive Packet Control Structure. ....	D-9
D.3.2.4	Receive Command Structure.....	D-10
D.3.3	EXAMPLES.....	D-11
D.3.3.1	Transmit a packet down stream (to the antenna).....	D-11
D.3.3.2	Explanation of example.....	D-12
D.3.3.3	Typical Exchange between a Service User and Service Provider.....	D-13
<b>D.4</b>	<b>SERVICE PRIMITIVES.....</b>	<b>D-14</b>
D.4.1	RECEIVE COMMAND BUILDING BLOCK.....	D-14
D.4.1.1	Receive.....	D-14
<b>D.5</b>	<b>PRECEDENCE OF SERVICE PRIMITIVES.....</b>	<b>D-15</b>
<b>D.6</b>	<b>SERVICE USER GUIDELINES. ....</b>	<b>D-15</b>
<b>D.7</b>	<b>SERVICE PROVIDER-SPECIFIC INFORMATION.....</b>	<b>D-15</b>
<b>D.8</b>	<b>IDL.....</b>	<b>D-15</b>
<b>D.9</b>	<b>UML. ....</b>	<b>D-15</b>

## Table of Figures

Figure 1. Service Definition Overview .....	D-1
Figure 2. Receive Command Building Block .....	D-6
Figure 3. Receive Command Sequence Diagram.....	D-6
Figure 4. Spread Spectrum Structure .....	D-7
Figure 5. XmitPacketControl structure .....	D-8
Figure 6. RxControl Structure.....	D-9
Figure 7. RcvPacketControl Structure .....	D-10
Figure 8. Creating Control structures.....	D-11
Figure 9. Example Class Diagram of Down Stream Interface.....	D-12
Figure 10. Typical Exchange .....	D-14

## Table of Tables

Table 1. Cross-Reference of Services and Primitives.....	D-3
--	-----

## D.1 INTRODUCTION.

### D.1.1 Overview.

The Physical Real-Time application-programming interface (API) provides standardized interfaces to the Physical layer. Physical Real-Time Services are grouped into Building Block (BB) classes and structures to define abstract services to foster software reuse and commonality between different Service Definition implementations. Physical Real-Time Services provide Service Users with methods to send real-time data and control information between software resource. Also provides methods to signal the Service User that an event has occurred. This document declares only one new BB (i.e., ReceiveCommand), however it provides structures and examples on how the Packet building block is used to build a Physical Real-Time API for a specific waveform. As pictured in Figure 1, this BB defines the **A** interface between the **Physical** and the **MAC**.

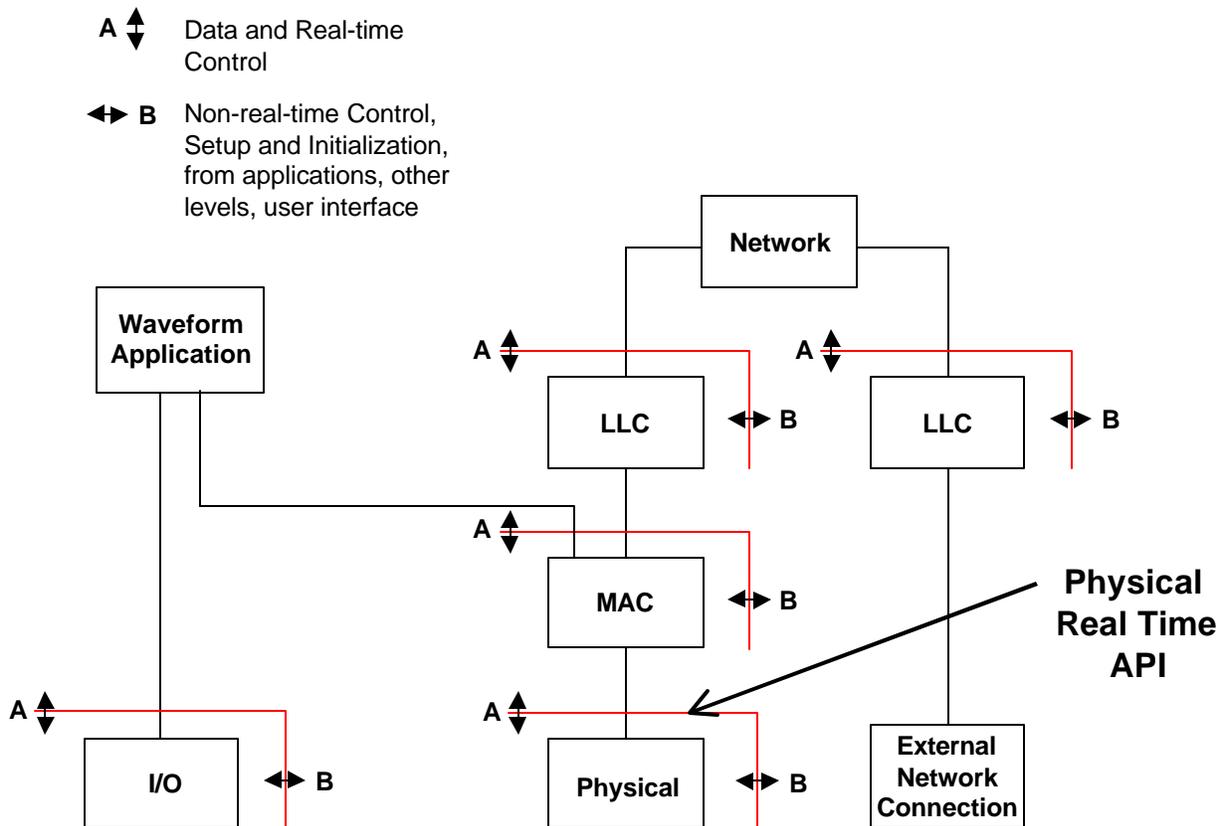


Figure 1. Service Definition Overview

### D.1.2 Service Layer Description.

The Physical Real-Time API Service user is the MAC layer and Service Provider is the Physical layer. The Physical Real-Time API is subdivided into the following Service Groups/BBs:

- Commanded Receive BB
- Control type/structure definitions

- Control Type Structure (general)
  - Transmit Packet Control Structure
  - Receive Packet Control Structure
  - Receive Command Structure
- Spread Spectrum structure

Each BB is an abstract Service Definition that must be instantiated with a concrete type to be realized. In the diagrams, gray colored classes indicate a realized BB. Some Waveform (WF) APIs can use all BBs and type definitions. Some may only use a few of the type definitions while others may create their own and use none of these type definitions.

#### D.1.3 Modes of Service.

There are no specific Modes of Service. This document describes a set of SCA building blocks and structures that provide a generalization of the Physical Real-time layer interface.

#### D.1.4 Service States.

Each building block realization is an instantiation of a parameterized building block, which subsumes the states enumerated in a waveform-specific concrete building block.

#### D.1.5 Referenced Documents.

Software Communication Architecture Specification, 2.0, November 17, 2000.
--

## **D.2 UUID.**

Not Applicable.

### D.3 SERVICES.

**Table 1. Cross-Reference of Services and Primitives**

<b>Service Group</b>	<b>Service</b>	<b>Primitives or Structure Attributes</b>
Command Physical Layer To Receive	Receive Command BB	oneway void receive (in RxCommandType control);
	RcvPacketControl structure	hopTimeOfFirstSymbol numOfSymbols symbolsPerSec frequencyInHz modulation noiseLevel_dBm
Transmit Packets (packets sent from MAC to physical layer to transmit)	Packet BB Instantiated only. Not defined in this API. See Service Definition Description Generic Packet Service Building Block	oneway void pushPacket (in octet priority, in ControlType control, in PayloadType payload);
	XmitPacketControl Control Structure	hopTimeOfFirstSymbol numOfSymbols symbolsPerSec frequencyInHz modulation outputPower_dBm

**Table 1. Cross-Reference of Services and Primitives – Continued**

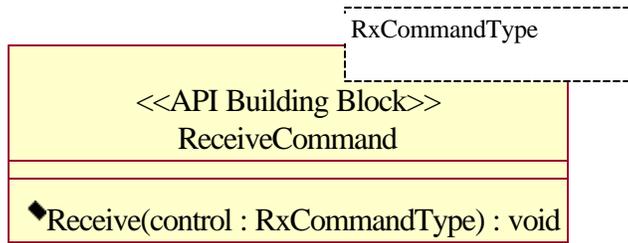
Service Group	Service	Primitives or Structure Attributes
Receive Packets (received packets sent from Physical layer to MAC layer.)	Packet BB Instantiated only. Not defined in this API. See Service Definition Description Generic Packet Service Building Block	oneway void pushPacket (in octet priority, in ControlType control, in PayloadType payload);
	RxControlType Structure	hopTimeOfFirstSymbol numOfSymbols symbolsPerSec frequencyInHz modulation noiseLevel_dBm
Spread Spectrum	Control Spread Spectrum Transmit structure	PNCode PNRate PNOffset
Flow Control	Stream Control structure Not defined in this API. See Service Definition Description Generic Packet Service Building Block	EndOfStream StreamID SequenceNum

**Table 1. Cross-Reference of Services and Primitives – Continued**

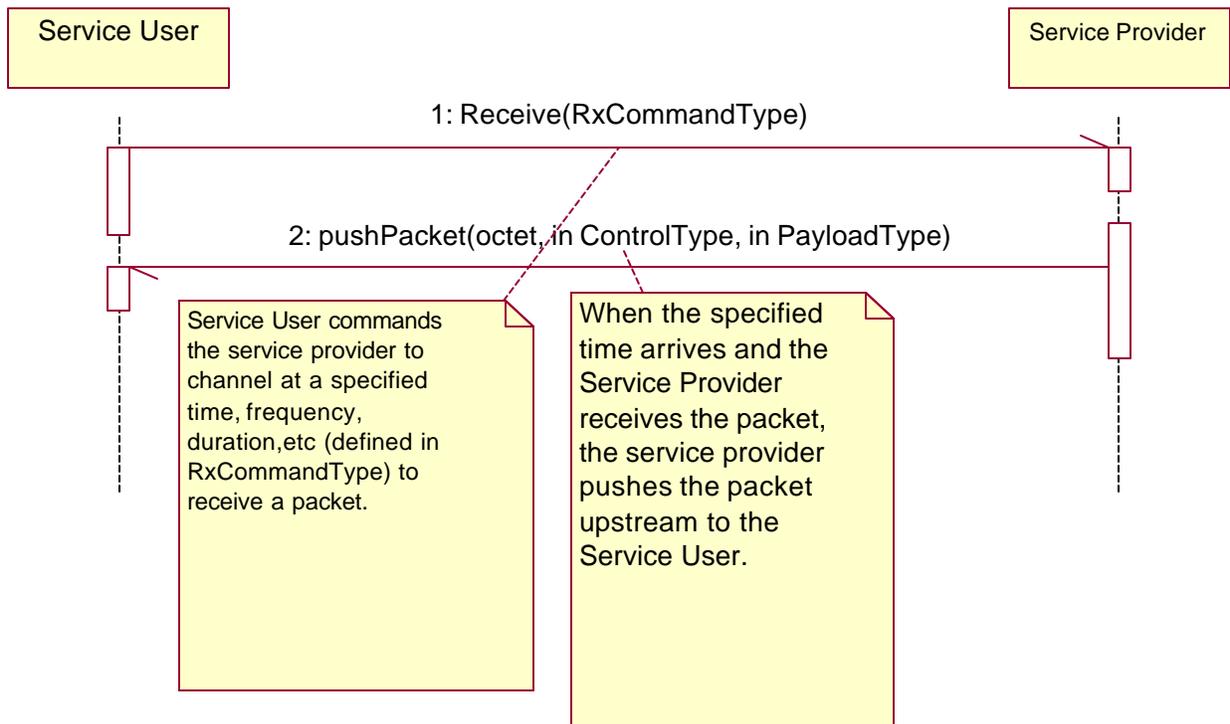
Service Group	Service	Primitives or Structure Attributes
	Packet BB & Signal BB Instantiated only. Not defined in this API. See Service Definition Description Generic Packet Service Building Block	readonly attribute unsigned short minPayloadSize;  readonly attribute unsigned short maxPacketSize;  readonly attribute unsigned short numOfPriorityQueues;  oneway void enableFlowControlSignals (in boolean enable);  unsigned short spaceAvailable (in octet priorityQueueID);  oneway void signalHighWatermark(in octet priorityQueueID );  oneway void signalLowWaterMark(in octet priorityQueueID);  oneway void signalEmpty ();
Quality Of Service	Packet BB & Signal BB Instantiated only. Not defined in this API. See Service Definition Description Generic Packet Service Building Block	oneway void setNumOfPriorityQueues ( in octet numOfPriorities);  oneway void enableFlowControlSignals ( in boolean enable);

D.3.1 Receive Command Building Block.

The Receive Command BB provides a method to command the Physical layer to receive. The specifics of when to receive, and what to receive are defined in ReceiveCommandType.



**Figure 2. Receive Command Building Block**

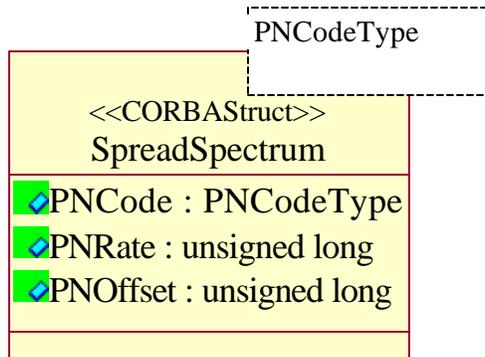


**Figure 3. Receive Command Sequence Diagram**

D.3.2 Control Type/Structure Definitions.

D.3.2.1 Spread Spectrum Structure.

This structure provides Service Users with a structure to communicate spread spectrum information associated with the transmission of a packet. Typically, the XmitPacketControl as part of the control parameter in *pushPacket*, would use this structure.



**Figure 4. Spread Spectrum Structure**

D.3.2.1.1 PNCODE.

PNCODE:PNCODETYPE

PNCODE is a pseudo-random code in terms of units or a table index as defined by the documentation for the specific physical radio.

PNCODETYPE should be defined by the inheriting API.

D.3.2.1.2 PNRATE.

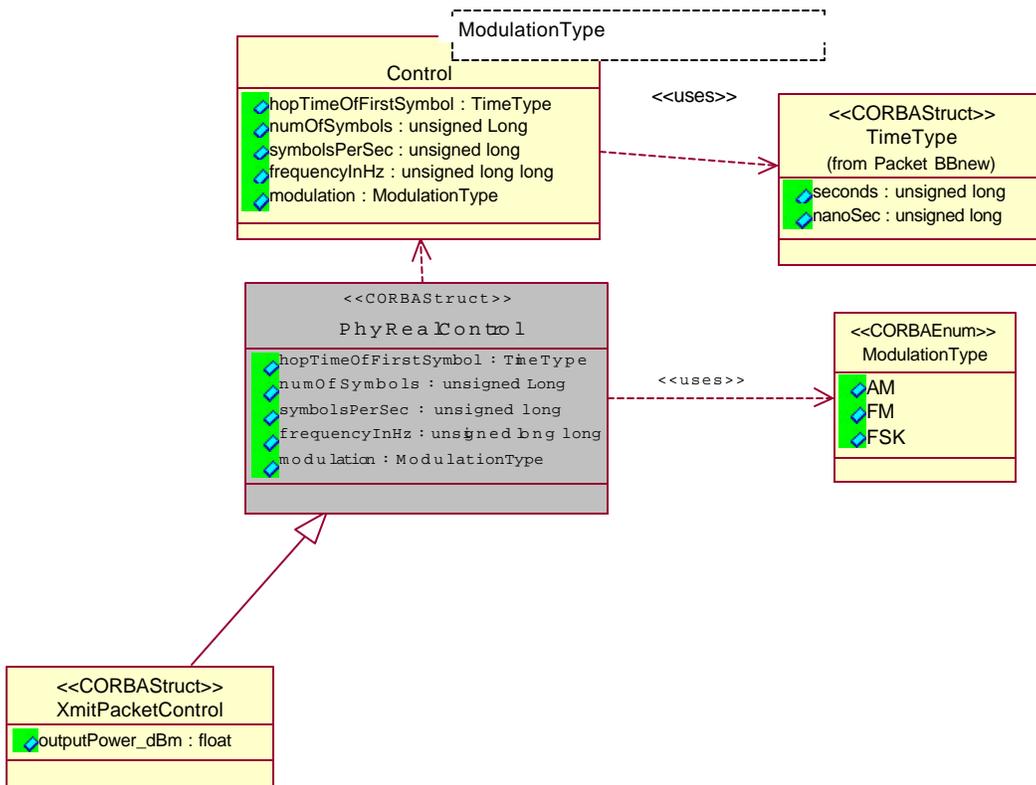
The pseudo-random code rate (e.g., chips/sec) in terms of units or a table index as defined by the documentation for the specific physical radio.

D.3.2.1.3 PNOFFSET.

The offset in a pseudo-random code sequence, in terms of units or a table index as defined by the documentation for the specific physical radio.

D.3.2.2 Transmit Packet Control Structure.

This structure can be used by *pushPacket* to transmit real time data to the Service Provider. Typically, the TxControlType as part of the control parameter in *pushPacket*, would use this structure.



**Figure 5. XmitPacketControl structure**

D.3.2.2.1 HopTimeOfFirstSymbol.

Time stamp in the future when transmit should begin.

D.3.2.2.2 NumOfSymbols.

Number of symbols in this packet.

D.3.2.2.3 SymbolsPerSec.

Number of symbols per second to be transmitted.

D.3.2.2.4 FrequencyInHz.

The frequency to tune for the duration of the hop.

D.3.2.2.5 Modulation.

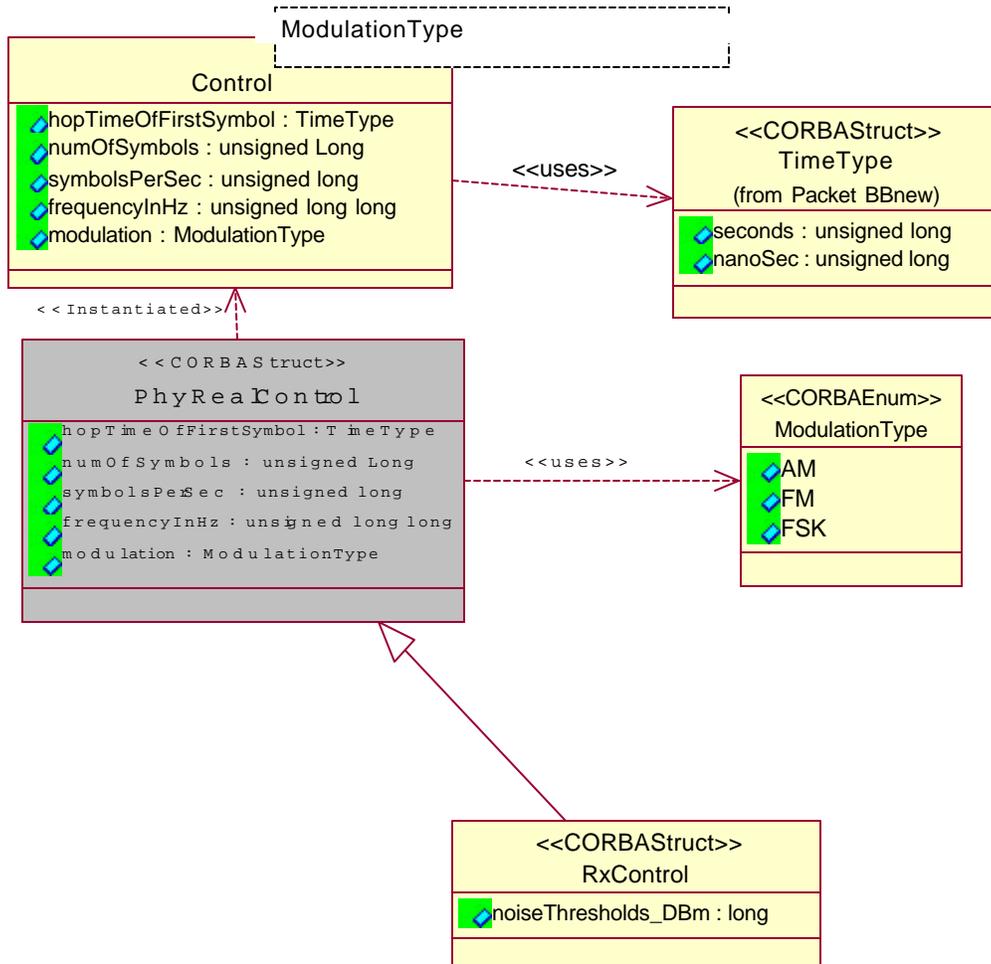
Defined by the inheriting API.

D.3.2.2.6 OutputPower\_dBm.

The RF output power to the antenna in dBm.

### D.3.2.3 Receive Packet Control Structure.

This structure can be used by *pushPacket* to send received data to the Service User. Typically, the RxControlType as part of the control parameter in *pushPacket*, would use this structure.



**Figure 6. RxControl Structure**

#### D.3.2.3.1 HopTimeOfFirstSymbol.

Time stamp in the future when receive should begin.

#### D.3.2.3.2 NumOfSymbols.

Number Symbols received. If the value is -1, there is no limit to the number of symbols to be received. Reception of a packet ends when this number of symbols is received or when the time to end the packet is reached.

D.3.2.3.3 SymbolsPerSec.

Number of symbols per second to be received.

D.3.2.3.4 FrequencyInHz.

The frequency to tune for the duration of the hop.

D.3.2.3.5 Modulation.

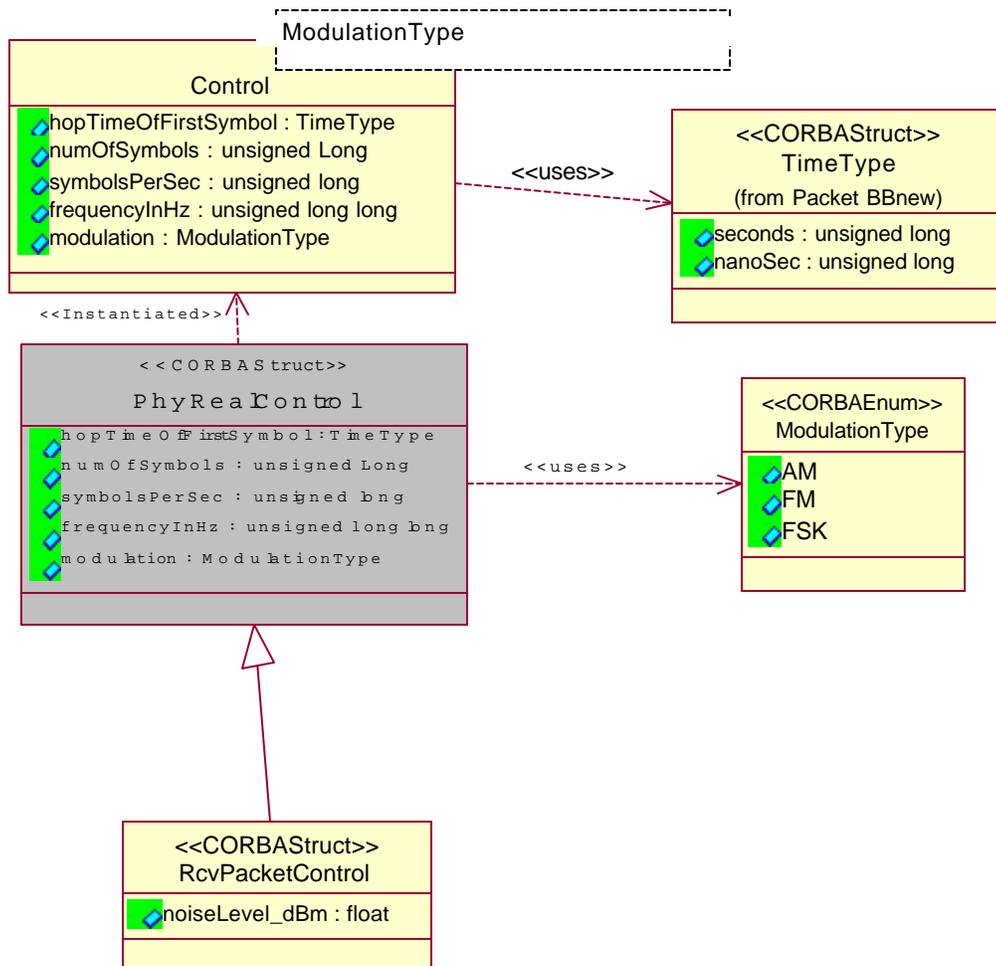
Defined by the inheriting API.

D.3.2.3.6 NoiseThresholds\_dBm.

Noise threshold levels to produce soft decisions.

D.3.2.4 Receive Command Structure.

This structure can be used by *pushPacket* to command the Physical Layer to receive at a specified time. Typically, the packet API as part of the control parameter in *pushPacket*, would use this structure.



**Figure 7. RcvPacketControl Structure**

D.3.2.4.1 HopTimeOfFirstSymbol.

Time stamp in the future when receive should begin.

D.3.2.4.2 NumOfSymbols.

Number Symbols to be received.

D.3.2.4.3 SymbolsPerSec.

Number of symbols per second to be received.

D.3.2.4.4 FrequencyInHz.

The frequency to tune for the duration of the hop.

D.3.2.4.5 Modulation.

Defined by the inheriting API.

D.3.2.4.6 NoiseLevel\_dBm.

Noise level in dBm.

D.3.3 Examples.

The following examples can be used for guidance.

D.3.3.1 Transmit a packet down stream (to the antenna).

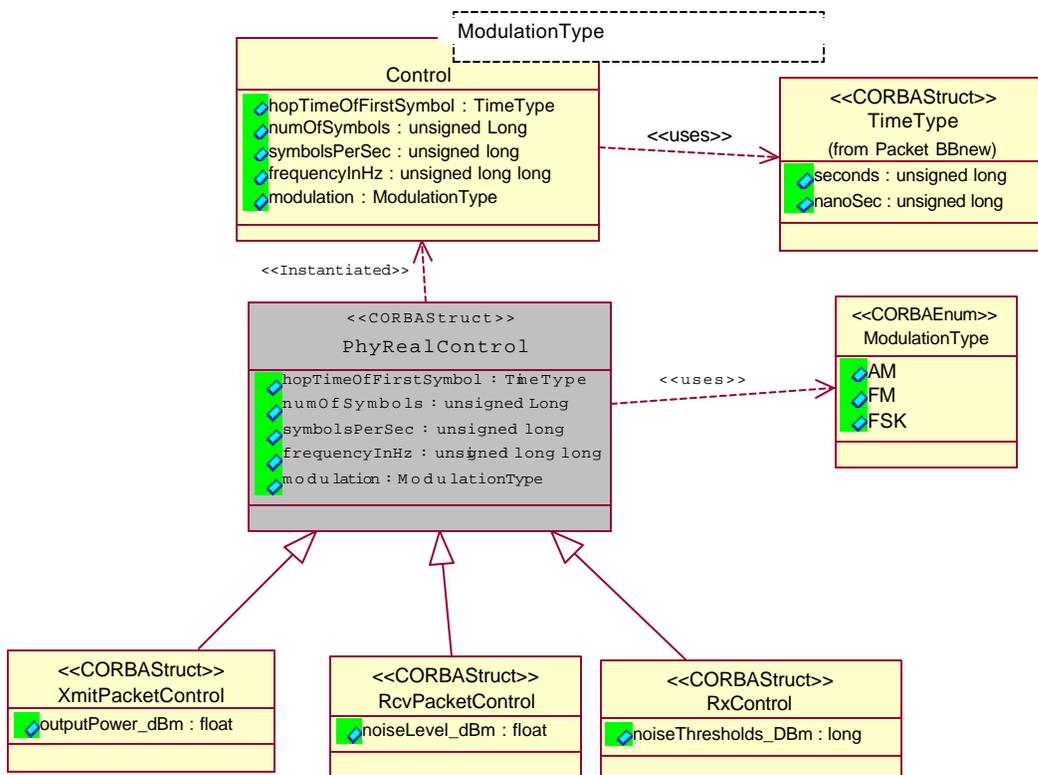
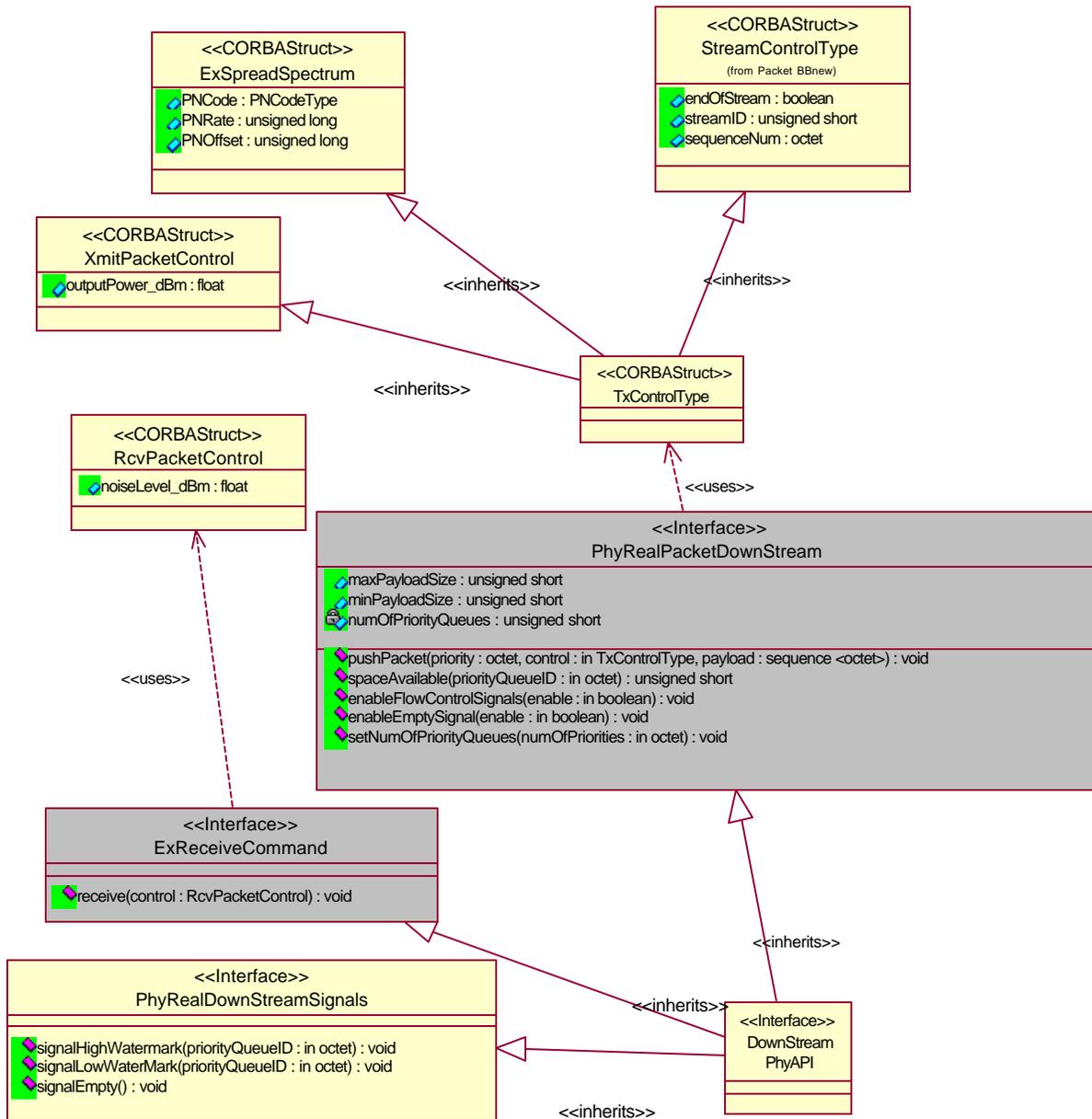


Figure 8. Creating Control structures



**Figure 9. Example Class Diagram of Down Stream Interface**

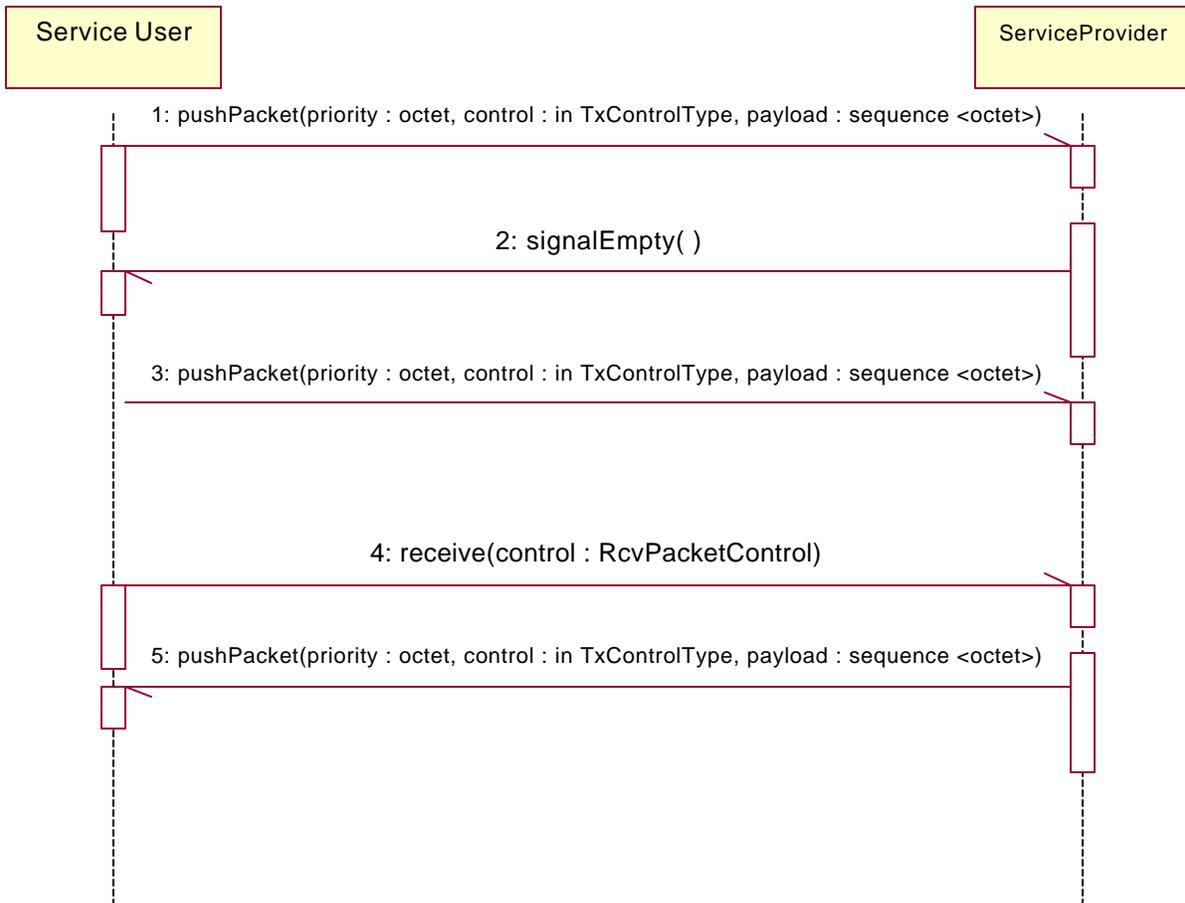
### D.3.3.2 Explanation of example.

In Figure 9, the Class *Control* can be used to build the control structure that will be used for pushing a packet downstream, commanding receive operations, and pushing a packet upstream. *XmitPacketControl* is a realization of *Control: ModulationType* is set to a type specific for this waveform. In Figure 9, *TxControlType* is created by inheriting the classes in order to command

this waveform to transmit a packet at the correct time, frequency, modulation (i.e., *XmitControlPacket*, *ExSpreadSpectrum*, and *StreamControlType*). *PhyRealPacketDownStream* is an instantiation of *Push Packet* that uses the newly created *TxControlType*.

#### D.3.3.3 Typical Exchange between a Service User and Service Provider.

1. In Figure 10, the Service User pushes a packet to the Service Provider to be transmitted. The *priority* field indicates the priority of the packet. The *control* field contains all the information necessary to transmit the packet (e.g., time to start transmitting, number of symbols to transmit, frequency).
2. When the Service Provider has transmitted the packet, the Service Provider notifies the Service User that the queue is empty.
3. The Service User pushes another packet to be transmitted.
4. The Service User commands the Service Provider to receive data. The *control* field contains all of the data necessary (e.g., start of hop time, frequency, modulation, etc) to receive the packet on the RF channel.
5. When the Service Provider has received the packet, the Service Provider pushes the received packet to the Service User. The *priority* field indicates the priority of the packet. The *control* field contains all of the information necessary (e.g., Time of reception, noiseLevel, etc.) for the Service User to process the packet.



**Figure 10. Typical Exchange**

## **D.4 SERVICE PRIMITIVES.**

### **D.4.1 Receive Command Building Block.**

#### **D.4.1.1 Receive.**

Receive provides the ability to command the PHYSICAL REAL-TIME Layer to receive data based on the control parameter.

##### **D.4.1.1.1 Synopsis.**

```
void receive(in RxCommandType control)
```

##### **D.4.1.1.2 Parameters.**

RxCommandType control

RxCommandType is defined by the inheriting API.

D.4.1.1.3 State.

Defined by inheriting API.

D.4.1.1.4 New State.

Defined by inheriting API.

D.4.1.1.5 Response.

No return.

D.4.1.1.6 Originator.

Service User.

D.4.1.1.7 Errors/Exceptions.

Defined by inheriting API.

## **D.5 PRECEDENCE OF SERVICE PRIMITIVES.**

Precedence of primitives is left to service definitions of full APIs.

## **D.6 SERVICE USER GUIDELINES.**

Service User guidelines are left to service definitions of full APIs.

## **D.7 SERVICE PROVIDER-SPECIFIC INFORMATION.**

Not applicable.

## **D.8 IDL.**

The IDL for an API is generated using the parameterized Classes of the building blocks to generate concrete classes with Waveform specific types and attributes. The Building Block documented herein is not intended to be instantiated directly in a UML diagram. The parameterized classes define the attributes and data types that are unique for each waveform.

The parameterized class is a template from which to generate user specific API definitions. To generate valid IDL from this Building Block, a concrete class must be generated that replaces the parameterized items with the user specific types and attributes. The completed UML diagram will not contain any reference to parameterized classes. Only concrete classes that were derived from them.

## **D.9 UML.**

This appendix includes the UML class diagrams for the Service Definition. The purpose for including these diagrams is to show the relationship between all the elements of the PHYSICAL REAL-TIME Services.

