JTRS HF ALE Physical API Service Definition

V1.0
December 15, 2000

Prepared for the
Joint Tactical Radio System (JTRS) Joint Program Office

Prepared by the
Modular Software-programmable Radio Consortium
Under Contract No. DAAB15-00-3-0001

Revision Summary

1.0	Initial release

Table Of Contents

1 INTRODUCTION 1
1.1 OVERVIEW 1
1.2 SERVICE LAYER DESCRIPTION 1
1.3 MODES OF SERVICE 2
1.4 SERVICE STATES 2
1.5 REFERENCED DOCUMENTS 3
2 UUID 3
3 SERVICES 3
3.1 TRANSCEIVER CONFIGURATION 3
3.2 REAL-TIME CONTROL 5
3.2.1 RxALEControl 6
3.2.2 TxALEControl 6
3.2.3 PhysicalToMACSignals 7
3.3 ALEMEDIA SETUP 7
3.3.1 ChannelSetup 7
3.3.2 GetRadioConfiguration 7
3.4 RXPACKET 9
3.4.1 RxPktControlType 10
3.4.2 ALEPayloadType 10
3.5 TXPACKET 11
3.5.1 TxPktControlType 11
3.5.2 ALEPayloadType 11
3.6 HF-ALE PHYSICAL LAYER API 12
4 SERVICE PRIMITIVES 13
4.1 TRANSCEIVER CONFIGURATION 13
4.1.1 SetUpReceiverParams Service 13
4.1.2 SetUpTramsitterParams Service 13
4.1.3 SetRFpathHomed Service 14
4.2 ALE MEDIA INTERFACE (INHERITS FROM ALE MEDIA SETUP) 15
4.2.1 setupMediaType Service 15
4.2.2 getRadioConfiguration Service 15
4.3 RX ALE CONTROL 16
4.3.1 Receive Service 16
4.4 TX ALE CONTROL 17
4.4.1 Set Frequency Service 17
4.4.2 TuneTxPath Service 17
4.4.3 GetTuneStatus Service 18
4.5 PHYSICAL TO MAC SIGNALS 18
4.5.1 TuneComplete Service 18
4.5.2 SquelchEvent Service 19
4.6 TX/RX PACKETS 19

4.6.1 Get minPayloadSize / Get MaxPayloadSize Service. 19
4.6.2 pushPacket Service 20
4.6.3 SpaceAvailable Service. 21
4.6.4 EnableFlowControl Service. 21
4.6.5 EnableEmptySignal Service. 22
4.6.6 setNumOfPriorityQueues Service. 22
4.7 TX/RX PACKET SIGNALS. 23
4.7.1 signal HighWatermark. 23
4.7.2 signal Low Watermark. 23
4.7.3 signal Empty. 24

5 ALLOWABLE SEQUENCE OF SERVICE PRIMITIVES. 25

6 PRECEDENCE OF SERVICE PRIMITIVES. 25

7 SERVICE USER GUIDELINES. 25

8 SERVICE PROVIDER-SPECIFIC INFORMATION. 25

9 IDL. 25

9.1 IDL FOR ALE MEDIA SETUP. 25
9.2 IDL FOR ALE PHYSICAL API. 26
9.3 IDL FOR ALE TRANSCEIVER INTERFACE. 27
9.4 IDL FOR PACKET SIGNALS INTERFACE. 28
9.5 IDL FOR PHYSICAL REAL TIME. 29
9.6 IDL FOR PHYSICAL RECEIVE PACKET. 30
9.7 IDL FOR PHYSICAL TRANSMIT PACKET. 31

10 UML. 34

List of Tables

Table 1. Cross-Reference of Services and Primitives 3

List of Figures

Figure 3-1. ALETranceiverConfig 4
Figure 3-2. Real-Time Control 6
Figure 3-3. HF-ALE Media Setup 9
Figure 3-4. Rx PacketInterface 10
Figure 3-5. HF-ALE Physical API 12
Figure 10-1. Physical Layer Tranceiver Config Interface for HF ALE 34
Figure 10-2. Physical Layer Media Setup Interface for HF ALE 35
Figure 10-3. Physical Layer Real-Time Control Interfaces 36
Figure 10-4. Physical Layer Upstream Packet Interface 37
Figure 10-5. Physical Layer Downstream Packet Interface 38
Figure 10-6. HF ALE Physical Layer API 39
Figure 10-7. HF ALE Component Diagram 40

1 INTRODUCTION.

1.1 OVERVIEW.

This document specifies an SCA conformant Physical Layer API for the HF-ALE waveform. This document provides the instantiation of the Physical Real-time and Physical Non-real-time building blocks to define an HF-ALE Physical Layer Service provider. The HF-ALE Physical Layer API provides the following services:

1. Transceiver Setup: This part of the API provides the setup parameters for the current Tx and Rx functions. These are generally non-real-time parameters that are configured at function start or infrequently during function operation.
2. Media Setup: This service allows configuration of each available RF channel for proper receipt and transmission of HF signals.
3. Receive Functions Control : This interface provides the real-time control of receive functions.
4. Transmit Functions Control : This interface provides the real-time control of transmit functions.
5. Physical-to-MAC Signals: This interface defines the asynchronous signals that the Physical Layer implementation will generate to the HF-ALE MAC Layer.
6. RxPacket : This provides the interface for the transfer of data from the Physical Layer to the HF-ALE MAC Layer.
7. TxPacket : This interface provides the interface for the transfer of data from the Physical Layer to the HF-ALE MAC Layer.

Packet interfaces are derived from the standard building blocks defined for data transfer in MSRC-5000SCA Packet Building Block.

1.2 SERVICE LAYER DESCRIPTION.

The HF-ALE Physical Layer API provides a common Physical Layer interface for all HF applications that can run in conjunction with ALE. The data interface is standardized to a sequence of 16-bit analog samples at a rate of 8 kHz.

The service layer provides for either independent sideband (ISB) or non-ISB operation. ISB operation allows for up to four ISB channels. Non-ISB operation supports one channel of either AM, FM or CW operation.

The ALE Media Setup interface provides the ability to read the capabilities of the Physical Layer, and configure the available radio channels for operation.

The real-time Physical Layer interfaces provide control and status of the transmit and receive functions during operation. This interface provides functionality such as setting the radio frequency and generating a squelch event to the HF-ALE MAC Layer.

Data is passed to and from the Physical Layer via instances of the Packet Building Block.

1.3 MODES OF SERVICE.

The HF-ALE Physical Layer can operate in one of the following four modes.

1. ISB (Independent Sideband)
2. AM (Amplitude Modulation)
3. FM (Frequency Modulation)
4. CW (Continuous Waveform)

1.4 SERVICE STATES.

Each mode of operation has a state of Transmit or Receive.

ISB mode can operate in any one of the following states.

1. One channel ISB
2. Two Channel ISB
3. Four Channel ISB

Each ISB channel can operate as an ISB channel or as an AME channel.

1.5 REFERENCED DOCUMENTS.

Document No.	Document Title
MSRC-5000SCA	Software Communications Architecture Specification

2 UUID.

The UUID for this API is `747eff90-d1d3-11d4-8cc8-00104b23b8a2.`

3 SERVICES.

The features of the interface are defined in terms of the services provided by the Service Provider, and the individual primitives that may flow between the Service User and Service Provider.

The services are tabulated in table 1 and described more fully in the remainder of this section.

Table 1. Cross-Reference of Services and Primitives

Service Group	Service	Primitives
Transceiver Configuration	Configuration Parameters	SetupReceiverParams
		SetupTransmitterParams
Real-time Control	Rx Control	Receive
	Tx Control	Set Frequency
		TuneTxPath
		GetTuneStatus
	Physical to MAC Signals	TuneComplete
		SquelchEvent
Media Setup	SetupMedia	SetupMediaType

3.1 TRANSCEIVER CONFIGURATION

The following describes the HF-ALE specific instantiation of the TransceiverConfig building block.

The ALETransceiverConfig interface (Figure 3-1) is constructed by instantiating the TransceiverSetup building block and extending that interface to include a method for "Homing" the RF path configuration.

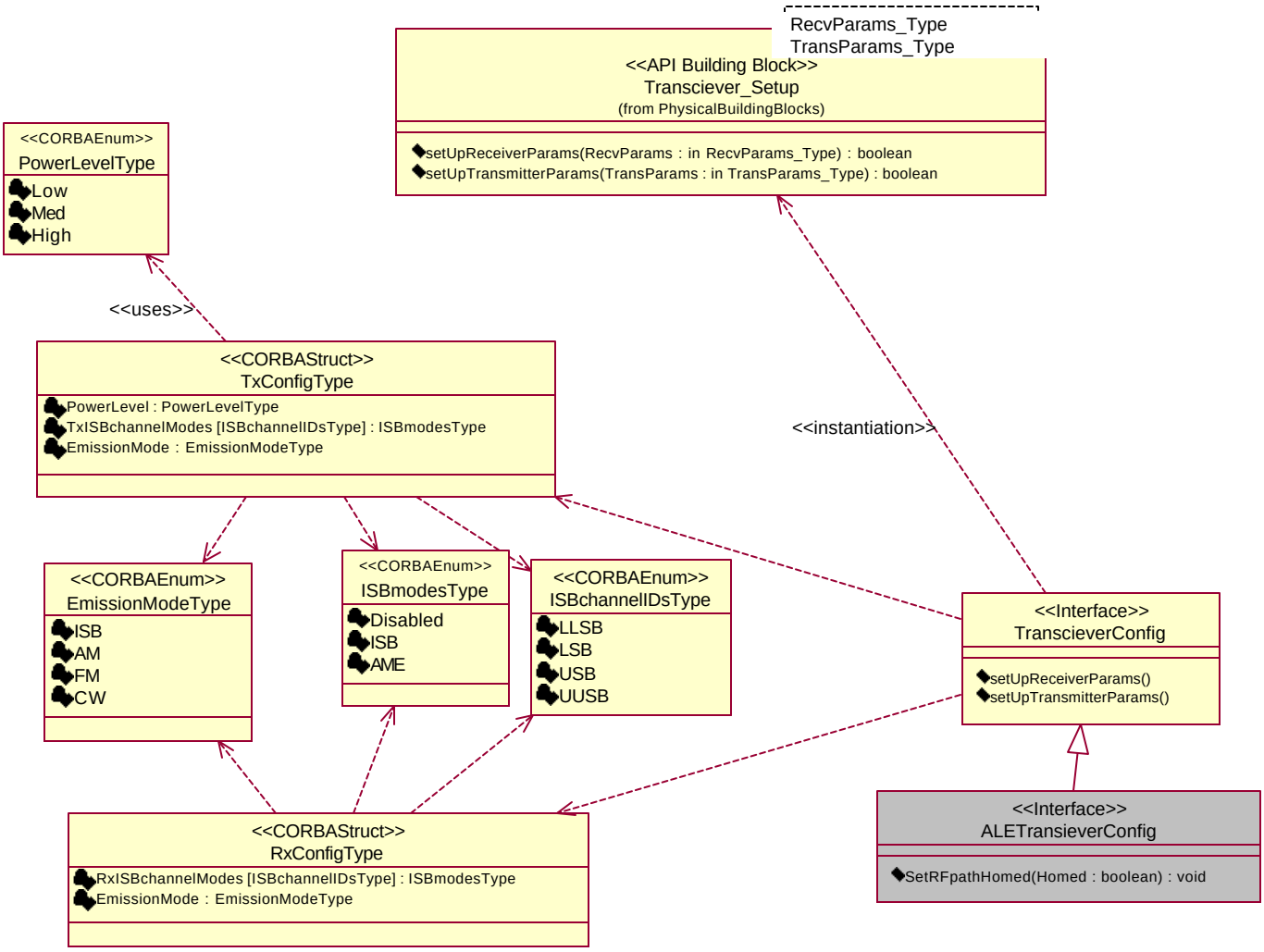

Figure 3-1. ALETranceiverConfig

TxConfigType: This type defines the transmitter configuration.

PowerLevel : Discrete power levels of High,Med,Low

TxISBchannelModes : This item allows configuration of each of the four possible ISB channels to a state of disabled, ISB or AME operation.

EmissionMode : This item configures the radio to ISB, AM, FM or CW operation.

RxConfigType: This type defines the receiver configuration.

RxISBchannelModes : This item allows configuration of each of the four possible ISB channels to a state of disabled, ISB or AME operation.

EmissionMode : This item configures the radio to ISB, AM, FM or CW operation.

If the emission mode is ISB, then the ISBchannelModes are in effect. The default channel mode is ISB for each of the four channels. If the emission mode is not ISB, then the radio operates on a single channel in the selected mode (AM, FM, or CW).

The extension "SetRFpathHomed (Homed : boolean) : void" places the physical layer in a state that allows reception of signals in the entire HF band. This is necessary because during transmit functions, the physical layer is tuned to a specific frequency. If the physical layer is not "homed", signals outside the tuned frequency will not be received.

3.2 REAL-TIME CONTROL.

Real-time control for HF-ALE has three components (see Figure 3-2).

1. RxALEControl: This interface is an instantiation of the RxControlType building block. The instantiation defines Frequency as the only Rx Control information.
2. TxALEControl: This interface is an HF-ALE specific interface. This interface allows setting of the Tx frequency, commanding the Physical Layer to tune the Tx path, and fetching the tune status.
3. PhysicalToMacSignals: This interface is specific to HF-ALE, and defines the asynchronous signals that the HF-ALE Physical Layer implementation will generate back to the HF-ALE MAC Layer implementation. The signals generated are "TuneComplete" and "SquelchEvent".

The details of these interfaces are described in the following paragraphs.

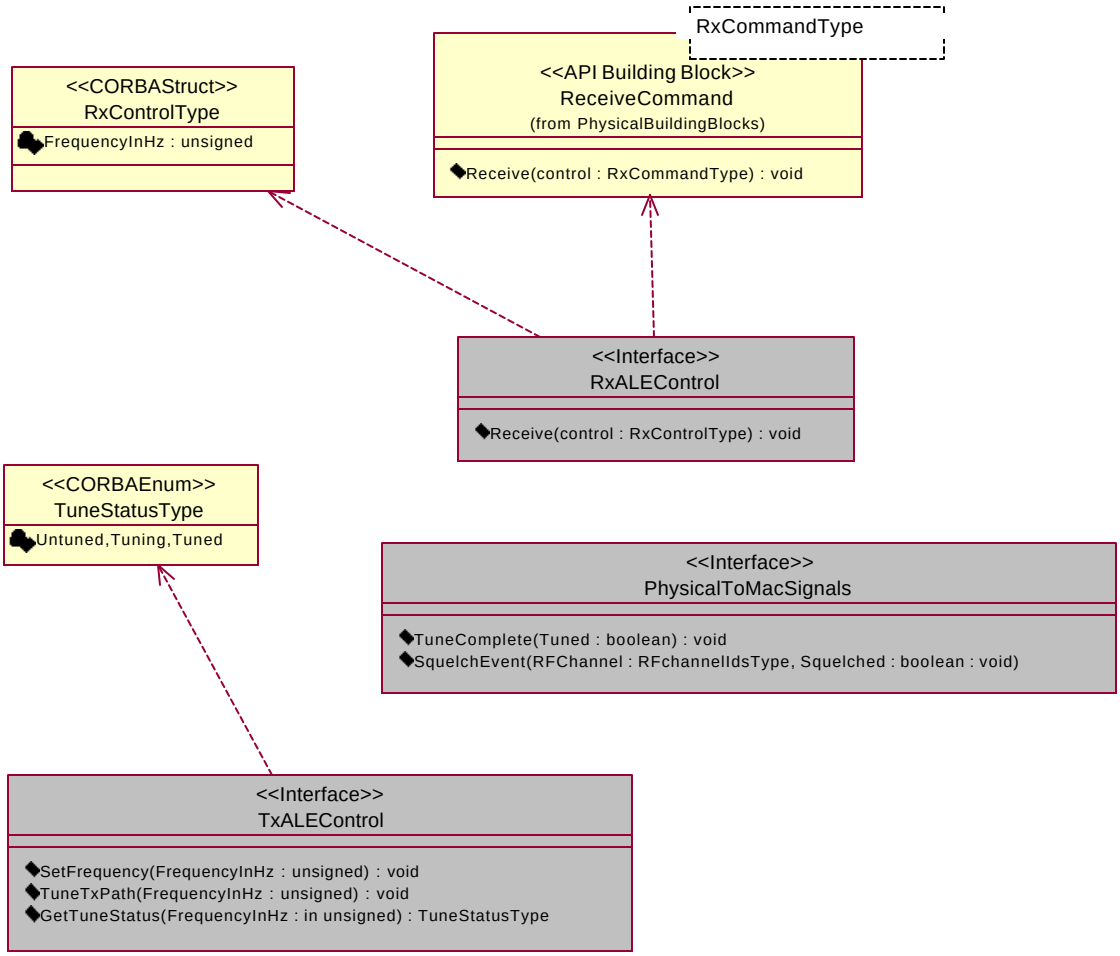

Figure 3-2. Real-Time Control

3.2.1 RxALEControl.

This interface defines the required control of the Rx channel. The only item controllable through this interface is the Rx frequency, specified in Hertz.

3.2.2 TxALEControl.

This interface defines the required control of the Tx channel - SetFrequency, TuneTxPath, and GetTuneStatus.

3.2.2.1 SetFrequency

This method causes the physical layer to tune the transmitter to the specified frequency.

3.2.2.2 TuneTxPath

This method forces the physical layer to tune the RF circuits to facilitate proper

tramsisison on the selected Tx Frequency.

3.2.2.3 GetTuneStatus

This method provides the tune status of the physical layer. The service user requests the tune status for a specific frequency. GetTuneStatus returns a boolean indicating whether or not the RF circuits are tuned to the requested frequency.

3.2.3 PhysicalToMACSignals.

This interface defines the asynchronous HF-ALE specific signals generated by the Physical layer, and serviced by the MAC layer.

3.2.3.1 TuneComplete

This method is called by the Physical Layer when a previously received tune command has beencompleted. A status is returned indicating whether or not the rune was complete.

3.2.3.2 SquelchEvent

This method is called by the Physical Layer when the squelch state of the indicated receive channel has changed.

3.3 ALEMEDIA SETUP.

This interface (Figure 3-3) provides configuration of each HF-ALE radio channel. The MediaSetup building block is instantiated with the ALE specific setup type ALEmediaParamsType.

Interface ALEMediaInterface extends the building block by adding the method "getRadioConfiguration". This method provides the installation configuration to the MAC layer, allowing the MAC layer to properly configure the radio based on the radios capabilities.

3.3.1 ChannelSetup

This type defines the setup values for the each of the four ISB channels and for the Non-Isb channel. Channel configuration consists of SquelchSensitivity and Datamode configuration.

3.3.2 GetRadioConfiguration

This method is used by the MAC layer to fetch the capabilities of the physical layer. The capabilities are returned as a structure in which each capability is tagged as either active or inactive. The following paragraphs describe the Radio’s configuration items.

3.3.2.1 NumOfISBchannels : boolean

This element denotes the number of ISB channels that are available when the radio is in ISB Mode. Possible values are One, Two or Four channels.

3.3.2.2 UpperSB : boolean

This element denotes whether or not the Upper Side Band channel is available for use as an ISB channel.

3.3.2.3 LowerSB : boolean

This element denotes whether or not the Lower Side Band channel is available for use as an ISB channel.

3.3.2.4 UpperUSB : boolean

This element denotes whether or not the Upper/Upper Side Band channel is available for use as an ISB channel.

3.3.2.5 LowerLSB : boolean

This element denotes whether or not the Lower/Lower Side Band channel is available for use as an ISB channel.

3.3.2.6 Continuous Waveform : boolean

This element denotes whether or not the radio is capable of CW communications when in non-ISB mode.

3.3.2.7 AM : boolean

This element denotes whether or not the radio is capable of AM communications when in non-ISB mode.

3.3.2.8 FM : boolean

This element denotes whether or not the radio is capable of FM communications when in non-ISB mode.

3.3.2.9 AME : boolean

This element denotes whether or not the radio is capable of AME communications when in non-ISB mode.

3.3.2.10 TxTuneInMicroSecs : unsigned

This element denotes the actual time overhead for transmit function hardware to tune to a specified frequency.

3.3.2.11 RTuneInMicroSecs : unsigned

This element denotes the actual time required for the receive function hardware to tune to a specified frequency.

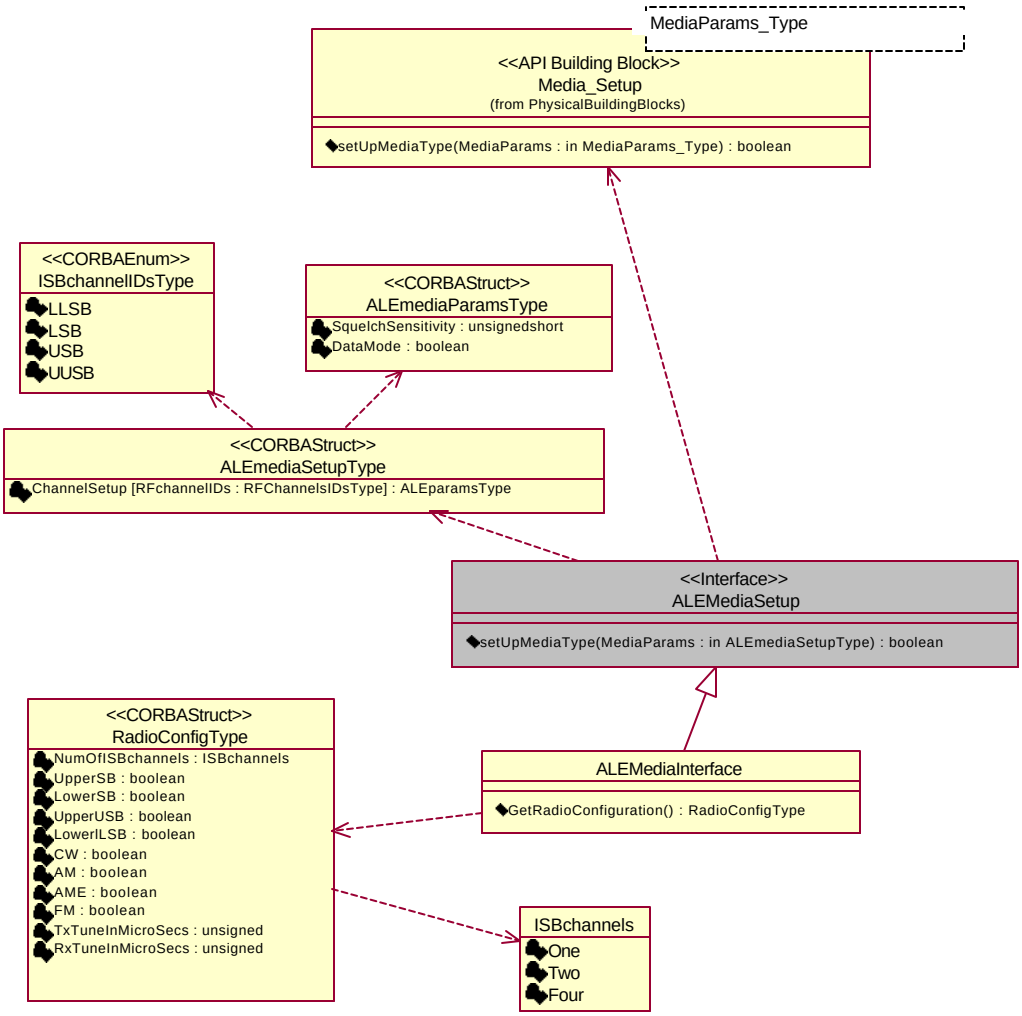

Figure 3-3. HF-ALE Media Setup

3.4 RXPACKET

This section defines the interface for passing data and related control from the ALE Physical layer to the ALE MAC layer, and the interface for the MAC layer to send Rx flow control signals back to the Physical layer (see Figure 3-4).

RxPacket is an instantiation of the Packet building block. The RxPacketSignals interface is realized by inheriting the PacketSignals building block directly.

The types specific to HF ALE are defined in the following paragraphs.

3.4.1 RxPktControlType

This type is a structure containing the RFChannel (non-ISB,LLSB,LSB,USB,UUSB) identifier and the time of the first sample.

3.4.2 ALEPayloadType

This type is defined as a sequence of unsigned short values.

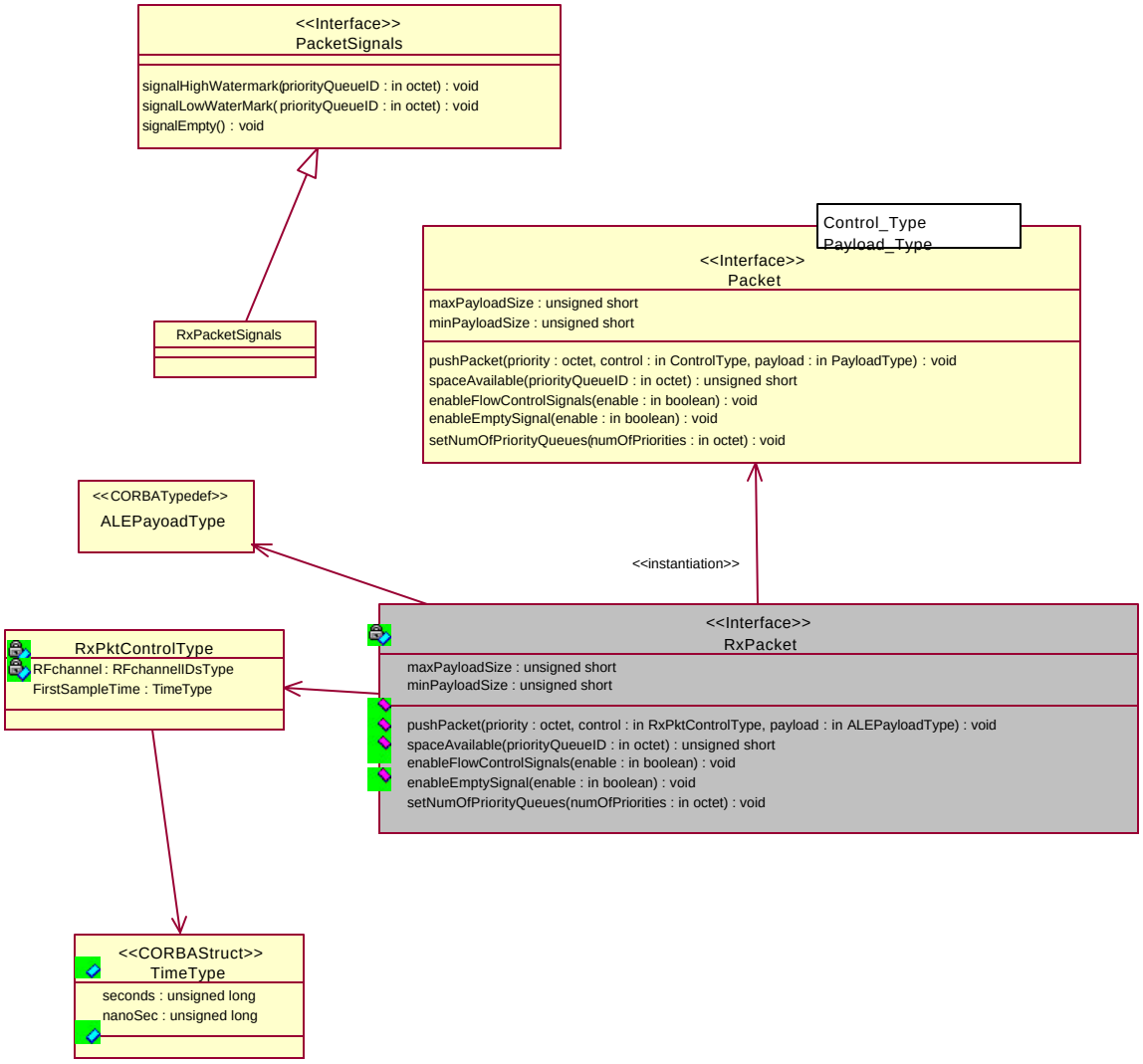

Figure 3-4. Rx PacketInterface

3.5 TXPACKET

This section defines the interface for passing data and related control from the ALE MAC layer to the ALE Physical layer, and the interface for the Physical layer to send Tx flow control signals back to the MAC layer.

TxPacket is an instantiation of the Packet building block. The TxPacketSignals interface is realized by inheriting the PacketSignals building block directly.

The types specific to HF ALE are defined in the following paragraphs.

3.5.1 TxPktControlType

This type is a structure containing the RFChannel (non-ISB,LLSB,LSB,USB,UUSB) identifier and the time of the first sample.

3.5.2 ALEPayloadType

This type is defined as a sequence of unsigned short values.

3.6 HF-ALE PHYSICAL LAYER API

Figure 3-5 shows the HF-ALE Physical Layer API.

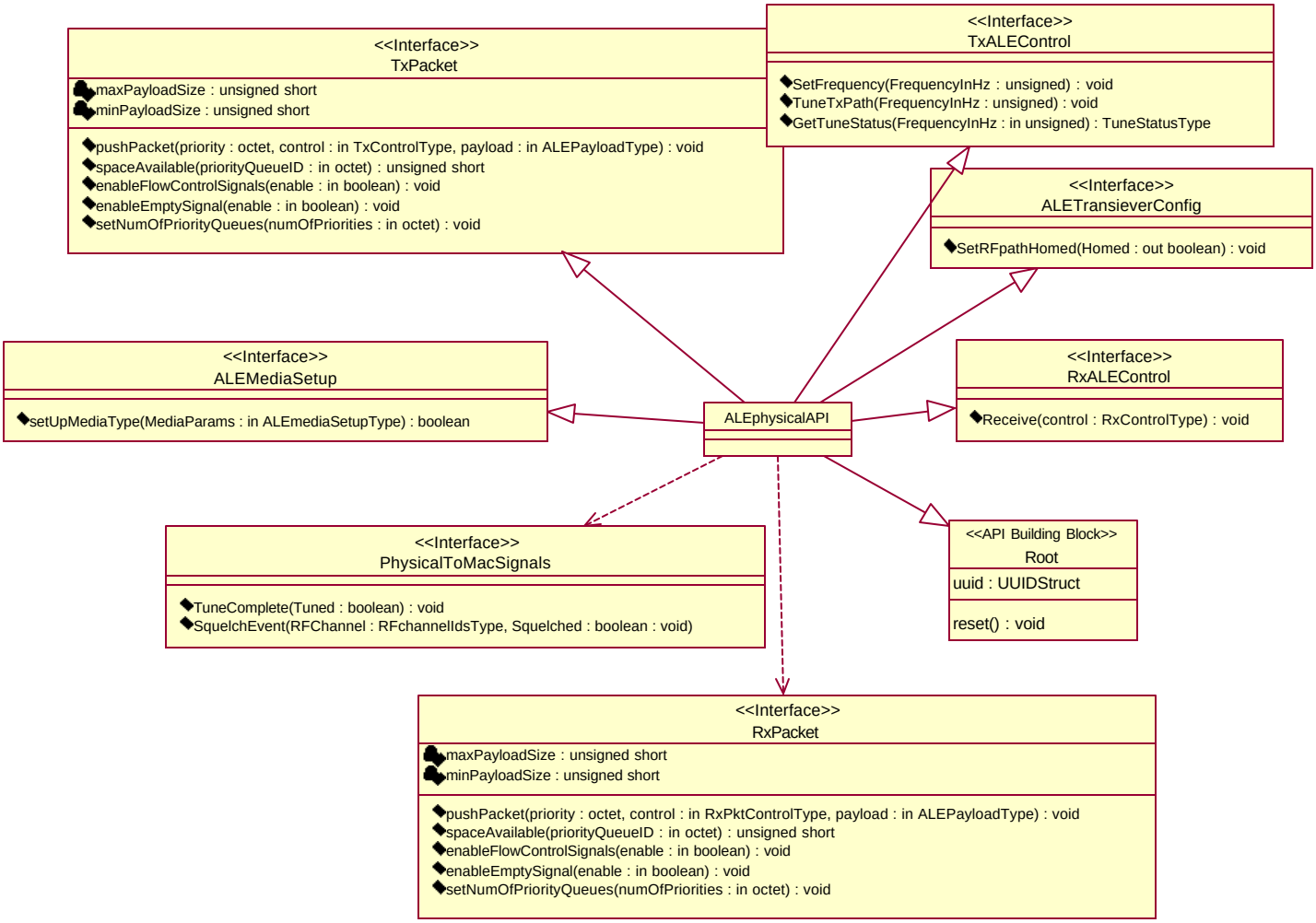

Figure 3-5. HF-ALE Physical API

4 SERVICE PRIMITIVES.

4.1 TRANSCEIVER CONFIGURATION.

4.1.1 SetUpReceiverParams Service.

This service is used to configure the receiver to operate as required by the MAC layer.

4.1.1.1 Synopsis.

SetUpReceiverParams (RecvParams : in RxConfigType)

4.1.1.2 Parameters.

RxConfigType defines the following parameters.

RxISBchannelModes [ISBchannelIdsType] : ISBmodesType

This type is defines the configuration for each ISB channel as Disabled, ISB or AME.

EmissionMode : EmissionModesType

This type defines the following emission modes; (ISB, AM, FM, CW)

4.1.1.3 State.

Any valid state as defined by the "parameters" parameter.

4.1.1.4 NewState.

Any valid state as defined by the "parameters" parameter.

4.1.1.5 Response.

A boolean value is returned indicating the status of the operation.

4.1.1.6 Originator.

HF-ALE MAC API.

4.1.1.7 Errors/Exceptions.

None defined.

4.1.2 SetUpTramsitterParams Service.

This service is used to configure the receiver to operate as required by the MAC layer.

4.1.2.1 Synopsis.

SetUpTransmitterParams (TransParams : in TxConfigType)

4.1.2.2 Parameters.

TxConfigType defines the following parameters.

PowerLevelType [Low,Med,High]

PowerLevel : PowerLevelType

TxISBchannelModes [ISBchannelIdsType] : ISBmodesType

This type is defines the configuration for each ISB channel as Disabled, ISB or AME.

EmissionMode : EmissionModesType

This type defines the following emission modes; (ISB, AM, FM, CW)

4.1.2.3 State.

Any valid state as defined by the "parameters" parameter.

4.1.2.4 NewState.

Any valid state as defined by the "parameters" parameter.

4.1.2.5 Response.

A boolean value is returned indicating the status of the operation.

4.1.2.6 Originator.

HF-ALE MAC API.

4.1.2.7 Errors/Exceptions.

None defined.

4.1.3 SetRFpathHomed Service.

This service resets the Physical layer such that the RF assests are not pretuned to any specific frequency. This allows proper reception of any signal in the HF band.

4.1.3.1 Synopsis.

SetRFpathHomed (Homed: boolean): void

4.1.3.2 Parameters.

Homed : This parameter denotes whether or not the home command was fully implemented.

4.1.3.3 State.

Any valid operational state.

4.1.3.4 NewState.

The SetRFpathHomed command forces the Physical layer to a state where a signal can be received anywhere within the HF band.

4.1.3.5 Response.

A boolean value is returned indicating the status of the operation.

4.1.3.6 Originator.

HF-ALE MAC API.

4.1.3.7 Errors/Exceptions.

None defined.

4.2 ALE MEDIA INTERFACE (INHERITS FROM ALE MEDIA SETUP).

4.2.1 setupMediaType Service.

This service defines the operational characteristics of each operating channel. Currently the channel setup is limited to Data vs. non-data operation and the squelch le vel.

4.2.1.1 Synopsis.

SetupMediaType (MediaParams : ALEMediaSetupType) : boolean

4.2.1.2 Parameters.

MediaParams : in ALEmediaSetupType

ALEmediaSetupType defines an array that contains the SquelchSensitivity and DataMode for each of the four ISB channels plus the non-ISB channel.

SquelchSensitivity adjusts the level at which squelch is broken. The level is relative with 0 being open squelch and 2^{16} representing the lowest sensitivity for the particular implementation.

DataMode is a boolean that tells the physical layer to process the incoming samples as either data samples or audio samples.

The radio channels are defined as Non-ISB, LLSB, LSB, USB and UUSB.

4.2.1.3 State.

Any valid operational state.

4.2.1.4 NewState.

The reconfigured state specified by the ALEmediaSetuptype.

4.2.1.5 Response.

A boolean value is returned indicating the status of the operation.

4.2.1.6 Originator.

HF-ALE MAC API.

4.2.1.7 Errors/Exceptions.

None defined.

4.2.2 getRadioConfiguration Service.

This operation is used by the MAC layer to discover the capabilities of the physical layer. This MAC layer uses this configuration to ensure proper usage of the physical layer.

4.2.2.1 Synopsis.

getRadioConfiguration () : RadioConfigType

4.2.2.2 Parameters.

RadioConfigType is a returned parameter. It is a record type defined as follows

{ NumOfISBchannels : [One,Two,Four],

UpperSB : booelan,

LowerSB : boolean,

UpperUSB : boolean,

LowerLSB : boolean

CW : boolean

AM : boolean

FM : boolean

TxTuneInMicrosecs : unsigned – Tune time of the transmitter.

RxTuneInMicrosecs : unsigned – Tune time of the receiver.}

4.2.2.3 State.

Any valid operational state.

4.2.2.4 NewState.

No change.

4.2.2.5 Response.

The method returns the Radio Configuration as defined in paragraph 4.2.2.2.

4.2.2.6 Originator.

HF-ALE MAC API.

4.2.2.7 Errors/Exceptions.

None defined.

4.3 RX ALE CONTROL.

4.3.1 Receive Service.

This service is used to configure the radio during normal runtime operations. The only configurable item in this interface is the frequency.

4.3.1.1 Synopsis.

Receive (Control: RxCommandType): void

4.3.1.2 Parameters.

"Control" provides the mechanism to set the frequency for receive functions.

Frequency is defined as an unsigned value specified in whole Hz.

4.3.1.3 State.

Any valid operational state.

4.3.1.4 NewState.

The receiver is tuned to the new receive frequency.

4.3.1.5 Response.

None.

4.3.1.6 Originator.

HF-ALE MAC API.

4.3.1.7 Errors/Exceptions.

None defined.

4.4 TX ALE CONTROL.

4.4.1 Set Frequency Service.

This service is used to tune the transmitter to the desired operational frequency.

4.4.1.1 Synopsis.

SetFrequency (FrequencyInHz : unsigned) : void

4.4.1.2 Parameters.

FrequencyInHz is defined as an unsigned value specified in whole Hz.

4.4.1.3 State.

Any valid operational state.

4.4.1.4 NewState.

The transmitter is tuned to the new frequency.

4.4.1.5 Response.

None.

4.4.1.6 Originator.

HF-ALE MAC API.

4.4.1.7 Errors/Exceptions.

None defined.

4.4.2 TuneTxPath Service.

This command pre-tunes the RF assets for proper operation on the given frequency. This operation does not tune the transmitter.

4.4.2.1 Synopsis.

TuneTxPath(FrequencyInHz : unsigned) : void

4.4.2.2 Parameters.

FrequencyInHz is defined as an unsigned value specified in whole Hz.

4.4.2.3 State.

Any valid operational state.

4.4.2.4 NewState.

The Tx path is tuned for operation on the specified frequency.

4.4.2.5 Response.

None.

4.4.2.6 Originator.

HF-ALE MAC API.

4.4.2.7 Errors/Exceptions.

None defined.

4.4.3 GetTuneStatus Service.

This command fetches the status of the previous tune command.

4.4.3.1 Synopsis.

GetTuneStatus(Status : TuneStatusType) : void

4.4.3.2 Parameters.

Status is defined as Untuned, Tuning or Tuned.

4.4.3.3 State.

Any valid operational state.

4.4.3.4 NewState.

No change.

4.4.3.5 Response.

None.

4.4.3.6 Originator.

HF-ALE MAC API.

4.4.3.7 Errors/Exceptions.

None defined.

4.5 PHYSICAL TO MAC SIGNALS.

4.5.1 TuneComplete Service.

This service is a signal used by the Physical layer top notify the MAC layer upon completion of a TuneTxPAth command.

4.5.1.1 Synopsis.

TuneComplete(Tuned : boolean) : void

4.5.1.2 Parameters.

Tuned denotes whether or not the previous TuneTxPath operation was successful.

4.5.1.3 State.

Any valid operational state.

4.5.1.4 NewState.

If the operation was successful then the state is Tuned for that frequency. If the operation was unsuccessful, then the state is Untuned for that frequency

4.5.1.5 Response.

None.

4.5.1.6 Originator.

HF-ALE Physical API.

4.5.1.7 Errors/Exceptions.

None defined.

4.5.2 SquelchEvent Service.

This service is a signal used by the Physical layer to notify the MAC layer of changes in the squelch state of the Rx channel.

4.5.2.1 Synopsis.

SquelchEvent(RfChannel : RfChannelIdsType, Squelched : boolean) : void

4.5.2.2 Parameters.

RfChannelIdsType denotes the channel for which the squelch event applies.

Squelched denotes the current state of the channel.

4.5.2.3 State.

Any valid operational state.

4.5.2.4 NewState.

No change.

4.5.2.5 Response.

None.

4.5.2.6 Originator.

HF-ALE Physical API.

4.5.2.7 Errors/Exceptions.

None defined.

4.6 TX/RX PACKETS.

Due to the commonality between TX and RX packets, they both are documented in the following paragraphs.

4.6.1 Get minPayloadSize / Get MaxPayloadSize Service.

These items are attributes of the Packet interface. The service definition is defined by the auto-generated read implementation.

4.6.1.1 Synopsis.

The functions used to read the Min and Max payload attributes are generated as a result of generating code from IDL.

4.6.1.2 Parameters.

MinPayloadSize denotes the minimum # of elements that are transferred in a Packet.

MaxPayloadSize denotes the maximum # of elements that are transferred in a Packet.

4.6.1.3 State.

Any valid operational state.

4.6.1.4 NewState.

No change.

4.6.1.5 Response.

None.

4.6.1.6 Originator.

HF-ALE MAC API.

4.6.1.7 Errors/Exceptions.

None defined.

4.6.2 pushPacket Service.

This service provides the ability to push data from a client process to a server process.

4.6.2.1 Synopsis.

PushPacket (priority : octet, control : in TXControlType, payload : in ALEPayloadType) : void

PushPacket (priority : octet, control : in RXControlType) : void

4.6.2.2 Parameters.

Priority denotes the priority with which the packet should be processed.

Tx/RxControlType is composed of a ChannelID, denoting the RF Channel for which the data is targeted, and the FirstSampleTime. FirstSampleTime is the time that either the first Rcv sample was received, or the time in which the first Tx sample should be transmitted.

4.6.2.3 State.

Any valid operational state.

4.6.2.4 NewState.

No change.

4.6.2.5 Response.

None.

4.6.2.6 Originator.

HF-ALE Physical API for Rx functions, HF-ALE MAC API for Tx functions.

4.6.2.7 Errors/Exceptions.

None defined.

4.6.3 SpaceAvailable Service.

This service is used to determine the amount of available queue space on a packet server.

4.6.3.1 Synopsis.

SpaceAvailable (priorityQueueId : in octet) : unsigned short

4.6.3.2 Parameters.

Priority denotes thequeue for which the operation applies.

4.6.3.3 State.

Any valid operational state.

4.6.3.4 NewState.

No change.

4.6.3.5 Response.

The number of elements in the servant queue is returned to the client.

4.6.3.6 Originator.

HF-ALE Physical API for Rx functions, HF-ALE MAC API for Tx functions.

4.6.3.7 Errors/Exceptions.

None defined.

4.6.4 EnableFlowControl Service.

This service is used to enable and disable flow control signals.

4.6.4.1 Synopsis.

EnableFlowControlSignals (enable : boolean) : void

4.6.4.2 Parameters.

Enable denotes whether or not the Packet server should generate signal calls for the Flow Control interface.

4.6.4.3 State.

Any valid operational state.

4.6.4.4 NewState.

Enabled = true forces the Packet server to a state that generates flow control signals.

Enabled = false forces the Packet server to a state that does not generate flow control signals.

4.6.4.5 Response.

None.

4.6.4.6 Originator.

HF-ALE Physical API for Rx functions, HF-ALE MAC API for Tx functions.

4.6.4.7 Errors/Exceptions.

None defined.

4.6.5 EnableEmptySignal Service.

This service is used to enable and disable generation of EmptySignal when a queue goes empty.

4.6.5.1 Synopsis.

EnableEmptySignal (enable : boolean) : void

4.6.5.2 Parameters.

Enable denotes whether or not the Packet server should generate a signal call for the Empty Signal interface.

4.6.5.3 State.

Any valid operational state.

4.6.5.4 NewState.

Enabled = true forces the Packet server to a state that generates the Empty signal.

Enabled = false forces the Packet server to a state that does not generate the Empty signal.

4.6.5.5 Response.

None.

4.6.5.6 Originator.

HF-ALE Physical API for Rx functions, HF-ALE MAC API for Tx functions.

4.6.5.7 Errors/Exceptions.

None defined.

4.6.6 setNumOfPriorityQueues Service.

This service sets the number of priorities that a pushPacket server will service.

4.6.6.1 Synopsis.

setNumOfPriorityQueues (NumOfPriorities : in octet) : void

4.6.6.2 Parameters.

NumOfPriorities denotes the number of priority segments that the Packet server will support.

4.6.6.3 State.

Any valid operational state.

4.6.6.4 NewState.

Same.

4.6.6.5 Response.

None.

4.6.6.6 Originator.

HF-ALE Physical API for Rx functions, HF-ALE MAC API for Tx functions.

4.6.6.7 Errors/Exceptions.

None defined.

4.7 TX/RX PACKET SIGNALS.

Due to the commonality between TX and RX packets, they both are documented in the following paragraphs.

4.7.1 signal HighWatermark.

"*signalHighWaterMark*" provides the ability to signal the service user when a queue has reached the high water mark: the queue is full for the specified priority.

4.7.1.1 Synopsis.

signalHighWatermark(priorityQueueID : in octet) : void

4.7.1.2 Parameters.

priorityQueueID : octet indicates the queue priority which has reached the high water mark. (See setNumOfPriorityQueues).

4.7.1.3 State.

Any state.

4.7.1.4 New State.

Same state.

4.7.1.5 Originator.

Service Provider.

4.7.1.6 Errors/Exceptions.

None.

4.7.2 signal Low Watermark.

"*signalLowWaterMark*" provides the ability to signal the service user when a queue has reached the low water mark: the queue is near empty for the specified priority.

4.7.2.1 Synopsis.

signalLowWatermark(priorityQueueID : in octet) : void

4.7.2.2 Parameters.

priorityQueueID : octet indicates the queue priority which has reached the low water mark. (See setNumOfPriorityQueues).

4.7.2.3 State.

Any state.

4.7.2.4 New State.

Same state.

4.7.2.5 Originator.

Service Provider.

4.7.2.6 Errors/Exceptions.

None.

4.7.3 signal Empty.

"*signalEmpty*" provides the ability to signal the service user when all priority queues are empty. (See setNumOfPriorityQueues).

4.7.3.1 Synopsis.

signalEmpty(void) : void

4.7.3.2 Parameters.

None.

4.7.3.3 State.

Any state.

4.7.3.4 New State.

Same state.

4.7.3.5 Originator.

Service Provider.

4.7.3.6 Errors/Exceptions.

None.

5 ALLOWABLE SEQUENCE OF SERVICE PRIMITIVES.

This section intentionally left blank.

6 PRECEDENCE OF SERVICE PRIMITIVES.

None.

7 SERVICE USER GUIDELINES.

This section is intentionally blank.

8 SERVICE PROVIDER-SPECIFIC INFORMATION.

This section is intentionally blank.

9 IDL.

The HF ALE Physical interface design depicted in IDL source code is shown in the following subsections.

9.1 IDL FOR ALE MEDIA SETUP.

```
//Source file: C:/Projects/JTRS/APIs/SCAWorkingGroup/Building_Blocks/HF-
ALE/Physical_API/IDL/ALEMediaSetup.idl

#ifndef __ALEMEDIASETUP_DEFINED
#define __ALEMEDIASETUP_DEFINED

/* CmIdentification
  %X% %Q% %Z% %W% */

struct ALEmediaParamsType {
      unsignedshort SquelchSensitivity;
      boolean DataMode;
};

enum ISBchannels {
      One,
      Two,
      Four
};

enum RFChannelIDType {
      NonISB,
      LSB,
      USB,
      UUSB,
      LLSB
};

struct RadioConfigType {
      ISBchannels NumOfISBchannels;
      boolean UpperSB;
```

```
      boolean LowerSB;
      boolean AME;
      boolean FM;
      unsigned TxTuneInMicroSecs;
      unsigned RxTuneInMicroSecs;
      boolean CW;
      boolean LowerlLSB;
      boolean UpperUSB;
      boolean AM;
};

struct ALEmediaSetupType {
      RFChannelsIDType] : ALEmediaParamsType ChannelSetup;
};

interface ALEMediaSetup {
      /*
      @roseuid 39EF5F850329 */
      boolean setUpMediaType (
            in ALEmediaSetupType MediaParams
            );

};

interface ALEMediaInterface : ALEMediaSetup {
      /*
      @roseuid 39F5FF2903E5 */
      RadioConfigType getRadioConfiguration ();

};

#endif
```

9.2 IDL FOR ALE PHYSICAL API.

```
//Source file: C:/Projects/JTRS/APIs/SCAWorkingGroup/Building_Blocks/HF-
ALE/Physical_API/IDL/ALEPhysicalAPI.idl

#ifndef __ALEPHYSICALAPI_DEFINED
#define __ALEPHYSICALAPI_DEFINED

/* CmIdentification
  %X% %Q% %Z% %W% */

#include "ALEMediaSetup.idl"
#include "PhysicalRealTime.idl"
#include "PhysicalTxPacket.idl"
#include "ALETransceiverInterface.idl"

interface RxPacket;

interface Root {
      readonly attribute UUIDStruct uuid;
```

```
      /* The reset operation is meant to return the resource to a default
condition.  What this default condition is, is specified by the particular
implementation.
      @roseuid 38EE2E6601F6 */
      oneway void reset ();

};

interface ALEphysicalAPI : ALEMediaSetup, TxALEControl, RxALEControl,
TxPacket, Root, ALETransieverConfig {
};

#endif
```

9.3 IDL FOR ALE TRANSCEIVER INTERFACE.

```
//Source file: C:/Projects/JTRS/APIs/SCAWorkingGroup/Building_Blocks/HF-
ALE/Physical_API/IDL/ALETransceiverInterface.idl

#ifndef __ALETRANSCEIVERINTERFACE_DEFINED
#define __ALETRANSCEIVERINTERFACE_DEFINED

/* CmIdentification
  %X% %Q% %Z% %W% */

#include "ALEMediaSetup.idl"

enum ISBmodesType {
      Disabled,
      AME,
      ISB
};

enum EmissionModeType {
      ISB,
      FM,
      CW,
      AM
};

struct RxConfigType {
      ISBmodesType rxISBchannelModes;
      EmissionModeType emissionMode;
};

enum PowerLevelType {
      Low,
      Med,
      High
};

struct TxConfigType {
      PowerLevelType powerLevel;
      EmissionModeType emissionMode;
      ISBmodesType txISBchannelModes;
};
```

```
interface TranscieverConfig {
      /*
      @roseuid 39EE22E80228 */
      boolean setUpReceiverParams (
            in RxConfigType RecvParams
            );

      /*
      @roseuid 39EE230103BE */
      boolean setUpTransmitterParams (
            in TxConfigType TransParams
            );

};

interface ALETransieverConfig : TranscieverConfig {
      /*
      @roseuid 39F0AFFE00A5 */
      void setRFpathHomed (
            out boolean Homed
            );

};

#endif
```

9.4 IDL FOR PACKET SIGNALS INTERFACE.

```
//Source file: C:/Projects/JTRS/APIs/SCAWorkingGroup/Building_Blocks/HF-
ALE/Physical_API/IDL/PacketSignalsInterface.idl

#ifndef __PACKETSIGNALSINTERFACE_DEFINED
#define __PACKETSIGNALSINTERFACE_DEFINED

/* CmIdentification
  %X% %Q% %Z% %W% */

interface PacketSignals {
      /* This operation is a call event back to the PacketAPI client
indicating that a queue has reach the high watermark.  If priority or
multiple queues are being supported then the priorityQueueID indicates which
queue has reached the high watermark.
      @roseuid 38F3442F01B8 */
      oneway void signalHighWatermark (
            in octet priorityQueueID
            );

      /* This operation is a call event back to the PacketAPI client
indicating that the queue has reach the low watermark.   If priority or
multiple queues are being supported then this indicates that the sum total of
all the queues has reached the low watermark.
      @roseuid 38F3446F025A */
      oneway void signalLowWaterMark (
            in octet priorityQueueID
            );
```

```
      /* This operation is a call event back to the PacketAPI client
indicating that the queue has emptied.   If priority or multiple queues are
being supported then this indicates that the sum total of all the queues has
reached zero.
      @roseuid 38FE26CF02FA */
      oneway void signalEmpty ();

};

#endif
```

9.5 IDL FOR PHYSICAL REAL TIME.

```
//Source file: C:/Projects/JTRS/APIs/SCAWorkingGroup/Building_Blocks/HF-
ALE/Physical_API/IDL/PhysicalRealTime.idl

#ifndef __PHYSICALREALTIME_DEFINED
#define __PHYSICALREALTIME_DEFINED

/* CmIdentification
  %X% %Q% %Z% %W% */

interface PhysicalToMacSignals {
      /*
      @roseuid 39EF74560390 */
      void tuneComplete (
             boolean Tuned
            );

      /*
      @roseuid 39F0ADF40221 */
      any squelchEvent (
             RFchannelIdsType RFChannel,
             boolean : void Squelched
            );

};

struct RxControlType {
      unsigned frequencyInHz;
};

interface RxALEControl {
      /*
      @roseuid 39EF709003D7 */
      void receive (
             RxControlType control
            );

};

enum TuneStatusType {
      Untuned,Tuning,Tuned
};

interface TxALEControl {
```

```
        /*
        @roseuid 39EF732802D7 */
        void setFrequency (
                unsigned FrequencyInHz
               );

        /*
        @roseuid 39EF735E02FD */
        void tuneTxPath (
                unsigned FrequencyInHz
               );

        /*
        @roseuid 39EF73D3034C */
        TuneStatusType getTuneStatus (
               in unsigned FrequencyInHz
               );

};

#endif
```

9.6 IDL FOR PHYSICAL RECEIVE PACKET.

```
//Source file: C:/Projects/JTRS/APIs/SCAWorkingGroup/Building_Blocks/HF-
ALE/Physical_API/IDL/PhysicalRxPacket.idl

#ifndef __PHYSICALRXPACKET_DEFINED
#define __PHYSICALRXPACKET_DEFINED

/* CmIdentification
  %X% %Q% %Z% %W% */

#include "PacketSignalsInterface.idl"
#include "PhysicalTxPacket.idl"
#include "PhysicalRealTime.idl"

interface RxPacketSignals : PacketSignals {
};

typedef sequence <unsigned> ALEPayoadType;

struct RxPktControlType {
        rfchannelIDsType rfchannel;
        TimeType firstSampleTime;
};

struct TimeType {
        unsigned long seconds;
        unsigned long nanoSec;
};

interface RxPacket {
        attribute unsigned short minPayloadSize;
```

```
      /* The maxPacketSize is a read only attribute set by the Packet Server
and the get operation reports back the maximum number of traffic units
allowed in one pushPacket call.  */

      attribute unsigned short maxPayloadSize;

      /* This operation is used to push Client data to the Server with a
Control element and a Payload element.
      @roseuid 39F08FCB01A7 */
      void pushPacket (
             octet priority,
            in RxPktControlType control,
            in ALEPayloadType payload
            );

      /* The operation returns the space available in the Servers queue(s) in
terms of the implementers defined Traffic Units.
      @roseuid 39F08FCB01FA */
      unsigned short spaceAvailable (
            in octet priorityQueueID
            );

      /* This operation allows the client to turn the High Watermark Signal
ON and OFF.
      @roseuid 39F08FCB01FC */
      void enableFlowControlSignals (
            in boolean enable
            );

      /* This operation allows the client to turn theEmpty Signal ON and OFF.
      @roseuid 39F08FCB0202 */
      void enableEmptySignal (
            in boolean enable
            );

      /*
      @roseuid 39F08FCB0204 */
      void setNumOfPriorityQueues (
            in octet numOfPriorities
            );

};

#endif
```

9.7 IDL FOR PHYSICAL TRANSMIT PACKET.

```
//Source file: C:/Projects/JTRS/APIs/SCAWorkingGroup/Building_Blocks/HF-
ALE/Physical_API/IDL/PhysicalTxPacket.idl

#ifndef __PHYSICALTXPACKET_DEFINED
#define __PHYSICALTXPACKET_DEFINED

/* CmIdentification
  %X% %Q% %Z% %W% */
```

```
#include "PacketSignalsInterface.idl"

interface TxPacketSignals : PacketSignals {
};

typedef sequence <unsigned> ALEPayoadType;

struct TimeType {
      unsigned long seconds;
      unsigned long nanoSec;
};

struct TxControlType {
      rfchannelIdsType rfchannel;
      TimeType firstSampleTime;
};

interface TxPacket {
      /* The maxPacketSize is a read only attribute set by the Packet Server
and the get operation reports back the maximum number of traffic units
allowed in one pushPacket call.  */

      attribute unsigned short maxPayloadSize;
      attribute unsigned short minPayloadSize;

      /* This operation is used to push Client data to the Server with a
Control element and a Payload element.
      @roseuid 39F0640A03C7 */
      void pushPacket (
             octet priority,
            in TxControlType control,
            in ALEPayloadType payload
            );

      /* The operation returns the space available in the Servers queue(s) in
terms of the implementers defined Traffic Units.
      @roseuid 39F0640A03D4 */
      unsigned short spaceAvailable (
            in octet priorityQueueID
            );

      /* This operation allows the client to turn the High Watermark Signal
ON and OFF.
      @roseuid 39F0640A03D6 */
      void enableFlowControlSignals (
            in boolean enable
            );

      /* This operation allows the client to turn theEmpty Signal ON and OFF.
      @roseuid 39F0640A03D8 */
      void enableEmptySignal (
            in boolean enable
            );

      /*
      @roseuid 39F0640A03DC */
      void setNumOfPriorityQueues (
```

```
            in octet numOfPriorities
            );

};

#endif
```

10 UML.

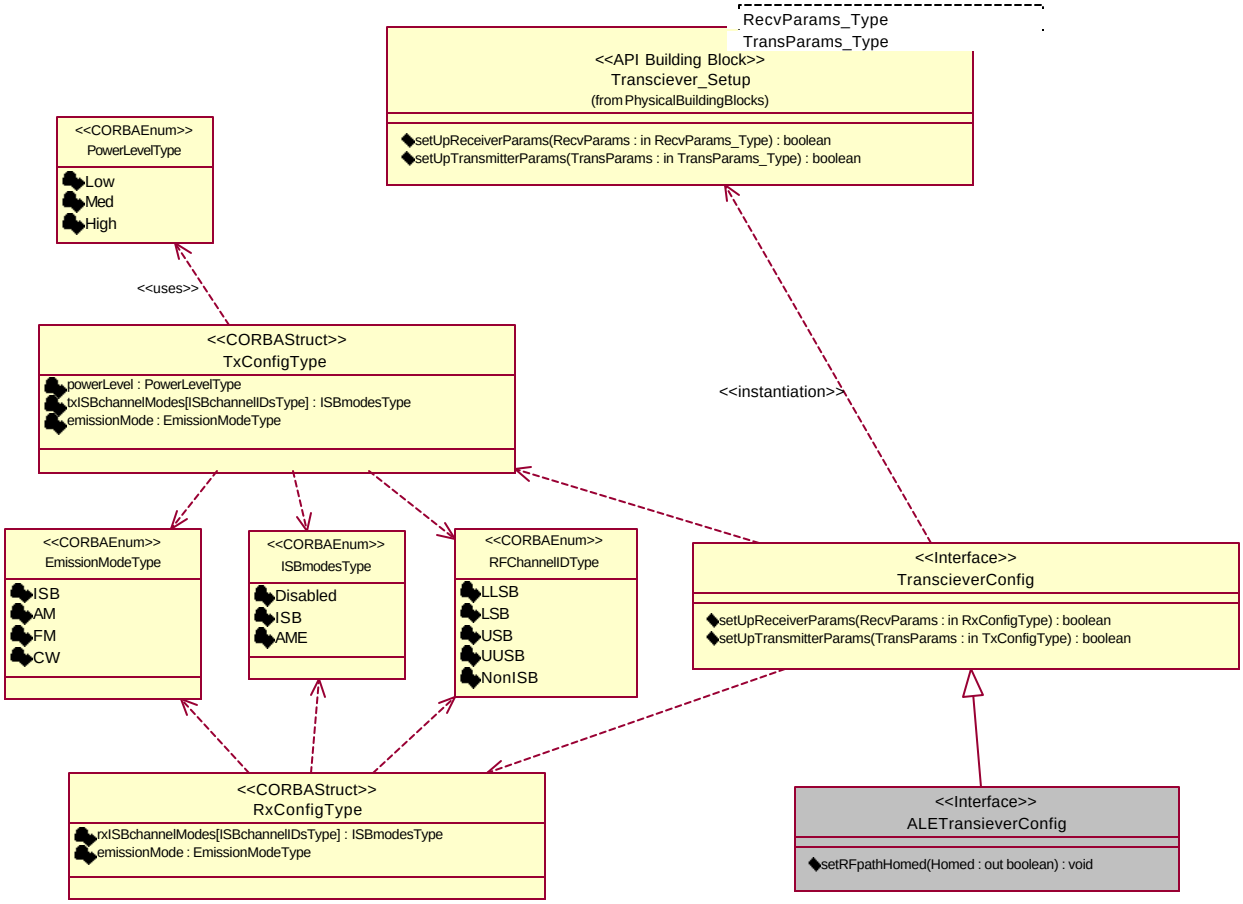

Figure 10-1. Physical Layer Tranceiver Config Interface for HF ALE

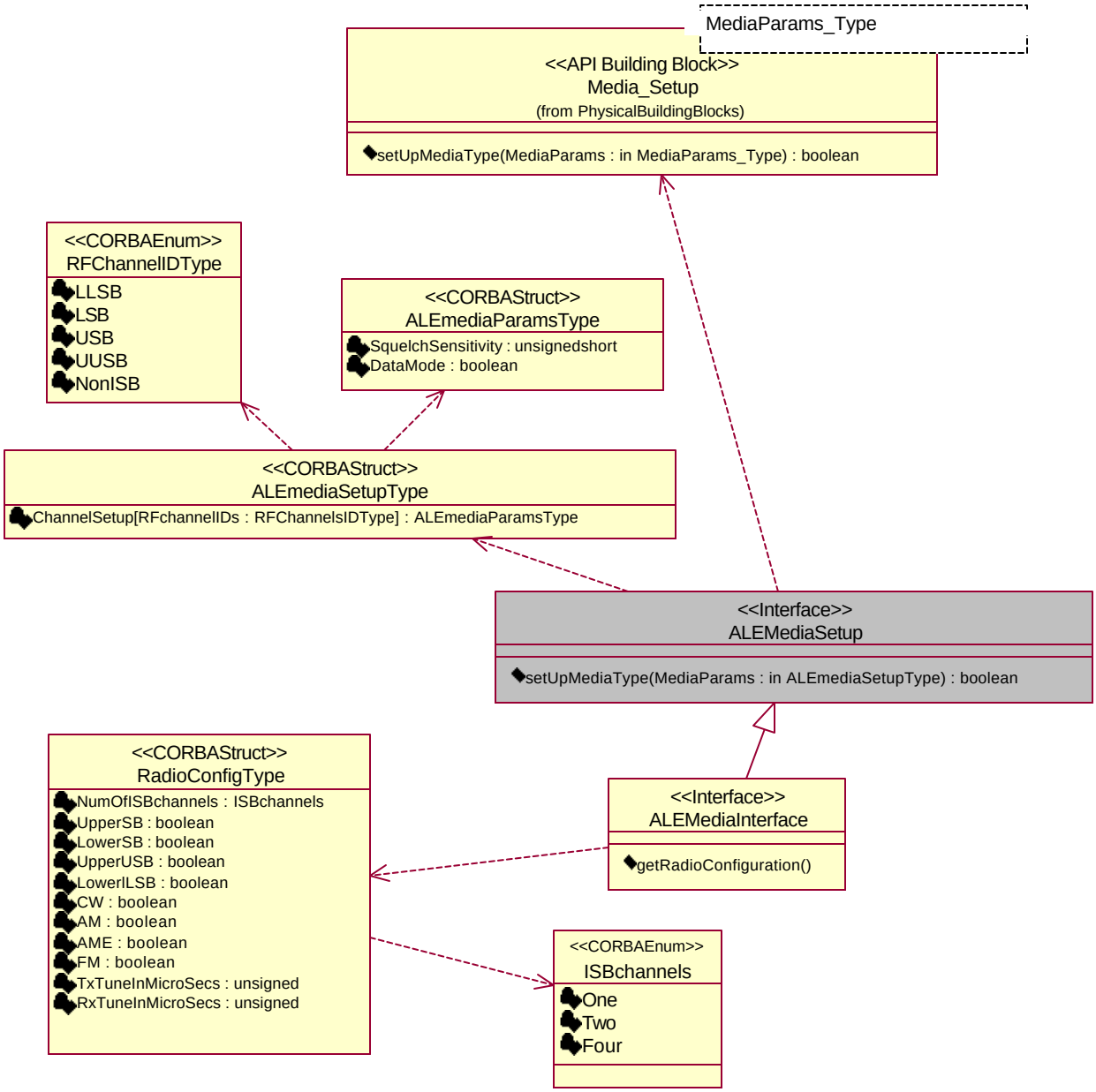

Figure 10-2. Physical Layer Media Setup Interface for HF ALE

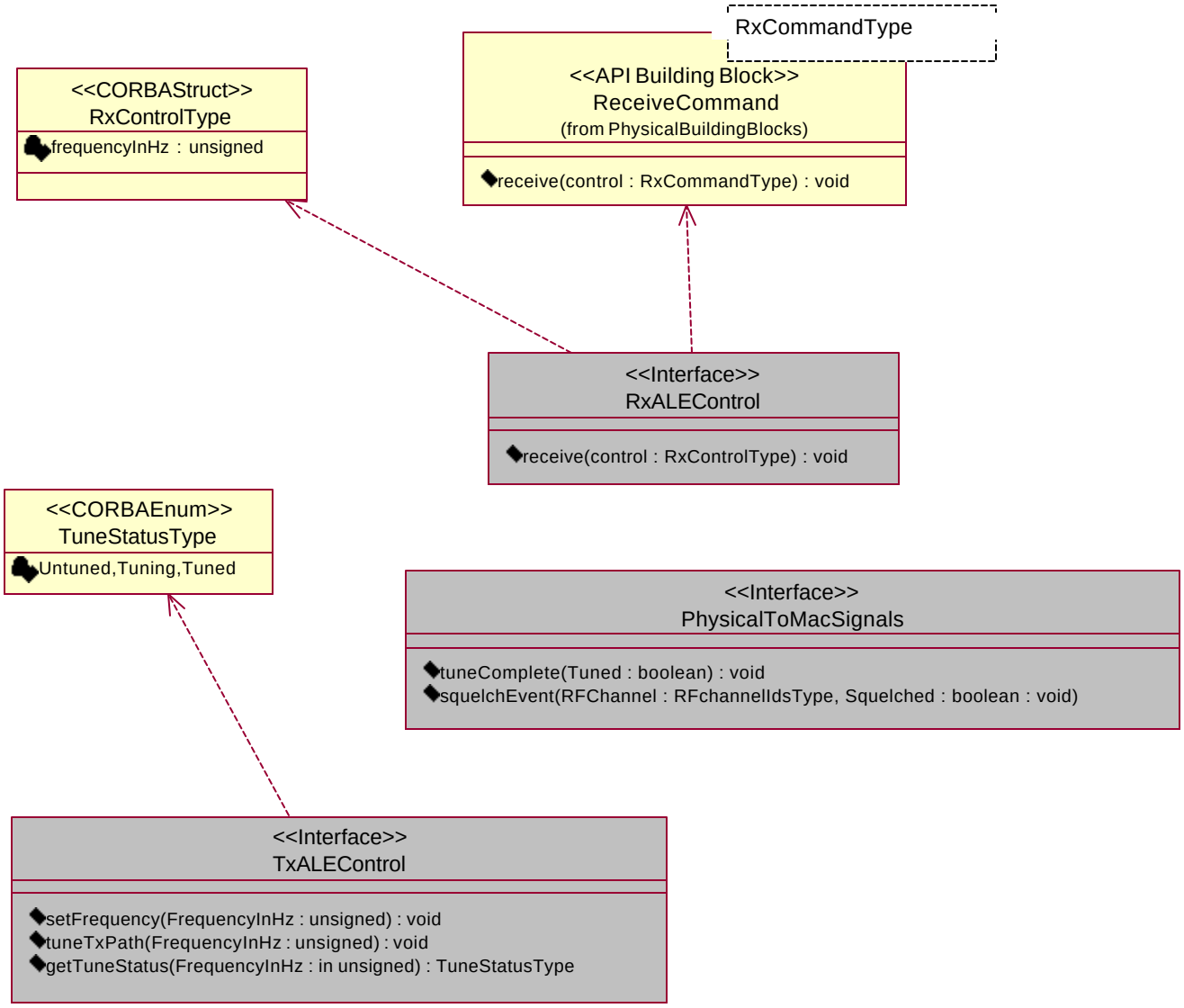

Figure 10-3. Physical Layer Real-Time Control Interfaces

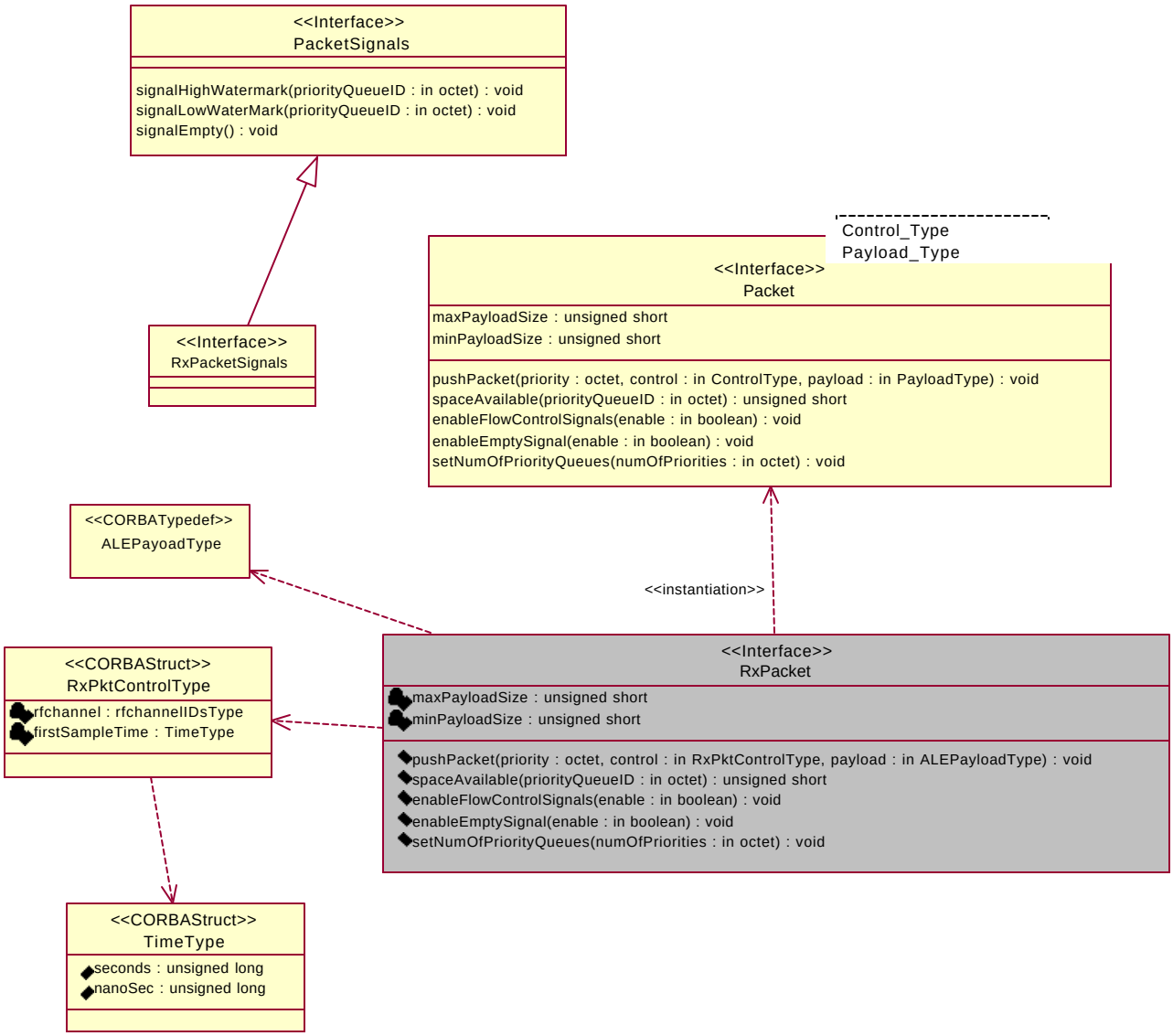

Figure 10-4. Physical Layer Upstream Packet Interface

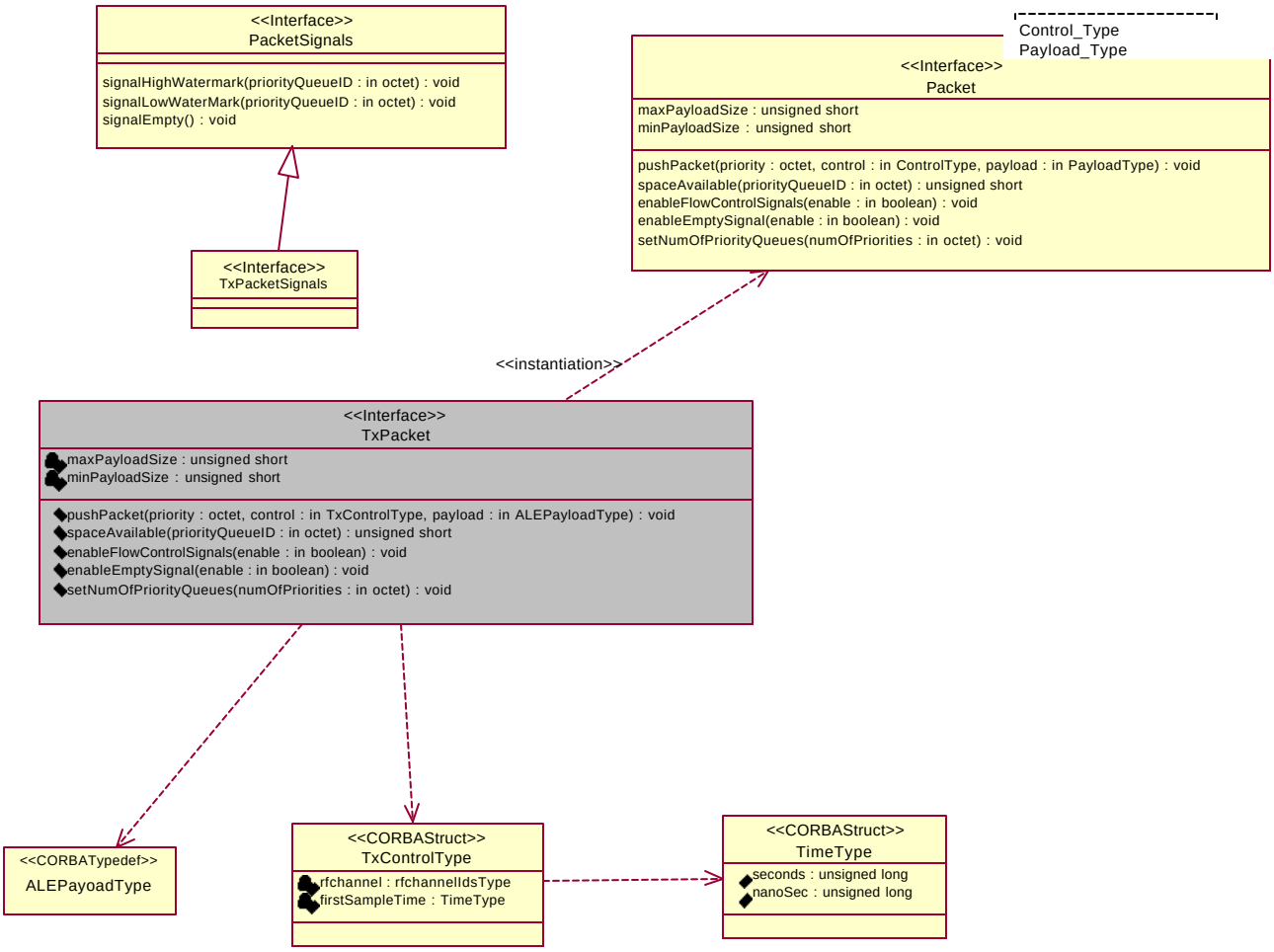

Figure 10-5. Physical Layer Downstream Packet Interface

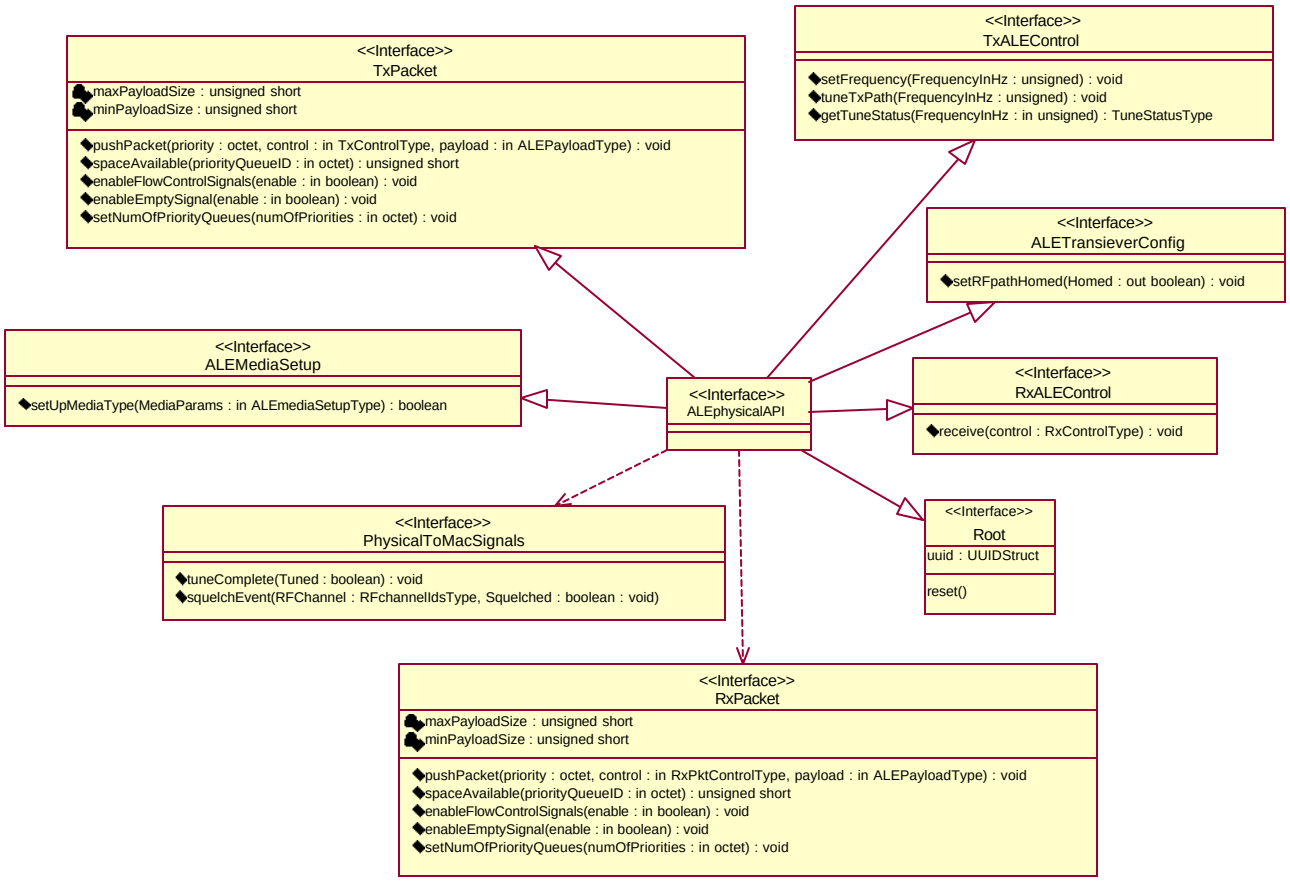

Figure 10-6. HF ALE Physical Layer API

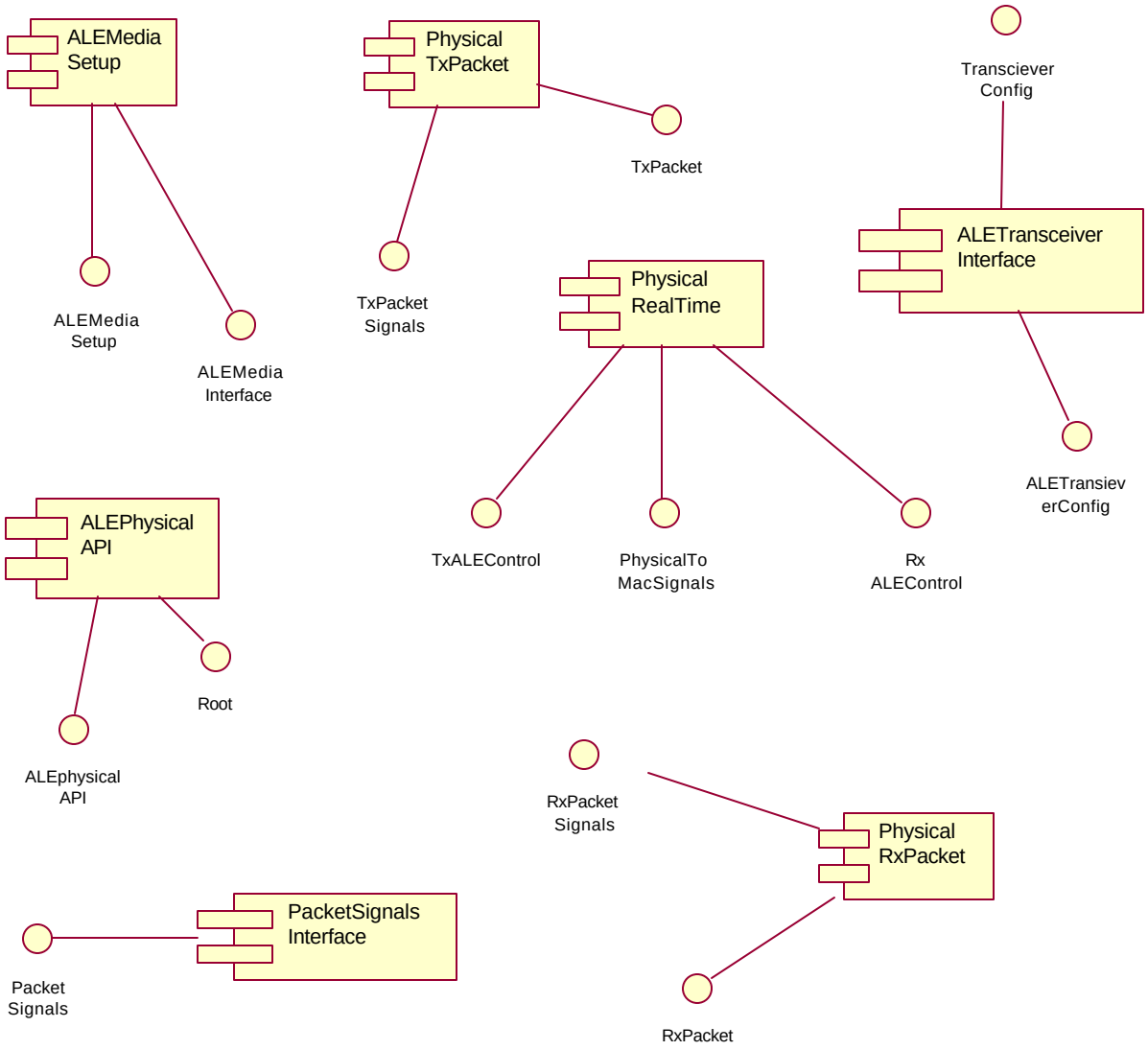

Figure 10-7. HF ALE Component Diagram