Date: March 2007

Communication Channel and Equipment Specification, v1.0

OMG Available Specification
formal/07-03-02

The PIM and PSM for Software Radio Components Specification (formal/07-03-01) is physically partitioned into 5 volumes:

Communication Channel and Equipment (formal/07-03-02)
Component Document Type Definitions (formal/07-03-03)
Component Framework (formal/07-03-04)
Common and Data Link Layer Facilities (formal/07-03-05)
POSIX Profiles (formal/07-03-06)

OBJECT MANAGEMENT GROUP

developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents, Report a Bug/Issue (*http://www.omg.org/technology/agreement.htm*).

Table of Contents

Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at *http://www.omg.org/*.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications.

OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM).

Platform Specific Model and Interface Specifications

- CORBAservices
- CORBAfacilities
- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications.

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

Note – Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/technology/agreement.htm*.

1 Scope

This specification responds to the requirements set by “Request for Proposals for a Platform Independent Model (PIM) and CORBA Platform Specific Model (PSM)” (swradio/02-06-02) of communication channel creation and interfaces for radio management and physical layer facilities that can be utilized in deployment of applications such as waveforms and the modeling of a radio set or radio system.

The specification is physically partitioned into three major chapters: UML Profile for Communication Channel, Comm Channel Facilities PIM, and PSM for CORBA IDL and XML. UML Profile for Communication Channel defines a language for modeling communication channels, communication equipment and radio management components by extending the UML language. The profile is specified independently from the underlying middleware technology and is applicable for other domains besides SDR.

Comm Channel Facilities provides a set of interfaces for interfacing with the communication equipment for data, control, and status, and for radio set control. This specification also provides a mechanism for transforming the elements of the profile and facilities PIM into the platform specific model for CORBA IDL and XML.

Finally, the specification provides different compliance points depending on the role the implementer of this specification plays. Those different roles and respective partitioning of this document is given in the Conformance (Chapter 2).

2 Conformance

There are two kinds of conformance with respect to the communication channel profile: conformance on the part of a model of a specific communication channel and channel manager, and conformance on the part of an MDA tool.

2.1 Conformance by a Model of a Specific Application

A UML model of a specific communication channel and channel manager either conforms to the communication channel profile or it does not. There are no categories of this kind of conformance. Such a UML model conforms to the communication channel profile if it satisfies all constraints imposed by the profile package.

2.2 Conformance by a Tool

2.2.1 Definition of Terms for Discussion of Tool Conformance

To support the discussion of conformance by an MDA tool, we define two terms: “identified subset of UML 2.0" and “all constructs defined by the profile.” The identified subset of UML 2.0 for the profile is the set of packages contained in the UML 2.0 Superstructure specification Part 1 (Structure). Part 1 includes the following packages and the transitive closure of all packages contained by these packages and of all packages upon which these packages depend:

- Classes
- Composite Structures
- Components
- Deployments

Hereafter we sometimes use the abbreviated term identified subset to refer to the identified subset of UML 2.0. The term all constructs defined by the profile is defined to mean all constructs that are part of the package's identified subset of UML 2.0, plus all extensions to that subset that the profile defines. Thus this term includes UML constructs that are part of the identified subset but that are not extended by the profile.

2.2.2 Categories of Tool Conformance

A tool is considered to be a conformant simple modeling tool for the communication channel profile if it does both of the following:

- Supports expression of all constructs defined by the profile, via UML 2.0 notation.
- Supports the UML 2.0 XMI exchange mechanism for the identified subset and for UML 2.0 profiles.

A tool is considered to be a conformant CORBA/XML-based forward engineering tool for the profile if it does the following:

- Supports the PIM-to-PSM Mapping defined in Chapter 9.
- Produces comm channel manager components PSMs that are conformant to the behavior defined in the PIM.

Alternately, if a tool only produces a component skeleton, the skeleton must not make it impossible for a full component based on the skeleton to qualify as a conformant component – in other words, the skeleton must be able to form the basis of a conformant component.

A forward engineering tool that targets a platform technology other than CORBA/XML can legitimately claim a degree of conformance to the communication channel profile and PIM derived from the Profile if it conforms to the PIM-to-PSM Mapping and produces components PSMs that are conformant components to the behavior in defined in the PIM, or produces component skeletons that can form the basis of conformant components. In practice this requires the definition of an alternate PIM-PSM mapping.

A forward engineering tool of this nature for the platform "X" is considered to be a conformant X-Based forward engineering tool for the profile.

2.3 Conformance on the part of a Component PSM

The interfaces and components as defined in sections 7 and 8 of this specification are not required to be used for a given platform or application. A platform or application uses the interfaces and component definitions that meet their needs. Conformance is at the level of usage as follows:

- A PSM implementation (no matter what language) of an interface defined in this specification needs to be conformant to the interface definition as described in the specification.
- A PSM implementation (no matter what language) of a component defined in this specification needs to be conformant to the component definition (ports, interfaces realized, properties, etc.) as described in the specification.

A component is considered to be a conformant for CORBA/XML platform if it does all of the following:

- Implements the CORBA interfaces that the component PSM defines
- Implements the XML serialization formats that the component PSM defines.
- Implements the semantics that the component PIM defines.

Note that the component PIM essentially defines the semantics for the CORBA interfaces and XML serialization formats. The semantics for a CORBA interface defined in the component PSM are defined by the semantics of the corresponding element(s) in the component PIM. It is possible to deduce the corresponding elements in the PIM for such a CORBA interface by reversing the PIM-PSM Mapping.

3 References

3.1 Normative References

3.1.1 UML and Profile Specifications

3.1.1.1 UML Language Specification

Unified Modeling Language (UML) Superstructure Specification, Version 2.1.1
Formal OMG Specification, document number: formal/07-02-03
The Object Management Group, February 2007
[http://www.omg.org]

Unified Modeling Language (UML) Infrastructure Specification, Version 2.1.1
Formal OMG Specification, document number: formal/07-02-04
The Object Management Group, February 2007
[http://www.omg.org]

3.1.1.2 OCL Language Specification

Object Constraint Language (OCL) Specification, Version 2.0
Formal OMG Specification, document number: formal/2006-05-01
The Object Management Group, May 2006
[http://www.omg.org]

3.1.1.3 UML Profile for CORBA Specification

UML Profile for CORBA Specification, Version 1.0
Formal OMG Specification, document number: formal/2002-04-01
The Object Management Group, April 2002
[http://www.omg.org]

3.1.1.4 UML Profile for Modeling QoS and FT Characteristics and Mechanisms Specification

UML Profile for Modeling QoS and FT Characteristics and Mechanisms, Version 1.0
Formal OMG Specification, document number: formal/06-05-02
The Object Management Group, May 2006
[http://www.omg.org]

3.1.1.5 MOF 2.0/XMI Mapping Specification

Meta Object Facility (MOF) 2.0 XMI Mapping Specification, Version 2.1
Formal OMG Specification, document number: formal/05-09-01
The Object Management Group, September 2005
[http://www.omg.org]

3.1.2 CORBA Core Specifications

3.1.2.1 CORBA Specification

Common Object Request Broker (CORBA/IIOP), Version 3.0.3
Formal OMG Specification, document number: formal/2004-03-01
The Object Management Group, March 2004
[http://www.omg.org]

3.1.2.2 Real-time CORBA Specification

Real-time - CORBA Specification, Version 1.2
Formal OMG Specification, document number: formal/2005-01-04
The Object Management Group, January 2005
[http://www.omg.org]

3.1.2.3 CORBA/e Specification

CORBA/e Specification
Draft Adopted OMG Specification, document number: ptc/06-05-01
The Object Management Group, May 2006
[http://www.omg.org]

3.1.3 UML Models

3.1.3.1 UML Profile for Communication Channel

UML Profile for Communication Channel XMI File
Formal OMG document number: dtc/2006-04-10
The Object Management Group, December 2006
[http://www.omg.org]

3.1.3.2 UML Profile for Component Framework

UML Profile for Component Framework XMI File
Formal OMG document number: dtc/2006-04-09
The Object Management Group, December 2006
[http://www.omg.org]

3.1.3.3 Common and Data Link Layer Facilities PIM

Common and Data Link Layer Facilities PIM XMI File
Formal OMG document number: dtc/2006-04-11
The Object Management Group, December 2006
[http://www.omg.org]

Note – See also formal/07-03-07, a .zip archive of associated schema files.

3.2 Non-normative References

3.2.1 Common and Data Link Layer Facilities Specification

Common and Data Link Layer Facilities Specification, Version 1.0
Formal OMG document number: formal/07-03-05
The Object Management Group, March 2007
[http://www.omg.org]

3.2.2 UML Profile for Component Framework Specification

Component Framework Specification, Version 1.0
Formal OMG document number: formal/07-03-04
The Object Management Group, March 2007
[http://www.omg.org]

3.2.3 Software Radio Facilities IDL

Software Radio Facilities IDL Files
Formal OMG document number:dtc/2006-04-14
The Object Management Group, December 2006
[http://www.omg.org]

3.2.4 Communication Channel XML Schema

Communication Channel XML Schema File
Formal OMG document number:dtc/2006-04-15
The Object Management Group, December 2006
[http://www.omg.org]

4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

Common Object Request Broker Architecture (CORBA)

An OMG distributed computing platform specification that is independent of implementation languages.

Component

A component can always be considered an autonomous unit within a system or subsystem. It has one or more ports, and its internals are hidden and inaccessible other than as provided by its interfaces. A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component exposes a set of ports that define the component specification in terms of provided and required interfaces. As such, a component serves as a type, whose conformance is defined by these provided and required interfaces (encompassing both their static as well as dynamic semantics).

Facility

The realization of certain functionality through a set of well defined interfaces.

Interface Definition Language (IDL)

An OMG and ISO standard language for specifying interfaces and associated data structures.

Mapping

The Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel.

Metadata

The Data that represents models. For example, a UML model; a CORBA object model expressed in IDL; and a relational database schema expressed using CWM.

Metamodel

A model of models.

Meta Object Facility (MOF)

An OMG standard, closely related to UML, that enables metadata management and language definition.

Model

A formal specification of the function, structure and/or behavior of an application or system.

Model Driven Architecture (MDA)

An approach to IT system specification that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform.

Platform

A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.

Platform Independent Model (PIM)

A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.

Platform Specific Model (PSM)

A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform.

Radio Platform

The Radio Platform is made of a Hardware Platform and a Software Platform.

Radio Set

A single radio set unit that can be ground fixed, mounted on a mobile platform or held by hand.

Radio System

A networked set of radio sets that provide wireless communication facilities between callers and callees.

Request for Proposal (RFP)

A document requesting OMG members to submit proposals to the OMG's Technology Committee. Such proposals must be received by a certain deadline and are evaluated by the issuing task force.

Unified Modeling Language (UML)

An OMG standard language for specifying the structure and behavior of systems. The standard defines an abstract syntax and a graphical concrete syntax.

UML Profile

A standardized set of extensions and constraints that tailors UML to particular use.

5 Symbols and abbreviated terms

Abbreviation	Definition
CORBA	Common Object Request Broker Architecture
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
GPP	General Purpose Processor
I/O	Input/Output
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
ISO	International Standards Organization
N/A	Not Applicable
OMG	Object Management Group
ORB	Object Request Broker
OS	Operating System
PIM	Platform Independent Model
PSM	Platform Specific Model
RF	Radio Frequency
SDR	Software Defined Radio
UML	Unified Modeling Language
XML	eXtensible Markup Language

6 Additional Information

6.1 Changes to Adopted OMG Specifications

The specifications contained in this document require no changes to adopted OMG specifications.

6.2 Guide to this Specification

This specification consists of three major parts, contained in the following chapters 7 to 9.

- Chapter 7 defines the modeling language used in this specification in form of a UML profile for Communication Channel, Communication Equipment and Radio Management components. The normative UML Profile specified in model referenced in Section 3.1.3 is used to generate the class diagrams shown throughout this specification.

- Chapter 8 contains the Radio Control Facilities Platform Independent Model (PIM). The UML language extended by the communication channel profile defined in Chapter 7 is used to specify this PIM.
- Chapter 8 contains a description of the mapping process from the Platform Independent Model (PIM) to a Platform Specific Model (PSM).

6.3 Acknowledgements

The following organizations (listed in alphabetical order) contributed to this specification:

- BAE Systems
- The Boeing Company
- Blue Collar Objects
- Carleton University
- Communications Research Center Canada
- David Frankel Consulting
- École de Technologie Supérieure
- General Dynamics Decision Systems
- Harris
- ISR Technologies
- ITT Aerospace/Communications Division
- L-3 Communications Corporation
- Mercury Computer Systems
- The MITRE Corporation
- Mobile Smarts
- Northrup Grumman
- PrismTech
- Raytheon Corporation
- Rockwell Collins
- SCA Technica
- Space Coast Communication Systems
- Spectrum Signal Processing
- THALES
- Virginia Tech University
- Zeligsoft
- 88solutions

7 UML Profile for Communication Channel

This non-normative section defines the UML Profile for Communication Channel, Communication Equipment, and Radio Management. This profile is an integral part of the PIM and PSM for Software Radio Components. The set of stereotypes defined in this profile constitutes the core language for the definition of the communication channel and its management. The UML Profile for Communication Channel extends the UML 2.0 meta-language, with emphasis on extensions to the Class and Components package of UML 2.0.

The goal of the UML Profile for Communication Channel is to enable the development of UML tools to support the development of platforms with communication channels. The objectives are not only to facilitate the modeling of communication channels, but also to enable the automatic generation of descriptor files (e.g., XML descriptor files) and code (or code skeletons) from UML models.

To address the issues of the different actors involved in product developments, the current profile has been developed with two main viewpoints in mind: the viewpoint of system developers and the viewpoint of infrastructure/middleware providers. These two viewpoints define distinct sets of concepts (and stereotypes) that are required in different contexts.

To be consistent with the two viewpoints introduced above, the UML Profile for Communication Channel is partitioned in three main packages: the Communication Equipment package, Communication Channel package, and the Radio Management package. Each package defines the set of concepts and UML stereotypes required to perform a specific role in the development of a product.

The Communication Equipment and Communication Channel packages define the set of concepts that are required to develop communication channels.

The Communication Equipment package defines the concepts that are required to model SWRadio equipment. This package defines stereotypes for the different types of hardware devices used in SWRadio. This package mainly contains a set of stereotypes that extends the UML 2.0 meta-classes Device and Port. This set of stereotypes includes I/O Device, Antenna, Amplifier, Frequency Converter, etc. For each Device stereotype specific characteristics are defined that are required by a waveform component for deployment behavior.

The Communication Channel package defines the concepts that are required to model the communication equipment that make up a channel and defines different parts of channel (security, I/O, processing, physical). This package mainly contains a set of stereotypes that extends the UML 2.0 meta-class Class.

The Radio Management package defines the concepts that are required to manage communication channels. This package mainly contains a set of stereotypes that extends the UML 2.0 meta-class Component. This set of stereotypes includes CommChannelComponent, RadioManagerComponent, and RadioSystemManager components.

The format for stereotype names to agree with resolution (e.g., commequipment) in the profile is all lower case letters. In this specification the stereotype names are shown with mixture of upper and lower case letters, where each word starts with an upper case letter (e.g., CommEquipment).

7.1 Communication Equipment

The Communication Equipment package contains device stereotypes that describe devices realized by a specific Communication Channel. The selected devices represent basic functions associated with software radio equipment. Additional stereotypes are defined for modeling the relationships between radio devices. However, this specification neither dictates, nor restricts, the arrangement of radio devices. Actual connection definitions between devices are left to the implementer.

The purpose of the Communication Equipment package is twofold. It defines a language to describe a specific hardware platform upon which applications execute. This description can be stored in XML files for automatic processing. This enables the deployment and configuration machinery to acquire knowledge about the platform capabilities. This information could be used to determine whether or not a platform has the required capabilities to run an application before instantiating it. On the other hand, this language is also useful from a system engineering point of view. By mapping the information contained in the model to a simulation language, the operating capabilities of a radio platform can be studied off-line. This greatly eases the application development and porting process since the actual hardware platform is not required for determining if a specific platform can support a specific waveform. The stereotypes providing this language are summarized in Table 7.1.

It must be noted that this package only provides basic definition for a software radio hardware device. Implementers can extend device definitions to meet their specific needs.

Table 7.1 - Communication Equipment Stereotypes

Stereotype	Base Class	Parent	Tags	Constraints	Description
Amplifier	Device	IODevice	dutyCycle, gain, maxGain, minGain	See constraints in section below	Increases the energy of signals passing through it.
AnalogInputPort	Port	N/A	inputImp, inputLevel, maxInputLevel, insertionLoss, inputVSWR	See constraints in section below	Receives an analog signal.
AnalogOutputPort	Port	N/A	maxOutputLevel, outputImp, outputVSWR	See constraints in section below	Transmits an analog signal.
Antenna	Device	IODevice	calibration, radiationPattern, polarization, type, maxRadiationPattern, minRadiationPattern, polarizationCapability	See constraints in section below	Converts an electrical signal into an electromagnetic wave and vice versa for carrying data over an air interface.
AntennaElement	Device	IODevice	polarization, positionInAntennaArray, radiationPattern, type, active	See constraints in section below	Translates electrical energy into an electromagnetic wave and vice-versa.
AudioDevice	Device	IODevice	N/A	There has to be at least one AnalogInput Port.	Converts electrical signals into sounds waves.

Table 7.1 - Communication Equipment Stereotypes

Stereotype	Base Class	Parent	Tags	Constraints	Description
CommEquipment	Device	N/A	equipmentInformation, equipmentSize, equipmentWeight, powerConsumption, maxOperatingTemperature, minOperatingTemperature, radiationCapability, meanTimeBetweenFailures, lastMaintenanceCheck, maintenancePeriod, temperatureStatus	See constraints in section below	Represents a radio communication device.
CommEquipment Communication Path	Communication Path	N/A	N/A	See constraints in section below	Represents an association between communication equipments through which signals and messages are exchanged.
CommEquipment Connector	Connector	N/A	N/A	See constraints in section below	Represents a link that enables communication between two or more instances of communication equipment ports.
Converter	Device	IODevice	converterKind, maxSampleRate, minSampleRate, phaseNoise, sampleRate, sampleSize	See constraints in section below	Converts an analog signal into a digital signal or/and vice versa.
CryptoDevice	Device	CommEquipment	algorithm, keyLength	See constraints in section below	Performs encryption and decryption on a set of data.
DigitalPort	Port	N/A	quantizationNoise, dataFlowDirection, streaming, maxThroughput	See constraints in section below	Receives or transmits a digital signal.
Filter	Device	IODevice	N/A	See constraints in section below	Alters the frequency spectrum of signals passing through it.

Table 7.1 - Communication Equipment Stereotypes

Stereotype	Base Class	Parent	Tags	Constraints	Description
Frequency Converter	Device	IODevice	currentInputFrequency, currentOutputFrequency, maxInputFrequency, minInputFrequency, maxOutputFrequency, minOutputFrequency, loInputLeakagePower, loOutputLeakagePower, outputToInputLeakage, phaseNoise, loStability	See constraints in section below	Performs frequency translation in such a manner that the output frequencies are higher/ lower in the spectrum than the input frequencies.
Hopping Frequency Converter	Device	FrequencyConverter	nextInputFrequency, nextOutputFrequency	N/A	Performs hopping frequency conversion.
IODevice	Device	CommEquipment	maxPowerHandling, minPowerHandling, noiseFigure, maxOperatingVSWR, freqResponse, tunedFrequency, maxFrequencyResponse, minFreqencyResponse, maxFrequency, minFrequency, amplitudePhaseResponse	N/A	Operates on a signal.
Microphone	Device	IODevice	N/A	There has to be at least one Analog OutputPort.	Converts sound waves into an electrical signal.
PowerSupply	Device	CommEquipment	type, efficiency	N/A	Provides electrical power to other devices.
Processor	Device	CommEquipment	processorArchitecture, maxOperatingFrequency, nonVolatileMemory Capacity, volatileMemoryCapacity	See constraints in section below	Processes digital or analog data.
Programmable LogicDevice	Device	Processor	logicUnitCapacity, reconfigurability, timeForReconfiguration	N/A	Uses hardware logic to process data.
RadiatingElement	Device	IODevice	active, radiationPattern, polarization, type, positionInAntennaArray	See constraints in section below	Represents the part of an antenna that actually emits and receives electromagnetic waves.

Table 7.1 - Communication Equipment Stereotypes

Stereotype	Base Class	Parent	Tags	Constraints	Description
SerialIODevice	Device	IODevice	N/A	N/A	Transmits and receives digital signals serially.
SoftwareProcessor	Device	Processor	operatingEnvironment	N/A	Uses software instructions to process digital data.
Switch	Device	IODevice	inputOutputIsolation, switchSetting	See constraints in section below	Connects two I/O ports to each other given a specific configuration.

7.1.1 Types and Exceptions

- `AmplitudePhaseResponse`
  An amplitude phase response is an array of PowerLevels. An amplitude phase response with one point represents a 1 dB compression point. In an amplitude phase response with two points, the first point represents the 1 dB compression point and the second point represents the IP3 (third order intercept) point. An amplitude phase response with more than two points represents the entire AM-to-AM. Typically, curves represent instantaneous power.

- `AntennaCalibration: OctetSequence`
  Antenna calibration data.

- `<<primitive>>AntennaType`
  AntennaType, a specialization of String, denotes the physical configuration of an antenna (e.g., OMNI, DIRECTIONAL, etc.).

- `<<primitive>>ArchitectureType`
  ArchitectureType, a specialization of String, denotes the architecture of the device (e.g., FPGA, CPLD, PPC, x86, etc.).

- `CartesianCoordinates(x: Meter, y: Meter, z: Meter)`
  Three dimensional coordinates. This type is used to specify the location of an object from a given reference point.

- `<<enumeration>> ConverterType (ATOD, DTOA, BOTH)`
  The ConverterType defines the type of the converter. A converter can be an analog to digital converter (ATOD), digital to analog converter (DTOA), or can have both functionalities (BOTH).

- `<<primitive>>CryptoAlgorithm`
  CryptoAlgorithm, a specialization of String, denotes the type of crypto algorithm (e.g., BLOWFISH, RSA, DES, 3DES, AES, HASH_MD5, etc.).

- `Date(ULong day, ULong month, ULong year)`
  Date in days, months, and years.

- `Decibel`
  Decibel, a specialization of Float, denotes the ratio between two voltages, currents, or signal power levels.

- `DeviceModelInformation(manufacturerName: String, modelName: String, modelNumber: String, modelDescription: String, serialNumber: String,`

majorRevision: String, minorRevision: String)
Generic information about a hardware device.

- <<primitive>>Degrees
  Degrees, a specialization of Float, denotes units of measurement for angles.

- <<enumeration>>Direction (INPUT, OUTPUT, BOTH)
  Direction of data flow.

- <<primitive>>DistributionType
  DistributionType, a specialization of String, specifies the type of probability distribution (e.g., GAUSSIAN, POISSON, RAYLEIGH, RICIAN, BINOMIAL, CHISQUARE, TDISTRIBUTION, WEIBULL, LOGNORMAL, NONE).

- FrequencyResponseType
  FrequencyResponseType is array of FrequencyResponsePoint(s).

- FrequencyResponsePoint(frequency: Hertz, amplitude: Decibel, phase: Degrees)
  A frequency response is the relation between signal amplitude and gain versus frequency. A frequency response with only one point represents a single-sided 3 dB bandwidth. A frequency response with more than one point is an arbitrary frequency response with an arbitrary resolution. A given frequency response has 0 dB gain and is centered at 0 Hz (it does not have to be symmetric).

- <<primitive>>Hertz
  Hertz, a specialization of Double, a unit of frequency equal to one cycle per second.

- Impedance (resistance: Float, reactance: Float)
  Impendance type denotes the opposition that a device offers to an electric current. Impedance is composed of two components, resistance and reactance.

- <<primitive>>LogicUnit
  LogicUnit, a specialization of ULong, denotes the description a basic logic blocks available inside the device.

- <<primitive>>Meter
  Meter, a specialization of Double, denotes the fundamental unit of length in the metric system.

- PatternOrientationType (elevation: Degrees, azimuth: Degrees)
  The pattern orientation is represented by an elevation angle which gives the vertical orientation and an azimuth angle which gives the horizontal orientation.

- <<primitive>>PhaseNoise
  PhaseNoise, a specialization of Short, denotes random and short duration fluctuations in the phase of a signal.

- <<enumeration>>PolarizationKind (VERTICAL, HORIZONTAL, RIGHT_CIRCULAR_POLARIZE, LEFT_CIRCULAR_POLARIZE)
  The orientation of the RF energy radiated from the device.

- <<primitive>>Power
  Power, a specialization of Float, denotes the Rate at which electrical energy is transformed to another type of energy.

- Powerlevels (inputPower: Float [0..1], outputPower: Float [0..1])
  The factor Power is a measure of the input (or output) signal strength expressed in dBm referenced to 50 Ohms.

- <<enumeration>>PowerSupplyType (AC_DC, DC_DC)
  If a device is of AC_DC type, it converts AC power to DC power. If the device is of DC_DC type, it converts DC power to DC power.

- `ProbabilityDensity (distribution: DistributionType, parameterList: double[*])`
  Specifies an exact or approximate value of a probability density function. In case distribution is NONE, parameterList refers to the expected values of the random variable E(x), E(x^2), E(x^3), ... etc. Otherwise, parameters list contains the parameters required by the distribution type.

- `QuantizationNoiseDensity`
  Distribution function estimating the quantization noise resulting from using a specific quantization process.

- `<<primitive>>RadiatingElementType`
  RadiatingElementType, a specialization of String, denotes the physical configuration of a radiating element (e.g., MONOPOLE, DIPOLE, PATCH, CONE, DISH, etc.).

- `<<primitive>>Radiation`
  Radiation, a specialization of Float, denotes information about a specific radiation environment.

- `RadiationPatternType ( azimuthPlane: RadiationPatternPoint [0..*], elevationPlane: RadiationPatternPoint [0..*])`
  Field intensity variation of an antenna as an angular function with respect to the azimuth and elevation axis.

- `RadiationPatternPoint (gain: Decibel, angle: Degrees)`
  A single point in the radiation pattern is made of a gain value and an angle value.

- `<<enumeration>>ReconfigurabilityType (STATIC, DYNAMIC)`
  STATIC reconfigurability means that the device is configured at the start of execution and remains unchanged for the duration of the application. DYNAMIC reconfigurability means the ability for partial reconfiguration of certain logic blocks while others are performing computations.

- `Size(Float x, Float y, Float z)`
  Represents the physical size of an object in a given unit.

- `SwitchMapping (inputPortNumber: UShort, outputPortNumber: UShort)`
  A SwitchMapping is the association of an input port with an output port; thus creating a connection inside the switch.

- `SwitchSettingType`
  SwitchSettingType is sequence of SwitchMapping that indicates the connections between the switch's ports.

- `Temperature`
  Temperature, a specialization of Float, represents the temperature of an object in a given unit (Celsius, Kelvin…).

- `VSWR`
  VSWR, a specialization of Float, denotes the ratio of the device operating impedance to a desired characteristic impedance (usually 50 ohm characteristic impedance reference).

- `Weight`
  Weight, a specializtion of Float, represents the physical weight of an object in a given unit.

7.1.2 CommEquipmentCommunicationPath

Description

The CommEquipmentCommunicationPath stereotype is an extension of the UML 2.0 CommunicationPath metaclass (from UML2.0::Deployments::Nodes). A CommEquipmentCommunicationPath is an association between two communication equipment elements, through which signals and messages may be exchanged.

Constraints

The association ends of a CommEquipmentCommunicationPath are of type CommEquipment.

7.1.3 CommEquipmentConnector

Description

The CommEquipmentConnector stereotype is an extension of the UML 2.0 Connector metaclass (from UML2.0::CompositeStructures::InternalStructures). A CommEquipmentConnector is a link that enables communication between two or more instances of communication equipments ports (see Section 7.1.4).

Constraints

The type attribute must be of CommEquipmentCommunicationPath type.

A CommEquipmentConnector connects compatible hardware ports. A set of compatible ports consists either one AnalogInputPort and one AnalogOutputPort or two DigitalPorts. In the case of two DigitalPorts, one DigitalPort must be the input port and the other must be the output port.

7.1.4 Port

Communication equipments communicate with each other through ports. Three extensions to the UML 2.0 Port metaclass (from UML2.0::CompositeStructures::Ports) are defined: AnalogInputPort, AnalogOutputPort, and DigitalPort. By using the port stereotype, the implementer can customize or extend a device with additional ports for exchanging control, status, or any other information. An example is an amplifier. Typically, when an amplifier has two ports, it is a fixed gain amplifier, when it has three ports it can be an AGC.

A bidirectional analog port can be constructed by aggregating one AnalogInputPort and one AnalogOutputPort. A bidirectional digital port can be constructed by aggregating two instances of DigitalPort.

7.1.4.1 AnalogInputPort

Description

The AnalogInputPort stereotype is an extension of the UML 2.0 Port metaclass (from UML2.0::CompositeStructures::Ports). The AnalogInputPort defined the attributes of an analog input port.

Attributes

- `<<characteristicproperty>>inputImpedance: Impedance`
  The inputImpedance attribute represents the impedance of the port.
- `<<characteristicproperty>>inputLevel: Power`
  The inputLevel attribute represents the power level currently at the input of the port.
- `<<characteristicproperty>>inputVSWR: VSWR [0..1]`
  The inputVSWR attribute represents the voltage standing wave ratio of the port.
- `<<characteristicproperty>>insertionLoss: Decibel`
  The insertionLoss attribute represents the loss occurring when a device is inserted in a transmission line. This value is the ratio between the signal powers on that end of the line after and before insertion of the device.

- `<<characteristicproperty>>maxInputLevel: Power`
  The maxInputLevel attribute represents the maximum input power that the port can sustain.

Constraints

An AnalogInputPort can only be connected to an AnalogOutputPort.

7.1.4.2 AnalogOutputPort

Description

The AnalogOutputPort stereotype is an extension of the UML 2.0 Port metaclass (from UML2.0::CompositeStructures::Ports). The AnalogOutputPort defines the attributes of an analog output port.

Attributes

- `<<characteristicproperty>>maxOutputLevel: Power`
  The maxOutputLevel attribute represents the maximum output power that the port can provide.

- `<<characteristicproperty>>outputImpedance: Impedance`
  The outputImpedance attribute represents the impedance of the port.

- `<<characteristicproperty>>outputVSWR: VSWR [0..1]`
  The outputVSWR attribute represents the voltage standing wave ratio of the port.

Constraints

An AnalogOutputPort can only be connected to an AnalogInputPort.

7.1.4.3 DigitalPort

Description

The DigitalPort stereotype is an extension of the UML 2.0 Port metaclass (from UML2.0::CompositeStructures::Ports). The DigitalPort defines the attributes of a digital port.

Attributes

- `<<characteristicproperty>>dataFlowDirection: Direction`
  The dataFlowDirection attribute indicates whether the port is an input port or an output port.

- `<<characteristicproperty>>maxThroughput: Float`
  The maximum throughput of the port.

- `<<characteristicproperty>>quantizationNoise: ProbabilityDensity`
  The quantizationNoise attribute represents the noise resulting from the approximation error in the quantization process. Quantization noise is related to the specific quantization process and the characteristics of the quantized signal.

- `<<characteristicproperty>>streaming: Boolean`
  The streaming attribute indicates if the port is a streaming port.

Constraints

A DigitalPort with the dataFlowDirection attribute equal to INPUT can only be connected to a DigitalPort with the dataFlowDirection attribute equal to OUTPUT and vice versa.

7.1.5 CommEquipment

Description

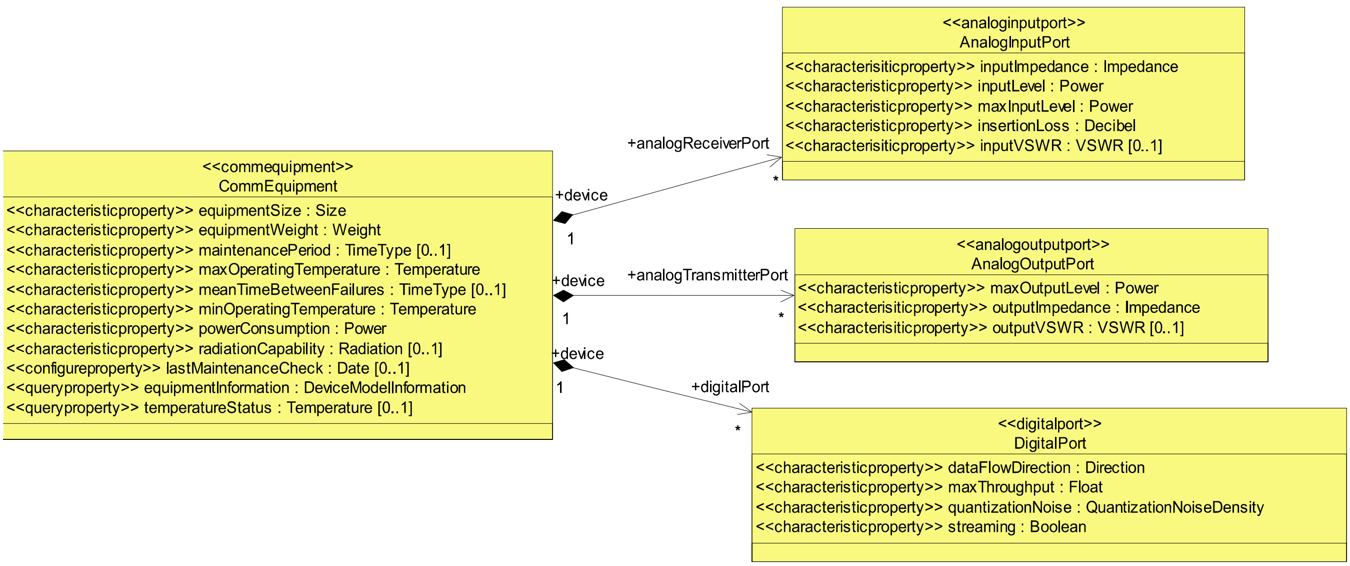

Figure 7.1 - CommEquipment M1 Illustration

The CommEquipment stereotype is an extension of the UML 2.0 Device metaclass (from UML2.0::Deployment::Nodes).

CommEquipment is the overall base class for describing that collection of devices, which are used to realize the Communication Channel. Antennas, amplifiers, CPUs, and FPGAs are example CommEquipment Elements. They all contain the attributes of the CommEquipment stereotype.

Attributes

- `<<characteristicproperty>>equipmentSize: Size`
  The size attribute indicates the size of the physical device.

- `<<characteristicproperty>>equipmentWeight: Weight`
  The weight attribute indicates the weight of the physical device.

- `<<characteristicproperty>>maxOperatingTemperature: Temperature`
  The maxOperatingTemperature attribute indicates the maximum sustainable operating temperature of the physical device.

- `<<characteristicproperty>>meanTimeBetweenFailures: TimeType [0..1]`
  The meantimeBetweenFailures attribute indicates the length of time a user may reasonably expect a component to work properly before an incapacitating fault occurs.

- `<<characteristicproperty>>minOperatingTemperature: Temperature`
  The minOperatingTemperature attribute indicates the minimum sustainable operating temperature of the physical device.

- `<<characteristicproperty>>powerConsumption: Power`
  The powerConsumption attribute indicates the power consumed by the device.

- `<<characteristicproperty>>radiationCapability: Radiation [0..1]`
  The radiationCapability attribute indicates the sustainable radiation level of the physical device. This attribute is useful for radiation hardened devices.

- `<<configureproperty>>lastMaintenanceCheck: Date [0..1]`
  The lastMaintenanceCheck attribute indicates the date at which the last maintenance check was performed. Could be used for devices requiring manual calibration.

- `<<queryproperty>>equipmentInformation: DeviceModelInformation`
  The equipmentInformation attribute gives descriptive information about the physical device. This information could be used in a plug and play hardware environment.

- `<<queryproperty>>maintenancePeriod: TimeType [0..1]`
  The maintenancePeriod attribute indicates the time interval between required maintenance check. Could be used for components requiring manual calibration.

- `<<queryproperty>>temperatureStatus: Temperature [0..1]`
  The temperatureStatus attribute indicates the internal temperature of the device.

Constraints

CommEquipment shall have a composite relationship to at least one of the following: AnalogInputPort, AnalogOutputPort, or DigitalPort.

7.1.5.1 CryptoDevice

Description

The CryptoDevice stereotype represents a dedicated device that performs encryption and decryption services for Communication Channels. Typically, these devices are used in military communication systems; there are also commercial devices that perform these functions.

Attributes

- `<<queryproperty>>keyLength: UShort [0..*]`
  The keyLength attribute indicates the length of the cipher key supported by the device. It could be either 1024 bits long or more for Public Key or 128 bits or more for Symmetric Key. More than one key length can be supported depending on the algorithm.

- `<<queryproperty>>algorithm: CryptoAlgorithm [0..*]`
  The algorithm attribute identifies the cryptographic algorithms supported by the device.

Constraints

A CryptoDevice shall have at least two DigitalPorts.

A CryptoDevice's DigitalPort shall have its dataFlowDirection attribute equal to INPUT and the other CryptoDevice's DigitalPort shall have its dataFlowDirection attribute equal to OUTPUT.

7.1.5.2 IODevice

Description

The IODevice stereotype represents the base stereotype for all devices that provide analog or digital input/output capability for the RadioSet.

The IODevice class not only applies to the subscriber-side of the radio but also to the RF-side. The term subscriber-side does not imply a human actor. From a higher perspective, both ends of a radio can be considered as I/O. Filters, amplifiers, etc., can be found on both the subscriber-side and RF-side of the equipment.

The members of the IODevice class were conceived with this flexibility in mind. This implies that all devices can operate at non DC frequencies. The IODevice class includes a "tunedFrequency" parameter which can have any frequency as a valid entry.

Elements inheriting the IODevice class can be used to construct more complex elements like receivers and exciters.

All of the attributes are optional to cover the specifics of both analog and digital IO devices.

Attributes

- `<<characteristicproperty>> noiseFigure: Decibel [0..1]`
  The noiseFigure attribute is the ratio of the noise power at the output to the noise power at the input, where the input noise temperature is equal to the reference temperature (290 K). The noise figure is expressed in decibels.

- `<<characteristicproperty>>amplitudePhaseResponse: AmplitudePhaseResponse [0..1]`
  The amplitudePhaseResponse attribute gives the amplitude/phase response plot for the device. The amplitude phase response contains two components. The first component is a representation of the output power versus the input power. The second component is a representation of the output phase versus input power. The purpose of the amplitude phase response is to describe any active element which cannot be described by an ideal relationship (non linearities) e.g., Power Amplifier.

- `<<characteristicproperty>>maxTunedFrequency: Hertz [0..1]`
  The maxTunedFrequency attribute is the maximum frequency of the bandwidth for which the device performance is rated.

- `<<characteristicproperty>>maxFrequencyResponse: FrequencyResponseType [0..1]`
  The maxFrequencyResponse attribute is the maximum frequency response the device is able to achieve. The maximum amplitude and/or phase at a given frequency.

- `<<characteristicproperty>>maxOperatingVSWR: VSWR [0..1]`
  The maxOperatingVSWR attribute is the ratio of the device operating impedance to a desired characteristic impedance (usually 50 ohm characteristic impedance reference).

- `<<characteristicproperty>>maxPowerHandling: Power [0..1]`
  The maxPowerHandling attribute is the maximum power the device can sustain.

- `<<characteristicproperty>>minTunedFrequency: Hertz [0..1]`
  The minTunedFrequency attribute is the minimum frequency of the bandwidth for which the device performance is rated.

- `<<characteristicproperty>>minFrequencyResponse: FrequencyResponseType [0..1]`
  The minFrequencyResponse attribute is the minimum frequency response the device is able to achieve. The minimum amplitude and/or phase at a given frequency.
- `<<characteristicproperty>>minPowerHandling: Power [0..1]`
  The minPowerHandling attribute is the minimum RF power the device must be supplied in order to work.
- `<<configureproperty>>freqResponse: FrequencyResponseType [0..1]`
  The freqResponse attribute represents the frequency response plot for the device.
- `<<configureproperty>>tunedFrequency: Hertz [0..1]`
  The tunedFrequency attribute corresponds to the center frequency of the frequency response.

Constraints

An IODevice shall have at least one AnalogInputPort or one AnalogOutputPort or one DigitalPort.

7.1.5.2.1 Amplifier

Description

The Amplifiers stereotype represents a device that provides gain. Amplifiers include but are not limited to base band, RF, power and low noise amplifiers. Different Amplifier types are differentiated by the values of their attributes.

Attributes

- `<<characteristicproperty>>dutyCycle: ULong`
  The dutyCycle attribute is the maximum continuous duty cycle the device can operate.
- `<<characteristicproperty>>maxGain: Decibel`
  The maxGain attribute is the maximum power amplification factor a device is able to apply to a signal.
- `<<characteristicproperty>>minGain: Decibel`
  The minGain attribute is the minimum power amplification factor a device is able to apply to a signal.
- `<<configureproperty>>gain: Decibel`
  The gain attribute is the current power amplification factor applied to the input signal by the device.

Constraints

An Amplifier shall have at least (one AnalogInputPort and one AnalogOutputPort) or two DigitalPorts.

7.1.5.2.2 Antenna

Description

The Antenna stereotype, shown in Figure 7.2, represents the RF radiating elements necessary for transmission/reception of radio energy through the ether. The Antenna class consists of both a simple passive radiating element as well as an antenna array with possibly some dedicated intelligence.

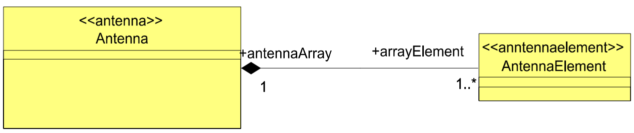

Figure 7.2 - Antenna M1 Illustration

Attributes

- `<<characteristicproperty>>maxRadiationPattern: RadiationPatternType`
  The maxRadiationPattern attribute indicates the maximum radiation pattern that the device is able to achieve.
- `<<characteristicproperty>>minRadiationPattern: RadiationPatternType`
  The minRadiationPattern attribute indicates the minimum radiation pattern that the device is able to achieve.
- `<<characteristicproperty>>polarizationCapability: PolarizationKind`
  The polarizationCapability attribute gives the orientation options of the RF energy radiated from the antenna.
- `<<characteristicproperty>>type: AntennaType`
  The type attribute indicates the physical type of the antenna.
- `<<configureproperty>>calibration: AntennaCalibration`
  The calibration attribute contains calibration data for the antenna.
- `<<configureproperty>>polarization:PolarizationKind[0..1]`
  The polarization attribute indicates the current orientation of the RF energy radiated from the antenna.
- `<<configureproperty>>radiationPattern: RadiationPatternType`
  The radiationPattern attribute represents the current radiation pattern configured in the device.

M1 Associations

- `arrayElement: AntennaElement [1..*]`
  The individual radiating element objects of the antenna.

Constraints

An Antenna shall have at least one AnalogInputPort or one AnalogOutputPort.

7.1.5.2.3 AntennaElement

Description

The AntennaElement stereotype represents a device that translates electrical energy into an electromagnetic wave and vice-versa. An AntennaElement is a passive element. The AntennaElement acts as the transducer between the electrical world and the air interface. Typical examples can be cones, patches, dipoles, dishes, etc.

Attributes

- `<<characteristicproperty>>polarization: PolarizationKind`
  The polarization attribute indicates the orientation of the RF energy radiated for the AntennaElement.
- `<<characteristicproperty>>positionInAntennaArray: CartesianCoordinates`
  The positionInAntennaArray attribute indicates its 3D position in array with repect to the geometric center of the array.
- `<<characteristicproperty>>radiationPattern: RadiationPatternType`
  The radiationPattern attribute represents the current radiation pattern for this single AntennaElement.
- `<<characteristicproperty>>type: RadiatingElementType`
  The type attribute indicates the physical configuration of the AntennaElement.
- `<<configureproperty>>active: Boolean`
  The active attribute indicates if the AntennaElement is currently active.

M1 Associations

- `antennaArray: Antenna [1]`
  The antenna object which the AntennaElement is part of.

Constraints

An AntennaElement shall have at least one AnalogInputPort and one AnalogOutputPort.

7.1.5.2.4 Converter

Description

The Converter stereotype represents a device that performs analog-to-digital and / or digital-to-analog conversion of transmit and / or receive signal.

Attributes

- `<<characteristicproperty>> converterKind: ConverterType`
  The converterKind attribute represents the type of converter. It can be an ATOD, DTOA, or BOTH.
- `<<characteristicproperty>>maxSampleRate: Hertz`
  The maxSampleRate attribute is the maximum sample rate the device is able to achieve.
- `<<characteristicproperty>>minSampleRate: Hertz`
  The minSampleRate attribute is the minimum sample rate the device is able to achieve.
- `<<characteristicproperty>>phaseNoise: ProbabilityDensity`
  The phaseNoise attribute represents the phase noise that the device introduces in the signal.

- `<<characteristicproperty>>sampleSize: ULong`
  The sampleSize attribute represents the size in bits of a sample.

- `<<configureproperty>>sampleRate: Hertz`
  The sampleRate attribute is the current number of samples per second converted by the device.

Constraints

A Converter shall have at least (one AnalogInputPort or one AnalogOutputPort) and one DigitalPort.

7.1.5.2.5 Filter

Description

The Filter stereotype provides selective frequency gain or attenuation to the Communication Channel in both analog and digital domains. Filters also provide signal shaping in both amplitude and phase to the Communications Channel.

In a Communication Channel, filters can often be called by other names depending on their location and / or functionality (e.g., duplexers, interference cancellers, equalizers). The filter class regroups all those devices regardless of their implementation, location, and function. The filter class recognizes that the functionality of all these devices is to attenuate/enhance some frequency components of the signal. Furthermore, since the frequency response is a configure property; the filter class can represent both fixed and adaptive filters.

Due to the large number of “filters” in a Communication Channel, the filter device can be found between every other type of device. It is certainly frequent to have filters before ADC and after DAC, before and / or after amplifiers, frequency converters, antennas/radiating elements, and switches.

Constraints

A Filter shall have at least one AnalogInputPort and one AnalogOutputPort.

7.1.5.2.6 FrequencyConverter

Description

The FrequencyConverter stereotype represents an analog or digital device that translates signals between one center frequency to another center frequency. When the output center frequency is higher than the input center frequency, the device is called an up converter otherwise it is called a down converter. Like filters frequency converters can take many names or forms (e.g., Direct RF, frequency hopping, harmonic, etc.).

In an analog FrequencyConverter, the local oscillator is assumed to be part of the device. Therefore the FrequencyConverter can be a device with two ports. This choice was made to support elegantly harmonic converters and other devices, which do not require an external local oscillator.

The FrequencyConverter device does not implement the entire exciter or receiver concept by itself. However, it is a key building block in the definition of higher level concepts.

Attributes

- `<<characteristicproperty>>loInputLeakagePower: Power [0..1]`
  The loInputLeakagepower attribute represents the local oscillator input leakage power.

- `<<characteristicproperty>>loOutputLeakagePower: Power [0..1]`
  The loOutputLeakagePower attribute represents the local oscillator output leakage power.

- `<<characteristicproperty>>loStability: UShort [0..1]`
  The loStability attribute represents the local oscillator stability expressed in PPM.

- `<<characteristicproperty>>maxInputFrequency: Hertz [0..1]`
  The maxInputFrequency attribute represents the maximum input signal frequency the device is able to handle.

- `<<characteristicproperty>>maxOutputFrequency: Hertz [0..1]`
  The maxOutputFrequency represents the maximum output signal frequency the device is able to handle.

- `<<characteristicproperty>>minInputFrequency: Hertz [0..1]`
  The minInputFrequency represents the minimum input signal frequency the device is able to handle.

- `<<characteristicproperty>>minOutputFrequency: Hertz [0..1]`
  The minOutputFrequency represents the minimum output signal frequency the device is able to handle.

- `<<characteristicproperty>>outputToInputLeakage: Decibel [0..1]`
  The outputToInputLeakage attribute indicates the amount of the output frequency which is found at the input.

- `<<characteristicproperty>>phaseNoise: PhaseNoiseType [0..1]`
  The phaseNoise attribute represents the phase noise that the device introduces in the signal.

- `<<configureproperty>>currentInputFrequency: Hertz [0..1]`
  The currentInputFrequency indicates the frequency of the signal currently at the input of the device.

- `<<configureproperty>>currentOutputFrequency: Hertz [0..1]`
  The currentOutputFrequency indicates the frequency of the signal currently at the output of the device.

Constraints

A FrequenceConverter shall have (at least one AnalogInputPort and one AnalogOutputPort) or (at least two DigitalPorts).

7.1.5.2.7 HoppingFrequencyConverter

Description

The HoppingFrequencyConverter stereotype represents a device that performs frequency conversion while switching between predefined frequencies. It is a specialization of the FrequencyConverter stereotype.

Attributes

- `<<configureproperty>>nextInputFrequency: Hertz [0..1]`
  The nextInputFrequency attribute represents the input frequency that the device will select after the next triggering event. This attribute is used for instantaneous frequency changes; typically in the context of frequency hopping and frequency scanning algorithms.

- `<<configureproperty>>nextOutputFrequency: Hertz [0..1]`
  The nextOutputFrequency attribute represents the output frequency that the device will select after the next triggering event. This attribute is used for instantaneous frequency changes; typically in the context of frequency hopping and frequency scanning algorithms.

7.1.5.2.8 Switch

Description

The Switch stereotype represents a device that provides routing of signals between different devices. A Switch may have many input and output ports and it connects the chosen input port to one or many output ports. It may also be programmed to turn off the signal transmission. In this case, no input would be connected to any output port.

Attributes

- `<<characteristicproperty>>inputOutputIsolation: Decibel`
  The inputOutputIsolation attribute represents the amount of input port leakage on all unselected output ports.
- `<<configureproperty>>switchSetting: SwitchSettingType`
  The switchSetting attribute indicates the current configuration matrix of the device.

Constraints

A Switch shall have (at least one AnalogInputPort and one AnalogOutputPort) or (at least two DigitalPorts).

7.1.5.3 PowerSupply

Description

The PowerSupply stereotype represents a device that provides electrical power to CommEquipment components. It is therefore associated with all other CommEquipment components. It must be noted that this specification does not address the issue of power management. It is expected that power management is the responsibility of a higher level application.

Attributes

- `<<characteristicproperty>>efficiency: UShort`
  The efficiency attribute is the ratio of signal power output to total power input.
- `<<characteristicproperty>>type: PowerSupplyType`
  The type attribute indicates if the device converts AC power to DC power or DC power to DC power.

Constraints

A PowerSupply shall have at least one AnalogInputPort representing the input voltage and one AnalogOutputPort for the output voltage.

The allowed PowerSupply input or output voltage value range for Efficiency is from 0 to 1 (it is expressed as a percentage).

7.1.5.4 Processor

Description

The Processor stereotype represents a device that provides computational functions along with supporting functions such as memory and I/O. Processor types include general purpose processors (such as PowerPCs, x86s, etc.), digital signal processors, field programmable gate arrays, application-specific integrated circuits configured for computational functions, and others.

Examples of devices that are considered as processors can include but are not limited to: digital down converter, codec, interconnect, RAID subsystem, memory subsystem, etc.

Due to the diverse nature of these devices, they are modeled by their communication capability, i.e., their ports and by their volatile and non-volatile memory capacities.

Attributes

- `<<characteristicproperty>>maxOperatingFrequency: Hertz [0..1]`
  The maxOperatingFrequency attribute indicates the maximum frequency at which the device is able to operate.
- `<<characteristicselectionproperty>>processorArchitecture: ArchitectureType`
  The architecture attribute indicates the specific type of programmable device.
- `<<queryproperty>>nonVolatileMemoryCapacity: ULong`
  The nonVolatileMemoryCapacity attribute indicates the total number of bytes of persistent memory available to the processor.
- `<<queryproperty>>volatileMemoryCapacity: ULong`
  The volatileMemoryCapacity attribute indicates the total number of bytes of volatile memory available to the processor.

Constraints

A Processor shall have at least one DigitalPort.

7.1.5.4.1 ProgrammableLogicDevice

Description

The ProgrammableLogicDevice stereotype represents a device that processes digital data using hardware logic. It is a specialization of the Processor class. Examples of programmable logic device (PLD) are FPGA and CPLD. This stereotype contains attributes specific to this type of device. Basic logic blocks are used to dynamically instantiate a particular function during device initialization.

Attributes

- `<<characteristicproperty>>logicUnitCapacity: LogicUnit`
  The logicUnitCapacity attribute is the total amount of logic units available inside the device.
- `<<characteristicproperty>>reconfigurability: ReconfigurabilityType`
  The reconfigurability attribute indicates whether the device is statically or dynamically reconfigurable.
- `<<characteristicproperty>>timeForReconfiguration: TimeType`
  The timeForReconfiguration attribute indicates the duration of the reconfiguration process.

7.1.5.4.2 SoftwareProcessor

Description

The SoftwareProcessor stereotype represents a device that executes software instructions in order to execute specific algorithms. GPP and DSP processor are example devices of this type.

Attributes

- <<characteristicsetproperty>>operatingEnvironment: NameVersionCharacteristic [1..*]
  The operatingEnvironment attribute contains information regarding the operating environment that the device is using.

7.2 Communication Channel

Description

A SWRadio provides a means to enable communications between physically separated users. A SWRadio has the capability to utilize its devices as needed for a particular communications scenario and to use them possibly in a different way in another instance. A Communication Channel is the data description for the collection and interconnection of the radio's devices necessary for a particular application to be able to provide communication.

The LogicalCommunicationChannel, shown in which inherits from an abstract Channel class is logically partitioned into three groups: the LogicalPhysicalChannel (e.g., Radio Frequency (RF)), the LogicalProcessingChannel, and the LogicalIOChannel. LogicalPhysicalChannel bundles devices that provide communication over the physical medium, LogicalIOChannel bundles devices that provide I/O functionality for the platform and LogicalProcessingChannel for signal processing needs. A radio may support one or many different logical communication channels. It may support multiple communication channels, but not all simultaneously due to the need for some device(s) by multiple waveforms. The CommChannelComponent as defined in Radio Management (Section 7.3) needs visibility into the capabilities needed by its applications and the capabilities provided by its devices in order to deploy a usable communication channel.

7.2.1 Channel

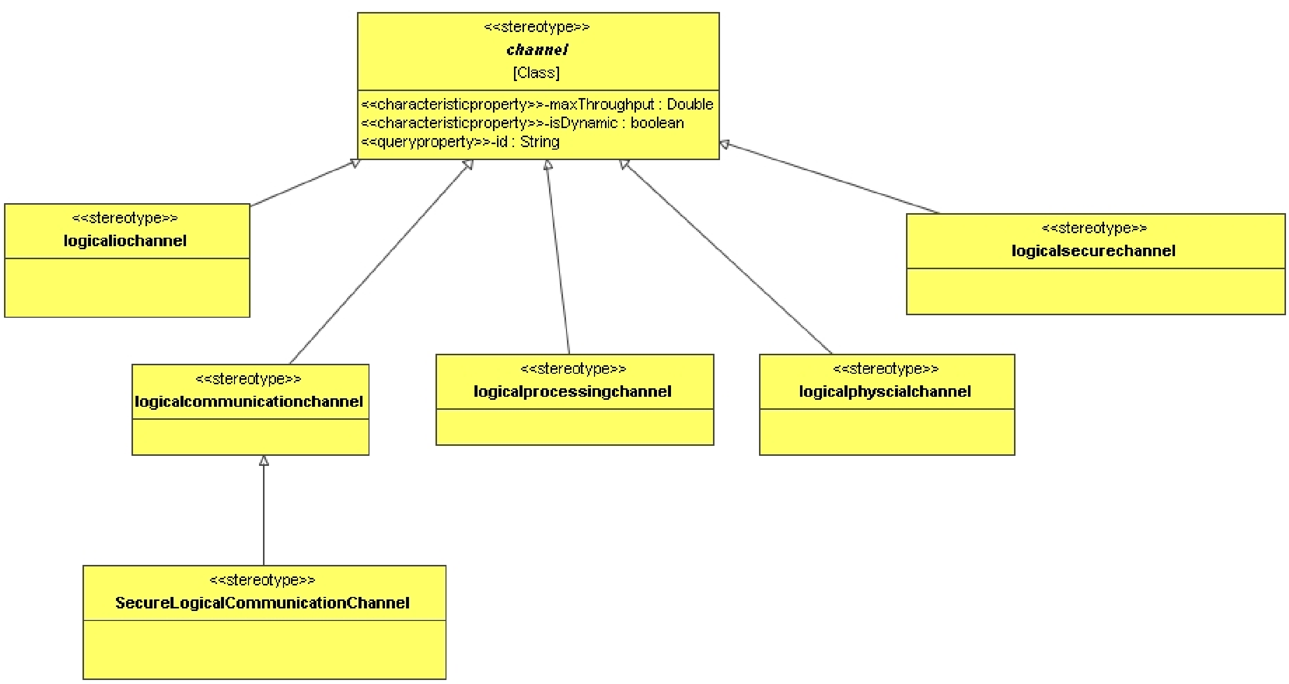

Figure 7.3 - Communication Channel Types Overview

Description

Channel provides an abstract class definition by extending the UML Class definition. This abstract class definition is specialized by all of the stereotype definitions in the Communication Channel section.

Attributes

- `<<queryproperty>>id: String`
  The id attributes represents the identification of the channel.
- `<<characteristicproperty>>isDynamic: Boolean`
  Specifies whether the channel is a dynamic channel or not. A Dynamic channel is one whose definition can be changed in run-time by the application.
- `<<characteristicproperty>>maxThroughput: Double`
  Data throughput of the channel.

M1 Associations

- `channel: Channel [*]`
  A channel can have associations to any number of channels.

7.2.2 LogicalCommunicationChannel

Description

LogicalCommunicationChannel stereotype is a specialization of the abstract Channel and is a data descriptor for different types sub-channels. It is an aggregate of LogicalProcessingChannel, LogicalIOChannel, and LogicalPhysical channel as shown in Figure 7.4.

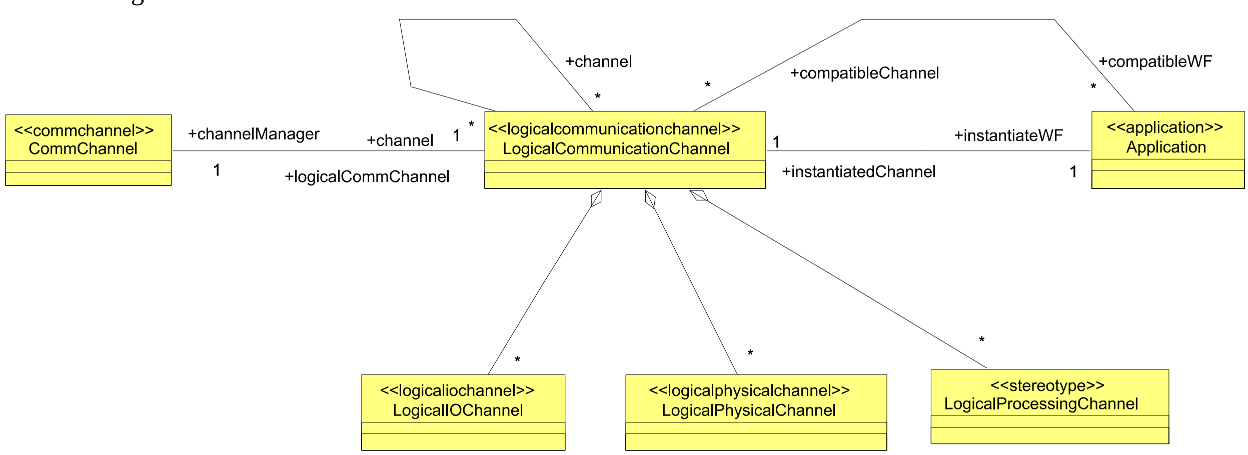

Figure 7.4 - LogicalCommunicationChannel M1 Illustration

M1 Associations

- `compatibleWF: Application [*]`
  A Logical Communication Channel may have all the capabilities required by a WaveformApplication.

- `instantiatedWF: Application [1]`
  An instantiated application runs on its associated CommunicationChannel.

- `commManager: CommChannel [1]`
  A LogicalCommunicationChannel is managed by a CommChannel.

Constraints

A LogicalCommunicationChannel requires at least one LogicalPhysicalChannel or a LogicalIOChannel and combination of any other channel type (LogicalPhysicalChannel, LogicalIOChannel, and LogicalProcessingChannel).

The model allows for realizations that do not require security nor any processing (i.e., a non-software defined radio). Further, a valid channel may be an RF relay, with no local I/O, or may include a router and require no RF capability.

7.2.3 LogicalIOChannel

Description

The LogicalIOChannel stereotype extends the abstract Channel and provides for the baseband connection to the radio and consists of the devices that format, encode, decode, etc. the communication signals at that interface. Figure 7.5 shows the description of the Logical I/O Channel.

M1 Associations

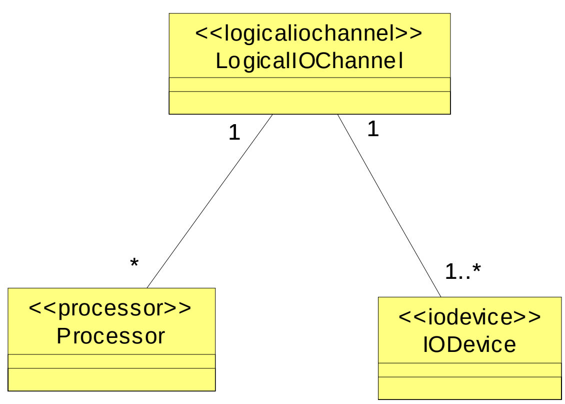

Figure 7.5 - LogicalIOChannel M1 Illustration

- `processor: Processor [*]`
  A LogicalIOChannel may be associated with zero or more processors.

- `ioDevice: IODevice [1..*]`
  Each LogicalIOChannel has only one IODevice.

Constraints

A LogicalIOChannel shall be associated with at least one IODevice.

Semantics

The LogicalIOChannel can include an IO algorithm which can be distinguished by the codec type and data conversion type. An IODevice and Processor can be associated with the algorithm that is a part of the channel. Processor acts as a data processor and an algorithm loader while the IODevice acts as the data processor that employs the IO algorithm to process data.

7.2.4 LogicalPhysicalChannel

Description

The LogicalPhysicalChannel stereotype extends the abstract Channel by consisting of all devices processing the analog signal after digitization, to and including the antenna(s). For convenience, A/D and D/A conversion devices, if used, are included here. The current state of the art is such that most of the operations of the interfaces realized by these devices are performed via hardware elements as opposed to software; nonetheless, the model does not force either implementation. Figure 7.6 shows the LogicalPhysicalChannel definition with attributes of its aggregated components.

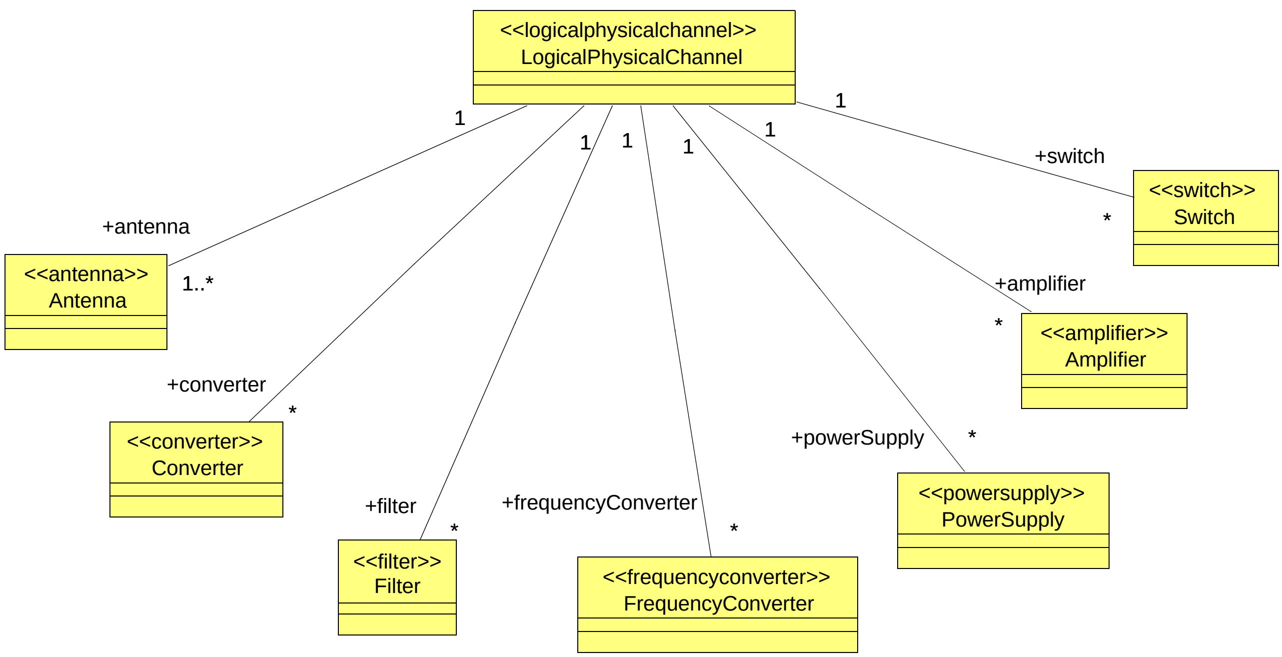

Figure 7.6 - LogicalPhysicalChannel M1 Illustration

M1 Associations

- `anntenna: Antenna [1..*]`
  LogicalPhysicalChannel can be associated with an Antenna.
- `converter: Converter [*]`
  LogicalPhysicalChannel can be associated with a Converter.
- `filter: Filter [*]`
  LogicalPhysicalChannel can be associated with a Filter.

- `amplifier: Amplifier [*]`
  LogicalPhysicalChannel can be associated with an Amplifier.

- `frequencyConverter: FrequencyConverter [*]`
  LogicalPhysicalChannel can be associated with a FrequencyConverter.

- `powerSupply: PowerSupply [*]`
  LogicalPhysicalChannel can be associated with a PowerSupply.

- `switch: Switch [*]`
  LogicalPhysicalChannel can be associated with a Switch.

Constraints

A LogicalPhysicalChannel shall be associated with at least one Antenna element.

7.2.5 LogicalProcessingChannel

Description

The LogicalProcessingChannel stereotype extends the abstract Channel and provides the processing nodes for applications and radio services used by the waveforms running on the processing channel's operating environment(s). An exception to this is the processing node(s) specific to supporting security functions, which are part of the LogicalSecurityChannel. Figure 7.7 shows the LogicalProcessingChannel definition.

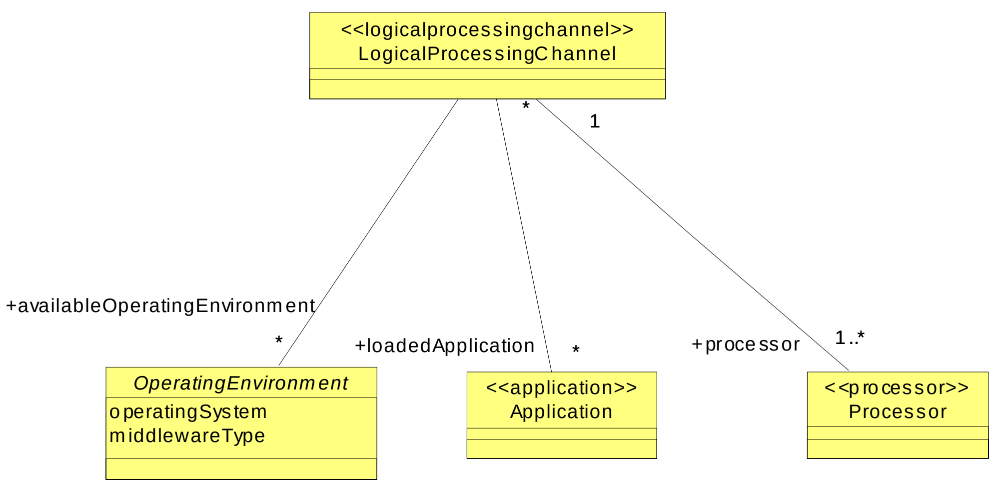

Figure 7.7 - LogicalProcessingChannel M1 Illustration

M1 Associations

- `availableOperatingEnvironment: OperatingEnvironment [*]`
  A LogicalProcessingChannel uses the interfaces provided by the operating environment.

- `loadedApplication: Application [*]`
  A LogicalProcessingChannel can run multiple Applications on it.

- `processor: Processor [1..*]`
  A LogicalProcessingChannel contains at least one processor to perform computations on.

Constraints

A LogicalProcessingChannel shall be associated with at least one processor.

7.2.6 LogicalSecurityChannel

Description

The LogicalSecurityChannel stereotype extends the abstract Channel and provides the processing node(s) for security applications applicable to communications. The LogicalSecurityChannel is present in a logical channel definition only if the channel has security requirements. This channel may be used for separating between secure and insecure sides of the communication (Red - Black separation). The LogicalSecurityChannel definition is shown in Figure 7.8.

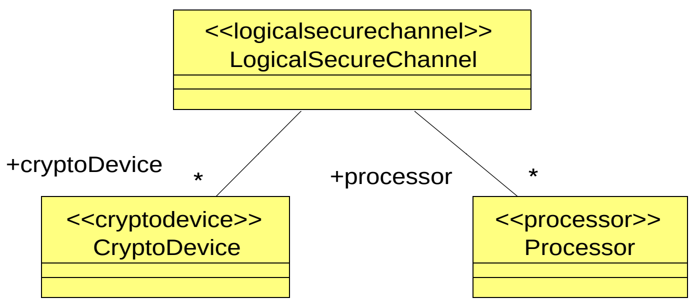

Figure 7.8 - LogicalSecurityChannel M1 Illustration

M1 Associations

- `cryptoDevice: CryptoDevice [1..*]`
  A LogicalSecurityChannel shall be associated with at least one CryptoDevice.
- `processor: Processor [*]`
  A LogicalSecurityChannel may be associated with any number of processors.
- `loadedAlgorithm: SecurityAlgorithm [1..*]`
  A LogicalSecurityChannel shall be associated with at least one SecurityAlgorithm. A security channel may support loading multiple algorithms at the same time.
- `loadedKey: SecurityKey [1..*]`
  A LogicalSecurityChannel shall be associated with at least one SecurityKey.
- `loadedPolicy: SecurityPolicy [*]`
  A LogicalSecurityChannel may be associated with any number of security policies that direct its actions.

Constraints

LogicalSecurity Channel has either a crypto device or a processor that runs a security algorithm; it may have a processor(s) for other functions.

The LogicalSecurityChannel runs security algorithms on either a Processor or a dedicated Crypto Device.

Semantics

A LogicalSecurityChannel uses the Crypto and the Processor to provide security features of a waveform. A LogicalSecurityChannel may provide a security algorithm, security keys, and a security policy in order to facilitate those features.

7.2.7 SecureLogicalCommunicationChannel

Description

The SecureLogicalCommunicationChannel stereotype is an extension of LogicalCommunicationChannel stereotype and adds another aggregation relationship to the LogicalSecurityChannel stereotype as shown in Figure 7.6. It includes the relationships inherited from the LogicalCommunicationChannel, this stereotype provides a data descriptor definition that can be made of four different types of sub-channels: LogicalProcessingChannel, LogicalIOChannel, LogicalPhysicalChannel, and LogicalSecurityChannel.

Constraints

A SecureLogicalCommunicationChannel shall have at least one LogicalSecureChannel associated with it.

7.2.8 RadioSet

Description

A RadioSet, an extension of UML Class, defines a radio set and its associated channels.

M1 Associations

- `channel: Channel [*]`
  A RadioSet can have associations to any number of Channel(s).

7.2.9 RadioSystem

Description

A RadioSystem, an extension of UML Class, defines a radio system and its associated RadioSets.

M1 Associations

- `radioSet: RadioSet [*]`
  A RadioSystem can have associations to any number of RadioSet(s).

7.3 Radio Management

This section defines the stereotypes for radio management. Radio management involves the management of the radio, inclusive of its devices and services. The radio management stereotypes depicted in Table 7.2 are extensions of the UML 2.0 Component (UML2.0::Components::BasicComponents). Figure 7.9 depicts the relationships of the radio management components to each other and the elements that they manage.

Table 7.2 - Radio Management Stereotypes

Stereotype	Base Class	Parent	Tags	Constraints	Description
CommChannelComponent	Component	Component			Represents a component that manages a LogicalCommunicationChannel.
RadioManagerComponent	Component	DomainManager Component			Represents a component that manages a RadioSet.
RadioSystemManager	Component	N/A			Represents a component that manages RadioSystems.

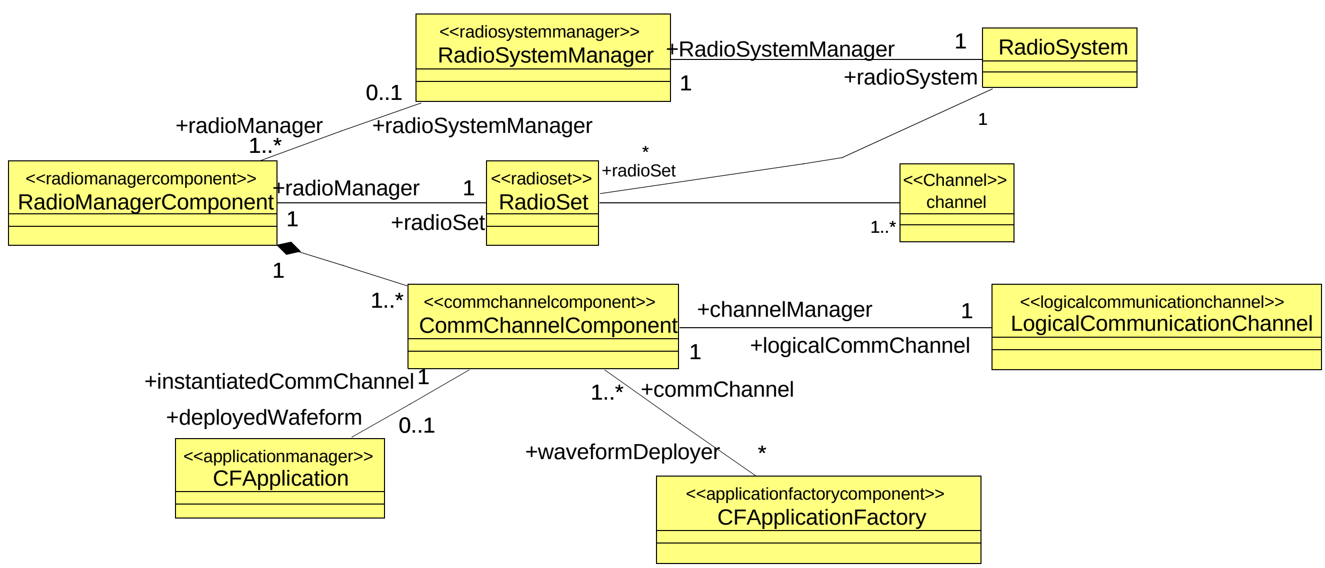

Figure 7.9 - Radio Management M1 Illustration

7.3.1 CommChannelComponent

Description

The CommChannelComponent, as shown in an association in Figure 7.9, represents a component that provides communication channel management.

M1 Associations

- `deployedWaveform: ApplicationManager [0..1]`
  The deployedWaveform represents the waveform deployed on the communication channel.
- `logicalCommunicationChannel: LogicalCommunicationChannel [1]`
  The LogicalCommunicationChannel represents the set of devices that provide the communication path for the communication channel.

- `waveformDeployer:ApplicationFactoryComponent[*]`
  The ApplicationFactory that deploys the waveform onto the communication channel.

Semantics

The CommChannelComponent may be associated with static or a dynamic LogicalCommunicationChannel. The devices associated with a static LogicalCommunicationChannel do not vary over the life cycle of the communication channel. For dynamic LogicalCommunicationChannel the devices can vary during the life cycle of the communication channel.

7.3.2 RadioManagerComponent

Description

The RadioManagerComponent, as shown in Figure 7.9, describes the definition and relationships that are common for RadioSet manager. The RadioManager extends the DomainManagerComponent by providing communication channel management within the RadioSet.

M1 Associations

- `radioSet: RadioSet [1]`
  A RadioManagerComponent is associated with one RadioSet.

7.3.3 RadioSystemManager

Description

The RadioSystemManager component, as shown in Figure 7.9, describes the definition and relationships that are common for RadioSystem managers. The RadioSystemManager is responsible for control and management tasks for the RadioSystem. It may be associated with one or more RadioManagers, which are used to control the RadioSets the RadioSystem consists of.

M1 Associations

- `radioManager: RadioManagerComponent[1..*]`
  The associated RadioManager provides the capability to manage a RadioSet within a RadioSystem.

- `radioSystem: RadioSytem [1]`
  The RadioSystem provides a set of a RadioSets.

8 Communication Channel Facilities PIM

The PIM specified in this section is a non-normative specification of the physical layer and radio control facilities. The model referenced in Section 3.1.3.1 is the normative definition. It may be realized using many technologies. The CORBA reference PSM in Chapter 9 is one such realization.

The Communication Channel Facilities PIM is made of:

- Physical Layer Facilities – The set of interfaces that define the functionality to convert the digitized signal into a propagating RF wave, and conversely, to convert a propagating RF wave into a digitized signal for processing. The facilities also include frequency tuning, filters, interface cancellation, analog digital conversion, up/down conversion, gain control, synthesizer, etc. functionality. Physical layer facilities also include functionality for baseband I/O such as serial and audio devices.
- Radio Control Facilities – The set of interfaces that define the functionality to manage the radio domain and channels within the radio.

8.1 Physical Layer Facilities

According to Open System Interconnection (OSI) model, the purpose of the physical layer is "…provides the mechanical, electrical, functional, and procedural means to activate, maintain, and de-activate physical-connections for bit transmission between data-link-entities.'' It is the stated goal of the physical layer facilities to provide the necessary interfaces required to implement the functionality specified by the OSI physical layer. Due to the proposed facilities partitioning, interfaces in the Common Layer Facilities and in the Common Radio Facilities may be required to achieve this objective. Depending on waveform complexity, interfaces defined in this package may have to be combined with higher layer facilities such as the Data Link Layer Facilities. Various types of components, such as resources or devices, can implement an interface. As with all other interfaces, this specification does not restrict a particular implementation. Finally although the Physical Layer Facilities were designed with the OSI model as framework, it does not impose such a layering on any waveform implementation.

The approach supporting this specification separates the interfaces required to implement the physical layer into two sets of facilities. The data flow facilities, which are common to many layers, and the setup and control facilities that are specific to the physical layer.

Like in the communication equipment package, the Physical Layer Facilities provide the required interfaces to interact with both the subscriber-side and RF-side of the radio. Protocol specific combination, like Ethernet, USB, RS232, GSM, CDMA2000, Bluetooth, etc. can be built using the facilities presented here in conjunction with the other higher layer services.

8.1.1 Data Transfer

The data transfer services required by the physical layer are provided via the Common and Data Link Layer Facilities::Common Layer Facilities. A physical layer component must realize data transfer interfaces to communicate with the upper OSI layers.

8.1.2 Control

The Physical Layer Facilities package contains interfaces used to configure and control components performing the physical layer functions of a waveform. Interfaces are defined using high-level concepts. This degree of abstraction enables waveform developers to create waveform applications while abstracting away many of the low-level details of the supporting platform. This increases the ease and speed at which waveform applications can be developed.

Two functionally separate facilities are part of the Physical Layer Facilities. The first set of facilities is responsible for modem operation. The modem includes all signal processing components involved in the translation of bits into symbols and vice-versa. In this context, bits are composed of data and any and all overhead information. Symbols are defined as the points of any n dimensional constellation. This definition applies on both the subscriber-side and the RF-side. Common subscriber-side modulations include Manchester, Non-Return to Zero (NRZ), Non-Return to Zero Inverted (NRZI), Return-to Zero (RZ), etc. Common RF-side modulations include Amplitude Modulation (AM), Frequency Modulation (FM), Quadriture Amplitude Modulation (QAM), Phase Shift Keying (PSK), Continuous Phase Modulation (CPM), etc. Channel coding also equally applies to both sides.

The second set of facilities is used to control the basic devices of the channel adaptation chain. This is called the Radio Frequency/Intermediate Frequency (RF/IF) chain. The chain's purpose is to adapt the symbol stream to the transmission channel by adjusting the frequency response, power, and centre frequency of the signal. In subscriber-side applications this may translate in adjusting the pitch, doing echo cancellation, and setting the volume. On the subscriber-side the center frequency is Direct Current (DC) or 'close' to DC. On the RF-side this can be pulse shaping, equalization, and power control. On the RF-side the center frequency is normally not DC.

8.1.2.1 Modem Facilities

The modem facilities include all digital signal processing elements required to convert bits into symbols and vice versa. None of these elements perform pulse shaping or any filtering required to meet the mask. In addition, they do not perform equalization or any other form of channel estimation. These functions are viewed as part of the RF/IF facilities described in Section 8.1.2.2.

The modem is not concerned with the functionalities of the Medium Access Control (MAC) layer. It will, in some cases (CDMA), provide services (i.e., PN sequence generator), which can be used by the various MACs to achieve their objectives.

The modem should have the facilities to implement all of today's baseband and passband digital modulation schemes: RZ, NRZ, Manchester, Direct sequence spread spectrum, QAM, PSK, Frequency Shift Keying (FSK), Amplitude Shift Keying (ASK), CPM, Gaussian Minimum Shift Keying (GMSK), Orthogonal Frequency Division Multiplex (OFDM), and Multiple-Input/Multiple-Output (MIMO) as well as digitally represented analog modulation schemes: Amplitude Modulation (AM), Frequency Modulation (FM), and Phase Modulation (PM). The preceding list is not exhaustive and is not intended to limit the scope of the modem functionality.

The modem facilities include more than just simple modulation; they also provide support for Forward Error Correction (FEC), differential encoding, interleaving, direct sequence spreading, scrambling, and Fourier Transforms. In the implementation of a particular modulation scheme, some or even all of these interfaces may be required. Furthermore, the order is flexible. For example, Trellis Coded Modulation (TCM) is viewed as a special arrangement of modulation and FEC.

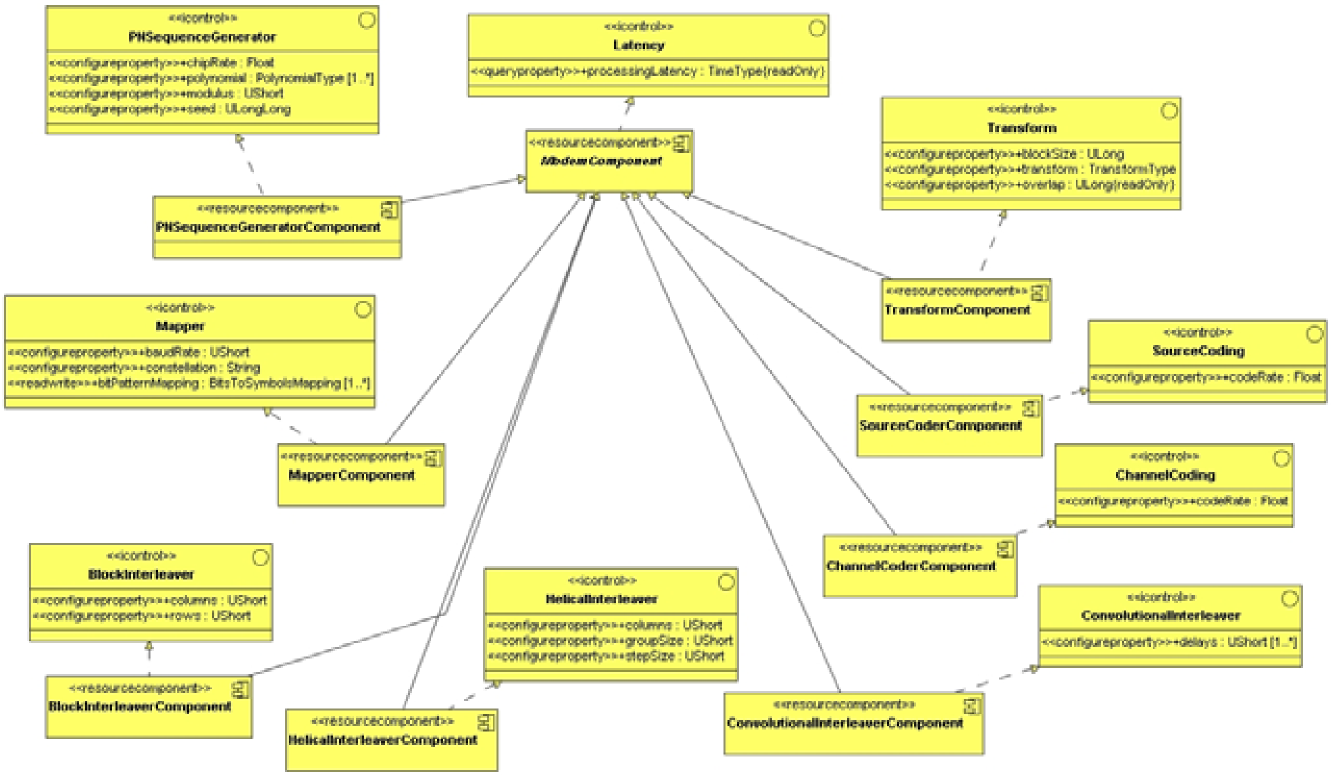

Figure 8.1 - Modem Facilities Overview

8.1.2.1.1 ModemComponent

Description

The ModemComponent component is an abstract component that realizes the Latency interface. All components in the modem facilities inherit from this component. Components are stereotyped as <<resourcecomponent>> to indicate that they could either be implemented purely in software or in hardware via a <<devicecomponent>> component.

Constraints

ModemComponent shall provide one ControlPort and at least one DataControlPort or DataPort.

8.1.2.1.2 Latency

Description

The Latency interface is used for specifying the processing latency of all digital signal processing components.

Attributes

- `<<queryproperty>>processingLatency: TimeType`
  The processingLatency attribute represents the time it takes for an input data element to be carried out to the output of the component.

8.1.2.1.3 BlockInterleaver

Description

This interface is used to control a block interleaver / deinterleaver. An interleaver permutes the incoming bit stream. It does not change the bit rate. Interleavers can be found after any component of the modulation chain. This can be at the input, after forward error correction, spreading, mapping, and even after having applied a transformation.

Attributes

- `<<configureproperty>>columns: UShort`
  The columns attribute is the number of columns of the block interleaver.
- `<<configureproperty>>rows: UShort`
  The rows attribute is the number of rows of the block interleaver.

8.1.2.1.4 ConvolutionalInterleaver

Description

This interface is used to control a convolutional interleaver / deinterleaver. An interleaver permutes the incoming bit stream. It does not change the bit rate. Interleavers can be found after any component of the modulation chain. This can be at the input, after forward error correction, spreading, mapping, and even after having applied a transformation.

Attributes

- `<<configureproperty>>delays: UShort [1..*]`
  The delays attribute is the delays applied by the convolutional interleaver to the bit stream.

8.1.2.1.5 HelicalInterleaver

Description

This interface is used to control a helical interleaver / deinterleaver. An interleaver permutes the incoming bit stream. It does not change the bit rate. Interleavers can be found after any component of the modulation chain. This can be at the input, after forward error correction, spreading, mapping, and even after having applied a transformation.

Attributes

- `<<configureproperty>>columns: UShort`
  The columns attribute is the number of columns of the helical interleaver.
- `<<configureproperty>>groupSize: UShort`
  The groupSize attribute is the size of each group of input symbols.
- `<<configureproperty>>stepSize: UShort`
  The stepSize attribute is the number of rows between consecutive input groups in their respective columns.

8.1.2.1.6 Mapper

Description

This interface is used to control a mapper. The mapper executes the transformation from bits, coded or not, to symbols. This transformation can be described completely by a mathematical expression relating the bit input pattern to the corresponding output symbol. It is important to note the units representing the location of the output symbols need not always be amplitudes (e.g., (X, Y) coordinates). For example, an FSK mapper would have frequencies as output.

The output of the mapper is at the baud rate.

Attributes

- `<<configureproperty>>baudRate: UShort`
  The baudRate attribute represents the current baud rate.
- `<<configureproperty>>constellation: String`
  The constellation attribute is the constellation type used by the mapper.
- `<<readwrite>>bitPatternMapping: BitsToSymbolsMapping [1..*]`
  The bitPatternMapping attribute represents the actual definition of the constellation. Each input bit pattern is mapped to one or more dimensional quantities.

Types and Exceptions

- `BitsToSymbolsMapping ( bitPattern: ULong,dimensions: UShort [1..*])`
  bitPattern: The actual bit pattern as a UShort.
  dimensions: The quantity to which the bit pattern is mapped.

8.1.2.1.7 PNSequenceGenerator

Description

This interface is used to control a Pseudo Noise (PN) Sequence Generator. PN sequences are commonly used in scramblers, spreaders, and data sources. The output rate of the PN sequence will be called chip rate. Note that when used as a scrambler, the chip rate matches the data rate + overhead rate, when used as a data source, then it is simply the data rate.

The interface assumes that the PN sequence generated multiplies another incoming ‘data’ stream to produce the output ‘data’ stream.

There are many techniques used to generate PN sequences. Much like interleavers, each technique has its own mathematical description method. The generic formula for describing a random sequence generator is:

$$X_n = (a_1 X_{n-1}^{j1} + a_2 X_{n-2}^{j2} \ldots + a_k X_{n-k}^{jk}) \bmod m$$

Figure 8.2 - Sequence number generator formula

Attributes

- `<<configureproperty>>chipRate: Float`
  The chipRate attribute represents the rate of encoding of the spreader. In other words, the chip rate is the rate at which the information bits are transmitted as a pseudo-random sequence of chips.
- `<<configureproperty>>polynomial: PolynomialType [1..*]`
  The polynomial attribute is the polynomial used to generate the pseudo-random sequence.
- `<<configureproperty>>modulus: UShort`
  The modulus attribute represents the value by which the polynomial is divided (i.e., m in Figure 8.2).
- `<<configureproperty>>seed: ULongLong`
  The seed attribute is the first value (X0) used to calculate the remaining pseudo-random sequence.

Types and Exceptions

- `PolynomialType ( multiplier: ULongLong, exponent: ULongLong )`
  (refer to Figure 7.28 - - Sequence number generator formula)
  multiplier: a
  exponent: j

8.1.2.1.8 Transform

Description

This interface is used to control the transform. The transformations included at this point are Fast Fourier Transform (FFT) and Inverse Fourier TRansform (IFFT). These transformations are commonly used for the generation and reception of OFDM and Coded OFDM (COFDM) waveforms as well as for frequency domain filtering.

Attributes

- `<<configureproperty>>blockSize: ULong`
  The blockSize attribute is the block size used by the transform.
- `<<configureproperty>>transform: TransformType`
  The transform type attribute indicates which type of transform is performed by the implementation.
- `<<configureproperty>>overlap: ULong`
  The overlap attribute is the amount of overlap of the transform in number of points.

Types and Exceptions

- `<<enumerationproperty>>TransformType (FFT: UShort = 1, IFFT: UShort = 2)`
  FFT: Fast Fourier Transform
  IFFT: Inverse Fast Fourier Transform

8.1.2.1.9 ChannelCoding

Description

This interface is used to represent a channel coder or decoder. A coder applies some transformation on the incoming data. Common examples of coders are differential encoders, and Forward Error Correction (FEC) encoders. Decoders reverse the transformation. The output of the coder is a coded sequence. In differential encoders the output rate is usually the same as in input rate where as in FEC coders, the output rate is higher than the input rate. The code rate is the ratio of the input rate over the output rate.

Channel coders and decoders have very different structures and mathematical formulas that describe them. Due to these differences, the interface provided here is not sufficient for a waveform developer. The goal of the interface is to provide system simulator with the minimum number of parameters to be able to model the communication path.

Attributes

- `<<configureproperty>>codeRate: Float`
  The code rate attribute, R, represents the ratio of the input rate, N, over the output rate K. R = N / K.

8.1.2.1.10 SourceCoding

Description

This interface is used to control a source coder or decoder. Source coding essentially represents the compression of input data for better efficiency during transmission.

Source coders and decoders have very different structures and mathematical formulas that describe them. Due to these differences, the interface provided here is not sufficient for a waveform developer. The goal of the interface is to provide system simulator with the minimum number of parameters to be able to model the communication path.

Attributes

- `<<configureproperty>>codeRate: Float`
  The code rate attribute, R, represents the ratio of the input rate, N, over the output rate K. R = N / K.

8.1.2.2 RF/IF Facilities

The RF/IF Facility is used to configure and control the basic devices of the communication channel. The granularity at which these interfaces are implemented is not specified. For example, at the highest granularity level, the FrequencyResponse interface can be implemented by a single component for the whole communication channel. The underlying API implementation could then break-up the frequency parameter into smaller frequency responses for configuring individual devices. The waveform application is unaware of the individual devices that make up the communication channel. The interaction point between the waveform and the platform is via this single interface. On the other hand, at the lowest granularity, each device that makes up the communication channel could implement the interface. In this case, the waveform could elect to configure each device with the correct frequency response. These design choices are left to the implementer. The same scenario could be applied to all interfaces defined in this package.

The components of the RF/IF Facilities maps to the concepts defined in the CommEquipment package. Components stereotyped as <<resourcecomponent>> indicates that they are either implemented in software or via hardware devices. Components stereotyped as <<devicecomponent>> indicates that they are implemented via hardware devices.

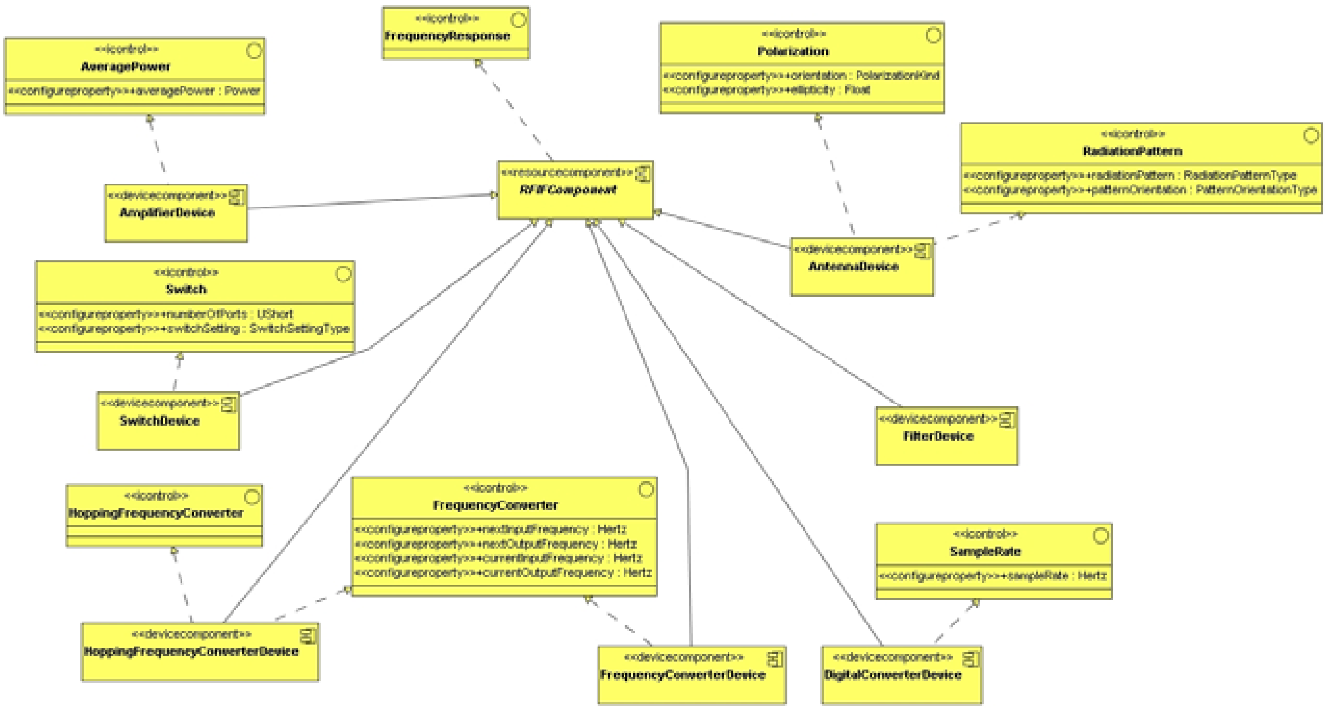

Figure 8.3 - RF/IF Facilities Overview

8.1.2.2.1 RFIFComponent

Description

The RFIFComponent component is an abstract component that realizes the FrequencyResponse interface. All components in the RF/IF Facilities inherit from this component.

Constraints

RFIFComponent shall provide one ControlPort and at least one DataControlPort or DataPort.

8.1.2.2.2 FrequencyResponse

Description

This interface is used to configure the frequency response of a specific component. There are multiple ways of specifying the frequency response. For example, a 1-point frequency response could indicate the 3 dB cut-off of a symmetric spectrum. A 2-point frequency response could be the upper and lower 3 dB cut-off locations for a nonsymmetric spectrum. The number of points, the location of the points, and the attenuation and / or phase vary from filter to filter. In some cases, it is the pass band of the filter that is critical while in others it is the stop band. It is left to the designer to specify the key points of each filter with the degree of precision required.

Examples of components whose frequency response could be set with this API are pulse shaping filters and equalizers.

Attributes

- `<<configureproperty>>freqResponse: FrequencyResponseType`
  The frequencyResponse attribute is the frequency response of the device. The frequency response specified is centered at 0 Hz.

- `<<configureproperty>>tunedFrequency: Hertz`
  The tunedFrequency attribute is the frequency at which the frequency response is centered.

8.1.2.2.3 RadiationPattern

Description

This interface is used to configure and/or control the radiation pattern of an antenna. The radiation pattern of an antenna is usually represented by the azimuth plane and elevation plane plots. The radiation pattern is represented with respect to the True North (0 degree) and 0 degree elevation. The orientation of the antenna is also represented with those same measurements.

Attributes

- `<<readwrite>>radiation_Pattern: RadiationPatternType`
  The radiationPattern attribute represents the radiation pattern of the device.

- `<<configureproperty>>patternOrientation: PatternOrientationType`
  The patternOrientation attribute is the actual pattern orientation, which is represented by an azimuth angle and an elevation angle. The antenna can be moved without having to change the radiation pattern.

8.1.2.2.4 Polarization

Description

This interface is used to configure and / or control the polarization parameters of an antenna.

Attributes

- `<<configureproperty>>orientation: PolarizationKind`
  The orientation attribute represents the polarization of the antenna.

- `<<configureproperty>>ellipticity: Float`
  The ellipticity attribute is the ratio between the minor and major axis of the ellipse. In the case of right hand or left hand circular. If the ellipticity is 1, this means that the polarization is a perfect circle.

8.1.2.2.5 FrequencyConverter

Description

This interface is used to configure and / or control a frequency converter. The frequency converter can either be an up converter or a down converter.

Attributes

- `<<configureproperty>>nextInputFrequency: Hertz`
  The nextInputFrequency attribute is the input frequency that the device will select after the next triggering event. This

attribute is used for instantaneous frequency changes. Typically in the context of frequency hoping and frequency scanning algorithms.

- `<<configureproperty>>nextOutputFrequency: Hertz`
  The nextOutputFrequency attribute is the output frequency that the device will select after the next triggering event. This attribute is used for instantaneous frequency changes. Typically in the context of frequency hoping and frequency scanning algorithms.
- `<<configureproperty>>currentInputFrequency: Hertz`
  The currentInputFrequency attribute is the frequency of the signal currently at the input of the device.
- `<<configureproperty>>currentOutputFrequency: Hertz`
  The currentOutputFrequency attribute is the frequency of the signal currently at the output of the device.

8.1.2.2.6 SampleRate

Description

This interface is used to configure and /or control the sample rate of a specific device. Typically, the device is either an analog to digital converter (ADC) or a digital to analog converter (DAC).

Attributes

- `<<configureproperty>>sampleRate: Hertz`
  The sampleRate attribute represents the number of samples the device takes per second.

8.1.2.2.7 AveragePower

Description

This interface is used to configure and / or control the power of a specific device. Typically, the device will be either the power amplifier or a variable gain amplifier used as part of an Automatic Gain Control (AGC) loop. Note that it is assumed that all other devices are average power neutral (i.e., they have a gain of 0 dB).

Attributes

- `<<configureproperty>>averagePower: Power`
  The averagePower attribute represents the average power of the device.

8.1.3 IO Facilities

Inside a radioset, the IO subsystem has mission to establish bidirectional connections, termed IO channels, between a waveform stack and a physical radioset wired line. Those wired IOs may serve several purposes such as:

- connecting the radioset with a human operator through a microphone and a headset,
- linking a radioset with a Local Area Network (LAN) to provide a bridge between LAN stations and mobile equipment,
- connecting peripheral sensor devices to the radioset,
- clustering radiosets together to offer scalable and/or fault-tolerant capabilities,
- providing a means to upload/download software from/to the radioset.

Currently, waveforms stacks are plugged to dedicated serial lines using single or half-duplex protocols and each additional IO physical channel require an additional physical slot (one-to-one relationship). Now, wired radiosets can be connected to multiplexed serial lines and/or buses (Ethernet, USB) and a single physical IO slot may virtually support an unlimited number of virtual channels (one-to-many relationship).

This package defines two types of IO mechanisms: Serial IO and Audio IO.

8.1.3.1 Serial IO Package

The Serial IO services are realized by a SerialIODeviceComponent that provides and uses the following set of interfaces:

- SerialIOSignals
- SerialIODevice
- SerialIOControl

Those interfaces are summarized in Figure 8.4:

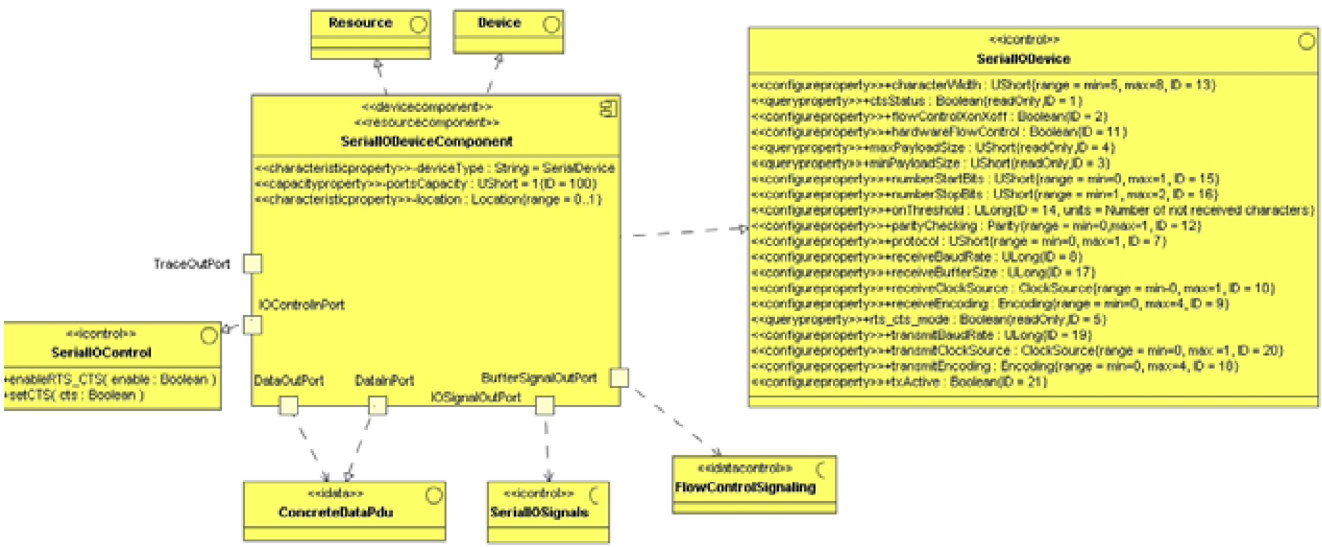

Figure 8.4 - Serial IO Framework

8.1.3.1.1 Serial IO Control Interfaces

8.1.3.1.1.1 SerialIOControl

This interface is used for in-band control of Serial IO.

Description

The SerialIOControl interface is used to control flow on customer’s side.

Operations

- `enableRTS_CTS (in enable : Boolean)`
  Enable Clear To Send (CTS) and Request To Send (RTS).

- `setCTS (in cts : Boolean)`
  Force CTS.

8.1.3.1.1.2 SerialIOSignals

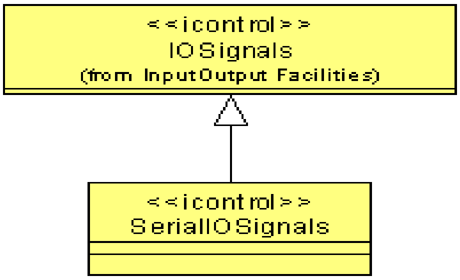

Figure 8.5 - Serial IO Signal

Description

This interface is used by IO device to signal clients when RTS signal is up.

8.1.3.1.1.3 SerialIODevice

Description

The Serial IODevice is the control interface used for out-band control of serial lines.

Attributes

- `<<configureproperty>> characterWidth : UShort`
  (Asynchronous protocol only) Number of bits in character (5, 6, 7, or 8).
- `<<queryproperty>> ctsStatus : Boolean`
  Indicates the CTS status.
- `<<configureproperty>> flowControlXonXoff : Boolean`
  Controls whether flow Control signals should be generated. True means Xon and False means Xoff.
- `<<configureproperty>> hardwareFlowControl : Boolean`
  To enable/disable use of RTS/CTS hardware signals used for flow control.
- `<<queryproperty>> maxPayloadSize : UShort`
  Maximum size of payload for the pushPDU() method in ConcreteDataPDU interface.
- `<<queryproperty>> minPayloadSize : UShort`
  Minimum size of payload for the pushPDU() method in ConcreteDataPDU interface.
- `<<configureproperty>> numberStartBits : UShort`
  (Asynchronous protocol only) Number of start bits (0 or 1).
- `<<configureproperty>> numberStopBits : UShort`
  (Asynchronous protocol only) Number of stop bits (1 or 2).
- `<<configureproperty>> onThreshold : ULong`
  Optional, used only for receive flow control. IDLE time that Serial I/O waits before data received through the serial port must be forwarded to the component connected to the DataOutPort. IDLE time in number of not received characters unit.

- `<<configureproperty>> parityChecking : Parity`
  Type of parity checking (Even = 0, Odd = 1).

- `<<configureproperty>> protocol: UShort`
  Sets asynchronous serial data protocol (Asynchronous=0 and Synchronous = 1).

- `<<configureproperty>> receiveBaudRate: ULong`
  Baud rate for Receive data

- `<<configureproperty>> receiveBufferSize : ULong`
  Size of packets to buffer before any data is written to device caller.

- `<<configureproperty>> receiveClockSource : ClockSource`
  Clock source for Receive data: internal Receive baud rate generator, external clock line, and Transmit clock source, respectively. Predefined values for coding scheme are 0=Internal Receive and 1=External clock.

- `<<configureproperty>> receiveEncoding : Encoding`
  Sets the encoding method for Transmission of serial data to NRZ, NRZI Mark, FM0, Manchester, and Differential Manchester, respectively. Predefined values for coding scheme are 0=NRZ, 1=NRZI Mark, 2=FM0, 3=Manchester, and 4=Differential Manchester, respectively.

- `<<queryproperty>> rts_cts_mode: Boolean`
  Retrieves the RTS/CTS mode.

- `<<configureproperty>> transmitBaudRate : ULong`
  Baud rate for transmit data.

- `<<configureproperty>> transmitClockSource : ClockSource`
  Clock source for Transmission of data: internal Transmit baud rate generator, external clock line, Receive clock source, and clock recovery, respectively. Predefined values for coding scheme are 0=Internal Receive and 1=External clock.

- `<<configureproperty>> transmitEncoding : Encoding`
  Sets the encoding method for Transmission of serial data to NRZ, NRZI Mark, FM0, Manchester, and Differential Manchester, respectively. Predefined values for coding scheme are 0=NRZ, 1=NRZI Mark, 2=FM0, 3=Manchester, and 4=Differential Manchester, respectively.

- `<<configureproperty>> txActive : Boolean`
  Set if on-going transmission.

Types and Exceptions

- `<<enumerationproperty>>ClockSource`
  The Clock source for receive and transmit data: internal baud rate generator, external clock line. Predefined values for coding scheme are 0=Internal Receive and 1=External clock.

- `<<enumerationproperty>>Encoding`
  The encoding method for transmission or reception of serial data to NRZ, NRZI Mark, FM0, Manchester, and Differential Manchester, respectively. Predefined values for coding scheme are 0=NRZ, 1=NRZI Mark, 2=FM0, 3=Manchester, and 4=Differential Manchester, respectively

- `<<enumerationproperty>>Parity`
  Type of parity checking (Even = 0, Odd = 1).

8.1.3.1.2 SerialIODeviceComponent

Description

The <<devicecomponent>> and <<resourcecomponent>> SerialIODeviceComponent contains the basic definition, ports and properties, for a logical serial I/O device.

Attributes

- `<<characteristicproperty>> DeviceType : String = "SerialDevice"`
  Defines the type of device.

- `<<capacityproperty>> portsCapacity : UShort = 1`
  Specifies the number of serial ports for a device.

- `<<characteristicproperty>> location : Location [0..1]`
  Defines if the device is on red (unencrypted boundary) or black side (encrypted boundary) of an encryption boundary (Black/Encrypted = 0, Red/Unencrypted = 1).

Types and Exceptions

- `<<enumerationproperty>> Location (Black_Encrypted : UShort = 0, Red_Unencrypted : UShort = 1)`
  Defines if the device is on red (unencrypted boundary) or black side (encrypted boundary) of an encryption boundary (Black/Encrypted = 0, Red/Unencrypted = 1.)

Ports

Table 8.1 - SerialIODeviceComponent Required Ports

Required Port Name	Required Interface	Connections	Purpose
DataOutPort	<<idata>> ConcreteDataPDU	SerialIODeviceComponent can only be connected to one component by this port.	This port is used by SerialIODeviceComponent to connect to a component to which data, coming from a host connected to the serial port, are forwarded.
BufferSignalOutPort	<<icontrol>> FlowControlSignaling interface (Common and Data Link Layer Facilities::Common Layer Facilities::Flow Control Facilities)	SerialIODeviceComponent can be connected to any number of components by this port.	This port is used by SerialIODeviceComponent to connect to components in order to notify serial data buffer signal events.
IOSignalOutPort	<<icontrol>>SerialIOSignals	SerialIODeviceComponent can be connected to any number of components by this port.	This port is used by SerialIODeviceComponent to connect to components in order to notify Request To Send (RTS) change.
TraceOutPort	OMG LightWeight Log Service	SerialIODeviceComponent can only be connected to one log service by this port.	This port is used by SerialIODeviceComponent to connect to log service in order to send log information.

Table 8.2 - SerialIODeviceComponent Provided Ports

Provided Port Name	Provided Interface	Purpose
DataInPort	<<idata>>ConcreteDataPDU (Common and Data Link Layer Facilities::Common Layer Facilities::PDU Facilities)	The SerialIODeviceComponent provides this port so that components can send data to this component using this port.
IOControlInPort	SerialIOControl	The SerialIODeviceComponent provides this port so that clients can control RTS/ Clear To Send (CTS) signals on the serial device.

Semantics

The SerialIODeviceComponent provides a basic standard definition of a logical serial I/O device. The <<icontrol>> SerialIODevice interface defines configuration and query properties based upon industry. The PropertySet interface (UML Profile for Component Framework::Application and Device Components::Resource Components) is used to configure and query these properties for a serial device, which can occur at initial setup of the serial I/O device or during runtime by application using the serial device.

The SerialIODeviceComponent supports a provided port named DataInPort and a required port named DataOutPort, which are both based upon the same <<idata>> ConcreteDataPDU interface (Common and Data Link Layer Facilities::Common Layer Facilities::PDU Facilities).

The SerialIODeviceComponent supports RTS/CTS management by a provided port named IOControlInPort and a required port named IOSignalOutPort.

The SerialIODeviceComponent also uses the <<icontrol>> FlowControlSignaling interface (Common and Data Link Layer Facilities::Common Layer Facilities::Flow Control Facilities) to indicate the serial data buffer state as follows:

- signalHighWatermark - The signalHighWatermark indicates that the serial I/O data buffer is full and that no more data can be processed until its state is changed to data buffer not full data buffer empty.
- signalLowWatermark - The signalLowWatermark indicates that the serial I/O data buffer is capable of receiving and processing more data.
- signalEmpty - The signalEmpty indicates that the serial I/O data buffer is empty and capable of receiving and processing more data.
- signalCongestion - The signalCongestion indicates that the serial I/O data buffer is full and data is being dropped not processed.

8.1.3.2 Audio Interfaces

The Audio IO services are realized by an AudioIODeviceComponent that provides and uses the following set of interfaces:

- AudioIOControl
- AudioIODevice

8.1.3.2.1 Audio Control Interfaces

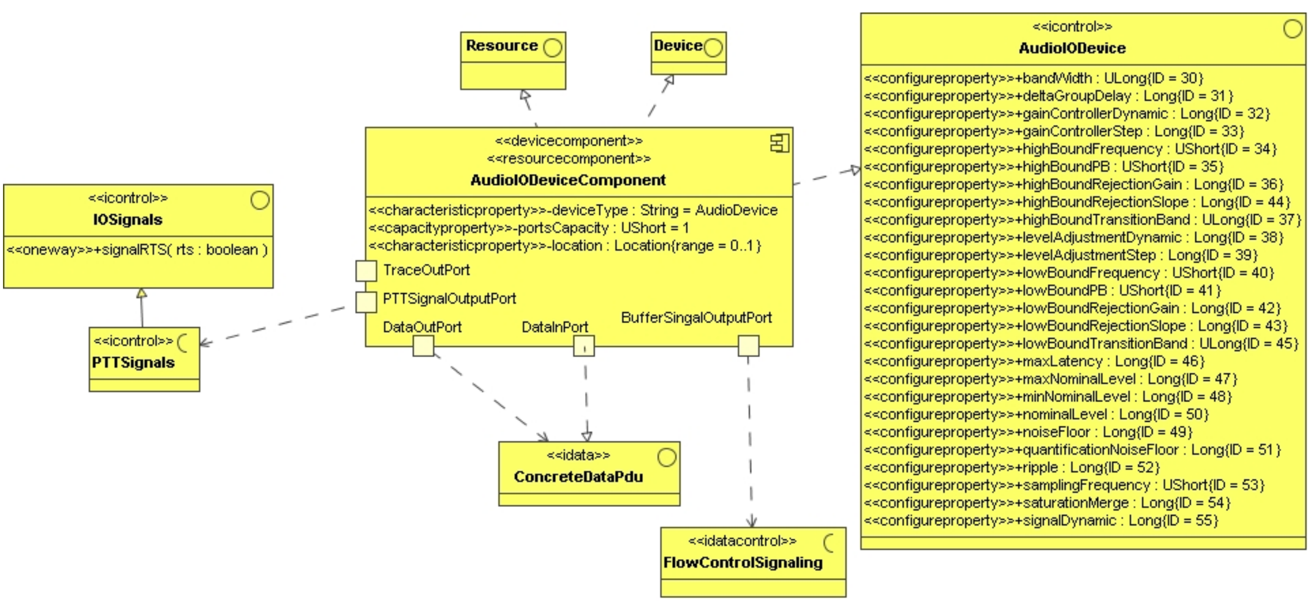

Figure 8.6 - Audio Framework

8.1.3.2.1.1 AudioIODevice

Description

The AudioIODevice is the control interface used to configure and control Acquisition and Restitution Audio device.

Attributes

- `<<configureproperty>> bandWidth : ULong`
  Width of frequency band.
- `<<configureproperty>> deltaGroupDelay : Long`
  Delta group delay.
- `<<configureproperty>> gainControllerDynamic : Long`
  Define Gain
- `<<configureproperty>> gainControllerStep : Long`
  Defines granularity of gain.
- `<<configureproperty>> highBoundFrequency : UShort`
  High bound sampling frequency in order to satisfy the Shannon sampling criterion.
- `<<configureproperty>> highBoundPB : UShort`
  Defines the high bound rejection limit in low frequencies to avoid continuous component (pass band).
- `<<configureproperty>> highBoundRejectionGain : Long`
  High bound of rejection gain.
- `<<configureproperty>> highBoundRejectionSlope : Long`
  High bound of rejection slope.
- `<<configureproperty>> highBoundTransitionBand : ULong`
  High bound of transition band.
- `<<configureproperty>> levelAdjustmentDynamic : Long`
  Capability of the gain.
- `<<configureproperty>> levelAdjustmentStep : Long`
  Granularity of the gain.
- `<<configureproperty>> lowBoundFrequency : UShort`
  Low bound sampling frequency in order to satisfy the Shannon sampling criterion.
- `<<configureproperty>> lowBoundPB : UShort`
  Defines the low bound rejection limit in low frequencies to avoid continuous component (pass band).
- `<<configureproperty>> lowBoundRejectionGain : Long`
  Low bound of rejection gain.
- `<<configureproperty>> lowBoundRejectionSlope : Long`
  Low bound of rejection slope.
- `<<configureproperty>> lowBoundTransitionBand : ULong`
  Low bound of transition band.

- `<<configureproperty>> maxLatency : Long`
  Maximum allowed latency.

- `<<configureproperty>> maxNominalLevel : Long`
  Defines maximum bound of nominal level.

- `<<configureproperty>> minNominalLevel : Long`
  Defines minimal bound of nominal level.

- `<<configureproperty>> nominalLevel : Long`
  Defines the instruction for output analog signal nominal level.

- `<<configureproperty>> noiseFloor : Long`
  Defines the level of noise (assumed white) present in audio frequency samples as inputting inside (resp. being output from) Audio. Expressed in dBFS/Hz. Possible spurious are integrated in this value.

- `<<configureproperty>> quantificationNoiseFloor : Long`
  Defines the level of quantification noise present in digital samples as inputting inside (resp. being output from) ADC. Expressed in dBFS.

- `<<configureproperty>> ripple : Long`
  Ripple.

- `<<configureproperty>> samplingFrequency : UShort`
  Defines the sampling frequency of the audio frequency signal.

- `<<configureproperty>> saturationMerge : Long`
  Avoid gain saturation (in dBfs).

- `<<configureproperty>> SignalDynamic : Long`
  Expresses the expected variations of signal magnitude around the nominal level.

8.1.3.2.1.2 PTTSignals

Description

This interface is used by audio IO device to signal clients when Pushed-To-Talk (PTT) is pushed or released.

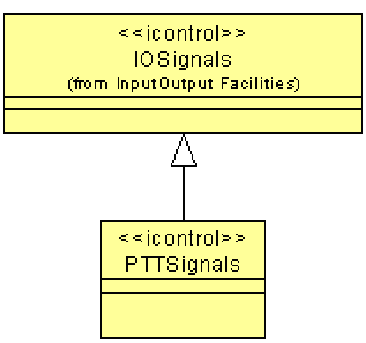

Figure 8.7 - PTTSignals

8.1.3.2.2 AudioIIODeviceComponent

Description

The <<devicecomponent>> and <<resourcecomponent>> AudioIODeviceComponent contains the basic definition, ports and properties, for a logical audio I/O device.

Attributes

- `<<characteristicproperty>> DeviceType : String = "AudioDevice"`
  Defines the type of device.
- `<<capacityproperty>> portsCapacity : UShort = 1`
  Specifies the number of audio ports for a device.
- `<<characteristicproperty>> location : Location [0..1]`
  Defines if the device is on red (unencrypted boundary) or black side (encrypted boundary) of an encryption boundary (Black/Encrypted = 0, Red/Unencrypted = 1).

Ports

Table 8.3 - AudioIODeviceComponent Required Ports

Required Port Name	Required Interface	Connections	Purpose
DataOutPort	<<idata>> ConcreteDataPDU (Common and Data Link Layer Facilities::Common Layer Facilities::PDU Facilities)	AudioIODeviceComponent can only be connected to one component by this port.	This port is used by AudioIODeviceComponent to connect to a component to which data, coming from a host connected to the audio port, are forwarded.
BufferSignalOutPort	<<icontrol>> FlowControlSignaling interface (Common and Data Link Layer Facilities::Common Layer Facilities::Flow Control Facilities)	SerialIODeviceComponent can be connected to any number of components by this port.	This port is used by AudioIODeviceComponent to connect to components in order to notify audio data buffer signal events.
PTTSignalOutPort	<<icontrol>>PTTSignals	AudioIODeviceComponent can be connected to any number of components by this port.	This port is used by AudioIODeviceComponent to connect to components in order to notify Push To Talk (PTT) change.
TraceOutPort	OMG LightWeight Log Service	AudioIODeviceComponent can only be connected to one log service by this port.	This port is used by AudioIODeviceComponent to connect to log service in order to send log information.

Table 8.4 - AudioIODeviceComponent Provided Ports

Provided Port Name	Provided Interface	Purpose
DataInPort	<<idata>>ConcreteDataPDU (Common and Data Link Layer Facilities::Common Layer Facilities::PDU Facilities)	The AudioIODeviceComponent provides this port so that components can send data to this component using this port.

Semantics

The AudioIODeviceComponent provides a basic standard definition of a logical Audio I/O device. The <<icontrol>> AudioIODevice interface defines configuration and query properties based upon industry. The PropertySet interface (UML Profile for Component Framework::Application and Device Components::Resource Components) is used to configure and query these properties for a Audio device, which can occur at initial setup of the Audio I/O device or during runtime by application using the Audio device.

The AudioIODeviceComponent supports a provided port named DataInPort and a required port named DataOutPort, which are both based upon the same <<idata>> ConcreteDataPDU interface (Common and Data Link Layer Facilities::Common Layer Facilities::PDU Facilities).

The AudioIODeviceComponent supports PTT management by a required port named PTTSignalOutPort.

The AudioIODeviceComponent also uses the <<icontrol>> FlowControlSignaling interface (Common and Data Link Layer Facilities::Common Layer Facilities::Flow Control Facilities) to indicate the Audio data buffer state as follows:

- signalHighWatermark - The signalHighWatermark indicates that the serial I/O data buffer is full and that no more data can be processed until its state is changed to data buffer not full data buffer empty.
- signalLowWatermark - The signalLowWatermark indicates that the serial I/O data buffer is capable of receiving and processing more data.
- signalEmpty - The signalEmpty indicates that the serial I/O data buffer is empty and capable of receiving and processing more data.
- signalCongestion - The signalCongestion indicates that the serial I/O data buffer is full and data is being dropped not processed.

8.1.3.3 IOSignals

Description

This interface is used by IO device to signal clients when a request to send data condition has occurred.

Operations

- `<<oneway>> signalRTS (in rts: Boolean)`
  The signalRTS operation indicates whether a request to send data condition exists. True means the condition does exist to send data. False means the condition does not exist for sending data.

8.2 Radio Set Facilities

This section defines the facilities for RadioSet channel management as depicted in Figure 8.8. The facilities defined for a RadioSet extends the component definitions as defined in the UML Profile for Communication Channel::Infrastructure::Radio Management. The types of facilities offered by RadioSet are as follows:

1. Zeroize Control - provides the mechanism for zeroizing the RadioSet's classified or secure information.
2. Transmission Control - provides the mechanism for the controlling the transmission of the RadioSet's transmission or communication channel.
3. Communication Channel Control - provides the mechanism for managing a RadioSet's communication channel (unmanaged, managed, secure, and managed secure).
4. RadioSet Control - provides the mechanism for managing a RadioSet (unmanaged, managed, secure, and managed secure).
5. Waveform Instantiation - provides the mechanism of instantiation a waveform on a channel.

8.2.1 CommChannel

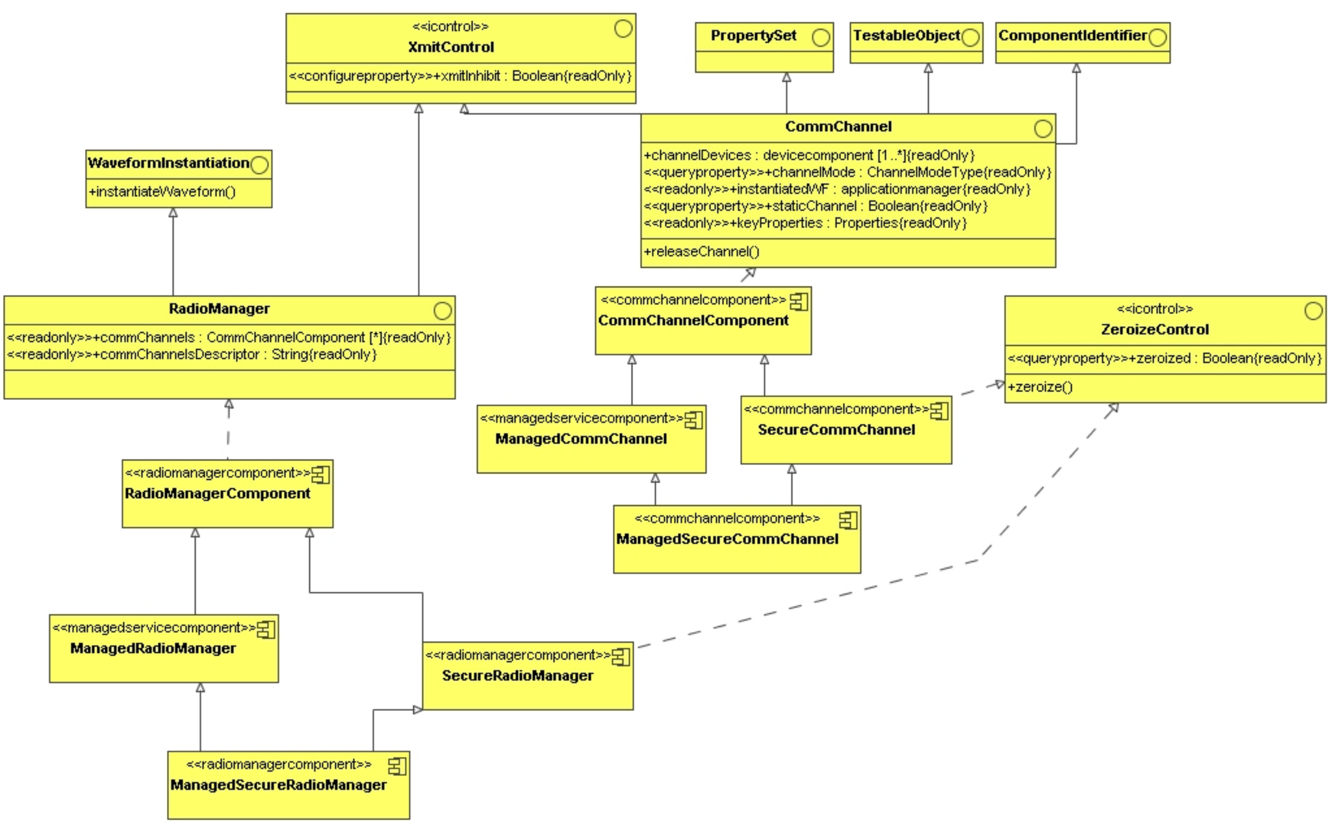

Figure 8.8 - Radio Set Facilities Overview

Description

The CommChannel interface provides additional attributes and operations for managing a Channel.

Attributes

- `<<readonly>>channelDevices : DeviceComponent [1..*]`
  The channelDevices attribute contains the DeviceComponents associated with this CommChannel. The devices could vary depending on the type of channel (static or not) and if instantiated.

- `<<queryproperty>>channelMode: ChannelModeType`
  The channelMode attribute indicates the capability of the channel. The values for channelMode are:
  1 means FULL_DUPLEX
  2 means RECEPTION_ONLY (half duplex)
  3 means XMIT_ONLY (half duplex)

- `<<readonly>>keyProperties: Properties`
  The keyProperties attribute contains information about each key associated with the channel.

- `<<readonly>>instantiatedWF: ApplicationManager`
  The instantiatedWF attribute contains the deployed waveform application associated with the instantiated channel. The instantiatedWF is a nil reference when the channel is not instantiated.

- `<<queryproperty>>staticChannel: Boolean`
  The staticChannel attribute indicates if the channel is static. A static channel means the channelDevices does not change and the communication path from baseband I/O to antenna is completely defined.

Operations

- `releaseChannel(): {raises = (releaseError)}`
  The releaseChannel operation provides the mechanism of uninstantiating the channel. The releaseChannel operation shall remove the deployed waveform as specified in the instantiatedWF attribute from the channel. The releaseChannel operation shall destroy the deployed waveform as specified in the instantiatedWF attribute. The releaseChannel operation shall raise the ReleaseError exception when the channel cannot be successfully released due to internal processing error(s).

Types and Exceptions

- `<<exception>>ReleaseError`
  The ReleaseError exception, specialization of SystemException, is raised when the releaseChannel operation is unsuccessful due to internal processing errors. The error number indicates an ErrorNumberType value (e.g., E2BIG, ENAMETOOLONG, ENFILE, ENODEV, ENOENT, ENOEXEC, ENOMEM, ENOTDIR, ENXIO, EPERM). The message is component-dependent, providing additional information describing the reason for the error.

- `<<enumerationproperty>>ChannelModeType (FULL_DUPLEX : UShort = 1, RECPETION_ONLY : USHORT = 2, XMIT_ONLY : USHORT = 3)`
  The ChannelModeType indicates the capability of the channel. The UShort values for channelMode are:
  1 means FULL_DUPLEX
  2 means RECEPTION_ONLY (half duplex)
  3 means XMIT_ONLY (half duplex).

8.2.2 CommChannelComponent

Description

The <<commchannelcomponent>> CommChannelComponent takes on additional functionality for managing a communication channel by realizing the CommChannel interface.

8.2.3 ManagedCommChannel

Description

The <<managedservicecomponent>> ManagedCommChannel component takes on the definition as described in the UML Profile for Component Framework::Infrastructure::Radio Services in addition to the specialization of the CommChannelComponent. The ManagedCommChannel provides the mechanism for a managed CommChannelComponent with state behavior.

Semantics

The ManagedCommChannel's operational state is based upon the operational state of its communication channel's devices. The ManagedCommChannel's usage state is IDLE when the communication channel has not been instantiated with a waveform. The ManagedCommChannel's usage state becomes BUSY when a waveform is instantiated on the communication channel. If the ManagedCommChannel's administrative state is SHUTTING_DOWN or LOCKED, then the communication channel is unavailable for waveform instantiation. This administrative state of the communication channel's devices may also be affected upon ManagedCommChannel admin state changes. Some devices may be shareable across communication channels, which may not affect their admin states when communication channel admin state changes. While other devices are only associated with one communication channel, which will effect their admin states.

Whenever the adminState attribute changes, a StateChangeEventType (Component Framework::Infrastructure::Domain Management::Event Channels) event may be issued to an event channel. The StateChangeEventType event data shall be populated as follows when issued:

1. The producerId field is the identifier attribute of the RadioManager component.
2. The sourceId field is the identifier attribute of the CommChannel component.
3. The stateChangeCategory field is ADMINISTRATIVE_STATE_EVENT.
4. The stateChangeFrom and stateChangeTo fields reflect the adminState attribute value before and after the state change, respectively.

Whenever the operationalState attribute changes, a StateChangeEventType event may be issued to an event channel. The event data shall be populated as follows when issued:

1. The producerId field is the identifier attribute of the RadioManager component.
2. The sourceId field is the identifier attribute of the CommChannel component.
3. The stateChangeCategory field is OPERATIONAL_STATE_EVENT.
4. The stateChangeFrom and stateChangeTo fields reflect the operationalState attribute value before and after the state change, respectively.

Whenever the usageState attribute changes, a StateChangeEventType event may be issued to an event channel. The StateChangeEventType event data shall be populated as follows when issued:

1. The producerId field is the identifier attribute of the RadioManager.
2. The sourceId field is the identifier attribute of the CommChannel.
3. The stateChangeCategory field is USAGE_STATE_EVENT.

The stateChangeFrom and stateChangeTo fields reflect the usageState attribute value before and after the state change, respectively.

8.2.4 ManagedRadioManager

Description

The <<managedservicecomponent>> ManagedRadioManager component takes on the definition as described in the UML Profile for Component Framework::Infrastructure::Radio Services in addition to the specialization of the RadioManager. The ManagedRadioManager provides the mechanism for a managed RadioManagerComponent with state behavior.

Semantics

The ManagedRadioManager’s operational state shall be based upon the operational state of its communication channels and devices. The Manager RadioManager’s usage state shall be IDLE when all of its communication channels are IDLE. The ManagedRadioManager’s usage state becomes ACTIVE when any of its communication channel is not IDLE. The ManagedRadioManager's usage state shall be BUSY when all of its communication channels are not IDLE. If the ManagedRadioManager’s administrative state is SHUTTING_DOWN or LOCKED, then its communication channels shall be unavailable for waveform instantiation.

Whenever the adminState attribute changes, a StateChangeEventType (Component Framework::Infrastructure::Domain Management::Event Channels) event may be issued to an event channel. The StateChangeEventType event data shall be populated as follows when issued:

1. The producerId field is the identifier attribute of the ManagedRadioManager component.
2. The sourceId field is the identifier attribute of the ManagedRadioManager component.
3. The stateChangeCategory field is ADMINISTRATIVE_STATE_EVENT.
4. The stateChangeFrom and stateChangeTo fields reflect the adminState attribute value before and after the state change, respectively.

Whenever the operationalState attribute changes, a StateChangeEventType event may be issued to an event channel. The event data shall be populated as follows when issued:

1. The producerId field is the identifier attribute of the ManagedRadioManager component.
2. The sourceId field is the identifier attribute of the ManagedRadioManager component.
3. The stateChangeCategory field is OPERATIONAL_STATE_EVENT.
4. The stateChangeFrom and stateChangeTo fields reflect the operationalState attribute value before and after the state change, respectively.

Whenever the usageState attribute changes, a StateChangeEventType event may be issued to an event channel. The StateChangeEventType event data shall be populated as follows when issued:

1. The producerId field is the identifier attribute of the ManagedRadioManager.
2. The sourceId field is the identifier attribute of the ManagedRadioManager.
3. The stateChangeCategory field is USAGE_STATE_EVENT.
4. The stateChangeFrom and stateChangeTo fields reflect the usageState attribute value before and after the state change, respectively.

8.2.5 ManagedSecureCommChannel

Description

The <<commchannelcomponent>> ManagedSecureCommChannel is a specialization of the SecureCommChannel and ManagedCommChannel components.

Semantics

This type of communication channel provides both managed and secure capability.

8.2.6 ManagedSecureRadioManager

Description

The <<radiomanagercomponent>> ManagedSecureRadioManager component is a specialization of the SecureRadioManager and ManagedRadioManager.

Semantics

This type of radio manager provides both managed and secure capability.

8.2.7 RadioManager

Description

The RadioManager provides additional attributes and operations for managing a RadioSet's channels.

Attributes

- `<<readonly>>commChannels: CommChannelComponent [1..*]`
  The commChannels attribute shall contain the set of communication channels for a RadioSet that this RadioManager is managing.
- `<<readonly>>commChannelsDescriptor: String`
  The commChannelsDescriptor attribute contains the URL of the descriptor that describes the RadioSet's Channels.

Semantics

The RadioManager's instantiateChannel operation instantiates one of its CommChannels using the input parameters.

8.2.8 RadioManagerComponent

Description

The <<radiomanagercomponent>> RadioManagerComponent takes on additional functionality for managing a RadioSet by realizing the RadioManager interface.

8.2.9 SecureRadioManager

Description

The <<radiomanagercomponent>> SecureRadioManager component takes on the definition as described in the UML Profile for Communication Channel::Radio Management in addition to the specializations of the RadioManagerComponent and the interfaces realized by this component. The SecureRadioManager component provides the mechanism of managing a secure radio manager.

Semantics

The usage of a radio manager after it has been zeroized is unspecified.

8.2.10 SecureCommChannel

Description

The <<commchannelcomponent>> SecureCommChannel component takes on the definition as described in the UML Profile for Communication Channel::Radio Management in addition to the specializations of the CommChannelComponent and the interfaces realized by this component. The SecureCommChannel provides the mechanism of managing a secure communication channel.

Semantics

The usage of a communication channel after it has been zeroized is unspecified.

8.2.11 WaveformInstantiation

Description

The WaveformInstantiation interface provides the mechanisms for instantiation a waveform application onto a communication channel.

Operations

- `instantiateWaveform (in waveformName: String, in instanceWFName: String, in wfProperties : Properties, in channelProperties: in Properties, return CommChannelComponent): {raises = (InstantiationError, InvalidChannelProperties, InvalidWFProperties, UnknownWaveform)}`
  The instantiateWavform operation deploys a waveform application onto a channel.

Types and Exceptions

- `<<exception>>InstantiationError`
  The InstantiationError exception, specialization of SystemException, is raised when the instantiateWaveform

operation is unsuccessful due to internal processing errors. The error number indicates an ErrorNumberType value (e.g., E2BIG, ENAMETOOLONG, ENFILE, ENODEV, ENOENT, ENOEXEC, ENOMEM, ENOTDIR, ENXIO, EPERM). The message is component-dependent, providing additional information describing the reason for the error.

- `<<exception>>InvalidChannelParameters (invalidProperties: Properties)`
  The InvalidChannelParameters exception is raised when the input channelParameters parameter is invalid.
- `<<exception>>InvalidWFParameters (invalidProperties: Properties)`
  The InvalidWFParameters exception is raised when the input wfParameters parameter is invalid.
- `<<exception>>UnknownWaveform`
  The UnknownWaveform exception indicates the waveform is not known.

Semantics

The instantiateWaveform operation shall deploy the waveform as specified by the input waveformName parameter onto a Channel. The instantiateWaveform operation shall return a CommChannelComponent when the instantiateWaveform operation successfully instantiated the waveform application onto the Channel. The instantiateWaveform operation shall use the input wfParameters for the initial configuration of the deployed waveform. The instantiateChannel shall use the channelParameters for the initial setup of the instantiated CommChannel.

The instantiateWaveform operation shall raise the UnknownWaveform exception when the input waveformName is not known. The instantiateWaveform operation shall raise the InstantiateError exception when the Channel cannot be successfully instantiated due to internal processing error(s). The instantiateWaveform operation shall raise the InvalidChannelProperties exception when the input channelParameters parameter is invalid. The InvalidChannelProperties identifies the properties that are invalid. The instantiateWaveform operation shall raise the InvalidWFProperties exception when the input channelParameters parameter is invalid. The InvalidWFProperties identifies the properties that are invalid.

8.2.12 XmitControl

Description

The XmitControl interface provides the mechanism to control a component's transmission such as transmission of radio frequencies.

Attributes

- `<<configureproperty>>xmitInhibit: Boolean`
  The xmitInhibit attribute is used to control and return the status of the transmission state of a component.

Semantics

The xmitInhibit attribute when a configuration value of "True" means the component shall inhibit transmission, otherwise the component can transmit.

8.2.13 ZeroizeControl

Description

The ZeroizeControl interface provides the mechanism to zeroize a component's environment or information.

Attributes

- `<<queryproperty>>zeroized: Boolean`
  The zeroize attribute is used to return the status of the zeroized state of a component. A True value indicates the component is zeroized, otherwise the component is not zeroized.

Operations

- `zeroize ()`
  The zeroize operation is used to command the component to zeroize its environment. The information that gets zeroized is component dependent.

9 Platform Specific Model

The PSM consists of CORBA and XML that are based upon the PIM and UML Profile for Communication Channel. The PIM to PSM transformation rules are not universal rules for creating *any* PSM, but only used for the purpose of this specification. This section defines a non-normative reference PSM. Non-CORBA PSMs may also be fully compliant to this specification as a whole.

The rule set for transforming Comm Channel Facilities PIM (UML packages, interfaces, types, and exceptions) into CORBA constructs is as follows:

1. UML interfaces and interface extensions are mapped to CORBA interfaces. The CORBA interface names are without the prefix "I" in the interface name as used in the radio Management PIM Facilities.
2. UML attributes with readonly and readwrite map to CORBA attributes in CORBA interfaces.
3. UML attributes with configureproperty, queryproperty, and testproperty do not map to CORBA attributes in CORBA interfaces. Instead XML definitions are used that follow the Property types as defined in UML Profile for Component Framework::Application and Device Components::Properties section.
4. UML classes without operations that are not stereotyped and used for type definitions map to CORBA Struct stereotypes in the CORBA interfaces and modules. The parent classes do not get translated into CORBA types, instead the parent class attributes are added to the subclass in the CORBA definition.
5. UML <<datatype>> map to CORBA basic types. Primitive types are mapped to CORBA primitive types and primitive sequence types are mapped to CORBA Typedef of primitive sequence types.
6. UML exceptions and exception extensions map to CORBA exceptions. There is no specialization of exceptions in CORBA so the (UML Profile for Component Framework::Application and Device Components::BaseTypes) SystemException definition does not appear in the generated CORBA interfaces but all the specialization exceptions of SystemException are in the CORBA interfaces with the same attributes as defined for SystemException.
7. UML attributes that have a cardinality of many [*] map to a CORBA Typedef of sequence types.
8. UML operations and <<optional>> operations map to operations in the CORBA interfaces.
9. Transformations are only performed for concrete classes, not for template classes. Concrete classes that bind to template classes are used in the PSM.
10. For Interfaces that reference a component stereotype for a type, the “component” qualifier is removed from the name. For Example, FileManagerComponent would become FileManager as the type for the parameter or attribute.
11. UML attributes with constant stereotype map to CORBA constants in CORBA interfaces.
12. Basic types (e.g., Any, Object) map to CORBA types.

Other non-CORBA PSM transforms (e.g., XML) are as follows:

1. The UML Profile for Communication Channel maps to Channel and Communication Equipment are transformed to XML using the following rules:
   - Properties are transformed as described in the Component Framework Specification (see Section 3.2.2, “UML Profile for Component Framework Specification for more information).

- Communication Equipment
  - Each CommEquipment stereotype or UML Device definition maps to the CommEquipment XML element definition. The CommEquipment name and stereotype names map to the name and stereotypeName elements of the CommEquipment XML element.
  - All properties of the CommEquipment map to the properties of the CommEquipment XML element as specified in item 1 (Properties) above.
  - All ports (AnalogInputPort, AnalogOutputPort, and DigitalPort map to the ports element of the CommEquipment XML element.
  - The properties of all communication equipment ports map to the properties of the Port XML element as specified in item 1 (Properties) above.
  - The Port name and stereotype name map to the name and stereotypeName elements of the Port XML element.
- Communication Channel
  - All Channel stereotypes map to the Channel XML element.
  - The properties of a Channel map to the properties element of the Channel XML element as specified above.
  - The Channel name and stereotype name map to the name and stereotypeName elements of the Channel XML element.
  - Associated Channels (LogicalPhysicalChannel, LogicalIOChannel, LogicalProcessingChannel, LogicalSecurityChannel) map to the subchannels XML element of the Channel XML element as references to their Channel XML element.
- Associated CommEquipments map to the commEquipments element of the Channel XML element as references to their CommEquipment XML element.
- Channel Connections map to connections element of the Channel XML element. A CommEquipmentConnector maps to the CommEquipmentConnector XML element.

The top most CORBA is called DfSWRadio which maps to the PIM Facilities package. There is a PhysicalLayer CORBA module within the DfSWRadio that encompasses all physical layer facilities PIM. SerialIO and AudioIO are CORBA modules within the PhysicalLayer module. There is also a RadioControl CORBA module within the DfSWRadio that encompasses all the Radio Control Facilities PIM. The DfSWRadio maps to existing IDL definition used in industry, therefore the IDL does not follow all of the OMG CORBA guidelines (e.g., operation, attribute, and parameter names), in order to reduce impact on industry.

Annex A   Software Radio Reference Sheet

The Software Radio specification responds to the requirements set by "Request for Proposals for a Platform Independent Model (PIM) and CORBA Platform Specific Model (PSM)" (swradio/02-06-02). The original specification (dtc/05-10-02) has been reorganized into 5 volumes, as follows:

Volume 1. Communication Channel and Equipment

This specification describes a UML profile for communication channel. The profile provides definitions for creating communication channel and communication equipment definitions. The specification also provides radio control facilities and physical layer facilities PIM for defining interfaces and components for managing communication channels and equipment for a radio set or radio system. Along with the profile and facilities is a platform specific model transformation rule set for transforming the communication channel into an XML representation and CORBA interfaces for the radio control facilities.

Volume 2. Component Document Type Definitions

This specification defines the content of a standard set of Data Type Definition (DTD) files for applications, components, and domain and device management. The complete DTD set is contained in Section 7, Document Type Definitions. XML files that are compliant with these DTD files will contain information about the service components to be started up when a platform is power on and information for deploying installed applications.

Volume 3. Component Framework

This specification describes a UML profile for component framework. The profile provides definitions for applications, components (properties, ports, interfaces, etc.), services, artifacts, logical devices, and infrastructure domain management components. In the profile are also library packages that contain interfaces for application, service, logical device, and infrastructure domain management components. Along with the profile is a platform specific model transformation rule set for transforming the profile model library interfaces into CORBA interfaces.

Volume 4. Common and Data Link Layer Facilities

This specification describes a set of facilities PIM for application and component definitions. The set of facilities are common and data link layer facilities that can be utilized in developing waveforms and platform components, which promote the portability of waveforms across Software Defined Radios (SDR). Along with the facilities PIM is a platform specific model transformation rule set for transforming the facilities into CORBA interfaces.

Volume 5. POSIX

This specification defines the application environment profiles for embedded constraint systems, based on Standardized Application Environment Profile - POSIX® Realtime Application Support (AEP), IEEE Std 1003.13-1998.

INDEX

R

S

T

V

W

X

Z