



**Smart Antenna API Specification**

**Approved 8 November 2007**

**SDRF-07-S-0016-V1.0.0**

## Table of Contents

<b>1. Scope</b> .....	<b>3</b>
<b>2. Conformance</b> .....	<b>3</b>
2.1. Conformance by a Model of a specific Application .....	3
2.2. Conformance by a Tool .....	3
2.3. Conformance on the part of a Component PSM.....	4
<b>3. Reference</b> .....	<b>5</b>
3.1. Normative References.....	5
3.2. Non-normative References .....	6
<b>4. Term and Definitions</b> .....	<b>7</b>
<b>5. Symbols and abbreviated terms</b> .....	<b>8</b>
<b>6. Additional Information</b> .....	<b>9</b>
6.1. Changes to Adopted OMG Specifications.....	9
6.2. How to Read this Specification.....	9
6.3. Acknowledgements.....	9
6.4. Security and Regulatory.....	9
6.5. Smart Antenna System.....	10
6.6. Classification of smart antennas .....	10
<b>7. UML Profile for Smart Antenna API(SA API)</b> .....	<b>11</b>
7.1. Types.....	11
7.2. CommEquipment for SA-API.....	12
<b>8. Platform Independent Model (PIM)</b> .....	<b>12</b>
8.1. Control Facilities.....	15
8.2. Synchronization Facilities.....	17
8.3. Algorithm Facilities .....	19
<b>9. PSM for Smart Antenna API</b> .....	<b>25</b>
9.1. Mapping Rule.....	25
9.2. IDL Mapping .....	26

## 1. Scope

This Specification responds to the requirements set by “Request for Proposals of PIM and PSM for Smart Antenna” (sbc/06-12-10) of smart antenna subsystem that can be utilized for expanding a single antenna system to array antenna system.

The Smart Antenna API specification is physically partitioned into three major chapters: UML Profile for Smart Antenna API (SA API), SA API PIM, and SA API PSM.

UML Profile for SA API defines a language for modeling smart antenna system by expanding the UML language. SA API PIM provides a set of interfaces for interfacing with the signal processing module, RF module, and controller module. SA API PSM provides a rule for transforming the elements of the profile and SA API PIM into the platform specific model for CORBA IDL and XML

The SA API specification is related to “Communication Channel and Equipment Specification (formal/07-03-02)” volume in such a way that stereotypes and class that have not been commented in the SA API specification are defined in it.

## 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. Conformance by a Model of a Specific Application

### 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:

#### 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. Reference

### 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>]

#### 3.1.2. CORBA Core Specification

##### 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>]

## 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. Term and Definitions

For the purposes of this specification, the terms and definitions given in the normative reference and the following 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.

### **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.

### **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
IDL	Interface Definition Language
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
SAAPI	Smart Antenna API
UML	Unified Modeling Language
XML	eXtensible Markup Language
DOA	Direction Of Arrival
SWRadio	Software Radio Components Specification

## 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. How to Read 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 smart antenna components. Chapter 8 contains the Smart Antenna Facilities Platform Independent Model (PIM). The UML language defined in Chapter 7 is used to specify this PIM. In Chapter 9, the mapping process from the Platform Independent Model (PIM) to a Platform Specific Model (PSM) is described.

### 6.3. Acknowledgements

The following companies submitted and/or supported parts of this specification:

- Alcatel
- Borland Software Corporation
- EDS
- HY-SDR research center, Hanyang University
- L-3 communications
- Lockheed Martin
- MITRE
- Mercury
- OIS
- Office of the Ministry of Defence
- PrismTech
- SDR Forum
- SELEX SI
- Virginia Tech. University

### 6.4. Security and Regulatory

There are two subjects that are not included herein: Security and Regulatory. These subjects impact the broader software radio (and other device-bound) topics (sensors, robotics, etc). As such, this document will, in a future release, address a specialization of the general solution adopted by the broader security and regulatory standardization community. Therefore the following caveats apply:

- Security

This architecture document does not include the functionality for security regarding certification of the source code or over the air delivery of new characteristics for the Smart Antenna or security functional protection mechanisms. The addition of such mechanisms is not expected to alter the architecture defined herein.

- Regulatory

This architecture document does not include the functionality for managing regulatory requirements for the Smart Antenna. The Smart Antenna herein described functionality could most likely require the addition of finer control and explicit regulatory controls. The addition of such controls is not expected to alter the architecture defined herein; although finer power/bandwidth/ERP may be required to meet regulations.

## 6.5. Smart Antenna System

A *Smart antenna* is an antenna array system that is aided by a processing system that processes the signals received by the array or transmitted by the array using suitable array algorithms to improve wireless system performance. An antenna array consists of a set of distributed antenna elements (dipoles, monopoles or directional antenna elements) arranged in certain geometry (e.g., linear, circular or rectangular grid) where the spacing between the elements can vary. The signals collected by individual elements are coherently combined in a manner that increases the desired signal strength and reduces the interference from other signals. A smart antenna can be viewed as a combination of antenna elements, using some form of RF, IF or baseband array combination, that transmit or receive RF signals using "smart" algorithms. A software defined smart antenna is a smart antenna in which certain operating characteristics, such as the field of regard, frequency of operation, access mode, or transmit/receive waveforms can be altered by firmware or software download after its manufacture.

## 6.6. Classification of smart antennas

Base on the signal processing technique followed at the baseband output of the antenna array smart antennas can be group into four basic types based on : 1) Beamforming 2) Space time equalization 3) Diversity combining 4) Multiple input multiple output(MIMO) processing.

- 1) **Beamforming:** Through Beamforming, a smart antenna algorithm can receive predominantly from a desired direction(direction of the desired source) compared to some undesired direction(direction of interfering sources). This implies that the digital processing has the ability to shape the radiation pattern for both reception and transmission and to adaptively steer beams in the direction of the desired signals and put nulls in the direction of the interfering signals. This enables low co-channel interference and large antenna gain to the desired signal. Beamforming systems can be implemented in two ways; fixed beamforming systems or fully adaptive systems. A fixed beamforming system has a beamforming network(BFN) followed by RF switches which operate in the RF/analog domain. The switches are controlled by a control logic which selects a particular beam. Here the processing required is minimal as the control logic has to choose one of the predetermined set of weights to select a beam. In adaptive beamforming, the antenna gains or weights are chosen adaptively through running array algorithms in the digital domain.
- 2) **Space-time equalization:** The preceding two techniques usually assume that the signal of interest is a narrowband signal compared to the coherence bandwidth of the channel and is thus subjected to flat fading across the bandwidth of the signal. Multipath fading in wireless communication can also introduce frequency distortion to the received signal.

By introducing temporal processing in each antenna element to remove the effect of frequency distortion and doing a spatial combining described above results in mitigating channel induced frequency selective fading and providing antenna gain. Such schemes are called space-time adaptive processing (STAP) or equalization.

- 3) Diversity combining: A major limiting factor in wireless communication is multipath fading where the amplitude of the received signal fluctuates over time. The occurrence of a deep fade where the signal amplitude becomes very small can impair the communications link for a conventional or a single antenna system. When multiple antennas are used it becomes less likely that two or more antennas undergo deep fades at the same time. This diversity in the received signal, for the same transmitted information, is exploited by smart antenna processing schemes. Many simple algorithms, such as maximal ratio combining, equal gain combining, and selection diversity have been developed to take advantage of using antenna arrays to exploit diversity reception in wireless systems. These algorithms weight the received signal similar to beamforming but based on a different criterion used in the algorithm.
- 4) Multiple-Input Multiple-Output (MIMO): As the name suggests this scheme requires array processing at the transmitter and receiver. There are two different types of MIMO schemes: one uses spatial multiplexing to enhance data rate for a given bandwidth (thus, the spectral efficiency) and the other uses space time coding using diversity combining techniques to combat fading. In the multiplexing scheme, data is serial to parallel converted and transmitted simultaneously over multiple antenna elements. The receiver also uses multiple antenna elements to receive the signal and applies a maximum likelihood (ML) algorithm to retrieve the simultaneously transmitted symbols. One key assumption in this case is that the propagation environment has to provide rich scattering; in other words, the propagation channel has to include a large number of scattering objects that will generate independent fading at the antenna elements. In the case of space-time coding, symbols to be transmitted are coded over multiple antennas and symbol time durations in such a way that the receiver can easily regenerate the transmitted signals by doing a linear processing on received signal. The space-time codes rely on the orthogonality present in the coded symbols for proper detection, and additionally they require the fading to be independent between the antenna elements for best performance results.

## 7. UML Profile for Smart Antenna API(SA API)

This non-normative section defines the UML Profile for only Smart Antenna API. The set of stereotypes and types that are not described in this section are defined in UML Profile for SWRadio components.

### 7.1. Types

- Complex (real: Float, imag: Float)  
Complex data type denote a general complex number.
- ComplexSequence  
ComplexSequence is an unbounded sequence of Complex (s).

- `<<primitive>>ArrayAntennaType`  
ArrayAntennType, a specialization of String, denote the physical configuration of an array antenna (e.g., Phased Array, Circular Array, etc.)

## 7.2. CommEquipment for SA API

### 7.2.1. ArrayAntenna

#### Description

The ArrayAntenna stereo type, shown in Figure 1, represents an antenna array which is a multiple of active antennas coupled to a common source or load to produce a directive radiation pattern. The ArrayAntenna class owns many AntennaElements.



Figure 1-ArrayAntenna 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.
- `<<configureproperty>>radiationPattern: RadiationPatternType`  
The radiationPattern attribute represents the current radiation pattern configured in the device.
- `<<characteristicproperty>>type: ArrayAntennaType`  
The type attribute indicates the physical type of the array antenna
- `<<configureproperty>>maxMutualCoupling: Decibel`  
The maxMutualCoupling is maximum mutual coupling value between antenna elements.
- `<<queryproperty>>bandwidth: Hertz`  
The bandwidth attribute indicates the bandwidth of physical array antenna.
- `<<queryproperty>>interElementDistance: Meter`  
The interElementDistance attribute represents physical distance between antenna elements.

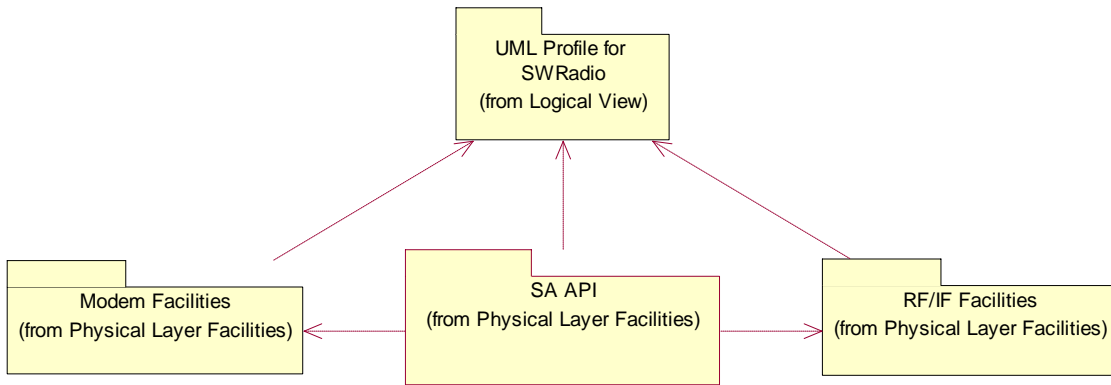
#### M1 Association

- arrayElemnt: AntennaElement [1..\*]  
The individual radiating element object of the ArrayAntenna.

## 8. Platform Independent Model (PIM)

The SA API provides interfaces used to configure and control a Smart Antenna System. In order to specify a SA API, we have to first define a standard PIM because a SA API specification should be valid regardless of platform types. Note that a PIM itself is a unique model of the SA API. This PIM of the SA API consists of three facilities each of which has been defined in accordance with their functions. These three facilities operate in conjunction with each other to

define the UML Profile for the SWRadio. Figure 2 is a Package Diagram illustrating this relation.

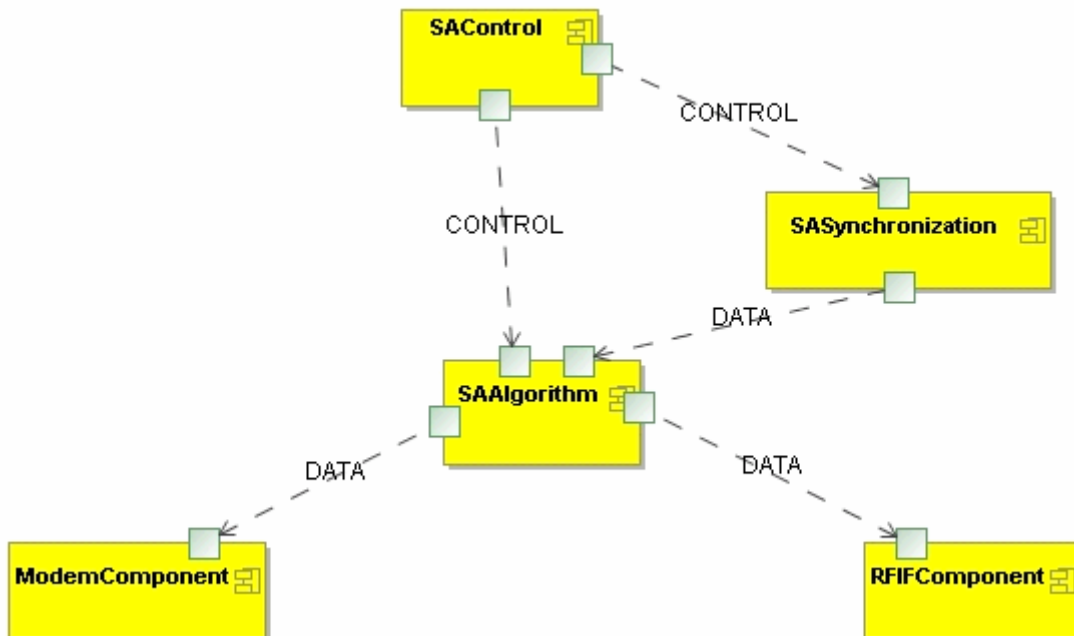


**Figure 2-Package Diagram**

Three facilities are used to control the entire Smart Antenna System:

- SAControl
- SASynchronization
- SAAAlgorithm

The control facilities are used to control all algorithm operations performed in the digital signal processing parts and RF/IF operations such as analog to digital or digital to analog conversion. The synchronization facilities are used to synchronize signals required for RF chain calibration and channel estimation. These synchronization facilities have been adopted in order to support the calibration and channel estimation. Note that the channel estimation, which is provided in SWRadio components as well, has been included in the synchronization facilities in order to fully exploit the merit of Smart Antenna System. The algorithm facilities are used to execute all the algorithms that are needed for the Smart Antenna System to provide superb performance compared to Single Antenna System. Algorithm facilities process signals fed by RF/IF facilities and transfer the results into the Modem facilities. Figure 3 illustrates data flows among components in the three facilities of SA API, Modem facilities, and RF/IF facilities.



**Figure 3 – Data flows among components in SA API and SWRadio components**

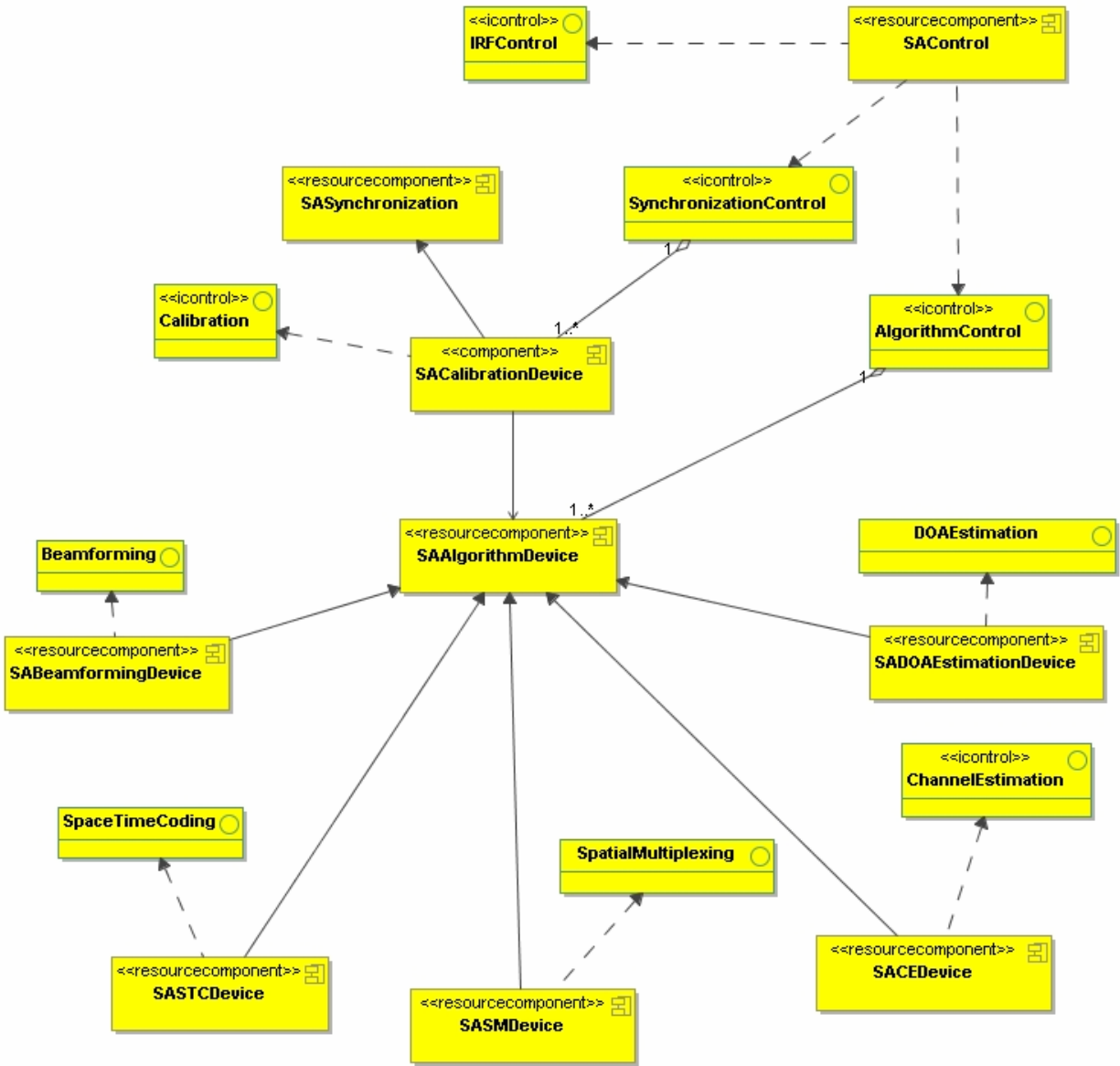


Figure 4-SA API Overview

Figure 4 illustrates a SA API overview that has been determined in accordance with the relation of the three facilities of SA API. More detailed explanations about interfaces among facilities in the SA API shown in Figure 4 are provided in the following sections.

### 8.1. Control Facilities

In this section, each function and interface provided by the Control facilities is described. Figure 5 illustrates control facilities that include SAControl component, Synchronization Control interface, Algorithm Control interface, and RF Control interface. Interfaces required in the SAControl component are shown in Figure 6. It can be observed from Figures 5 that RFControl interface, SynchronizationControl interface, and AlgorithmControl interface have been inherited into SAControl component, in order for SAControl component to control RF/IF component, SASynchronization component, and SAAgorithm component, respectively, according to the functions to be performed in the SAControl component.

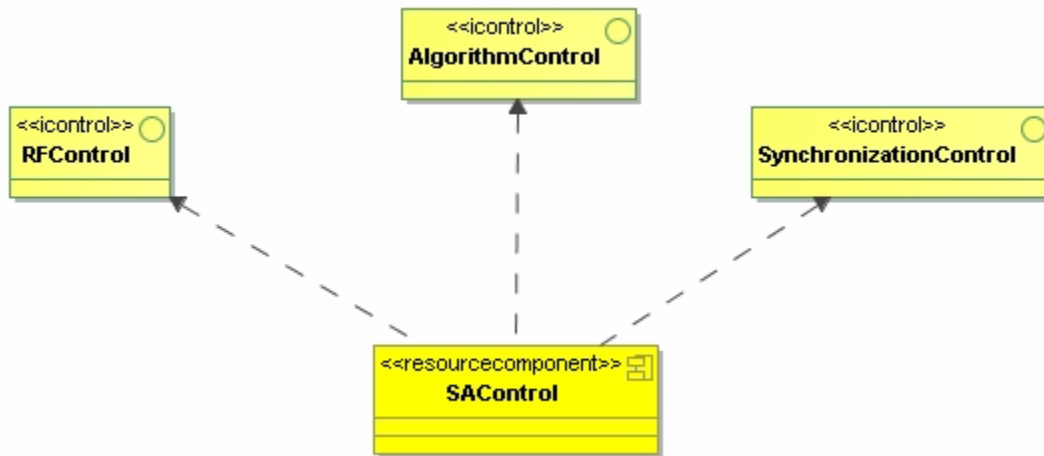


Figure 5-SAControl Facilities

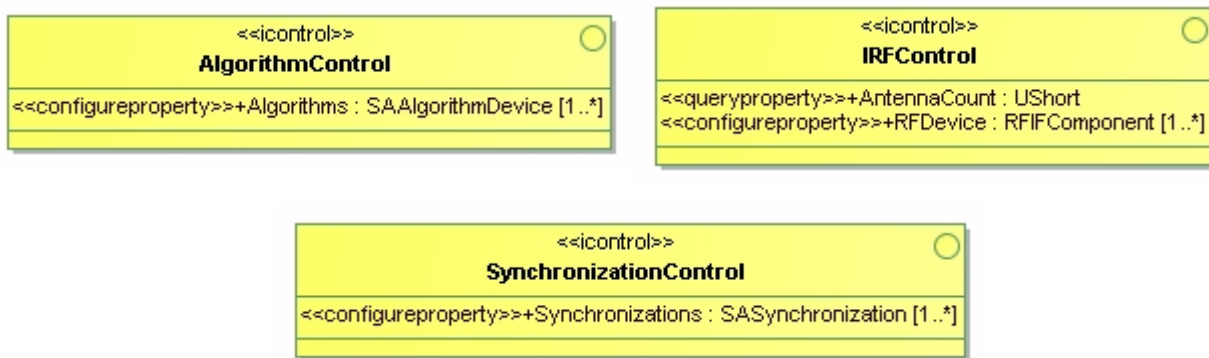


Figure 6-Interface of SAControl

### 8.1.1. SAControl

#### Description

The SAControl component is used to control not only all the three facilities of the SA API but also RF/IF facilities of SWRadio components.

### 8.1.2. AlgorithmControl

#### Description

The Algorithm interface is used to control SAAAlgorithmDevice components.

#### Attributes

- <<configureproperty>>Algorithms: SAAAlgorithmDevice [1..\*]  
The Algorithms attribute shall contain the set of SAAAlgorithmDevice references. The SAAAlgorithmDevice references can be used to control plural SAAAlgorithmDevice components.

### 8.1.3. SynchronizationControl

#### Description

The Synchronization is used to control the SASynchronization components.

#### Attributes

- <<configureproperty>>Synchronizations: SASynchronization [1..\*]  
The Synchronizations attribute shall contain the set of SASynchronization references. The SASynchronization references can be used to control plural SASynchronization components.

### 8.1.4. RFControl

#### Description

The RFControl interface is use to control RF/IF components.

#### Attributes

- <<configureproperty>>RFDevices: RFIFComponent [1..\*]  
The RFDevices attribute shall contain the set of RFIFComponent references. The RFIFComponent references can be used to control plural RFIFComponent's.
- <<configureproperty>>AntennaCount: Integer  
The AntennaCount attribute represents the number of Antenna elements.

## 8.2. Synchronization Facilities

In this section each function and interface in Synchronization facilities is described. Figure 7 illustrates SASynchronization facilities that include the SASynchronization component and the Calibration, Latency. The Latency interface has been defined in the Modem Facilities of SWRadio components [1]. Interfaces required in the SASynchronization component are shown in Figure 8 according to the functions to be performed in the SASynchronization component.

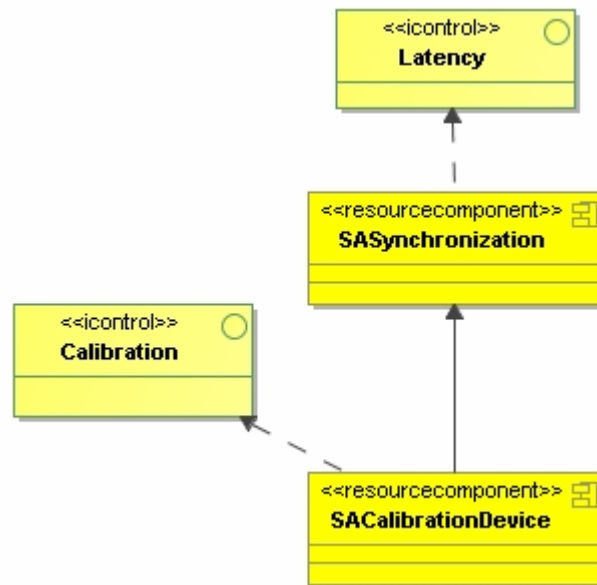


Figure 7-SASynchronization Facilities

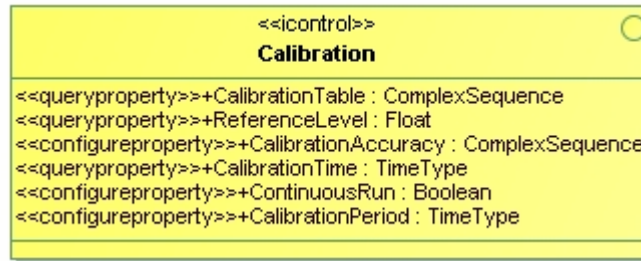


Figure 8 - Interface of SASynchronization

### 8.2.1. SASynchronization

#### Description

The SASynchronization component realizes the Latency interface. In order to perform the calibration, this component acquires the RF/IF signal from SAAAlgorithmDevice component. The calibration obtained by the SACalibrationDevice component is transferred back to the SAAAlgorithmDevice component for compensation.

#### Constraint

The SASynchronization component shall provide one ControlPort and one DataPort.

### 8.2.2. Calibration

#### Description

The Calibration interface is used to calibrate each of the smart antenna array elements.

#### Attributes

- <<queryproperty>>CalibrationTable: ComplexSequence  
The CalibrationTable attribute represents the output of Calibration.
- <<queryproperty>>ReferenceLevel: Float  
The ReferenceLevel attribute represents the value normalized to a unity power or some non-unity input level.
- <<configureproperty>>ContinuousRun: Boolean  
The ContinuousRun attribute indicates whether or not calibration is executed continuously.
- <<queryproperty>>CalibrationAccuracy: ComplexSequence  
The CalibrationAccuracy attribute represents the required accuracy of calibration. The required accuracy shall be configured in both amplitude and phase.
- <<configureproperty>>CalibrationPeriod: TimeType  
The CalibrationPeriod attribute is used to control calibration period.
- <<queryproperty>>CalibrationTime: TimeType  
The CalibrationTime attribute return the time required for a single calibration pass using the active settings.

### 8.2.3. SACalibrationDevice

#### Description

The SACalibrationDevice is a concrete component of SACalibration and SASynchronization. The calibration is to compensate for amplitude and phase differences of the signal path associated with each antenna in transmit and receive mode. The problem of calibration has arisen because the amplitude and phase characteristics of signal path associated with each antenna are

different from each other. Especially even if the optimal weight vector is computed from the received signal such that the uplink communication of the smart antenna system can fully exploit the enhancements in both communication capacity and cell coverage, downlink beam-forming can never be optimized without accurate calibration. In other words, the objective of calibration is to compensate for the mutual coupling effects between antenna array elements as well as for the mismatches of channel amplitude and/or channel phase in smart antenna systems.

### 8.3. Algorithm Facilities

In this section each function and interface in Algorithm facilities is described. Figure 9 illustrates Algorithm facilities. The Latencyinterface has been defined in Modem Facilities of SWRadio components [1]. Interfaces required in the SAControl component are shown in Figure 10 in accordance with the functions to be performed in the algorithm component.

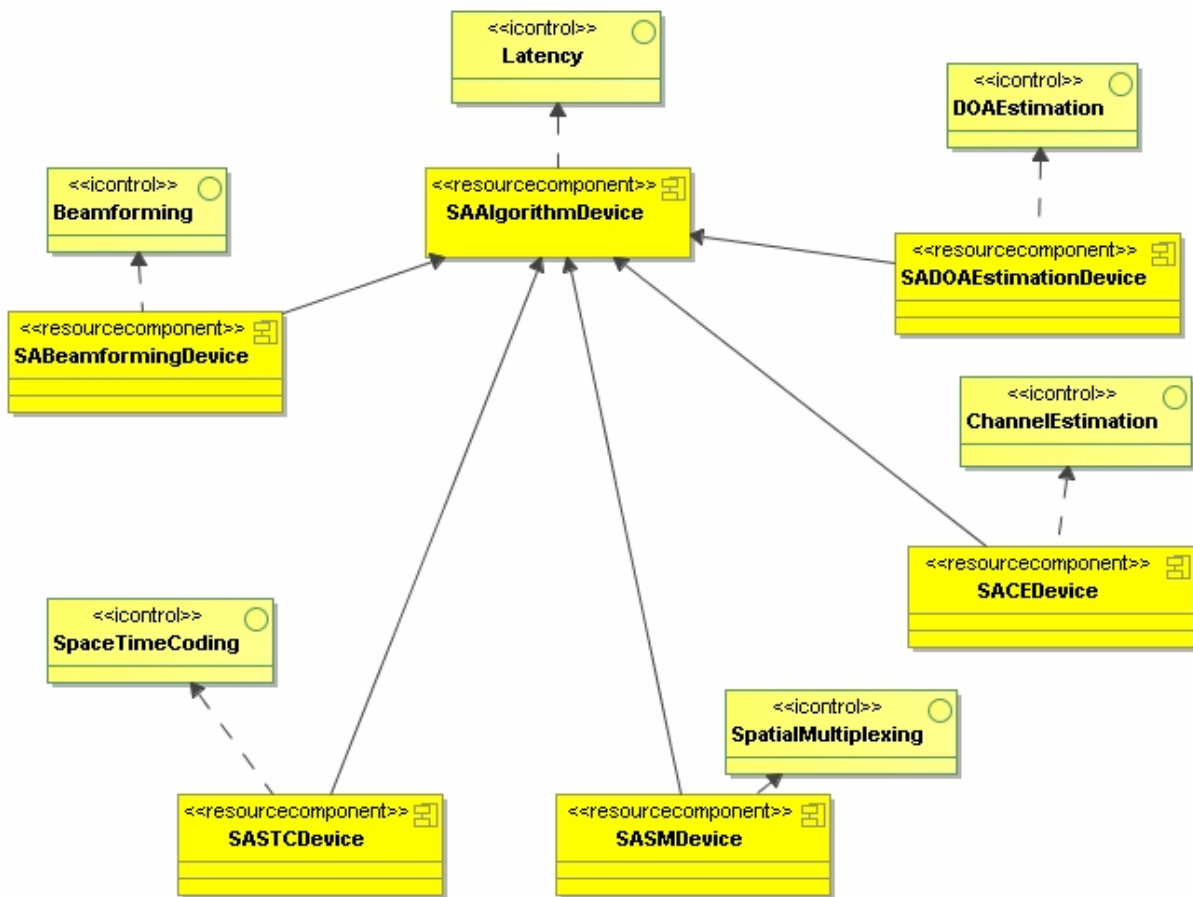


Figure 9-Algorithm Facilities

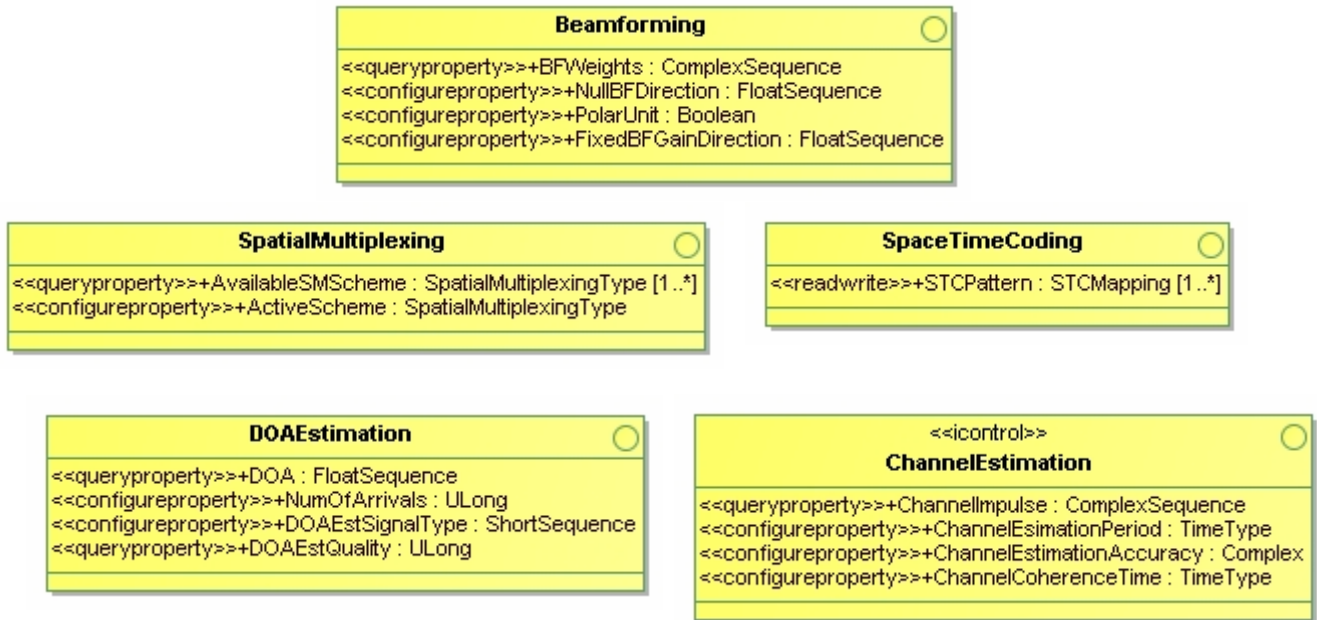


Figure 10-Interface of SAAAlgorithmDevice

### 8.3.1. SAAAlgorithmDevice

#### Description

The SAAAlgorithmDevice interface is a control interface from which all the algorithm components in the Smart Antenna Subsystem, i.e., SASTCDevice, SABeamformingDevice, SACEDevice, SASMDevice, and SADOAEstimationDevice component, have to be inherited. In other words, this control interface provides every interface used to control all the algorithm components.

#### Attributes

- <<configureproperty>>Active: Boolean  
The Active attribute indicates that the SAAAlgorithmDevice component is activated.
- <<EnumerationProperty>>OperationMode: AlgorithmOperationMode  
The OperationMode attribute is an operation mode of SAAAlgorithmDevice components.
- <<queryproperty>> Delays: TimeType  
The SAAAlgorithmDelay attribute is delay of signal processing executed by the SAAAlgorithm Device.
- <<queryproperty>> PowerConsumption: Float  
The SAAAlgorithmPowerConsumption attribute is power consumption of the SAAAlgorithmDe vice.
- <<configureproperty>>RequiredThroughput: ULong  
The RequiredThroughput attribute is a throughput (bits/sec) required by a user requesting ent ry into the network. This attribute may be used for selecting an appropriate algorithm.
- <<configureproperty>>TolerableDelay: TimeType  
The TolerableDelay attribute is a tolerable delay for a single alorithm excution. This attribut e may be required as a reference for selecting a proper algorithm.
- <<configureproperty>>RequiredBandwidth: Hertz  
The RequiredBandwidth attribute is a required bandwidth for link connection. The attribute may be used for selecting an appropriate algorithm.

- <<configureproperty>>TolerablePowerConsumption: Float  
The tolerablePowerConsumption attribute is tolerable power consumption for algorithm execution. This attribute may be used for selecting an appropriate algorithm.
- <<configureproperty>>MaxFER: Float  
The MaxFER attribute is tolerable FER for maintaining a link performance. This attribute may be used for selecting an appropriate algorithm.
- <<queryproperty>>Convergence: Boolean  
The Convergence attribute indicates whether the algorithm is confident that its link quality is high enough to satisfy the configured maximum FER and throughput configured by the corresponding attributes.
- <<queryproperty>>AvailableAlgorithm: AlgorithmType[1..\*]  
The AvailableAlgorithm attribute represents a list of the algorithms available for use on SAAAlgorithmDevice.
- <<configureproperty>>ActiveAlgorithm: AlgorithmType  
The ActiveAlgorithm attribute set algorithm or get activated algorithm.
- <<queryproperty>>OrderedEigenvalues: ShortSequence  
The OrderedEigenvalues attribute presents the eigen channel quality metrics used in SAAAlgorithmDevice,
- <<queryproperty>>OrderedEigenvectors: ComplexSequence  
The OrderedEigenvectors attribute presents the eigen channel quality metrics used in SAAAlgorithmDevice,
- <<configureproperty>>MaxEigenChannels: Short  
The MaxEigenChannels attribute configures maximum number of eigen channels to be used in SAAAlgorithmDevice.

### Operation

- getPreData(return ComplexType)  
The getPreData operation is provided to command the SAAAlgorithmDevice component to get pre-processing data such as pre-despreading data and pre-FFT data, etc.
- getPostData(return ComplexType)  
The getPreData operation is provided to command the SAAAlgorithmDevice component to get post-processing data such as post-despreading data and post-FFT data, etc.

### Types and Exceptions

- <<enumeration>>AlgorithmOperationMode ( CONTINUOUS, SINGLE\_BURST, REPEATED\_BURST )  
The AlgorithmOperationMode defines the operation mode of data processing.  
CONTINUOUS: SAAAlgorithmDevice components process input signals continuously.  
SINGLE\_BURST: SAAAlgorithmDevice components process the single burst input signal.  
REPEATED\_BURST: SAAAlgorithmDevice components process repetitively the single burst input signal.
- <<primitive>>AlgorithmType  
The AlgorithmType, a specialization of String, denotes the type of algorithm used for array processing.( e.g., Lagrange Multiplier based Algorithm, Eigen Space Based Algorithm, Maximum Likelihood Algorithm, Least Square Algorithm .Least Mean Square Algorithm, Zero Forcing Algorithm, etc.).

### 8.3.2. Beamforming

#### Description

The Beamforming interface is used to control the SABeamformingDevice.

#### Attributes

- <<queryproperty>>BFWeights: ComplexSequence  
The BFWeights attribute is weight vectors computed by the SABeamformingDevice. When this attribute is read, the SABeamformingDevice computes a new value from received signals. When this attribute is set up, the SABeamformingDevice applies a given value that provides the desired radiation pattern.
- <<configureproperty>>PolarUnit: Boolean  
The PolarUnit attribute is used to switch between real/imag mode and mag/phase mode of Weights attribute.
- <<configureproperty>>NullBFDirection: FloatSequence  
The NullBFDirection attribute is used to specify directions of nulls in degree to block known source of interference.
- <<configureproperty>>FixedBFGainDirection: FloatSequence  
The FixedBFGainDirection attribute is used to specify fixed gains (dB) in fixed direction (degree) to amplify weak signals in known direction.
- <<configureproperty>>SideLobeLevel: FloatSequence  
The SideLobeLevel attribute is used to limit the side lobe level in decibel (dB).

### 8.3.3. SABeamformingDevice

#### Description

The SABeamformingDevice is a concrete component of SAAlgorithmDevice and Beamforming. A beamforming algorithm in the SABeamformingDevice computes the weight vectors for both RX and TX operations. The weight vectors adaptively steer beams along the direction of desired signals and puts nulls along the direction of interfering signals.

### 8.3.4. SpaceTimeCoding

#### Description

The SpaceTimeCoding interface is used to control SASTCDevice.

#### Attributes

- <<readwrite>>STCPattern: STCMapping[1..\*]  
The STCPattern attribute represents the actual definition of the STC mapping. Each input symbol of SASTCDevice is mapped to one of transmit antennas according to STCMapping.

#### Types and Exceptions

- STCMapping ( NumAnt: ULong, codePattern: UShort [1..\*])  
NumAnt: Number of transmit antenna.  
codePattern: The actual space time code pattern as a UShort.

### 8.3.5. SASTCDevice

#### Description

The SASTCDevice is a concrete component of SAAlgorithmDevice and SpaceTimeCoding. The SASTCDevice is for Space Time Coding (STC) processing. A Space Time Coding (STC) is a method employed to improve the reliability of data transmission in wireless communication systems using multiple transmit antennas. STCs rely on transmitting multiple, redundant copies of a data stream to the receiver in the hope that at least some of them may survive the physical path between transmission and reception in a good enough state to allow reliable decoding.

### 8.3.6. SpatialMultiplexing

#### Description

The SpatialMutiplexing interface is used to control SASMDevice.

#### Attributes

- <<queryproperty>>AvailableSMScheme: SpatialMultiplexingType[1..\*]  
The AvailableSMScheme attribute represents a list of the spatial multiplexing schemes available for use on SASMDevice.
- <<configureproperty>>ActiveSMScheme: SpatialMultiplexingType  
The ActiveSMScheme attribute set SpatialMultiplexingType or get activated SpatialMultiplexingType.

#### Types and Exceptions

- <<primitive>> SpatialMultiplexingType  
The SpatialMultiplexingType, a specialization of String, denotes the type of algorithm used for or spatial multiplexing.(e.g.,V-BLAST, D-BLAST, H-BLAST, etc.).

### 8.3.7. SASMDevice

#### Description

The SASMDevice is a concrete component of SAAlgorithmDevice and SpatialMultiplexing. SASMDevice is for spatial multiplexing. The spatial multiplexing is a transmission technique in MIMO wireless communication to transmit independent and separately encoded data signals from each of the multiple transmit antennas.

### 8.3.8. ChannelEstimation

#### Description

The ChannelEstimation interface is used to control the channel estimation functions for Smart Antenna System.

#### Attributes

- <<queryproperty>>ChannelImpulse: ComplexType  
The ChannelImpulse attribute represents the channel estimation vector that is calculated by the channel estimator realized by the SASynchronization component.
- <<configureproperty>>ChannelCoherenceTime: Microsecond  
The ChannelCoherenceTime attribute represent channel coherence time. This attribute is necessary in selecting an appropriate algorithm.

- <<configureproperty>>ChannelEstimationPeriod: Millisecond  
The ChannelEstimationPeriod attribute is used to control channel estimation period. This attribute would be especially important to trade overhead time and processing against the rate of change in the channel due to platform motion, etc.
- <<configureproperty>> ChannelEstimationAccuracy: ComplexType  
The ChannelEstimationAccuracy attribute represents the required accuracy of channel estimation.

### 8.3.9. SACEDevice

#### Description

The SACEDevice is a concrete component of SAAAlgorithmDevice and ChannelEstimation. Space-time equalizations system or diversity combining system can be implemented using the SACEDevice. The Space-time Equalization is receiving technique use temporal processing on the signals received from multiple spatially diverse antennas to correct frequency distortion in the received signal path. And the diversity combining is also receiving technique to mitigate the multipath fading effects inherent in many wireless networks by combining the signals from multiple spatially diverse antennas together to improve signal quality.

### 8.3.10. DOAEstimation

#### Description

The DOAEstimation interface is used to control the SADOAEstimationDevice.

#### Attributes

- <<queryproperty>>DOA: FloatSequence  
The DOA attribute represents direction of arrival (DOA) angle in degree.
- <<configureproperty>>NumOfArrivals: Integer  
The NumOfArrivals specifies how many DOA estimates are required allowing for estimation of the arrival of the same signal from multiple directions.
- <<configureproperty>>DOAEstSignalType: ShortSequence  
The DOAEstimationSignalType attribute specifies the character of the various signals to estimate.
- <<queryproperty>>DOAEstQuality: Integer  
The DOAEstQuality attribute indicates DOA estimation quality.

### 8.3.11. SADOAEstimationDevice

#### Description

The SADOAEstimationDevice is a concrete component of SAAAlgorithmDevice and DOAEstimation.

## 9. PSM for Smart Antenna API

### 9.1. Mapping Rule

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.

13. Object references map to the name of CORBA objects.

## 9.2. IDL Mapping

PIM Name	IDL FileName
SAControl	Not necessary
AlgorithmControl	Not necessary
RFControl	Not necessary
SynchronizationControl	Not necessary
Calibration	Not necessary
ChannelEstimation	Not necessary
SAAAlgorithmDevice	DfSWRadioSmartAntenna.idl
Beamforming	Not necessary
DOAEstimation	Not necessary
SpaceTimeCoding	DfSWRadioSmartAntenna.idl
SpatialMultiplexing	Not necessary

Table 1-IDL Mapping