Smart Antenna API

Version: v 0.4; 16 August, 1998

Authors:[1] Chane Fullmer (chane@rooftop.com) & Dave Beyer (dave@rooftop.com)
Rooftop Communications Corp.

Heinrich Foltz (hfoltz@panam3.panam.edu) & James McLean (mclean@ccsi.com)
University of Texas – Pan American

1 Purpose

Advanced software protocols for distributed packet radio networks are being designed, tested, and fielded by a variety of organizations including: Rooftop Communications, University of California, Santa Cruz, SRI International, Bolt Beranek and Newman, and the University of California, Los Angeles. Additionally, a variety of organizations including University of Texas - Pan American, Virginia Tech, and University of Wisconsin are developing a new generation or electronically programmable smart antennas for such networks. These future networks promise to support efficient, reliable, and secure communication of multimedia traffic over rapidly-deployed, multihop wireless infrastructures, that can serve as seamless extensions of the Internet.

This Smart Antenna API was developed, and continues to evolve, to facilitate both collaboration and independent development of the network protocols and smart antenna hardware. The intent is to allow protocol software and smart antennas to be easily integrated (along with next-generation digital radio modems) or "mixed and matched," into distributed packet radio products (or *Internet Radios*).

Specifically, this Smart Antenna API is intended to:

- Define a concise, platform-independent, interface specification between smart antennas and the network protocols,
- Foster cross-organization collaboration between protocol and smart antenna developers,
- Permit the implementation and testing of protocols in the absence of actual antenna hardware,
- Provide standard methods for permitting antenna-specific extensions, and
- Permit easy porting of protocols between multiple smart antennas, and vice versa.

This Smart Antenna API is defined using the API Framework specified in the "API Framework for Internet Radios" document.[2] Following this framework, the API is defined primarily by a set of generic *"primitives"* that can be mapped to various software or hardware implementations, as appropriate for the particular hardware/software environment. Many of the primitives for this API are inherited from the "Core API" defined in the framework document.

An example serial interface implementation for this API is presented as an Appendix to this document [TBD]**.** The "Generic Device Driver" specification presents an example software implementation.[3]

[1] This work was supported by the Defense Advanced Research Projects Agency (DARPA). Refer to the *Acknowledgments* section for details

[2] See the "API Framework for Internet Radios" document, available through the http://www.rooftop.com and http://www.erg.glomo.com web-sites.

[3] See the "Generic Device Driver" document, available through the http://www.rooftop.com and http://www.erg.glomo.com web-sites.

2 Architecture

The position of this Smart Antenna API is presented in Figure 1. The Smart Antenna API is concerned solely with the control and status interface with the antenna, and ***not*** the analog RF "transmit signal" interface with the antenna.[4] The Smart Antenna API is defined separately from the Radio Device API to allow flexibility in implementations. For example, the antenna API may be implemented either using an entirely separate interface to that used for the Radio Device API, or simply as an extension of the Radio Device API. Note that the Smart Antenna API may also be present at the interface between the physical radio and the antenna, with selected primitives being passed through to the interface to the network protocols. Like the Radio Device API, this API is being defined in a manner which avoids restricting how the antenna, protocols, and interface functions are implemented.

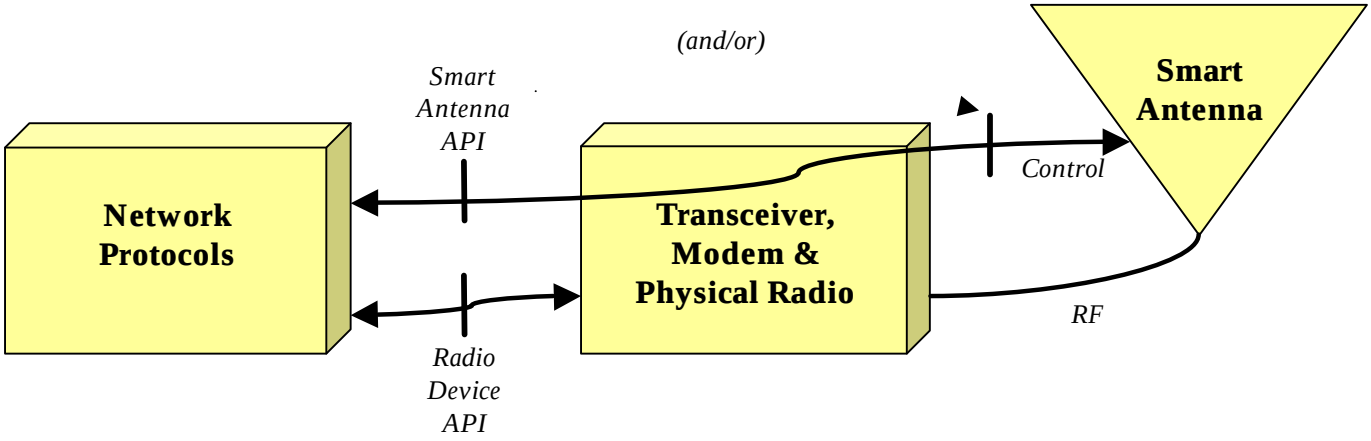

Figure 1: Position of Smart Antenna API

[4] In some cases, the antenna control and status information may actually be communicated using a separate carrier (or baseband) signal over the RF cable (separate from the RF signal to be transmitted). For these cases, this API is only concerned with the information flow in this separate carrier signal.

3 Logical Functionality

The logical functionality of the Smart Antenna API is defined using the API Framework defined in the "API Framework for Internet Radios" document. This Smart Antenna API inherits from, and extends, the "Core API" defined in that API Framework document.

In accordance with this API Framework, the logical functionality of the Smart Antenna API is defined by logical API primitives, qualifiers, and return codes. These are briefly summarized in this section. (Refer to the API Framework document for further information.) In addition, this section discusses how the basic packet handling procedures for this Smart Antenna API extend that of the Core API.

3.1 Primitives, Qualifiers and Return Codes

The Smart Antenna API is defined by four basic types of *primitives*, as described in the following table and illustrated in Figure 2.

Commands	Asynchronous protocols-to-device primitives for performing immediate, typically non-persistent actions. An example command primitive would be to reset the antenna.
Variables	Persistent antenna state or long-term measurement primitives that support one or more of the set, get, increment, and clear synchronous access operations. Control variables include the center frequency, band, bandwidth, polarization, and mode. Measurement variables include the received signal strength.
Responses	The synchronous device response to a protocol's command or variable operation. For a software based implementation (such as the Generic Device Driver implementation), this is typically handled using the return value from the command or variable function call (thus the dotted line in the diagram below).
Signals	Asynchronous device-to-protocols primitives for reporting recent, typically non-persistent events. The antenna device should support the selective enabling and disabling of each of these signals by the protocol software. An example would be a signal for an error condition.

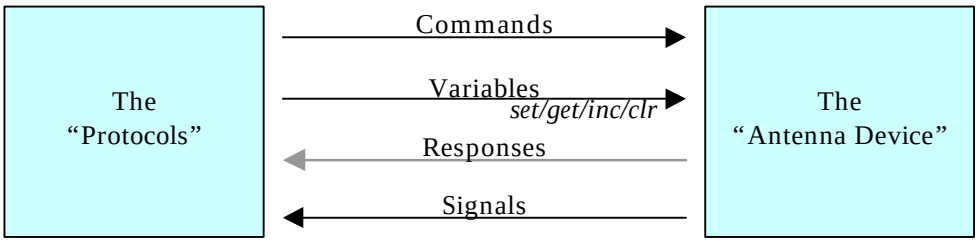

Figure 2: Basic Types of API Primitives

Each primitive can be *qualified* to give more specific instructions such as specifying the antenna "beam" to which the command should be applied (for antennae that support multiple beams. Of course, these qualifiers will only be relevant to antennae that support these capabilities.

The AntVarSignalEnabled variable is used by the protocols to enable or disable the generation of individual signals by the radio device. All signals can be enabled or disabled at once using the AntSigAll code (a code in the Signal space defined solely for this purpose).

The Antenna Device API also defines a set of return codes to provide a standard means for the Antenna device to indicate the success or failure status in each Response to Command and Variable operations, and in each asynchronous Signal delivered to the protocols. Examples of these codes include AntRetOk (request accepted or performed successfully), RetFail (general request failure), and RetInvCmd (command is invalid or has not been implemented by this antenna device).

4 Smart Antenna API Definition

This section gives the logical definition of the Smart Antenna API. It is divided according to the major API components, with subsections on:

- Qualifiers
- Primitives
- Return Codes

4.1 Qualifiers

Each primitive in the Smart Antenna API can be qualified by one or more of the qualifiers listed in this section. The following qualifiers are inherited from the Core API defined in the API Framework document.

Get	Primitive should support “get” operations.
Set	Primitive should support “set” operations.
Inc	Primitive should support “increment” operations.
Clr	Primitive should support “clear” operations.
Info	Used to retrieve capability information for variable primitives.
Isr	Tagged to signals running within an interrupt or high-priority thread.

In addition, the Smart Antenna API introduces the following qualifier.

Beam	Indicates that the primitive should be supported on a per beam basis, and allows the protocols to direct a particular operation to a specific beam. If no beam qualifier is specified, then the operation is applied to all beams. This qualifier is only valid for antennas that support multiple simultaneous directed (or omni-directional) beams.

4.2 Primitives

This identifies and describes the command, variable, response, and signal primitives that extend the Core API primitives for the Smart Antenna API, as well as summarizing and providing aliases[5] for the primitives inherited from the in the Core API.

4.2.1 Commands

The command primitives inherited from the Core API, along with their aliases, include:

Command, Alias	Rqmt	Qualifiers	Data
CmdReset, AntCmdReset	M		
CmdNativeConsole, AntCmdNativeConsole	O		User cmd string and response string.
CmdProcExec, AntCmdProcExec	O		Diagnostic, or other procedure, to exec.

The Smart Antenna API introduces no new commands.

4.2.2 Variables

The variable primitives inherited from the Core API, along with their aliases, include:

Variable, Alias	Rqmt	Qualifiers	Data
VarVersion, AntVarVersion	M	*get*	Version string (e.g., "2.0.1; 2 July 1998")
VarName, AntVarName	M	*get*	Name string given to lower module
VarSigEnabled, AntVarSigEnabled	H	*get/set*	Signal number to enable or disable
VarGroupSelect, AntVarGroupSelect	H	*set*	groupClassId, groupInstanceNum
VarGroupSettings, AntVarGroupSettings	H	*get (set)*	groupClass & instance, {var, value} pairs
VarGroupClassName, AntVarGroupClassName	H	*get*	Group class name string
VarGroupClassSize, AntVarGroupClassSize	H	*get*	Group class size (number of variables)
VarGroupClassInstances, AntVarGroupClassInstances	H	*get*	returns # of instances given groupClassId
VarGroupClassDefine, AntVarGroupClassDefine	O	*set*	groupClassId, name, size, instances, var list
VarGroupDefineNumMax, AntVarGroupDefineNumMax	O	*get*	# of dynamically-defined groups supported

The new variables introduced by this API are the following.

AntVarDirection Variable

Requirement: Mandatory

[5] Primitives in this Smart Antenna API can refer to inherited primitives either by the name provided in the Core API definition, or by the alias defined in this API.

Qualifiers: *Get/set/info/beam*
Data: For info: returns set of available directions
For get: returns current direction
: w/beam returns beam specific direction
For set: value for direction to set
: w/beam, direction to set for specific beam
NOTE – information may be supplied in degrees or angle, etc. In addition, values can be sets of tuples defining vectors (i.e., azimuth and elevation). The formatting used must be specified in the info set of values
Description: Get/set/query antenna direction

AntVarBand Variable

Requirement: Mandatory
Qualifiers: *Get/set/info/beam*
Data: For info: Returns set of available bands
For get: Returns current band (w/beam specific to beam)
For set: Value to set antenna or specific beam at
Description: Get/set/query antenna's operational band

AntVarMode Variable

Requirement: Mandatory
Qualifiers: *Get/set/info*
Data: For info: Returns set of mode values available
For get: Returns current mode of antenna
For set: Mode to set antenna into
Description: Get/set/query antenna's operational mode

AntVarBeamWidth Variable

Requirement: Highly desirable
Qualifiers: *Get/set/info/beam*
Data: For info: Returns set of available beam widths
(e.g., omni, directional, etc.)
For get: Returns current beamwidth for antenna
(or for a specific beam if given)
For set: value to set bandwidth at
Description: Get/set/query antenna beamwidth information

AntVarPolarization Variable

Requirement: Desirable
Qualifiers: *Get/set/info/beam*
Data: For info: Returns set of available polarization options
(e.g., Vertical, horizontal, circular, elliptical, etc.)
For get: Returns current polarization for antenna
(or for a specific beam if given)
For set: value to set for polarization

Description: Get/set/query antenna's polarization

AntVarBandwidth Variable

Requirement: Desirable
Qualifiers: *Get/set/info/beam*
Data: For info: Returns set of available bandwidths
For get: Returns current bandwidth for antenna
(or for a specific beam if given)
For set: value to set bandwidth to
Description: Get/set/query antenna's bandwidth values

AntVarFrequency Variable

Requirement: Optional
Qualifiers: *Get/set/info/beam*
Data: For info: Returns set of available frequencies
For get: Returns current center frequency for antenna
(or for a specific beam if given)
For set: value to set center frequency at
Description: Get/set/query antenna's center frequency

4.2.3 Signals

The signal primitives inherited from the Core API, along with their aliases, include:

Signal, Alias	Rqmt	Qualifiers	Data
SigAll, AntSigAll	M		
SigError, AntSigError	M	*isr*	Number indicating the error.
SigProcResults, AntSigProcResults	D	*isr*	Results of a CmdProcExec.

The Smart Antenna API introduces no new signals.

4.3 Return Codes

The Smart Antenna API inherits the return codes defined in the Core API, and introduces no new return code. However, aliases are defined in this API so that each of the Core API return codes can be referred to either by the name presented in that API definition, or by that name prefixed by "Ant." (For example, AntRetOk is identical to RetOk, and AntRetFail is identical to RetFail.)

4.4 Smart Antenna API Summary Tables

4.4.1 Smart Antenna API Primitives Summary

Table 2 summarizes the names, degree of requirement (M-Mandatory, H-Highly desirable, D-Desirable, O-Optional), qualifiers, and data for each of the logical primitives introduced by the Smart Antenna API (which extend those of the Core API).

Variables	**Rqmt**	**Qualifiers**	**Data**
AntVarDirection	M	*get/set/info/beam*	Direction value
AntVarBand	M	*Get/set/info/beam*	Band value
AntVarMode	M	*Get/set/info/beam*	Mode value
AntVarBeamWidth	H	*Get/set/info/beam*	Beam Width values
AntVarPolarization	D	*Get/set/info/beam*	Polarization values
AntVarBandwidth	D	*Get/set/info/beam*	Bandwidth value
AntVarFrequency	O	*Get/set/info/beam*	Center frequency value

5 Acknowledgments

The development of this document was initially supported by the Defense Advanced Research Projects Agency (DARPA) through the Global Mobile (GloMo) program's Wireless Internet Gateways (WINGS) project (contract no. DAAB07-95-C-D157), and by < *University of Texas – Pan American project*>**.** WINGS is a collaborative effort by the University of California, Santa Cruz and Rooftop Communications. This document has also benefited from the constructive review and feedback of the "GloMo Radio API Working Group."