CHILL Client-side Connection Protocols Specification

Created on October 11, 2007

Contributions by Jochen Deyke and Jim George

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to fully describe both the CHILL Archive Protocol and the CHILL Realtime Protocol from the perspective of the client-side. It will fully describe how to create connections with archive servers and realtime servers and how to interact with the servers once a connection is created. The document will contain full descriptions of all possible packets that may be sent or received; in addition, a list of all constants will be provided.

1.2 Conventions Used

All communication to and from servers is in network-byte order.

Data Types

Type	Number of bytes
short	2
int	4
unsigned int	4
long	8
float	4
UTF-8 String(#)	#

2 CHILL Connections

2.1 Control and Data Channels

Regardless of whether you are connecting to an archive server or to a realtime server, you will need a data channel. A data channel is just a connection with the server that receives all the various headers (see section 4) and the actual data.

To connect to an archive server, you will need a control channel in addition to a data channel; control channels must always be created before data channels. A control channel is just a connection with the archive server whose sole purpose is to issue various commands (see section 2.5) and receive various responses back in the form of Response Packets (see section 3.2).

2.1.1 Creating a Control Channel

To create a control channel connection, send the HELLO integer (see section 5.2) and then the ARCHIVE_CONTROL_CHANNEL integer (see section 5.2).

2.1.2 Creating a Data Channel

To create a data channel connection, send the HELLO integer (see section 5.2) and then the DATA_CHANNEL integer (see section 5.2). If connecting to an archive server, the DATA_CHANNEL integer will first need to be ORed with the session ID shifted left 16 bits; the session ID is received after creating a control channel.

2.2 Establishing a Realtime Connection

To establish a connection with a realtime server, create a data channel connection (see section 2.1.2). The server will then respond with one or more FIELD_TYPE_INFO headers (see section 4.3) on the data channel; servers are capable of providing up to 64 fields types.

2.3 Establishing an Archive Connection

To establish a connection with an archive server, first create a control channel connection (see section 2.1.1). Then send a Connect Command (see section 2.5.5) to the server on the newly created control channel; the server will respond back with a Response Packet (see section 3.2) that contains the new session ID in the extraInfo variable if the session creation was successful.

After the control channel connection is set up, you then need to establish a data channel connection (see section 2.1.2). Once a data channel is created, the server will then respond with one or more FIELD_TYPE_INFO headers (see section 4.3) on the newly created data channel.

2.4 Requesting Data

After establishing a connection, you will at that point have a list of all possible fields that you can request.

To actually request fields, send a long integer (64-bit) bitmask on the data channel that has the appropriate bits set for the fields desired. To determine which bit position in the bitmask corresponds to which field, look at the fieldNumber value in all FIELD_TYPE_INFO headers sent (see section 4.3).

After sending the bitmask, the data channel will then receive zero or more FIELD_TYPE_INFO headers (see section 4.3) corresponding to the fields that are actually available from the server. The server will also send a HOUSEKEEPING header (see section 4.4). The order in which the server sends all of the above headers is not guaranteed.

The server will then, for every ray, send a DATA header (see section 4.1) followed by the actual byte data. The data is arranged by gates (ie. the first gate of each field is sent before any second gate). The order of the fields in the byte data is specified by the fieldNumber value in the FIELD_TYPE_INFO headers (see section 4.3).

During this process of retrieving data, it is possible for any of the other various headers (see section 4) to arrive on the data channel. However, no headers will arrive in between a DATA header (see section 4.1) and the following byte data.

If, at any point, you want to request different fields or specify no fields at all, then simply change the long integer bitmask of desired fields and resend it to the server.

2.5 Archive Server Commands

The following are commands that can be sent on a control channel to an archive server to inform the server to perform a certain action. All commands are just Command Packets (see section 3.1) with particular values filled in for the various variables in the packet. Every command sent to the server will cause a corresponding Response Packet (see section 3.2) to be sent back.

2.5.1 Request Sweep

A Request Sweep Command is a Command Packet (see section 3.1) with the following values:

Name	Value
command	2 (request data for a sweep from a file)
subrequest	sweep number desired.
clientCode	0 (ignored)
majorRevision	0 (ignored)
minorRevision	0 (ignored)
unused	0 (Always 0)
inputString	”directory/file”

The server will respond back with the following Response Packet (see section 3.2):

Name	Value
status	256 (sending data) <u>Possible Errors:</u> 1 (error opening file) 2 (sweep requested was out-of-bounds) 9 (no sweeps in the file) 18 (bad command)
extraInfo	0
volumeNum	the first volume number (or -1)
sweepNum	the first sweep number (or -1)
rayNum	the first ray number (or 1)
scanMode	the scan mode of the sweep (or -1)
numSweeps	the number of sweeps available in the file (or 0)

The server will, at that point, start sending the data on the data channel (see section 2.4 for a description of how the data will arrive.

Finally, the server will send one more Response Packet (see section 3.2):

Name	Value
status	4 (end of volume) 5 (end of sweep) possibly bitwise ORed with 256 (sending data) 6 (end of file) Possible Errors: 3 (too many non-data blocks in file) 8 (error while trying to read the file) 22 (generic server failure)
extraInfo	0
volumeNum	the last volume number (or -1)
sweepNum	the last sweep number (or -1)
rayNum	the last ray number
scanMode	the scan mode of the sweep (or -1)
numSweeps	the number of sweeps available in the file (or 0)

2.5.2 Get File Details

A Get File Details Command is a Command Packet (see section 3.1) with the following values:

Name	Value
command	5 (get details on a file)
subrequest	0 (ignored)
clientCode	0 (ignored)
majorRevision	0 (ignored)
minorRevision	0 (ignored)
unused	0 (Always 0)
inputString	”directory/file”

The server will respond back with the following Response Packet (see section 3.2):

Name	Value
status	11 (file details) possibly bitwise ORed with 512 (calibration file) <u>Possible Errors:</u> 1 (error opening file) 9 (no sweeps in the file) 18 (bad command)
extraInfo	0
volumeNum	-1
sweepNum	-1
rayNum	-1
scanMode	-1
numSweeps	the number of sweeps available (or 1)

2.5.3 Halt Sweep

A Halt Sweep Command is a Command Packet (see section 3.1) with the following values:

Name	Value
command	6 (request server to stop sending the current sweep)
subrequest	0 (ignored)
clientCode	0 (ignored)
majorRevision	0 (ignored)
minorRevision	0 (ignored)
unused	0 (Always 0)
inputString	”directory/file”

The server will respond back with the following Response Packet (see section 3.2):

Name	Value
status	12 (stopped sending data) <u>Possible Errors:</u> 1 (error opening file) 9 (no sweeps in the file) 18 (bad command)
extraInfo	0
volumeNum	-1
sweepNum	-1
rayNum	-1
scanMode	-1
numSweeps	the number of sweeps available (or 1)

2.5.4 List Directory Contents

A List Directory Contents Command is a Command Packet (see section 3.1) with the following values:

Name	Value
command	8 (list directory contents)
subrequest	4 (list directory contents)
clientCode	0 (ignored)
majorRevision	0 (ignored)
minorRevision	0 (ignored)
unused	0 (Always 0)
inputString	”directory”

The server will respond back with the following Response Packet (see section 3.2):

Name	Value
status	14 (directory follows) Possible Errors: 18 (bad command)
extraInfo	length of directory listing
volumeNum	-1
sweepNum	-1
rayNum	-1
scanMode	-1
numSweeps	0

The directory contents will then be sent in the following format (names cannot contain spaces since spaces are used in the listing format to delimit information):

For files: "<file name>[<scan name>] <scan type>"
For directories: "<directory name> DIR"

Finally, the server will sent one more Response Packet (see section 3.2):

Name	Value
status	7 (directory sent)
extraInfo	0
volumeNum	-1
sweepNum	-1
rayNum	-1
scanMode	-1
numSweeps	0

2.5.5 Connect

A Connect Command is a Command Packet (see section 3.1) with the following values:

Name	Value
command	9 (initiate session with the server)
subrequest	0 (ignored)
clientCode	2 (Java VCHILL client)
majorRevision	the major revision of the client being used.
minorRevision	the minor revision of the client being used.
unused	0 (Always 0)
inputString	"username:password"

The server will respond back with the following Response Packet (see section 3.2):

Name	Value
status	16 (server ready) 14 (directory follows; this is the old message format) 21 (message follows) Possible Errors: 15 (server busy) 18 (bad command) 19 (bad username) 20 (bad password) 22 (generic server failure)
extraInfo	session ID
volumeNum	-1
sweepNum	-1
rayNum	-1
scanMode	-1
numSweeps	0

2.5.6 Disconnect

A Disconnect Command is a Command Packet (see section 3.1) with the following values:

Name	Value
command	10 (close session with the server)
subrequest	0 (ignored)
clientCode	0 (ignored)
majorRevision	0 (ignored)
minorRevision	0 (ignored)
unused	0 (Always 0)
inputString	””

No response is sent back from the server.

3 Control Channel Packets

3.1 Command Packet

Command Packets contain requests for the server to do something. For every Command Packet sent to the server, the client receives back a Response Packet (see section 3.2).

Command Packet contents		
Type	Name	Description
int	command	The command for the server to perform. 2 = Request data for a sweep from a file. 5 = Get details on a file. 6 = Request server to stop sending the current sweep. 8 = List directory contents. 9 = Initiate session with the server. 10 = Close session with the server.
short	subrequest	An additional code useful for some commands. 4 = List directory contents (only used with the list directory contents command). # = Sweep number (only used with request sweep data command).

Type	Name	Description
short	clientCode	The ID of the client being used. It is only used with the initiate session command; for all other commands, it is set to 0. 2 = Java VCHILL client (vs. old C client) (only used with initiate session command).
short	majorRevision	The major revision number of the client being used. It is only used with the initiate session command; for all other commands, it is set to 0.
short	minorRevision	The minor revision number of the client being used. It is only used with the initiate session command; for all other commands, it is set to 0.
int	unused	A reserved variable always set to 0.
UTF-8 String(100)	inputString	Contains various information that the server will need in order to perform the requested command. It can contain information like the username and password required for initiating a session or a directory name and/or file name for various other commands.

3.2 Response Packet

Response Packets contain various information regarding the status of commands issued through Command Packets (see section 3.1). For every Command Packet sent to the server, the client receives back a Response Packet.

Response Packet contents

Type	Name	Description
int	status	The status code of the command that this packet is in response to. 1 = Error opening file. 2 = Sweep number out of range. 3 = Too many non-data blocks in file. 4 = End of Volume. 5 = End of Sweep. 6 = End of File. 7 = Directory listing completed. 8 = Error while trying to read the file. 9 = No sweeps in the file. 11 = File details. 12 = Stopped sending data. 14 = Directory listing follows / old message format. 15 = Server busy. 16 = Server ready. 18 = Bad command. 19 = Bad username. 20 = Bad password. 21 = Message follows. 22 = Generic server failure. 256 = Sending data. 512 = Calibration file.
int	extraInfo	Additional information related to the command executed. When the command executed was an initiate session command, it contains the new session ID. When the command executed was a list directory contents command, it contains the number of bytes of the directory listing following this response packet.
int	volumeNum	The volume number (only used when the command executed was a request sweep data command); otherwise it is -1.
int	sweepNum	The sweep number (only used when the command executed was a request sweep data command); otherwise it is -1.
int	rayNum	The ray number of the first ray following (only used when the command executed was a request sweep data command); otherwise it is -1.

Type	Name	Description
int	scanMode	The scan mode (only used when the command executed was a request sweep data command); otherwise it is -1.
int	numSweeps	The number of sweeps (only used when the command executed was a get file details command or a request sweep data command); otherwise it is 0.

4 Data Channel Headers

All headers have common data, data specific to the particular header, and extra (most likely ignorable) data. The common header data comes first, followed by the specific header data (see the various header sections for details) and then finally the extra data is at the end.

The common header data includes:

Common Header contents		
Type	Name	Description
int	headerType	The type of the header being received.
int	headerLength	The length of the header including this common header and any extra data.

4.1 DATA

Data Headers contain various useful pieces of information regarding actual data that is about to be sent from the server; it corresponds to just a single ray.

Data Header contents			
Type	Name	Description	Units
long	requestedFields	Bitmask containing which fields were requested by the client.	
long	availableFields	Bitmask containing which fields are actually available from the server.	
int	startAz	Antenna azimuth position at the start of the integration cycle.	degrees
int	startEl	Antenna elevation position at the start of the integration cycle.	degrees
int	endAz	Antenna azimuth position at the end of the integration cycle.	degrees

Type	Name	Description	Units
int	endEl	Antenna elevation position at the end of the integration cycle.	degrees
int	numGates	The number of gates in the ray.	
int	startRange	The range to the first gate.	mm
unsigned int	dataTimeSecs	The time of the first transmit pulse in the ray	seconds since UNIX Epoch
int	dataTimeNSecs	Additional nanoseconds of the time of the first transmit pulse in the ray.	ns
int	rayNumber	The ray number of the ray within the sweep (counting starts at 1).	

4.2 EXTENDED_TRACKING

Extended Tracking Headers contain information useful for tracking the position of aircraft. It is a newer format than the Tracking Header (see section 4.10).

Extended Tracking Header contents

Type	Name	Description	Units
unsigned long	trackingTime	The time that the information in the header was gathered.	seconds since UNIX Epoch
float	posX	The vehicle position east of the radar.	km
float	poxY	The vehicle position north of the radar.	km
float	altitude	The vehicle altitude.	meters above ground level
float	heading	The vehicle heading if available.	degrees
UTF-8 String(32)	vehicleName	The name of the vehicle.	
UTF-8 String(32)	additionalInfo	Additional tracking information.	

4.3 FIELD_TYPE_INFO

Field Type Info Headers contain information about a particular field type that the server is capable of offering.

Field Type Info Header contents

Type	Name	Description
UTF-8 String(32)	fieldName	A short field name (ie. dBZ).
UTF-8 String(128)	fieldDescription	A description of the field type (ie. Reflectivity).
int	keyboardAccelerator	A keycode for a shortcut that the client can use for this field type.
UTF-8 String(32)	units	The units that the field type is specified in.
int	fieldNumber	The bit position of the field in the long integer sent to and from the server containing requested/available field types.
int	factor	actual data value =
int	scale	((value in byte data array) * scale + bias)
int	bias	/ factor.
int	maxFactorScaledValue	The maximum field value (multiplied by factor).
int	minFactorScaledValue	The minimum field value (multiplied by factor).
short	fieldDataFlags	Flags describing the data of the field type. Bit 0: set if the field has 8-bit signed data and unset otherwise. Currently, CHILL uses only unsigned 8-bit data. See the factor, scale, and bias variables for information about calculating actual values from the unsigned 8-bit data. Bits 1-3: reserved. Bit 4: set if the field's data may be unfolded.
short	colorMapType	An index specifying which color map to use since field types can be labeled differently among servers and the client can't know which field is say the Reflectivity field (it may be dBZ, Z, etc.).

4.4 HOUSEKEEPING

Housekeeping Headers contain various information that remains fairly constant in addition to information about sweeps. A housekeeping header is typically sent once per sweep.

Housekeeping Header contents

Type	Name	Description	Units
UTF-8 String(32)	radarID	The name/ID of the radar.	
int	radarLatitude	The latitude of the radar.	$degrees * 10^6$
int	radarLongitude	The longitude of the radar.	$degrees * 10^6$
int	radarAltitude	The radar altitude.	mm above mean sea level
int	antennaMode	The scan mode of the antenna. 1 and 4 = RHI scan modes. 0 and 3 = PPI scan modes.	
int	nyquistVel	The basic nyquist interval (2 * velocity range).	mm/sec
int	gateWidth	The gate spacing.	mm
int	pulses	The number of transmit pulses per integration cycle.	
int	polarizationMode	The polarization of the transmitters. 0 = Vertical only. 1 = Horizontal only. 2 = Alternating vertical and horizontal. 3 = Simultaneous vertical and horizontal.	
int	sweepNumber	The tilt/sweep sequence number (counting starts at 1 for each volume).	
int	saveSweep	The tilt/sweep number selected by the operator to be auto-saved as an image.	
int	angleScale	A value used to get the true angle from an integer angle. true angle = (integer angle * 360) / angleScale	degrees
unsigned int	sweepStartTime	The start time of the sweep.	seconds since UNIX Epoch

4.5 POWER_METERS_UPDATE

Power Meters Update Headers contain information regarding the power meters.

Power Meters Update Header contents

Type	Name	Description	Units
float	hPower	The horizontal peak power, assuming a rectangular pulse.	dBm
float	vPower	The vertical peak power, assuming a rectangular pulse.	dBm

4.6 PROCESSOR_INFO

Processor Info Headers contain information regarding the control and status of the signal processor.

Processor Info Header contents

Type	Name	Description	Units
int	polarizationMode	The polarization of the transmitters. 0 = Vertical only. 1 = Horizontal only. 2 = Alternating vertical and horizontal. 3 = Simultaneous vertical and horizontal.	
int	processingMode	Bitmask indicating the signal processing modes. Bit 0: Indexed Beam Mode. It uses the indexedBeamWidth and current antenna position to determine the integration cycle. Bit 1: Long Integration Mode. It decouples the signal processor's beam indexing from the transmitter controller's pulse numbering scheme, which allows for integration times that are much longer than the 256-hit limit imposed by the transmitter controller. Bit 2: Dual-PRT Mode. It makes use of the second PRT to enable staggered PRT. Bit 3: Phase Code Mode. Adds phase codes to the pulses sent from the radar to help detect 2nd-trip conditions.	
int	pulseType	The pulse type of the transmitters. 0 = 1 microsecond rectangular pulse. 1 = 200 nanosecond rectangular pulse. 2 = 1 microsecond gaussian-weighted pulse.	

Type	Name	Description	Units
int	testType	The radar calibration/test type used when performing calibration cycles. 0 = No test taking place. 1 = Clockwise calibration. 2 = Fixed sun-pointing. 3 = Scanning across the face of the sun. 4 = Noise source connected to the horizontal channel. 5 = Noise source connected to the vertical channel. 6 = Pointing at the blue-sky. 7 = Not a real test type. It indicates to compute nodes to save calibration parameters.	
int	integrationCyclePulses	The number of cycles integrated to give a single ray.	
int	clutterFilterNumber	The number of the clutter filter being used by the processor.	
int	rangeGateAveraging	The number of range gates to average.	
float	indexedBeamWidth	The beamwidth over which to integrate when in the indexed beam processing mode.	degrees
float	gateSpacing	The gate spacing (does not include effect of range averaging).	meters
float	prt	The PRT (Pulse Repetition Time).	μs
float	rangeStart	The range to start processing.	km
float	rangeStop	The range to stop processing.	km
int	maxGates	The number of gates for the digitizer to acquire.	
float	testPower	The power of the signal generator output when the test set is commanded to output 0 dBm.	dBm
float	unused	A reserved variable.	
float	unused	A reserved variable.	
float	testPulseRange	The range at which the test pulse is located.	km
float	testPulseLength	The length of the test pulse.	μs

4.7 RADAR_INFO

Radar Info Headers contain general radar information as well as fixed calibration terms.

Radar Info Header contents

Type	Name	Description	Units
UTF-8 String(32)	radarName	The name of the radar.	
float	radarLatitude	The current radar latitude.	degrees
float	radarLongitude	The current radar longitude.	degrees
float	radarAltitude	The current radar altitude.	meters
float	antennaBeamwidth	The antenna beamwidth.	degrees
float	radarWavelength	The wavelength of the radar.	cm
float	unused	A reserved variable.	
float	unused	A reserved variable.	
float	unused	A reserved variable.	
float	unused	A reserved variable.	
float	antennaHGain	The H polarization antenna gain from a reference point through the antenna.	dB
float	antennaVGain	The V polarization antenna gain from a reference point through the antenna.	dB
float	zdrCalBase	The operator-settable ZDR calibration base.	dB
float	phidpRotation	The operator-settable PhiDP rotation.	degrees
float	baseCalConstant	A base calibration constant that is used to calculate dBZ.	dB
float	firstGateOffset	The range offset to the first gate.	meters
float	powerHLoss	The loss on the horizontal channel from the reference point power to the power meter sensor.	dB
float	powerVLoss	The loss on the vertical channel from the reference point power to the power meter sensor.	dB
float	zdrVHSCalBase	The operator-settable ZDR calibration base when in simultaneous vertical and horizontal polarization mode	dB
float	testHPower	The power, on the H channel, into the directional coupler when the test set is commanded to output 0 dBm.	dB
float	testVPower	The power, on the V channel, into the directional coupler when the test set is commanded to output 0 dBm.	dB
float	dcHLoss	The directional coupler forward loss on the H channel.	dB

Type	Name	Description	Units
float	dcVLoss	The directional coupler forward loss on the V channel.	dB

4.8 SCAN_SEGMENT

Scan Segment Headers contain information that define a scan segment.

Scan Segment Header contents

Type	Name	Description	Units
float	manualAz	The manual azimuth position used for pointing and manual PPI/RHI modes.	degrees
float	manualEl	The manual elevation position used for pointing and manual PPI/RHI modes.	degrees
float	startAz	The starting azimuth angle; if it is greater than 360, then it implies we don't care what the starting azimuth is.	degrees
float	startEl	The starting elevation angle; if it is greater than 360, then it implies we don't care what the starting elevation is.	degrees
float	scanRate	The antenna scan rate.	degrees / sec
UTF-8 String(16)	segmentName	The name of the scan segment.	
float	rangeMax	A scan optimizer parameter indicating the maximum range.	km
float	heightMax	A scan optimizer parameter indicating the maximum height.	km
float	resolution	A scan optimizer parameter indicating the resolution.	meters
int	followMode	Indicates the object being followed (tracked) by the antenna. 0 = The radar isn't tracking anything. 1 = The radar is tracking the sun. 2 = The radar is tracking a vehicle.	

Type	Name	Description	Units
int	scanMode	The antenna scanning mode. 0 = PPI (Plan Position Indicator) mode. 1 = RHI (Range Height Indicator) mode. 2 = Fixed pointing angle mode. 3 = Manual PPI mode (elevation does not step automatically). 4 = Manual RHI mode (azimuth does not step automatically). 5 = Idle mode (radar is not scanning).	
int	scanFlags	Bitmask of flags describing the scan segment. Bit 0: Indicates this is the last segment in a volume. Bit 1: This is a sector scan. Bit 2: Radar initially turns clockwise. Bit 3: Radar initially turns counterclockwise. Bit 4: Record the time series data. Bit 5: RECORD_ENABLE_1 (unused) Bit 6: Record the processed data. Bit 7: RECORD_ENABLE_3 (unused)	
int	volumeNum	The volume number of the scan segment (increments linearly).	
int	segmentNum	The segment number within the current volume.	
int	timeLimit	The timeout for this scan, if nonzero.	seconds
int	saveSegment	Indicates which segment to auto-save, if nonzero.	
float	leftLimit	The left limit of a sector scan.	degrees
float	rightLimit	The right limit of a sector scan.	degrees
float	upLimit	The upper limit of a sector scan.	degrees
float	downLimit	The lower limit of a sector scan.	degrees
float	stepSize	The antenna step size used to increment azimuth (in RHI) and elevation (in RHI) if maxSegments is 0.	
int	maxSegments	The number of scan segments in this volume.	
int	clutterFilterBreakSegment	The segment at which the clutter filter switches from filter 1 to 2.	
int	clutterFilter1	The clutter filter number used for segments < clutterFilterBreakSegment.	
int	clutterFilter2	The clutter filter number used for segments >= clutterFilterBreakSegment.	

Type	Name	Description	Units
UTF-8 String(16)	projectName	The project name.	
float	currentFixedAngle	The current fixed angle (azimuth in RHI mode, elevation in PPI mode).	degrees

4.9 SWEEP_NOTICE

Sweep Notice Headers contain information about the status of a sweep/volume. They get sent out whenever a sweep or volume has either started or finished.

Sweep Notice Header contents

Type	Name	Description
int	flags	Bitmask of flags indicating the type of notice. Bit 0: End of Sweep. Bit 1: End of Volume. Bit 2: Start of Sweep.
int	cause	The reason the scan terminated. 0 = Scan completed normally. 1 = Scan has timed out. 2 = Timer caused this scan to abort. 3 = Operator issued an abort. 4 = Scan Controller detected an error. 5 = Communication fault with the antenna controller was recovered; restarting the scan.

4.10 TRACKING

Tracking Headers contain information useful for tracking the position of aircraft. It is an older format than the Extended Tracking Header (see section 4.2).

Tracking Header contents

Type	Name	Description	Units
float	posX	The vehicle position east of the radar.	km
float	posY	The vehicle position north of the radar.	km
float	altitude	The altitude above ground level.	km
unsigned int	trackingTime	The time that the information in the header was gathered.	seconds since UNIX Epoch

Type	Name	Description	Units
UTF-8 String(16)	vehicleName	The name of the vehicle being tracked.	

4.11 TRANSMITTER_INFO

Transmitter Info Headers contain information about the transmitters.

Transmitter Info Header contents

Type	Name	Description	Units
int	transmittersEnabled	Bitmask containing which transmitters are enabled. Bit 0: Horizontal transmitter enabled. Bit 1: Vertical transmitter enabled.	
int	polarizationMode	The polarization of the transmitters. 0 = Vertical only. 1 = Horizontal only. 2 = Alternating vertical and horizontal. 3 = Simultaneous vertical and horizontal.	
int	pulseType	The pulse type of the transmitters. 0 = 1 microsecond rectangular pulse. 1 = 200 nanosecond rectangular pulse. 2 = 1 microsecond gaussian-weighted pulse.	
float	prt	The PRT (Pulse Repetition Time).	μs
float	prt2	The second PRT for use in Dual-PRT mode.	μs

5 Constants

5.1 Header IDs

Header	ID
Data	0x9090
Extended Tracking	0x9494
Field Type Info	0x9292
Housekeeping	0x9191
Power Meters Update	0x5aa50004
Processor Info	0x5aa50003
Radar Info	0x5aa50001
Scan Segment	0x5aa50002
Sweep Notice	0x5aa50005
Tracking	0x9393
Transmitter Info	0x5aa50008

5.2 General Constants

number of bytes	name		value
4	HELLO	=	0xF0F00F0F
4	ARCHIVE_CONTROL_CHANNEL	=	12
4	DATA_CHANNEL	=	15