

# APRX Software Requirement Specification

## Table of Contents

1	APRX Software Requirement Specification.....	2
1.1	Purpose:.....	2
1.2	Usage Environments:.....	3
2	Treatment rules:.....	4
2.1	Basic IGate rules:.....	4
2.2	Low-Level Transmission Rules:.....	5
2.3	Low-Level Receiving Rules:.....	6
2.4	Additional Tx-IGate rules:.....	6
2.5	Digipeater Rules:.....	7
2.6	Radio Interface Statistics Telemetry.....	8
2.7	Individual Call-Signs for Each Receiver, or Not?.....	9
2.8	Beaconing.....	10
2.8.1	Radio Beaconing.....	10
2.8.2	Network beaconing.....	10

## 1 APRX Software Requirement Specification

This is *Requirement Specification* for a software serving in Amateur Radio APRS service.

Reader is assumed to be proficient with used terminology, and they are not usually explained here.

### 1.1 Purpose:

This describes algorithmic, IO-, and environmental requirements for a software doing any combination of following four tasks related to APRS service:

1. Listen on messages with a radio, and pass them to APRSIS network service
2. Listen on messages with a radio, and selectively re-send them on radio
3. Listen on messages with a radio, and selectively re-send them on radios on other frequencies
4. Receive messages from APRSIS network, and after selective filtering, send some of them on radio

Existing *aprx* software implements Receive-Only (Rx) IGate functionality, and the purpose of this paper is to map new things that it will need for extending functionality further.

23

## 24 **1.2 Usage Environments:**

25 The *aprx* software can be used in several kinds of environments to handle multiple tasks  
26 associated with local APRS network infrastructure tasks.

27 On following one should remember that amateur radio transmitters need a licensed  
28 owner/operator/license themselves, but receivers do not:

- 29 1. License-free Receive-Only (RX) IGate, to add more “ears” to hear packets, and to  
30 pipe them to APRSIS
- 31 2. Licensed bidirectional IGate, selectively passing messages from radio channels to  
32 APRSIS, and from APRSIS to radio channels, but not repeating packets heard on a  
33 radio channel back to a radio channel.
- 34 3. Licensed bidirectional IGate plus selectively re-sending of packets heard on radio  
35 channels back to radio channels
- 36 4. Licensed system for selectively re-sending of packets heard on radio channels back  
37 to other radio channels, and this without bidirectional IGate service.
- 38 5. Licensed system for selectively re-sending of packets heard on radio channels back  
39 to radio channels, and doing with with “receive only” IGate, so passing information  
40 heard on radio channel to APRSIS, and not the other way at all.

41

42 In more common case, there is single radio and single TNC attached to digipeating (re-  
43 sending), in more challenging cases there are multiple receivers all around, and very few  
44 transmitters. Truly challenging systems operate on multiple radio channels. As single-  
45 TNC and single-radio systems are just simple special cases of these complex systems,  
46 and for the purpose of this software requirements we consider the complex ones:

- 47 1. 3 different frequencies in use, traffic is being relayed in between them, and the  
48 APRSIS network.
- 49 2. On each frequency there are multiple receivers, and one well placed transmitter.
- 50 3. Relaying from one frequency to other frequency may end up having different rules,  
51 than when re-sending on same frequency: Incoming packet retains traced paths,  
52 and gets WIDEN-N/TRACEN-N requests replaced with whatever sysop wants.

53

54

## 2 Treatment rules:

Generally: All receivers report what they hear straight to APRSIS, after small amount of filtering of junk messages, and things which explicitly state that they should not be sent to APRSIS.

### 2.1 Basic IGate rules:

General rules for these receiving filters are described here:

<http://www.aprs-is.net/IGateDetails.aspx>

Gate all packets heard on RF to the Internet (Rx-IGate) EXCEPT

1. 3<sup>rd</sup> party packets (data type '}' ) should have all before and including the data type stripped and then the packet should be processed again starting with step 1 again. There are cases like D-STAR gateway to APRS of D-STAR associated operator (radio) positions.
2. generic queries (data type '?' ).
3. packets with TCPIP, TCPXX, NOGATE, or RFONLY in the header, especially in those opened up from a 3<sup>rd</sup> party packets.

Gate message packets and associated posits to RF (Tx-IGate) if

1. the receiving station has been heard within range within a predefined time period (range defined as digi hops, distance, or both).
2. the sending station has not been heard via RF within a predefined time period (packets gated from the Internet by other stations are excluded from this test).
3. the sending station does not have TCPXX, NOGATE, or RFONLY in the header.
4. the receiving station has not been heard via the Internet within a predefined time period.

A station is said to be heard via the Internet if packets from the station contain TCPIP\* or TCPXX\* in the header or if gated (3<sup>rd</sup> party) packets are seen on RF gated by the station and containing TCPIP or TCPXX in the 3<sup>rd</sup> party header (in other words, the station is seen on RF as being an IGate).

Gate all packets to RF based on criteria set by the sysop (such as call sign, object name, etc.).

With more advanced looking inside frames to be relayed, both the digipeater and Tx-IGate can use filtering rules, like “packet reports a position that is within my service area.”

92

93 From multiple receivers + single (or fewer) transmitter(s) follows, than when a more usual  
94 system does not hear what it sent out itself, this one will hear, and its receivers must have  
95 a way to ignore a frame it sent out itself a moment ago.

96 Without explicit “ignore what I just sent” filtering, an APRS packet will get reported twice to  
97 APRSIS:

98 rx  $\Rightarrow$  igate-to-aprsis + digi  $\Rightarrow$  tx  $\Rightarrow$  rx  $\Rightarrow$  igate-to-aprsis + digi (dupe filter stops)

99 Digipeating will use common packet duplication testing to sent similar frame out only once  
100 per given time interval (normally 30 seconds.)

101

## 102 **2.2 Low-Level Transmission Rules:**

103 These rules control repeated transmissions of data that was sent a moment ago, and other  
104 basic transmitter control issues, like persistence. In particular the persistence is fine  
105 example of how to efficiently use radio channel, by sending multiple small frames in quick  
106 succession with same preamble and then be silent for longer time.

107 1. Duplication detector per transmitter: Digipeater and Tx-IGate will ignore packets  
108 finding a hit in this subsystem.

109 2. A candidate packet is then subjected to a number of filters, and if approved for it,  
110 the packet will be put on duplicate packet detection database (one for each  
111 transmitter.) See Digipeater Rules, below.

112 3. Because the system will hear the packets it sends out itself, there must be a global  
113 expiring storage for recently sent packets, which the receivers can then compare  
114 against. (Around 100 packets of 80-120 bytes each.) This storage gets a full copy  
115 of packet being sent out – a full AX.25 frame.

116 Also, transmitters should be kept in limited leash: Transmission queue is less than T  
117 seconds ( < 5 ? ), which needs some smart scheduling coding, when link from computer to  
118 TNC is considerably faster.

119 Original KISS interface is defined as “best effort”: if TNC is busy while host sends a frame,  
120 the frame will be discarded, and “upper layers” will resend. In APRS Digipeating, the  
121 upper layer sends the packet once, and then declares circa 30 second moratorium on  
122 packets with same payload.

123

124

## 125 **2.3 Low-Level Receiving Rules:**

- 126 1. Received AX.25 packet is compared against “my freshly sent packets” storage, and  
127 matched ones are dropped. (Case of one/few transmitters, and multiple receivers  
128 hearing them.)
  - 129 2. Received packet is validated against AX.25 basic structure, invalid ones are  
130 dropped.
  - 131 3. Received packet is validated against Rx-IGate rules, forbidden ones are dropped  
132 (like when a VIA-field contains invalid data.)
  - 133 4. Packet may be rejected for Rx-IGate, but it may still be valid for digipeating!  
134 For example a 3<sup>rd</sup> party frame is OK to digipeat, but not to Rx-IGate to APRSIS!  
135 Also some D-STAR to APRS gateways output 3<sup>rd</sup> party frames, while the original  
136 frame is quite close to an APRS frame.
- 137 Divide packet rejection filters to common, and destination specific ones.

## 138 **2.4 Additional Tx-IGate rules:**

139 The Tx-IGate can have additional rules for control:

- 140 1. Multiple filters look inside the message, and can enforce a rule of “repeat only  
141 packets within my service area,” or to “limit passing message responses only to  
142 destinations within my service area”
  - 143 2. Basic gate filtering rules:
    - 144 1. the receiving station has been heard within range within a predefined time  
145 period (range defined as digi hops, distance, or both).
    - 146 2. the sending station has not been heard via RF within a predefined time period  
147 (packets gated from the Internet by other stations are excluded from this test).
    - 148 3. the sending station does not have TCPXX, NOGATE, or RFONLY in the header.
    - 149 4. the receiving station has not been heard via the Internet within a predefined time  
150 period.
- 151 A station is said to be heard via the Internet if packets from the station contain  
152 TCPIP\* or TCPXX\* in the header or if gated (3rd-party) packets are seen on RF  
153 gated by the station and containing TCPIP or TCPXX in the 3rd-party header (in  
154 other words, the station is seen on RF as being an IGate).
- 155 3. Optionally wait a few seconds (like a random number of seconds in range of 1 to 5  
156 seconds) before letting received packet out. This permits other systems to be faster  
157 than the Tx-IGate system, and thus to get their voice.

158

159

160

## 161 **2.5 Digipeater Rules:**

162 Digipeater will do following for each transmitter:

- 163 1. Compare candidate packet against duplicate filter, if found, then drop it. (Low-level  
164 transmission rules, number 1)
- 165 2. Count number of hops the message has so far done, and...
- 166 3. Figure out the number of hops the message has been requested to do  
167 (e.g. "OH2XYZ-1>APRS,OH2RDU\*,WIDE7-5: ..." will report that there was request  
168 of 7 hops, so far 2 have been executed – one is shown on trace path.)
- 169 4. If either of previous ones are over any of configured limits, the packet is dropped.
- 170 5. FIXME: WIDEn-N/TRACEn-N treatment rules: By default treat both as TRACE,  
171 have an option to disable TRACE mode. Possibly additional keywords for cross-  
172 band digipeating?
- 173 6. Multiple filters look inside the message, and can enforce a rule of "repeat only  
174 packets within my service area."
- 175 7. FIXME: Cross frequency digipeating? Treat much like Tx-IGate?  
176 Relaying from one frequency to other frequency may end up having different rules,  
177 than when re-sending on same frequency: Incoming packet retains traced paths,  
178 and gets WIDEn-N/TRACEn-N requests replaced with whatever sysop wants.
- 179 8. Cross band relaying may need to add both an indication of "received on 2m", and  
180 transmitter identifier: "sent on 6m":  
181 "OH2XYZ-1>APRS,RX2M\*,OH2RDK-6\*,WIDE3-2: ..."  
182  
183 This "source indication" may not have anything to do with real receiver identifier,  
184 which is always shown on packets passed to APRSIS.

185

186 The MIC-e has a weird way to define same thing as normal packets do with  
187 SRCCALL-n>DEST,WIDE2-2: ...

188 The MIC-e way (on specification, practically nobody implements it) is:  
189 SRCCALL-n>DEST-2: ...

190

191

192

## 193 **2.6 Duplicate Detector**

194 Normal digipeater duplicate packet detection compares message source, destination  
195 (without SSID!), and payload data against another in self-expiring storage called “duplicate  
196 detector”. Lifetime of this storage is commonly considered to be 30 seconds.

197 Practically the packet being compared at Duplicate Detector will be terminated at first CR  
198 or LF in the packet, and if there is a space character preceding the line end, also that is  
199 ignored when calculating duplication match. **However: The Space Characters are sent,**  
200 **if any are received, also when at the end of the packet!** (Some TNC:s have added one  
201 or two extra space characters on packets they digipeat...)

202

203

204

## 205 **2.7 Radio Interface Statistics Telemetry**

206 Current *aprx* software offers telemetry data on radio interfaces. It consists of following four  
207 things. Telemetry is reported to APRS-IS every 10 minutes:

- 208 1. Channel occupancy average in Erlangs over 1 minute interval, and presented as  
209 busiest 1 minute within the report interval. Where real measure of carrier presence  
210 on radio channel is not available, the value is derived from number of received  
211 AX.25 frame bytes plus a fixed Stetson-Harrison constant added per each packet  
212 for overheads. That is then divided by presumed channel modulation speed, and  
213 thus derived a figure somewhere in between 0.0 and 1.0.
- 214 2. Channel occupancy average in Erlangs over 10 minute interval. Same data source  
215 as above.
- 216 3. Count of received packets over 10 minutes.
- 217 4. Count of packets dropped for some reason during that 10 minute period.

218 Additional telemetry data points could be:

- 219 1. Number of transmitted packets over 10 minute interval
- 220 2. Number of packets IGate:d from APRSIS over 10 minute interval
- 221 3. Number of packets digipeated for this radio interface over 10 minute interval
- 222 4. Erlang calculations could include both Rx and Tx, but could also be separate.

223

224



225

## 226 **2.8 Individual Call-Signs for Each Receiver, or Not?**

227 Opinions are mixed on the question of having separate call-signs for each receiver (and  
228 transmitter), or not. Even the idea to use all 16 available SSIDs for a call-sign for  
229 something does get some opposition.

- 230 • There is no license fee in most countries for receivers, and there is no need to limit  
231 used call signs only on those used for the site transmitters.
- 232 • There is apparently some format rule on APRSIS about what a “call-sign” can be,  
233 but it is rather lax: 6 alphanumerics + optional tail of: “-” (minus sign) and one or two  
234 alphanumerics. For example OH2XYZ-R1 style call-sign can have 36 different  
235 values before running out of variations on last character alone (A to Z, 0 to 9.)
- 236 • Transmitter call-signs are important, and there valid AX.25 format call-signs are  
237 mandatory.

238

239 Transmitters should have positional beacons for them sent on correct position, and  
240 auxiliary elements like receivers could have their positions either real (when elsewhere), or  
241 actually placed near the primary Tx location so that they are separate on close enough  
242 zoomed map plot.

243 Using individual receiver identities (and associated net-beaconed positions near the real  
244 location) can give an idea of where the packet was heard, and possibly on which band. At  
245 least the *aprs.fi* is able to show the path along which the position was heard.

246

247

## 248 **2.9 Beacons**

249 Smallest time interval available to position viewing at aprs.fi site is 15 minutes. A beacon  
250 interval longer than that will at times disappear from that view. Default view interval is 60  
251 minutes.

252 Beacon transmission time **must not** be manually configured to fixed exact minute. There  
253 are large peaks in APRSIS traffic because of people are beaconing out every 5 minutes,  
254 and every 10 minutes, at exact 5/10 minutes. Beaconing algorithm must be able to  
255 spread the requests over the entire cycle time (10 to 30 minutes) evenly. Even altering  
256 the total cycle time by up to 10% at random at the start of each cycle should be considered  
257 (and associated re-scheduling of all beacon events at every cycle start). All this to avoid  
258 multiple non-coordinated systems running at same rhythm.

259 The Bresenham's line plotting algorithm can be used to find smooth integer time intervals –  
260 or the programmer can resort to floating point. Related algorithm is known as “Digital  
261 Differential Analyser”. Both can be implemented using integer arithmetic, which may be of  
262 interest on some cases.

263 Beaconing at quicker repetition rates is also possible by inserting same item multiple times  
264 into the cycle.

265 Receiver location beacons need only to be on APRSIS, transmitter locations could be also  
266 on radio.

### 267 **2.9.1 Radio Beacons**

268 “Tactical situation awareness” beaconing frequency could be 5-10 minutes, WB4APR does  
269 suggest 10 minutes interval. Actively moving systems will send positions more often.  
270 Transmit time spread algorithm must be used.

271 Minimum interval of beacon transmissions to radio should be 30 seconds. If more  
272 beacons need to be sent in this time period, use of Persistence parameter on TNCs (and  
273 KISS) should help: Send the beacons one after the other (up to 3?) during same  
274 transmitter activation, and without prolonged buffer times in between them. That is  
275 especially suitable for beacons *without* any sort of distribution lists.

276

### 277 **2.9.2 Network beaconing**

278 Network beaconing cycle time can be up to 30 minutes.

279 Network beaconing can also transmit positions and objects at much higher rate, than radio  
280 beaconing. Transmit time spread algorithm must be used.

281 Beacons could also be bursting similar to radio beacon Persistence – within a reason.

282

283