D. RTD Formats

D.1 Introduction

IRIS can output a real-time display data stream using UDP data packets. There are several formats provided, and the customer can also write their own. The source code for these transmitter programs is in ${IRIS_ROOT}/libs/rtq. The formats are documented in detail in the sig_rtdisp.h file.

Any computer running IRIS on that network can receive and process (display) the real time display data. One nice factor about the real time display is that it is completely scalable. If the IRIS Radar server is sending real time display data, the load induced on the network, and the servers CPU is constant no matter if none, one or a hundred machines are on that network receiving and processing the real time display data. This is made possible by the fact that the real time display is done in broadcast mode.

It should also be noted that UDP data does not get retransmitted upon packet loss, and at the application level in IRIS, there is no attempt done at detected packet losses nor asking for retransmissions. If such a loss occurs at the transmitter, on the network, or at one or more receiving hosts, then a gap would occur in the picture being drawn on the destination real time displays. This of course differs from IRIS product transmission which are done point-to-point via TCP and are reliable due to error checking and retransmissions.

The load induced on the network by a real time display server broadcasting is a function of the following: Number of moments being broadcast (up to 3 – Z, V and W), the number of bins per radial and the scan rate of the antenna. The total number of bins (combined for all moments) is approximately 1400 per radial. An overhead of about 100 bytes per ray exists. Each bin requires one byte of data. Thus this a scan rate of 5 RPM (30 radials per second), requires about 1500 * 30 = 45000 bytes per second. One must remember that the real time display bandwidth must be added to any other IRIS and other bandwidth being occupied on your network. Therefore, running the real time display on a remote radar connected via a 56K baud link would not be practical.

The real time display data stream can be configured to transmit using up to 6 different formats, IP addresses, and ports. See the **setup** utility documentation in the *SIGMET Utilities Manual*. Normally gateways, such as multi-homed hosts, bridges and routers will NOT pass IRIS real time display data. This is because such devices are built to filter broadcast data. This is the mechanism that keeps the real time display data on a single network. However, it is usually possible to configure such devices to pass broadcast data that is occurring on a specific port (like the real time display port). This may be convenient in some installations.

To receive real time display data, the receive process needs to issue a socket and a bind system call. Following this, the receiver can read messages from the socket as they are transmitted.

D.2 Rtd_v1_xmt

There are three different messages transmitted as part of the real time display data stream. The messages are distinguished by the the first two bytes (SHORT) in the message summarized in the following table (see definitions in sig_rtdisp.h):

RTRAY_TYPE_HEADER
RTRAY_TYPE_RAY
RTRAY_TYPE_END

The message RTRAY_TYPE_HEADER consists of the two byte ID followed by a struct rtd_v1_vol_header. This message is sent either at the beginning of a new elevation sweep, or periodically if a new sweep has not started recently.

The message RTRAY_TYPE_RAY begins with the two byte ID. What this is followed by depends on the type of data being transmitted by the real time display sender. This is chosen in the senders **setup** utility and can consists of 1, 2 or 3 data types being Z, V or W. The volume header indicates which data types are currently being transmitted and in which order the data types are being presented. For example, if only Z and V are being transmitted, the RTRAY_TYPE_RAY message will consist of its two byte message ID, followed by a struct rtd_v1_ray_header, followed by the data elements for Z, followed by the V elements. The Z and V elements are presented as one byte per range bin. The scaling of the elements is defined in section 3.3. For example, if the volume header indicates that 200 bins of each type are being transmitted, then the Z elements (and the V elements) would each be 200 bytes long each, with each byte representing one range bin. The spacing between the range bins is also defined by the volume header.

D.3 Rtd_v2_xmt

This format is designed to support the full number of range bins, almost the full number of data types, and both 8-bit and 16-bit data formats. As a result each radial must be split into potentially many individual 1500 byte UDP packets. Similar to rtd_v1_xmt, the volume header rtd_v2_vol_header is sent periodically. Every 25 rays, or when there is a change. Otherwise the rtd_v2_ray_header is sent followed by data.

D.4 Rtd_nids3_xmt

This format is designed to support a legacy data format used by Radtec. All packets are the same format, with minimal header information. Data can be either 8-bit or 4-bit format. There is a config file rtd_nids3_xmt.conf which controls some details.