Description of the new Product Display and Weather Tracking enhancements to the NWRT PAR software

Fall 2011

By David Priegnitz (CIMMS/NSSL)

As part of the fall 2011 software release for the NWRT PAR, two new capabilities have been added which are described in this document. The first is a product display enhancement, allowing an operator to view, at the RCI client, a reflectivity product for the lowest elevation cut. The second is a weather feature tracking algorithm, which commands the pedestal to move to a new position when the weather feature gets close to a scan sector boundary. The weather tracking algorithm is dependent on the product display enhancement.

PRODGEN

The new product display function, called "PRODGEN", required enhancements to the EP, RCI server, and RCI client components. It was developed to support the weather tracking algorithm but also provides NWRT PAR operators the capability of monitoring data collection without having to use WDSS-II as a display platform. This is especially important to those wanting to operate the radar from home.

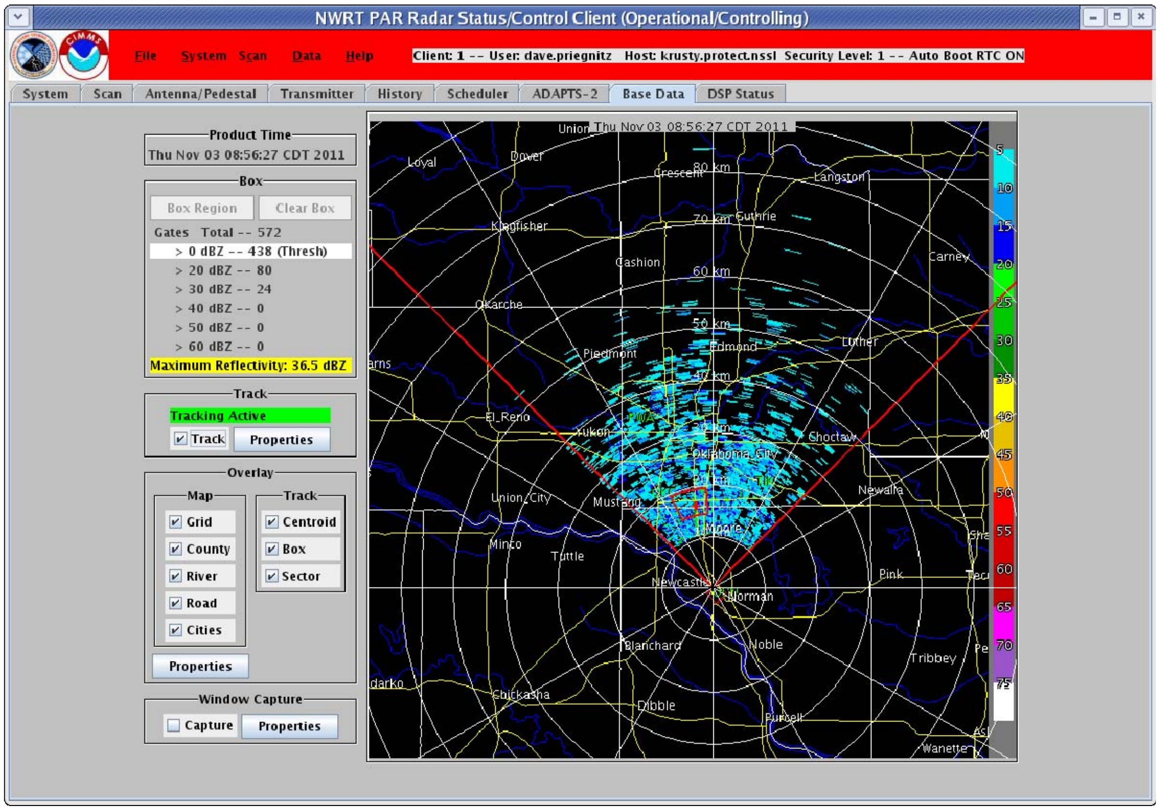
The first iteration of PRODGEN creates a single base reflectivity product which is required to support the weather tracking algorithm. It was kept simple at this time to minimize the work necessary for the fall 2011 release. Supporting multiple moments and elevations would have required considerable enhancements to the RCI server and client software which will most likely be addressed in future releases.

Inputs and Outputs

PRODGEN is run as a task on par4. It reads moment data from the censored moments LB and writes a prodgen_product message to the ep_comm LB. A new product is started when the start-of-volume flag is set in an input radial and completed on either an elevation angle change or when the end-of-volume flag is set. To minimize bandwidth issues when transferring products across the network, product data are compressed prior to being written to the output LB. The EP_COMM task reads the product from the LB and writes it to the RCI server/EP socket. The RCI server reads the prodgen_product message from the RCI server/EP socket and writes it to a file. When requested by an RCI client, the RCI server sends the message verbatim to the RCI client. Upon receipt, the RCI client uncompresses the product data contained in the prodgen_product message and displays it.

Display

When the Base Data tab is selected at the RCI client, the product display is automatically updated when a new product is received by the RCI server. The RCI client polls the RCI server for the time of the most recent product. When the product time changes the RCI client sends a product request to the RCI server. A sample Base Data window is presented in Figure 1.


NWRT PAR Radar Status/Control Client (Operational/Controlling)
File System Scan Data Help
System Scan Antenna/Pedestal Transmitter History Scheduler ADAPTS-2 Base Data DSP Status
Box Region Clear Box
Track Properties
Map Track
Grid County River Road Cities
Centroid Box Sector
Properties
Window Capture
Capture Properties
Loyal Dover Crescent Langston Kingfisher Guthrie Carney Cashion Okarche Piedmont Edmond Luther El Reno Yukon Choctaw Oklahoma City Union City Mustang Moore Newalla Minco Tuttle Newcastle Norman Pink Blanchard Noble Tribbey Chickasha Dibble Purcell Wanette
80 km 70 km 60 km 50 km 40 km
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75

Client: 1 -- User: dave.priegnitz Host: krusty.protect.nssl Security Level: 1 -- Auto Boot RTC ON

Product Time
Thu Nov 03 08:56:27 CDT 2011

Box
Gates Total -- 572
> 0 dBZ -- 438 (Thresh)
> 20 dBZ -- 80
> 30 dBZ -- 24
> 40 dBZ -- 0
> 50 dBZ -- 0
> 60 dBZ -- 0
Maximum Reflectivity: 36.5 dBZ

Track
Tracking Active
☑ Track

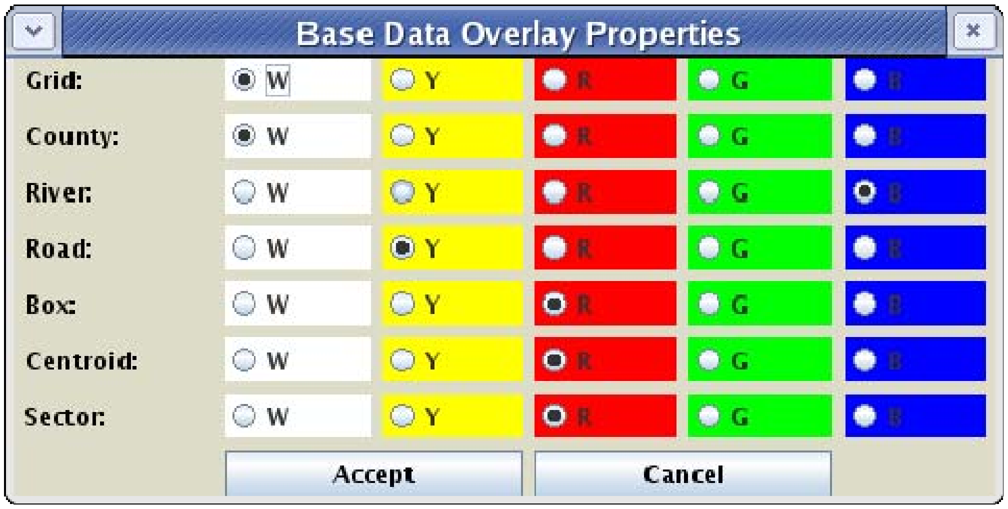
Overlay
☑ Grid ☑ County ☑ River ☑ Road ☑ Cities
☑ Centroid ☑ Box ☑ Sector
☐ Capture

Thu Nov 03 08:56:27 CDT 2011

Figure 2: Sample Base Data window

Overlay Data

The Base Data window provides a group of selections to control the overlay of various information. These include: Grid, County, River, Road, Cities, Centroid, Box and Sector. The Grid overlay selection toggles a polar grid over the reflectivity display. Spokes are drawn in 30 degree intervals. The number and spacing of range rings are dependent on the display magnification. The County, River, Road, and City selections provide geopolitical reference information. The specific Cities information that is displayed is dependent on the display magnification. The Centroid, Box and Sector selections are used to display information used/generated by the weather tracking algorithm. The Overlay Properties selection provides a menu to define overlay colors. A sample Base Data Overlay Properties menu is presented in Figure 2.



Grid:	[✓ W]	Y	R	G	B
County:	[✓ W]	Y	R	G	B
River:	W	Y	R	G	[✓ B]
Road:	W	[✓ Y]	R	G	B
Box:	W	Y	[✓ R]	G	B
Centroid:	W	Y	[✓ R]	G	B
Sector:	W	Y	[✓ R]	G	B

Figure 3: Sample Base Data Overlay Properties window

The colors for Cities overlay are defined in the data file that contains the city label and lat/lon coordinate (“nwrt_overlay_data”). There are also special lines and symbols that can also be defined in that file. Refer to the documentation at the beginning of the file for more information on how to edit/add/delete information.

Window Capture

The Base Data window also provides the capability to capture the contents of the data display region and save them as “png” format files. The Properties selection allows one to define a base directory to store the files. The file names match the product time label. If enabled, new files are created whenever a new reflectivity product is displayed. NOTE: The Base Data window must be visible in order to capture the data.

Mouse Actions

The display magnification and window position are controlled by the mouse buttons. Pressing the left mouse button moves the press coordinate to the window center. Pressing the middle mouse button moves the press coordinate to the window center and increases the zoom factor by 2. Pressing the right mouse button moves the press coordinate to the window center and decreases the zoom factor by 2. In a nutshell, this is how the buttons are defined:

Left Button	– Position
Middle Button	– Position and increase magnification
Right Button	– Position and decrease magnification

The Box and Track objects are tied to the weather feature tracking algorithm and are discussed in the next section.

Log File

When the PRODGEN task is running, certain textual output is written to the standard log error linear buffer (this is done for all tasks running at the DSP). Although the intended user of this file is the software developer/maintainer, it contains information about the task performance that may be useful to other users. A sample log for one scan iteration is shown below.

```
15:39:41    prodgen: Beginning of volume received
15:39:47    prodgen: Compressed 189954 product bytes down to 6430 bytes
15:39:47    prodgen: Write product for elevation 0.505371 with 109 of 109 radials from 314.829712 to 44.791260
15:39:47    prodgen: ===>Number of azimuths: 109
15:39:47    prodgen: ===>Number of gates:    1738
15:39:47    prodgen: ===>Size of product:    6446 bytes
15:39:47    prodgen: ===>Moment name:        Reflectivity
15:39:47    prodgen: Product message written to %RCI_COMM lb
15:40:34    prodgen: End of volume
15:40:34    prodgen: Write product for last elevation 52.899170 in volume with 109 radials, last elev 52.899170
```

This information is repeated for each volume scan. Please note that this file is currently recreated when the EP is restarted, so, if you are interested in preserving this information for future reference you will need to log into par4 and make a copy of it before restarting the EP (we are currently considering options to preserve these data so they are archived with the I/Q and moment data sets). The file name is "/home/nwrt/data/logs/prodgen.log". Also note that access to par4 is restricted, so, if you want a copy of this file preserved you will need to contact one of the software/test engineers.

WXTRACK

The new weather tracking algorithm, "WXTRACK", required enhancements to the DSP, RCI server, and RCI client components. The algorithm uses the reflectivity values of gates bounded by a user specified box to compute a centroid. In subsequent scans, the algorithm computes a new centroid position and adjusts the box to preserve the original centroid position relative to the box. When the box gets close enough to the edge of the scan sector, the algorithm determines an optimal antenna position for the next scan, builds a pedestal position message, and sends it to the RCI server. The RCI server, in turn, sends the message to the Real Time Controller (RTC) where it is executed before the next scan. Selections in the RCI client Base Data window are used to define the box surrounding the weather feature and algorithm properties. These are described later in this section.

The user defined box is kept in polar coordinates to match the format of the radar data. This makes the determination of which radials are inside or outside the box simple. When a new scan is detected by WXTRACK, each radial is checked to see if it is inside or outside the box. If outside, the radial is skipped. If inside, the reflectivity for each gate in the radial are stored in a temporary array.

Inputs and Outputs

WXTRACK runs as a task on par4. It reads moment data from the censored moments LB and *wxtrack_control* messages from the ep_comm LB. It writes *wxtrack_status* and *rtc_pedestal_control* messages to the rtc_comm LB. The EP_COMM task reads the messages in the rtc_comm LB and writes them to the RCI server/EP socket. The RCI server reads the messages from the RCI server/EP socket and processes them accordingly. The *wxtrack_status* messages are sent to RCI clients upon request. The *rtc_pedestal_control* message is written to the RCI server/RTC socket by the RCI server. The RTC reads the *rtc_pedestal_control* message from the RCI server/RTC socket and executes the pedestal move upon completion of the current active scan.

Algorithm Description

When a start-of-volume flag is detected the following action is performed:

1) start saving radial data until a new elevation cut is detected
2) keep track of the leftmost and rightmost azimuths; these will define the scan sector boundaries

When a new elevation cut is detected the following actions are performed:

1) throw out all gates inside the box that are below threshold
2) weight the remaining gates by subtracting the threshold, squaring the result, and adding 1
3) sum the weights along each radial and multiply the result by the azimuth angle
4) sum the weights at each unique range and multiply the result by the range
5) sum the weighted azimuths and divide by the total number of gates above threshold (centroid azimuth)
6) sum the weighted ranges and divide by the total number of gates above threshold (centroid range)
7) tally the number of gates inside box
8) determine the number of gates >= threshold inside box
9) determine the number of gates >= 20 dBZ inside box
10) determine the number of gates >= 30 dBZ inside box
11) determine the number of gates >=30 dBZ inside box
12) determine the number of gates >= 40 dBZ inside box
13) determine the number of gates >= 50 dBZ inside box
14) determine the number of gates >= 60 dBZ inside box
15) keep track of the maximum reflectivity inside box
16) update sector properties
17) update box position (not for 1st scan)
18) update pedestal position (not for 1st scan)
19) build status message and write it to the rtc_comm LB

While scanning is active and not at the lowest elevation cut, the algorithm continues reading radials, looking for a start- of-volume flag, which at that time starts a new centroid calculation and repeats the previous 18 steps.. In addition, the algorithm is constantly reading the ep_comm LB, looking for *wxtrack_control* messages sent from the RCI server.

Algorithm Control

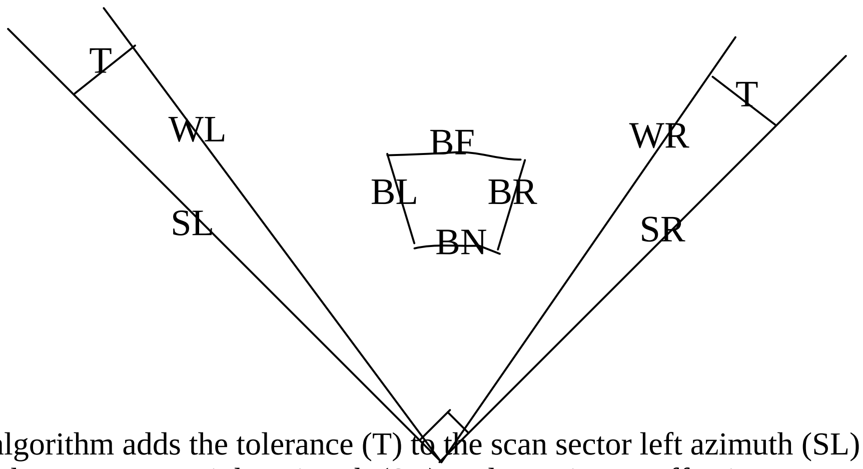
Currently there are only two commands that affect algorithm processing: ON and OFF. It is expected that when a control ON message is received, the desired box coordinates, tolerance, and threshold are included in the message.

The *wxtrack_control* message contains the following information:

1) Command (0 = OFF; 1 = ON)
2) Box left azimuth (deg)
3) Box right azimuth (deg)
4) Box near range (km)
5) Box far range (km)
6) Reflectivity threshold (dBZ)
7) Sector azimuth tolerance (deg)

Pedestal Control

The pedestal is commanded to a new position when one of the box sides moves outside the scan window. How the box and tolerance fields are defined relative to the scan sector is illustrated below.



The algorithm adds the tolerance (T) to the scan sector left azimuth (SL) and subtracts it from the scan sector right azimuth (SR) to determine an effective sector window (WL and WR). If the box left azimuth (BL) is less than or equal to the sector window left azimuth (WL) the pedestal is commanded counterclockwise to a position where the box right azimuth (BR) is 4 degrees inside the sector window right azimuth (WR). A value of 4 degrees was chosen to add some cushion between the box right azimuth (BR) and the new sector window right azimuth (WR). Correspondingly, if the box right azimuth (BR) is greater than or equal to the sector window right azimuth (WR) the pedestal is commanded clockwise to a position where the box left azimuth (BL) is 4 degrees inside the sector window left azimuth (WL).

The box near range (BN) and box far range (BF) have no effect on pedestal control. One thing to remember is that the difference (BF-BN) remains constant between scans; (BR-BL) does not.

When a pedestal position command is sent to the RCI server (pedestal position message is written to the rtc_comm LB and relayed to the RTC by the ep_comm task), it is invoked by the RTC following the completion of the current scan. After the pedestal move is completed, scanning is resumed. One scan must be completed before another pedestal move command can be issued by the algorithm.

Algorithm Status

At the end of each action a *wxtrack_status* message is built and written to the **rtc_comm**

LB and relayed to the RCI server by the **ep_comm** task.

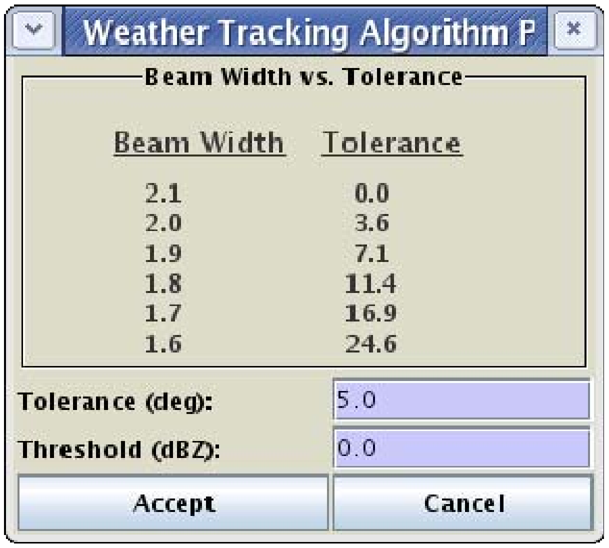
The *wxtrack_status* message contains the following information:

1) Algorithm status (0 = OFF; 1 = ON; 2 = ON/Pedestal Commanded)
2) sector left azimuth
3) sector right azimuth
4) azimuth tolerance
5) box left azimuth
6) box right azimuth
7) box near range
8) box far range
9) reflectivity threshold
10) centroid azimuth
11) centroid range
12) azimuths inside box
13) total gates inside box
14) gates at or above threshold inside box
15) maximum reflectivity inside box
16) number of gates at or above 20dBZ
17) number of gates at or above 30dBZ
18) number of gates at or above 40dBZ
19) number of gates at or above 50dBZ
20) number of gates at or above 60dBZ
21) algorithm action (0 = no action; 1 = terminated - box too close; 2 = terminated - box too wide; 3 = terminated - box outside; 4 = terminated - no gates above threshold; 5 = commanded pedestal move)

At the RCI client, the Base Data window contains a number of selection that are used for tracking control and status. Only the controlling client can turn tracking on/off, define the box region and edit track properties. However, all clients can monitor tracking status.

Defining input parameters

The input parameters for WXTRACK are defined at the controlling RCI client in the Base Data window (refer to Figure 1). To "box" a feature of interest in the display window the user first selects the "Box Region" button. Pressing the left mouse button in the display region defines the first box coordinate. While keeping the left mouse button pressed, drag the cursor to the desired second box coordinate. Releasing the left mouse button sets the second box coordinate. To set the azimuthal tolerance and reflectivity threshold select the "Properties" button in the Track group. These fields can be defined in the Weather Tracking Algorithm Properties window (Figure 3)



Beam Width	Tolerance
2.1	0.0
2.0	3.6
1.9	7.1
1.8	11.4
1.7	16.9
1.6	24.6

Tolerance (deg): 5.0

Threshold (dBZ): 0.0

Figure 4: Weather Tracking Algorithm Properties window

A reference table is included which relates beam width to tolerance. For example, if you wanted to ensure that all beams inside the box have beam widths less than or equal to 1.9 you would specify a tolerance of 7.1 degrees. This would result in an effective scan window of 75.8 degrees (90 – 2*7.1).

NOTE: To see the box in the display window once it is defined, the "Box" selection in the Overlay group must be selected. Once tracking is active use the "Centroid" selection in the Overlay group to display the updated centroid position. Also, you must consider the effective scan window size when defining the box. You want the box width (BR-BL) to be smaller than the effective scan window size (WR-WL). The algorithm will terminate tracking when the updated box width is greater than or equal to the effective scan window size.

Task Log File

When the wxtrack algorithm is running, certain textual output is written to the standard log error linear buffer (this is done for all tasks running at the DSP). Although the intended user of this file is the software developer/maintainer, it contains information about the algorithm performance that may be useful in subsequent data analysis by other users. A sample log for one scan iteration is shown below.

```
14:52:42   wxtrack: New scan detected so start gathering data for new product
14:52:42   wxtrack: elev [0.505371]
14:52:42   wxtrack: az [314.829712]
14:52:42   wxtrack:     Range resolution 0.239834 km
14:52:42   wxtrack:     beginning range  10.092963 km
14:52:42   wxtrack:     Data offset      66.000000
14:52:42   wxtrack:     Data scale       2.000000
14:52:42   wxtrack: start_elevation 0.505371 deg
14:52:48   wxtrack: new elevation 0.895386 deg
14:52:48   wxtrack: New elevation 0.895386 detected so compute new centroid
14:52:48   wxtrack: computeCentroid()
14:52:48   wxtrack: computeCentroid().azimuths = 109
14:52:48   wxtrack: computeCentroid().azimuth = 342.410889
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 343.196411
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 343.976440
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 344.756470
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 345.536499
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 346.305542
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 347.080078
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 347.849121
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 348.612671
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 349.381714
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 350.145264
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 350.903320
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 351.661377
14:52:48   wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48   wxtrack: computeCentroid().azimuth = 352.419434
```

```
14:52:48    wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48    wxtrack: computeCentroid().azimuth = 353.177490
14:52:48    wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48    wxtrack: computeCentroid().azimuth = 353.930054
14:52:48    wxtrack: boxNearRangeIndex = 56 ... boxFarRangeIndex = 84
14:52:48    wxtrack: computeCentroid().maxReflectivity = 21.000000
14:52:48    wxtrack: computeCentroid().boxAzimuths = 16
14:52:48    wxtrack: computeCentroid().centroidAzimuth = 350.415314
14:52:48    wxtrack: computeCentroid().centroidRange   = 26.407810
14:52:48    wxtrack: computeSectorProperties()
14:52:48    wxtrack: computeSectorProperties().azimuthLeft   [314.829712]
14:52:48    wxtrack: computeSectorProperties().azimuthRight  [44.791260]
14:52:48    wxtrack: computeSectorProperties().azimuthCenter [359.810486]
14:52:48    wxtrack: updateBoxPosition()
14:52:48    wxtrack: updateBoxPosition().controlData.boxCenterAzimuth  = 350.207184
14:52:48    wxtrack: updateBoxPosition().controlData.boxLeft  = 344.110626
14:52:48    wxtrack: updateBoxPosition().controlData.boxRight = 356.303741
14:52:48    wxtrack: updateBoxPosition().controlData.boxNear  = 23.674749
14:52:48    wxtrack: updateBoxPosition().controlData.boxFar = 30.348198
14:52:48    wxtrack: updatePedestalPosition()
14:52:48    wxtrack: updatePedestalPosition(): leftLimit [319.829712] ... rightLimit [39.791260]
14:52:48    wxtrack: sendStatus()
14:52:48    wxtrack: Status written to %RCI_COMM lb
```

This information is repeated for each volume scan whenever tracking is active. In addition, other messages pertinent to algorithm performance are included (i.e., commanded pedestal position, boundary checks, tracking termination). Please note that this file is currently recreated when the EP is restarted, so, if you are interested in preserving this information for future reference you will need to log into par4 and make a copy of it before restarting the EP (we are currently considering options to preserve these data so they are archived with the I/Q and moment data sets). The file name is "/home/nwrt/data/logs/wxtrack.log".

Also note that access to par4 is restricted, so, if you want a copy of this file preserved you will need to contact one of the software/test engineers.