# Thunderstorm Identification Tracking Analysis and Nowcasting
# TITAN Data Model
# Storm and Track Properties in NetCDF

Michael Dixon

2025/08/12

## 1. Overview

The Titan application reads radar volume scans sequentially in time, identifies the storms in each volume, and tracks those storms from one time to the next.

On the input side, the radar volume scans are stored in Cartesian coordinates, in NetCDF files that follow the CF conventions and conform to the NCAR MDV (Meteorological Data Volume) format.

On the output side, the current version of Titan writes the storm and track property data into daily binary files. Specific Titan-related applications convert those binary files as follows:

- Tstorms2Ascii converts to CSV text files.

- Tstorms2XML converts to XML text files.

- Tstorms2NetCDF converts to NetCDF-4 files, using groups.

This document details the data model used for the NetCDF files.

NOTE: the scan and storm properties are discussed first, followed by the track properties. However, after using the track sections to locate track entries you will need to go back to the storm sections to locate the storm properties associated with the track entries.

## 2. Storm identification and properties

Titan identifies storms as contiguous volumes with reflectivity exceeding a specified threshold. Figure 2.1 (a) below shows an example with a reflectivity threshold of 35 dBZ. The storm overall outline is shown by the cyan polygon.

Global properties (***gprops***) are computed for the storm as a whole.

Layer properties (***lprops***) are computed for each height in the Cartesian volume. The vertical section (Figure 2.1 (b)) shows the Cartesian layers for which the layer properties are computed.

For each storm Titan computes the reflectivity distribution as a histogram (***hist***), both in 3D and 2D. The 2D properties are generally a 2D column maximum of the 3D values. Figure 2.2 (b) shows a time history of the reflectivity histogram.
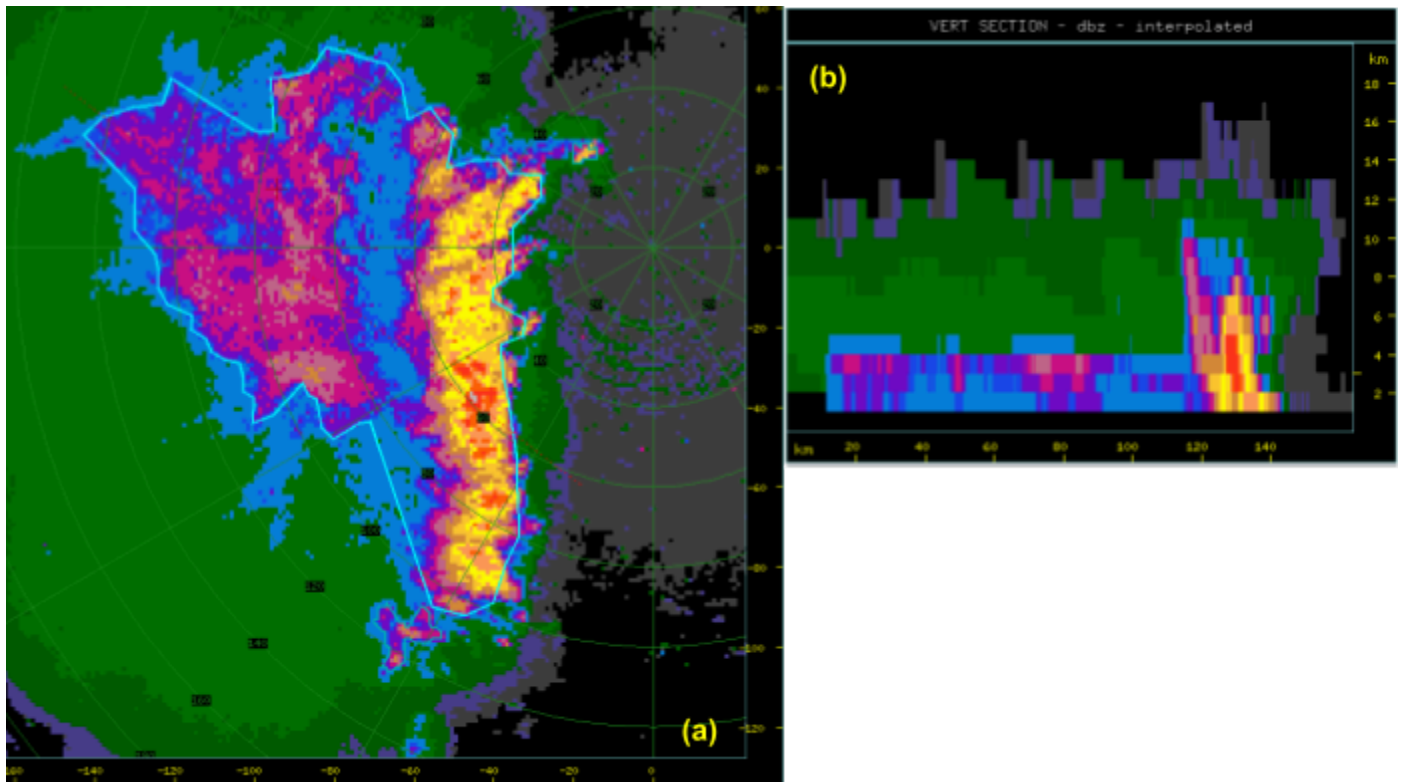
**Figure 2.1: storm identification (a) and vertical section through the storm (b). The storm is moving west to east, with a strong line behind which is a trailing stratiform region.**
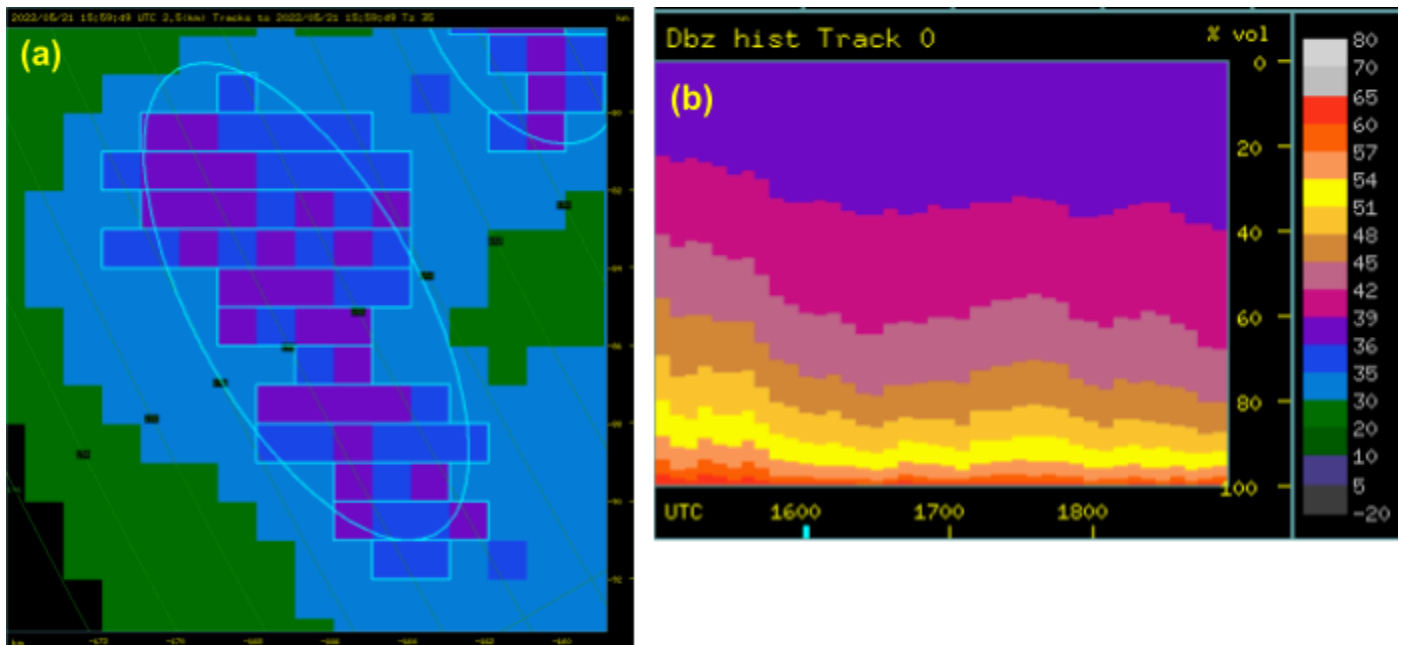


**Figure 2.2: (a) Grid runs (rectangles) plotted on top of the reflectivity for a storm.**
**(b) Example of a time history of the reflectivity distribution (2D histogram) for a storm track.**

Titan also saves the 3D grid locations (*runs*) that make up each storm. We refer to these as 'runs', which comprise a starting (x, y, z) location and a run length. Figure 2.2 (a) shows the runs plotted for an example storm, overlain by an ellipse that represents the shape and orientation of the storm.

In addition to the 3D runs, there is a set of 2D runs called **proj_runs** (for projected-area runs). These are the runs computed by collapsing the 3D runs in the vertical column to produce a 2D version of the storm envelope.

# 3. Tracking

## 3.1 Simple tracks

Titan tracks the storms over time, from one volume scan to the next.

A **simple track** refers to a track that has no mergers or splits. See Figure 3.1. This storm is moving from SW to NE.
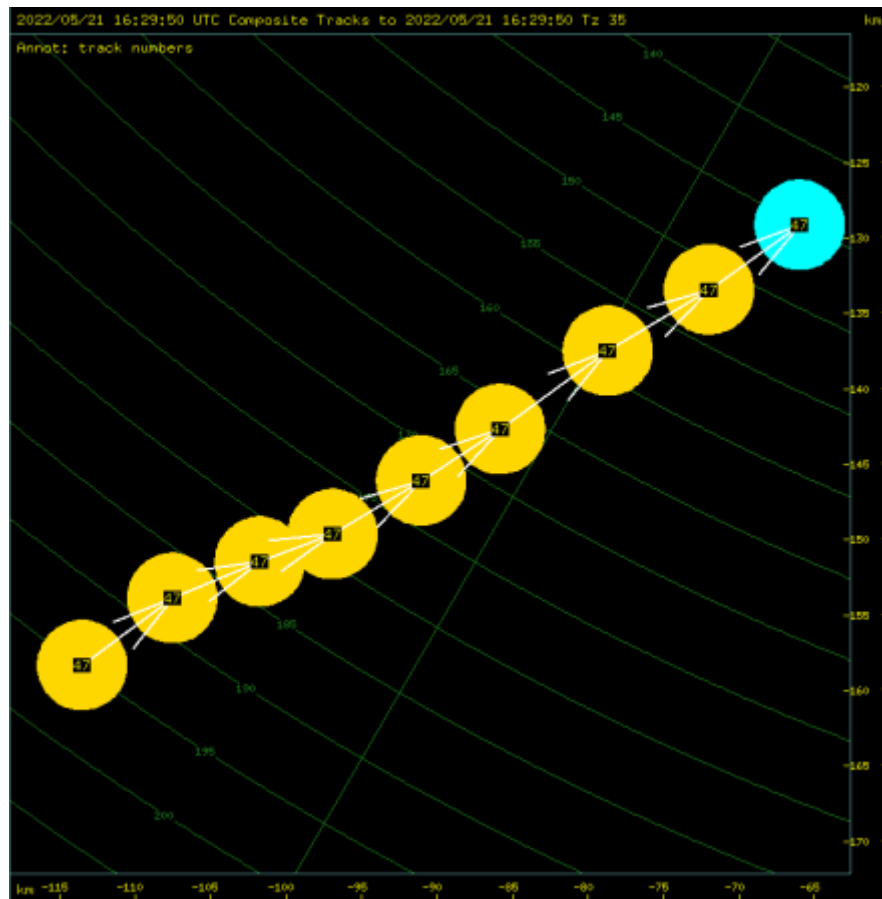


**Figure 3.1: example of simple track (number 47) with no mergers or splits over its lifetime.**

## 3.2 Complex tracks

Most storm tracks include mergers and/or splits during their lifetime. A **complex track** is made up of a set of simple tracks. Each individual simple track has no mergers or splits, but may have multiple parents and/or multiple children.

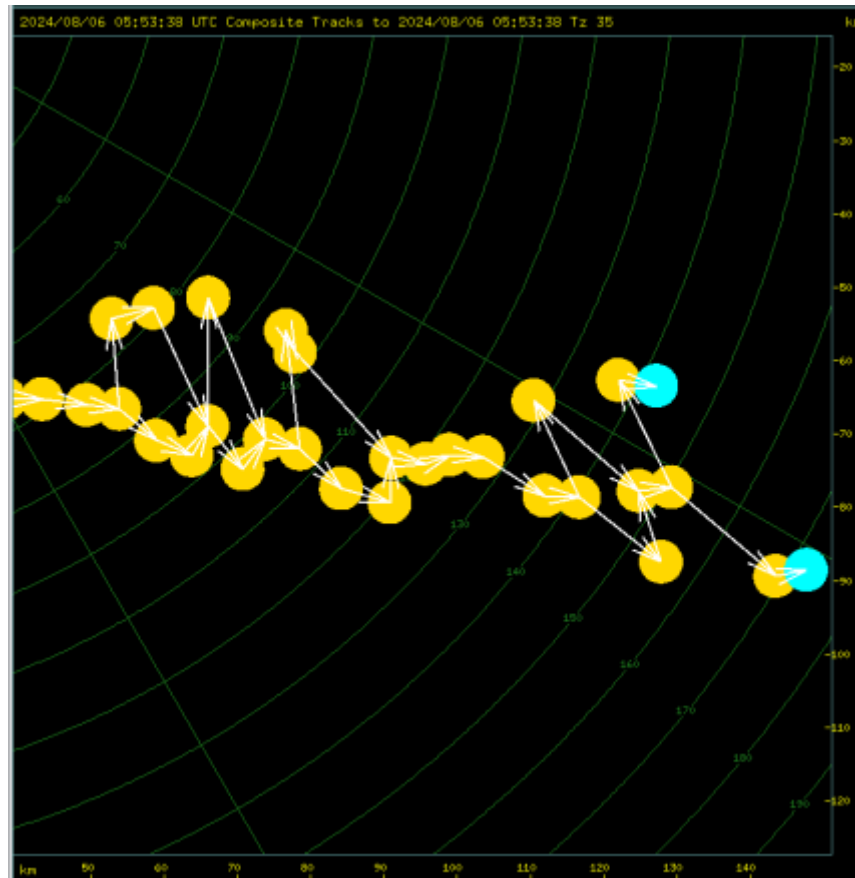Figure 3.2 shows an example of a complex track. The motion is from NW to SE.



**Figure 3.2: example of complex track (number 0), comprising a set of simple tracks containing multiple parents and children.**

## 3.3 Track numbers

Each of the nodes (filled circles) shown in Figure 3.2 is referred to as a **track entry**. There is a one-to-one correspondence between track entries and the storms identified in a scan.

A simple track is a sequence of one or more entries.

When a simple track is first identified, it is assigned a **simple_track_number**, the next in the sequence. There are no gaps in the sequence of simple track numbers.

At its start a simple track is also assigned a **complex_track_number**. This is initially set equal to the simple track number. A track with no mergers or splits in its lifetime will have the same simple and complex track number throughout.

When a merger does occur, the parent tracks are combined to form a complex track. Each parent in a merger will have a unique complex track number. Titan selects the lowest of these and that becomes the complex track number of the merged track. The other complex numbers are discarded. Therefore the sequence of complex numbers will contain gaps.

Figure 3.3 below shows the same track as figure 3.2, but with the track numbers plotted for each node. The numbers are shown as complex/simple. The figure shows a single complex track with number 0 - i.e. the first track identified in this tracking example. The first number in the pair is this complex track number. The second number is the simple track number, which obviously changes from one simple track to the next. At the extreme left of the plot you will see nodes with just '0' as the label. These are for simple track 0. If the simple and complex track numbers are the same only one number is plotted.

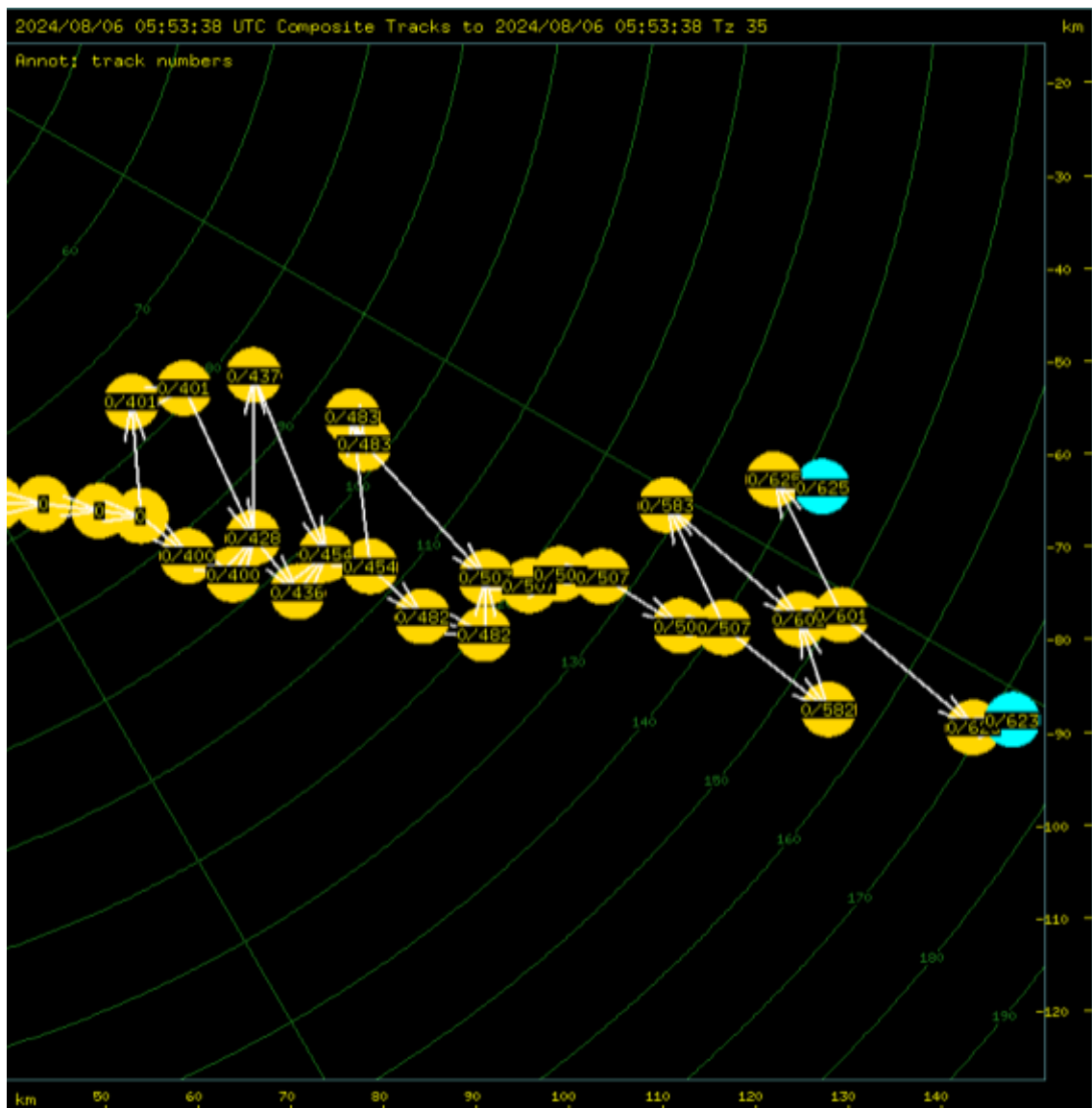Figure 3.3: the same complex track as in Figure 3.2, but with the track numbers shown. The first number is the complex track number (in this case 0), and the second is the simple track number.

As mentioned above this is a single complex track, number 0. It is made up of simple track numbers 0, 400, 401, 428, 436, 437, 454, 482, 483, 507, 582, 583, 601, 623 and 625. The cyan circles represent the final nodes in this track.

# 4. Data model for storms and tracks

## 4.1 Group hierarchy

Figure 4.1 shows the hierarchy of groups in a Titan netCDF data file:
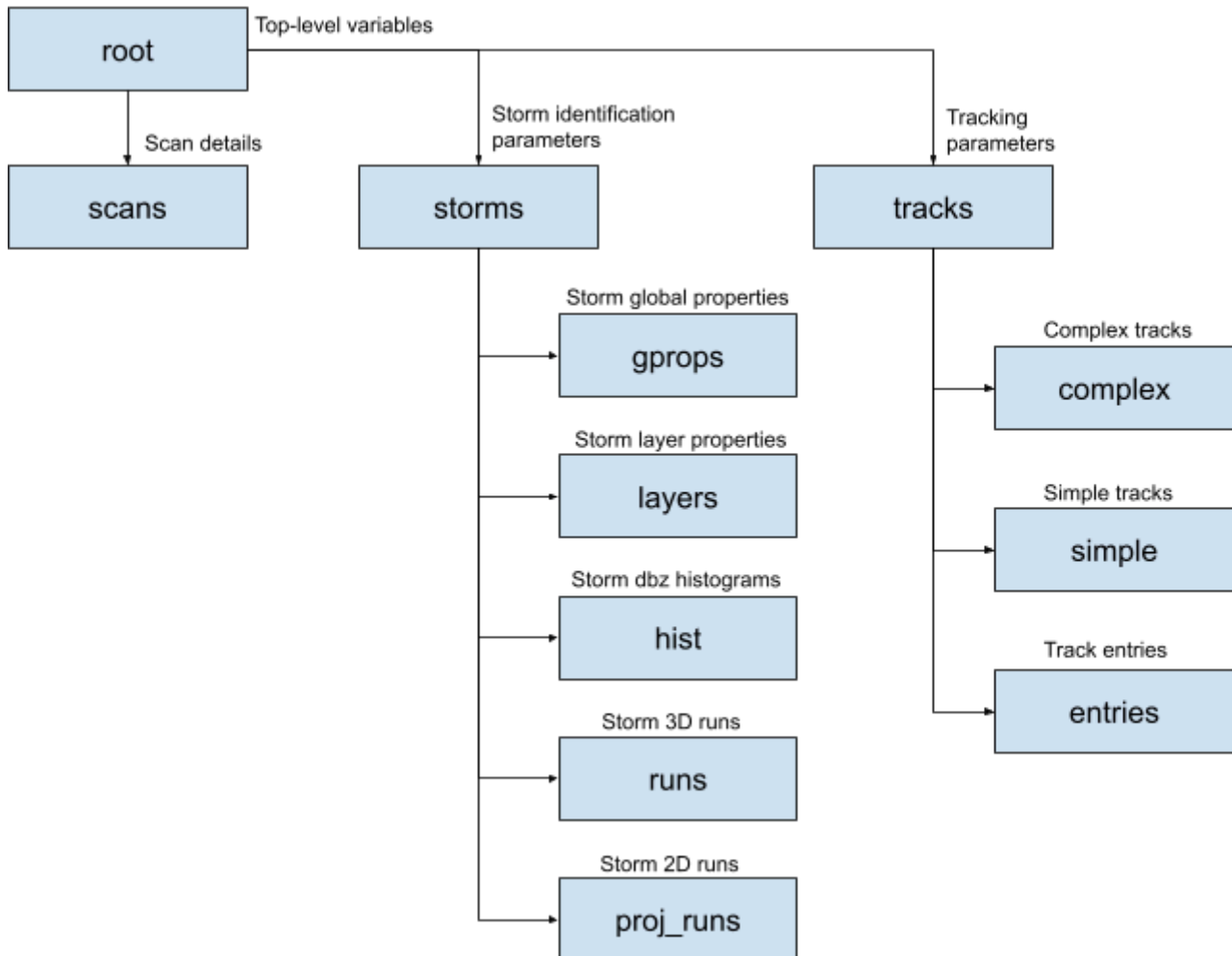


**Figure 4.1: hierarchy of groups in Titan data netCDF file**
**The numbers for the relevant sections below are listed in parentheses.**

## 4.2 root group

The **root** group has the top-level variables and the global attributes for the data set.

The variables include the **start_time** and **end_time** of the data set, and **n_scans** analyzed. These are all scalars.

## 4.3 scans group

The **scans** group contains 1-D arrays for the details of the volume scans analyzed. For each scan one or more storms may be identified. The properties of each storm are stored in the **gprops** group. In the **scans** group you will find the **scan_nstorms** and **scan_gprops_offset** variables. You use these variables to navigate to the global properties in the **gprops** group.

## 4.4 storms group

The **storms** group contains the parameters used for identifying the storms. These are scalars.

It also contains the following sub-groups: **gprops**, **layer**, **hist**, **runs** and **proj_runs**.

## 4.5 gprops group

The **gprops** group contains concatenated 1-D arrays with properties for the storms identified, for all of the scans combined. We need to locate the storm properties for a selected scan.
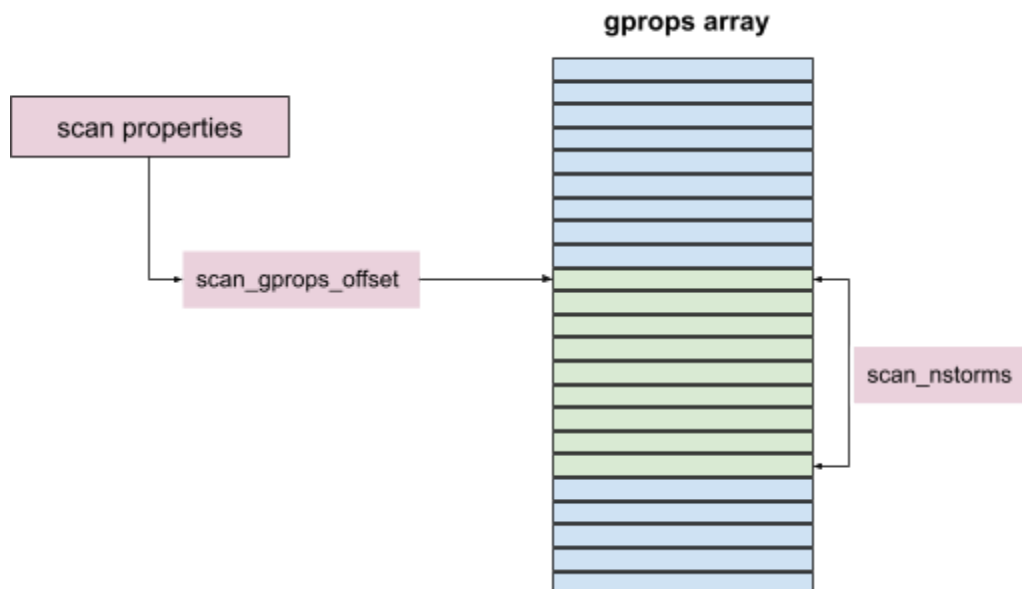


**Figure 4.2: locating the storm properties from the scan properties.**

## 4.6 layers group

The **layers** group contains concatenated 1-D arrays of layer properties, for all of the storms identified in all of the scans combined. We need to locate the layer properties for a selected storm.

**Figure 4.3: locating the layer properties from the storm global properties.**

## 4.7 hist group

The **hist** group contains concatenated 1-D arrays for the dbz histogram entries, for all of the storms identified in all of the scans combined. We need to locate the histogram entries for a selected storm.
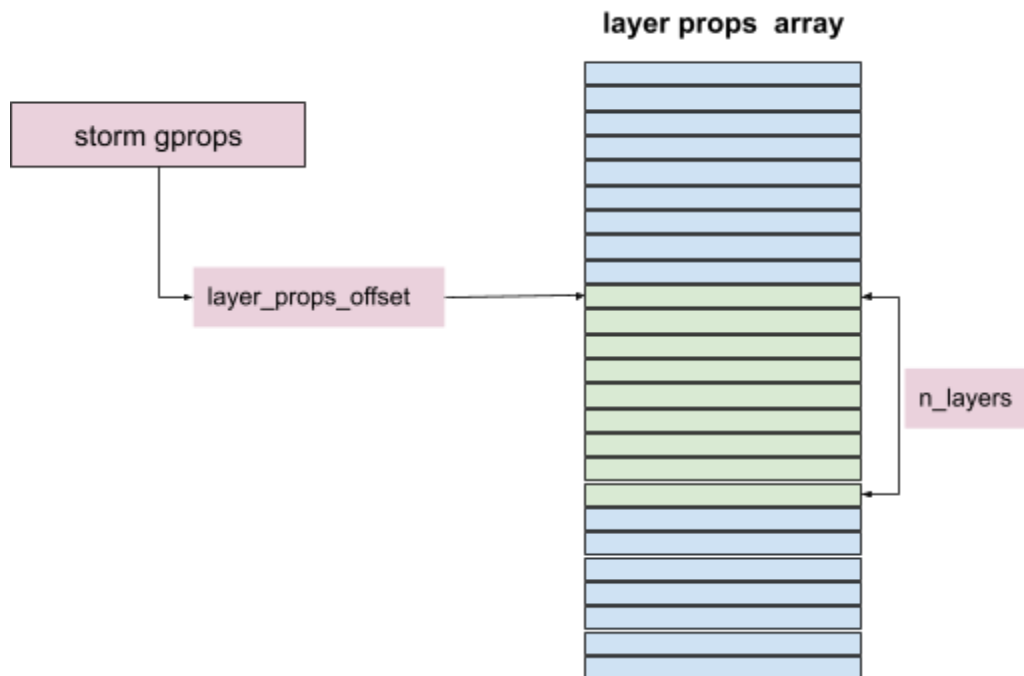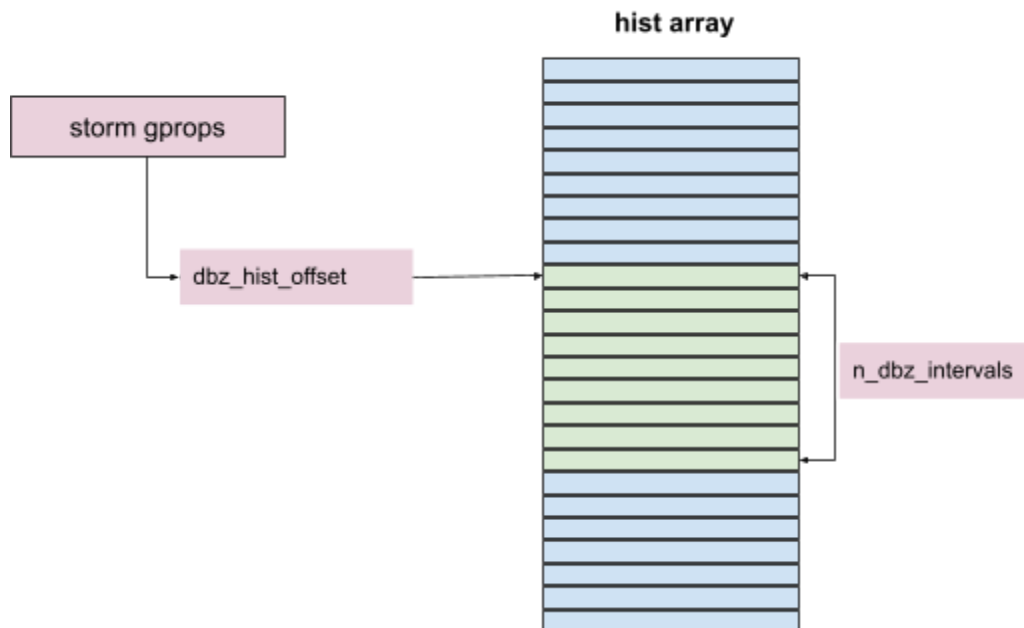


**Figure 4.4: locating the histogram properties from the storm global properties.**

## 4.8 runs group

The **runs** group contains concatenated 1-D arrays for the 3D storm runs, for all of the storms identified in all of the scans combined. We need to locate the **runs** entries for a selected storm.
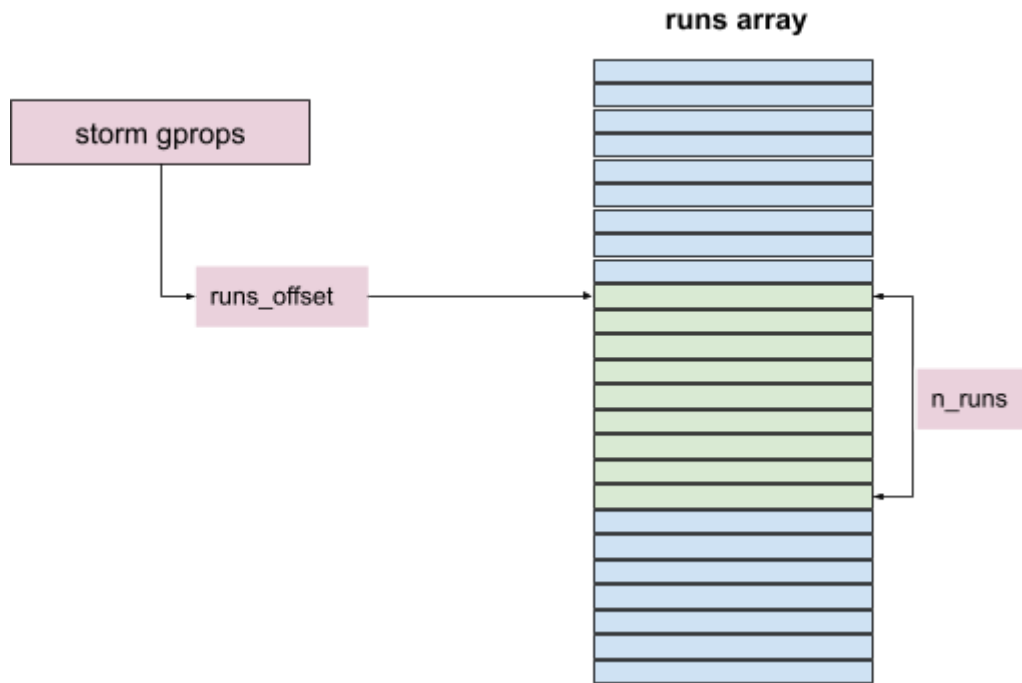


**Figure 4.5: locating the runs array from the storm global properties.**

## 4.9 proj_runs group

The **proj_runs** group contains concatenated 1-D arrays for the 2D storm projected area runs, for all of the storms identified in all of the scans combined. We need to locate the **proj_runs** entries for a selected storm.
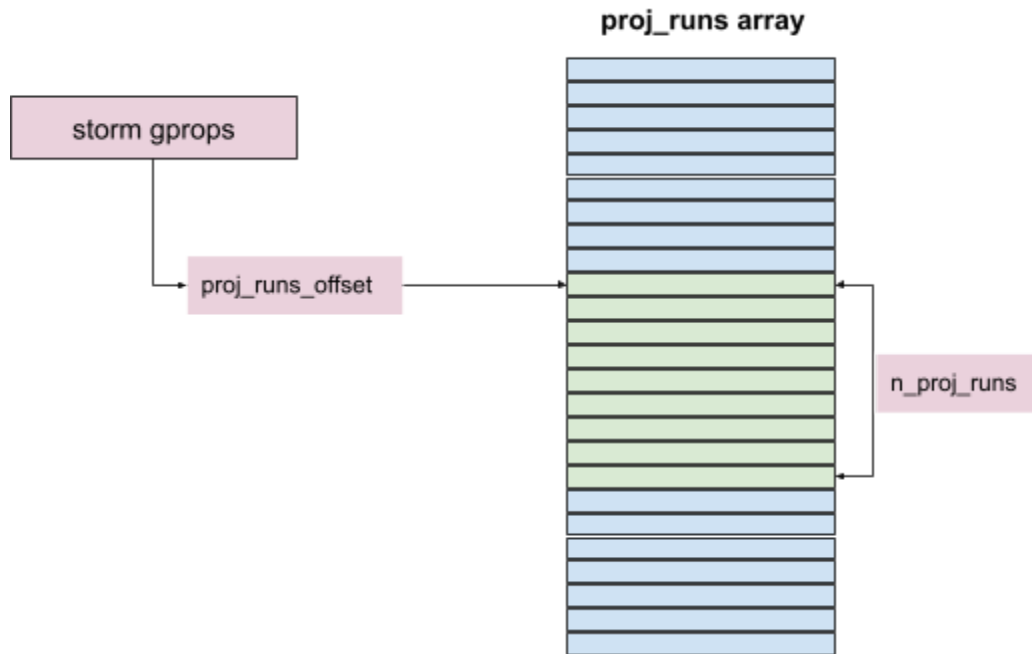


**Figure 4.6: locating the proj_runs array from the storm global properties.**

## 4.10 tracks group

The **tracks** group contains the parameters used for tracking storms from one scan to the next, and for producing forecasts. These are scalars, except for **forecast_weights**, which is a 1-D array containing the weights used for computing the forecast.

Also at the top-level are:

- n_complex_tracks
- n_simple_tracks

**n_complex_tracks** is always <= **n_simple_tracks**.

The group also contains the following sub-groups: **complex, simple** and **entries**.

## 4.11 complex group

The **complex** group contains the arrays of properties for the complex tracks in the data set. These are 1-D arrays.

The complex group contains two array types:

- the complex_track_nums array for determining the complex track numbers in the data set.
- arrays that contain the properties of each complex track.

The **complex_track_nums** array contains the numbers of the complex tracks. It has a logical length of **n_complex_tracks** (the size in the NetCDF file may be larger). This is a contiguous array with no gaps. To traverse the set of complex tracks, read in this array and then process the tracks one by one, i.e. for each complex track.

All other arrays in this group have a logical length of **n_simple_tracks** (the size in the NetCDF file may be larger). These are sparse arrays, with gaps (containing the missing value), because the complex track numbers are not contiguous. You use the complex_track_num as the index into these arrays.
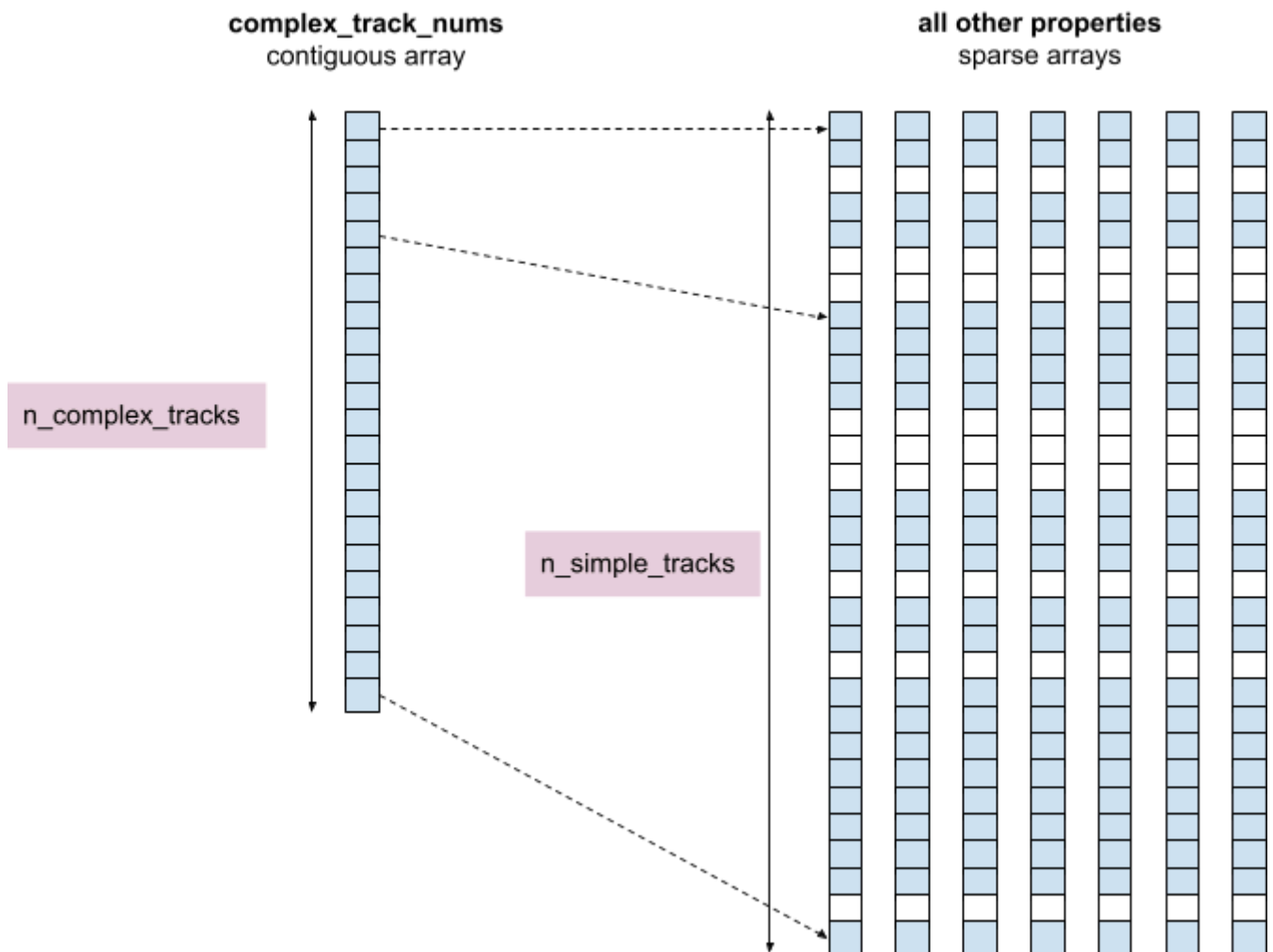


**Figure 4.7: locating the complex track properties using the complex_track_nums array.**

## 4.12 simple group

The simple group contains two array types:

- navigation arrays for determining the simple track numbers that make up a complex track

● arrays that contain the properties of each simple track.

## Locating the simple track properties

3 arrays are used to determine the simple track numbers for a specified complex track:

- n_simples_per_complex
- simples_per_complex_offsets
- simples_per_complex
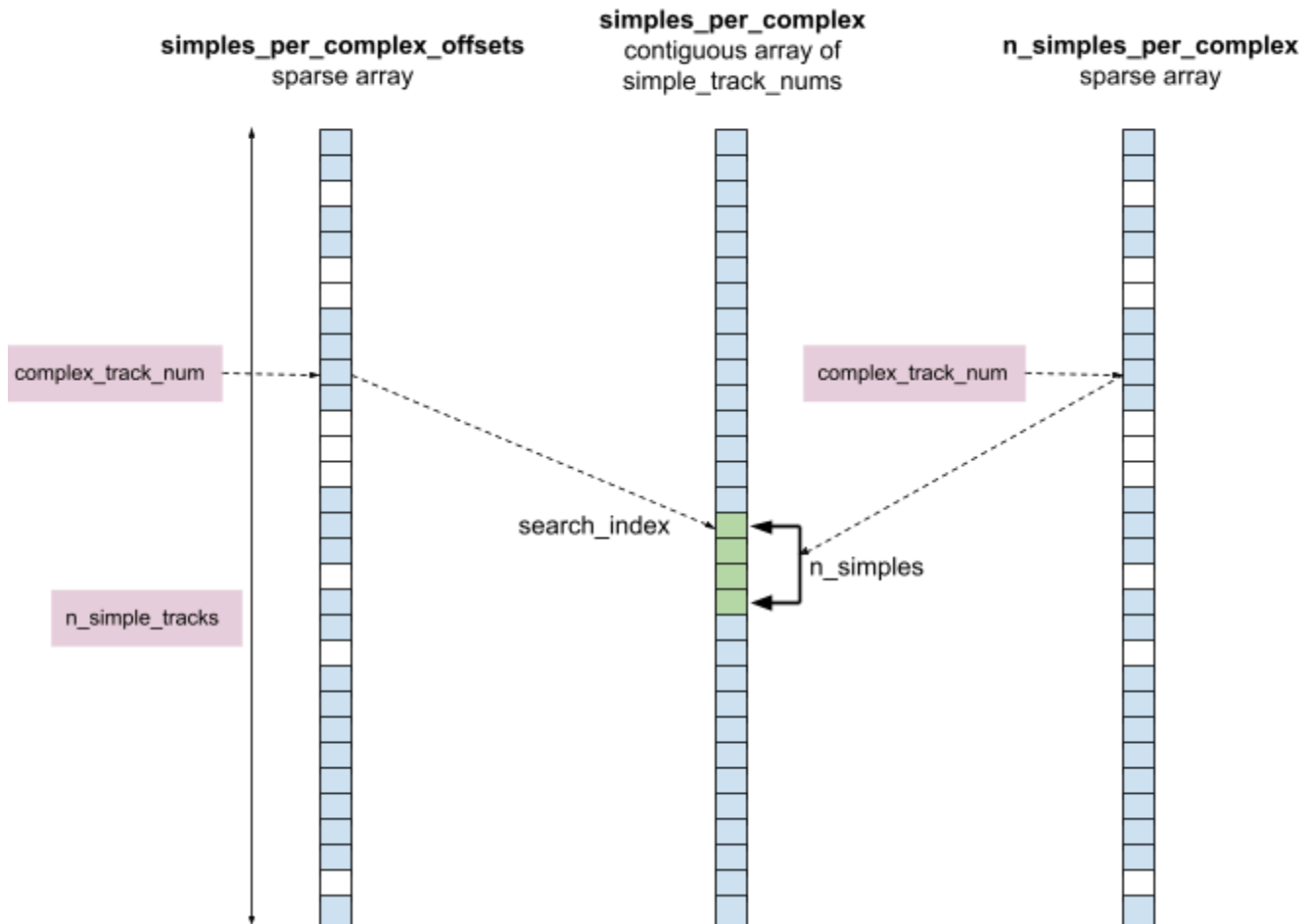


**Figure 4.8: locating the simple track numbers for a complex track**

We start with the **complex_track_num** determined from the complex group (see Figure 4.7).

For that **complex_track_num** we look up:

- the offset from the **simples_per_complex_offsets** array.
- the number of simple tracks from the **n_simples_per_complex** array.

Using those values as the index and count, we locate the set of simple track numbers in the **simples_per_complex** array (see green cells in Figure 4.8).

## Simple track properties

Using the procedure described above, you can locate the **simple_track_num** for each simple track in the complex track. You use the simple track numbers as indices into the simple track properties arrays.

The **simple** group contains the arrays for the properties for all of the simple tracks in the data set. These are 1-D arrays, except for the **parent** and **child** arrays, which are 2-D arrays of maximum length 8. Each track can have multiple parents or children (up to a maximum of 8). The properties **nparents** and **children** indicate the number of parents and/or children.

A simple track comprises one or more track **entries**. We use the simple track properties to locate the properties of the track entries, which in turn allow us to locate the relevant storm properties for each entry. See below for details.

## 4.13 entries group

The **entries** group contains arrays of the track entry properties for all of the entries in the data set.

There is a 1-to-1 correspondence between the track entries and the storms that are identified. We can use the track entry properties to locate the associated storm properties.

Section 4.12 above shows how to find the properties of each simple track. In those properties are 2 items that allow us to locate the entries:

- first_entry_offset - the index of the first entry in the simple track
- duration_in_scans - the number of entries in the simple track
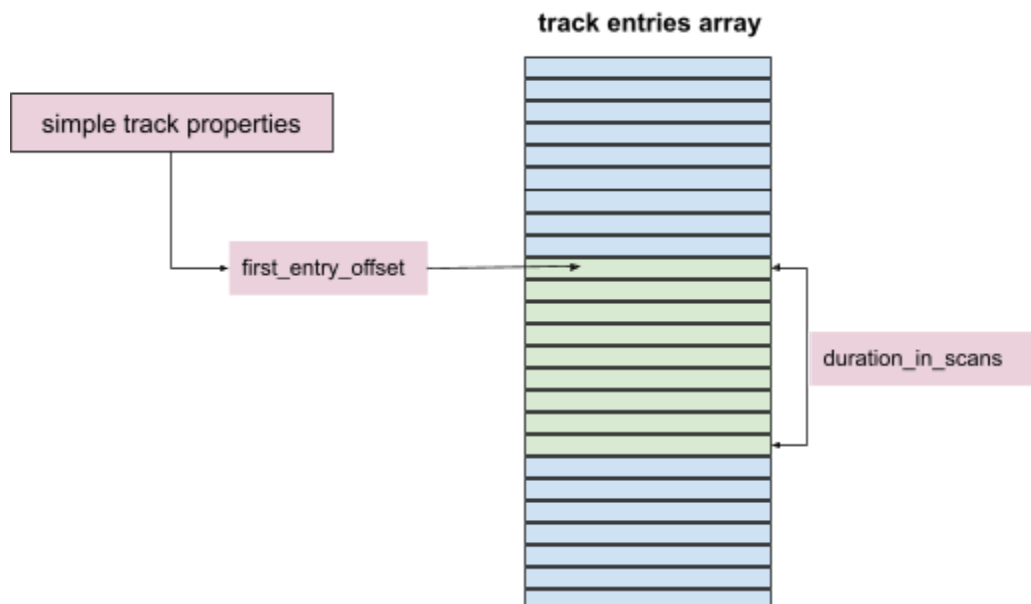


**Figure 4.9: locating the track entries from the simple track properties**

The track entry properties contain the following items:

- scan_num
- storm_num

These allow us to locate the storm properties (section 4.3, 4.5).

# 5. Details of group contents

In this section we describe the variables that are included in each group.

Note that table entries that are highlighted are used for locating properties across the groups.

## 5.1 root group

## Top level variables

| Var type | Var name | Description |
|---|---|---|
| int64 | file_time | unix time for last file modification<br>units = "seconds since 1970-01-01T00:00:00" ; |
| int64 | start_time | unix time of first scan |
| int64 | end_time | unix time of last scan |
| int32 | n_scans | number of scans |
| int32 | sum_storms | total number of storms, summed for all scans |
| int32 | sum_layers | total number of layer entries, summed for all storms |
| int32 | sum_hist | total number of histogram entries, summed for all storms |
| int64 | sum_runs | total number of grid runs (3D), summed for all storms |
| int64 | sum_proj_runs | total number of grid runs (2D), summed for all storms |
| int32 | max_simple_track_num | highest simple track number |
| int32 | max_complex_track_num | highest complex track number |

## 5.2 scans group

## Scan properties

| Type | Name | Description |
|---|---|---|
| int64 | scan_time | unix time of scan in seconds since 00:00 on 1970/01/01 |
| int32 | scan_num | scan number |
| int32 | scan_nstorms | number of storms in this scan |
| int64 | scan_gprops_offset | index of first storm gprops in scan |
| fl32 | scan_min_z | mean sea level height of lowest layer in storm (km) |
| fl32 | scan_delta_z | layer thickness (km) |

| fl32 | scan_ht_of_freezing | height of freezing level in km |
|------|---------------------|-------------------------------|
| fl64 | grid_minx | x coord of center of cell in SW corner of grid |
| fl64 | grid_miny | y coord of center of cell in SW corner of grid |
| fl64 | grid_minz | z coord of center of lowest layer of grid |
| fl64 | grid_dx | delta x for grid |
| fl64 | grid_dy | delta y for grid |
| fl64 | grid_dz | delta z for grid |
| fl64 | grid_sensor_x | x coord for radar |
| fl64 | grid_sensor_y | y coord for radar |
| fl64 | grid_sensor_z | z coord for radar |
| fl64 | grid_sensor_lat | latitude of radar (deg) |
| fl64 | grid_sensor_lon | longitude of radar (deg) |
| int32 | proj_type | projection type<br>● LAT_LON = 0<br>● LAMBERT_CONFORMAL = 3<br>● MERCATOR = 4<br>● POLAR_STEREO = 5<br>● POLAR_STEREO_ELLIP = 6<br>● CYLINDRICAL_EQUIDIST = 7<br>● AZIMUTHAL_EQUIDIST = 8<br>● POLAR_RADAR = 9<br>● OBLIQUE_STEREOGRAPHIC = 12<br>● TRANSVERSE_MERCATOR = 15<br>● ALBERS_EQUAL_AREA = 16<br>● LAMBERT_AZIMUTHAL = 17 |
| int32 | dz_constant | flag to indicate delta z is constant in the grid |
| int32 | grid_nx | number of grid cells in x |
| int32 | grid_ny | number of grid cells in y |
| int32 | grid_nz | number of grid cells in z |
| string | unitsx | grid units for x coordinate |
| string | unitsy | grid units for y coordinate |
| string | unitsz | grid units for z coordinate |
| fl64 | proj_origin_lat | projection origin latitude |
| fl64 | proj_origin_lon | projection origin longitude |
| fl64 | proj_rotation | rotation of grid relative to TN |

| Type | Name | Description |
|------|------|-------------|
| fl64 | proj_lat1 | latitude-1 applies to selected projections |
| fl64 | proj_lat2 | latitude-2 applies to selected projections |
| fl64 | proj_tangent_lat | tangent-latitude applies to selected projections |
| fl64 | proj_tangent_lon | tangent-longitude applies to selected projections |
| fl64 | proj_pole_type | north or south pole applies to selected projections |
| fl64 | proj_central_scale | central-scale applies to selected projections |

## 5.3 storms group

## Storm identification parameters

| Type | Name | Description |
|------|------|-------------|
| fl32 | low_dbz_threshold | dbz - low limit for dbz values |
| fl32 | high_dbz_threshold | dbz - high limit for dbz values |
| fl32 | dbz_hist_interval | dbz - bin interval for dbz histograms |
| fl32 | hail_dbz_threshold | dbz - threshold above which precip is assumed to be hail |
| fl32 | base_threshold | km - min ht for storm base - echo below this is ignored |
| fl32 | top_threshold | km - max ht for storm top - echo above this is ignored |
| fl32 | min_storm_size | km2 or km3<br>min size for storm definition (km2 for 2D data km3 for 3D data) |
| fl32 | max_storm_size | km2 or km3<br>max size for storm definition (km2 for 2D data km3 for 3D data) |
| fl32 | morphology_erosion_threshold | threshold to which morphology erosion is performed (km) |
| fl32 | morphology_refl_divisor | The morphology field is obtained by adding the euclidean distance to storm edge (km) to the reflectivity excess (above threshold) divided by this value (dbz/km) |
| fl32 | min_radar_tops | (km) min tops for valid radar data - if checked |
| fl32 | tops_edge_margin | (km) margin placed on min_tops field to allow for tilted storms |
| fl32 | z_p_coeff | Z - precip coefficient |
| fl32 | z_p_exponent | Z - precip exponent |
| fl32 | z_m_coeff | Z - mass coefficient |
| fl32 | z_m_exponent | Z - mass exponent |
| fl32 | sectrip_vert_aspect | vertical aspect ratio threshold above which storm is considered to be second trip |

| fl32 | sectrip_horiz_aspect | horizontal aspect ratio threshold above which storm is considered to be second trip |
|---|---|---|
| fl32 | sectrip_orientation_error | deg - error in ellipse orientation from radar within which storm is considered second trip |
| fl32 | poly_start_az | deg azimuth from T.N. for first polygon point |
| fl32 | poly_delta_az | deg azimuth delta between polygon points (pos is counterclockwise) |
| int32 | check_morphology | check_morphology flag |
| int32 | check_tops | check_tops flag |
| int32 | vel_available | vel data availability flag |
| int32 | n_poly_sides | number of sides in storm shape polygons |
| fl32 | ltg_count_time | number of seconds over which the lightning strikes are counted. |
| fl32 | ltg_count_margin_km | margin from storm edge to describe outer region for counting ltg strikes |
| fl32 | hail_z_m_coeff | Z - mass coefficient |
| fl32 | hail_z_m_exponent | Z - mass exponent |
| fl32 | hail_mass_dbz_threshold | dbz - threshold above which hail mass is calculated |
| fl32 | tops_dbz_threshold | dbz threshold for computing storm tops |
| fl32 | precip_plane_ht | CAPPI ht for which precip is computed (km MSL). See precip_computation_mode |
| fl32 | low_convectivity_threshold | used if Ecco convectivity thresholds applied, otherwise missing |
| fl32 | high_convectivity_threshold | used if Ecco convectivity thresholds applied, otherwise missing |
| int32 | precip_computation_mode | PRECIP_FROM_COLUMN_MAX = 0<br>PRECIP_AT_SPECIFIED_HT = 1<br>PRECIP_AT_LOWEST_VALID_HT = 2<br>PRECIP_FROM_LOWEST_AVAILABLE_REFL = 3 |

## 5.4 gprops group

## Storm global properties

| Type | Name | Description |
|---|---|---|
| int32 | storm_num | storm number |
| int32 | base_layer | the layer number of the storm base |
| int32 | n_layers | the number of layers in this storm |

| | | |
|---|---|---|
| int64 | layer_props_offset | index of layer props data in layer group |
| int32 | n_dbz_intervals | the number of intervals in the dbz distribution histograms |
| int64 | dbz_hist_offset | index of dbz hist data in hist group |
| int64 | n_runs | number of runs above threshold for this storm |
| int64 | runs_offset | index of storm runs data in runs group |
| int64 | n_proj_runs | number of runs in the projected area for this storm |
| int64 | proj_runs_offset | index of projected area runs data in proj_runs group |
| fl32 | vol_centroid_x | km or deg |
| fl32 | vol_centroid_y | km or deg |
| fl32 | vol_centroid_z | km |
| fl32 | refl_centroid_x | km or deg |
| fl32 | refl_centroid_y | km or deg |
| fl32 | refl_centroid_z | km |
| fl32 | top | km |
| fl32 | base | km |
| fl32 | volume | km3 |
| fl32 | area_mean | km2 |
| fl32 | precip_flux | m3/s |
| fl32 | mass | ktons |
| fl32 | tilt_angle | deg |
| fl32 | tilt_dirn | degT |
| fl32 | dbz_max | dbz |
| fl32 | dbz_mean | dbz |
| fl32 | dbz_max_gradient | dbz/km |
| fl32 | dbz_mean_gradient | dbz/km |
| fl32 | ht_of_dbz_max | km |
| fl32 | rad_vel_mean | m/s |
| fl32 | rad_vel_sd | m/s |
| fl32 | vorticity | /s |
| fl32 | precip_area | km2 |
| fl32 | precip_area_centroid_x | km or deg |

| fl32 | precip_area_centroid_y | km or deg |
|---|---|---|
| fl32 | precip_area_orientation | degT |
| fl32 | precip_area_minor_radius | km or deg - minor radius of precip area ellipse |
| fl32 | precip_area_major_radius | km or deg - standard dev of precip area along the major axis of the precip area shape |
| fl32 | proj_area | km2 |
| fl32 | proj_area_centroid_x | km or deg |
| fl32 | proj_area_centroid_y | km or deg |
| fl32 | proj_area_orientation | degT |
| fl32 | proj_area_minor_radius | km or deg - minor radius of proj area ellipse |
| fl32 | proj_area_major_radius | km or deg - standard dev of proj area along the major axis of the proj area shape |
| fl32 | proj_area_polygon[72] | in grid units the length of rays from the proj area centroid to the polygon vertices |
| int32 | top_missing | flag to indicate that top of storm was not sampled: set to 1 of top missing 0 otherwise |
| int32 | range_limited | flag to indicate that storm was not fully sampled because of range limitations: set to 1 for range-limited 0 otherwise |
| int32 | second_trip | flag to indicate that storm is probably second trip |
| int32 | hail_present | flag to indicate dBZ values above the hail threshold |
| int32 | anom_prop | flag to indicate anomalous propagation |
| int32 | bounding_min_ix | bounding box min x index in grid coords |
| int32 | bounding_min_iy | bounding box min y index in grid coords |
| int32 | bounding_max_ix | bounding box max x index in grid coords |
| int32 | bounding_max_iy | bounding box max y index in grid coords |
| fl32 | vil_from_maxz | vertically-integrated liquid from max dbz per height layer |
| fl32 | ltg_count | ltg strike count in x minutes before storm time with x km of storm |
| fl32 | convectivity_median | used if convectivity thresholds applied |
| int32 | hailFOKRcategory | category 0-4 |
| fl32 | hailWaldvogelProb | 0 <= probability <= 1.0 |
| fl32 | hailMassAloft | ktons |
| fl32 | hailVertIntgMass | kg/m2 from maxz |

| fl32 | hda_poh | Waldvogel Probability % |
|------|---------|--------------------------|
| fl32 | hda_shi | Severe Hail Index J.m-1.s-1 |
| fl32 | hda_posh | probability of severe hail % |
| fl32 | hda_mehs | Maximum Expected Hail Size mm |

## 5.5 layers group

## Storm layer properties

| Type | Name | Description |
|------|------|-------------|
| fl32 | vol_centroid_x | km or deg |
| fl32 | vol_centroid_y | km or deg |
| fl32 | refl_centroid_x | km or deg |
| fl32 | refl_centroid_y | km or deg |
| fl32 | area | km2 |
| fl32 | dbz_max | dbz |
| fl32 | dbz_mean | dbz |
| fl32 | mass | ktons |
| fl32 | rad_vel_mean | m/s - radial velocity mean |
| fl32 | rad_vel_sd | m/s - radial velocity standard deviation |
| fl32 | vorticity | /s - rotation around the storm centroid |
| fl32 | convectivity_median | median convectivity if Ecco is used to constrain the storm to convective regions |

## 5.6 hist group

## Storm DBZ histograms

| Type | Name | Description |
|------|------|-------------|
| fl32 | percent_volume | Bin values for histogram of reflectivity by volume |
| fl32 | percent_area | Bin values for histogram of reflectivity by area |

## 5.7 runs group

### Storm grid 3-D runs

| Type | Name | Description |
|------|------|-------------|
| int32 | run_ix | Starting x coord for run in grid location |
| int32 | run_iy | Starting y coord for run in grid location |
| int32 | run_iz | Starting z coord for run in grid location |
| int32 | run_len | Length of run in grid counts |

## 5.8 - proj_runs group

### Storm projected area grid 2-D runs

| Type | Name | Description |
|------|------|-------------|
| int32 | run_ix | Starting x coord for run in grid location |
| int32 | run_iy | Starting y coord for run in grid location |
| int32 | run_len | Length of run in grid counts |

## 5.9 tracks group

### Tracking header

| Var type | Var name | Description |
|----------|----------|-------------|
| int32 | n_complex_tracks | number of complex tracks in data set |
| int32 | n_simple_tracks | number of simple tracks in data set |

### Tracking algorithm parameters

| Type | Name | Description |
|------|------|-------------|
| int32 | max_children | max number of children in a storm split (8) |
| int32 | max_parents | max number of parents in a storm merge (8) |
| fl32 | forecast_weights[10] | weights for forecast regression |
| fl32 | weight_distance | weight for distance moved in tracking algorithm |

| | | |
|---|---|---|
| fl32 | weight_delta_cube_root_volume | weight for delta of cube root of volume in tracking algorithm |
| fl32 | merge_split_search_ratio | ratio of search radii used when searching for mergers or splits |
| fl32 | max_tracking_speed | km/hr |
| fl32 | max_speed_for_valid_forecast | km/hr |
| fl32 | parabolic_growth_period | number of seconds for positive growth - this is used with the PARABOLA trend growth forecast |
| fl32 | smoothing_radius | km |
| fl32 | min_fraction_overlap | This is the min individual fraction overlap for a valid match |
| fl32 | min_sum_fraction_overlap | When determining the overlap of a storm at successive times we sum the overlap as a fraction of the storm area at time1 and time2. This is the min sum for a valid match |
| int32 | scale_forecasts_by_history | flag for scaling the forecasts based on the ratio of the history to min_history_for_valid_forecast |
| int32 | use_runs_for_overlaps | flag for using runs to find the overlaps for mergers and splits |
| int32 | grid_type | See section 5.2 |
| int32 | nweights_forecast | number of weighted points to be used in the forecast data |
| int32 | forecast_type | FORECAST_BY_TREND = 1 FORECAST_BY_PARABOLA = 2 FORECAST_BY_REGRESSION = 3 |
| int32 | max_delta_time | secs |
| int32 | min_history_for_valid_forecast | secs |
| int32 | spatial_smoothing | TRUE or FALSE |

## 5.10 complex group

## complex_track_nums array

The **complex_track_nums[n_complex_tracks]** array holds the track number for each complex track. This is a contiguous array with no gaps. The logical length is n_complex_tracks. The actual length in the NetCDF file may exceed the logical length.

## Complex track properties

All other arrays in the complex group have a logical length of **n_simple_tracks**. These are sparse arrays, with gaps because some complex tracks have more than one simple track. To locate the complex properties look up the **complex_track_num** in the **complex_track_nums[]** array, and then use that as the index into all of the other arrays in this group.

| Type | Name | Description |
|------|------|-------------|
| int64 | start_time | time for first scan of complex track |
| int64 | end_time | time for last scan of complex track |
| int32 | complex_track_num | number of this complex track |
| int32 | n_simple_tracks | number of simple tracks in this complex track |
| int32 | start_scan | start scan number of complex track |
| int32 | end_scan | end scan number of complex track |
| int32 | duration_in_scans | duration in scans for track - 1 or more |
| int32 | duration_in_secs | duration in seconds for track |
| fl32 | volume_at_start_of_sampling | km3 - volume at start of track |
| fl32 | volume_at_end_of_sampling | km3 - volume at end of track |
| int32 | n_top_missing | number of scans for which top was not completely sampled |
| int32 | n_range_limited | number of scans for which storm sampling was incomplete due to range |
| int32 | start_missing | flag to indicate storm existed when sampling began - start volume stored |
| int32 | end_missing | flag to indicate storm existed when sampling ended - end volume stored |
| int32 | n_samples_for_forecast_stats | number of samples used to compute statistics for forecast verification |

## 5.11 simple group

## Simple track properties

| Type | Name | Description |
|------|------|-------------|
| int64 | start_time | for the simple track only |
| int64 | end_time | for the simple track only |
| int64 | time_origin | time for first storm in any branch which led to this entry |
| int64 | last_descendant_end_time | end time for the last descendant of this storm |
| int32 | duration_in_secs | duration of this simple track in secs |
| int32 | duration_in_scans | duration of this simple track in scans |
| int64 | first_entry_offset | file offset of first track entry struct |
| int32 | simple_track_num | number of this simple track |

| int32 | complex_track_num | number of track complex if track is part of a complex |
|---|---|---|
| int32 | nparents | number of parents, max 8. In the case of a merge this will be > 1. |
| int32 | nchildren | number of children, max 8. In the case of a split this will be > 1. |
| int32 | parent[8] | simple track numbers of nparents |
| int32 | child[8] | simple track numbers of nchildren |
| int32 | last_descendant_simple_track_num | simple track num of last descendant of this storm |
| int32 | start_scan | first scan in this simple track |
| int32 | end_scan | last scan in this simple track |
| int32 | last_descendant_end_scan | end scan of last descendant of this storm |
| int32 | scan_origin | scan for first storm in any branch which led to this simple track |
| int32 | history_in_scans | number of scans of history data for making forecast - in the case of a merger or a split this will be longer than the duration in scans |
| int32 | history_in_secs | time in secs since the start of the earliest branch of this storm |

## 5.12 entries group

There is a 1-to-1 correspondence between track entries and storms. Each entry in a simple track is associated with a single storm in a single scan.

## Track entry

| Type | Name | Description |
|---|---|---|
| int64 | time | time for this entry - i.e. scan time |
| int64 | time_origin | time for scan of first storm in any branch which led to this entry |
| int32 | scan_num | scan number in storm file |
| int32 | storm_num | storm number in scan |
| int64 | prev_entry_offset | offset to previous entry in track (doubly linked list) |
| int64 | this_entry_offset | offset to this entry |
| int64 | next_entry_offset | offset to next entry in track (doubly linked list) |
| int64 | next_scan_entry_offset | offset to next entry in this scan |

| int32 | scan_origin | scan number for first storm in any branch which led to this entry |
|---|---|---|
| int32 | simple_track_num | simple track number for this entry |
| int32 | complex_track_num | complex track number for this entry |
| int32 | history_in_scans | number of scans of history data for making forecast - in the case of a merger or a split - this will be longer than the duration in scans |
| int32 | history_in_secs | time in secs since the start of the earliest branch of this track |
| int32 | duration_in_scans | duration of the simple track in scans |
| int32 | duration_in_secs | duration of the simple track in seconds |
| int32 | forecast_valid | flag to indicate if the forecast is valid |