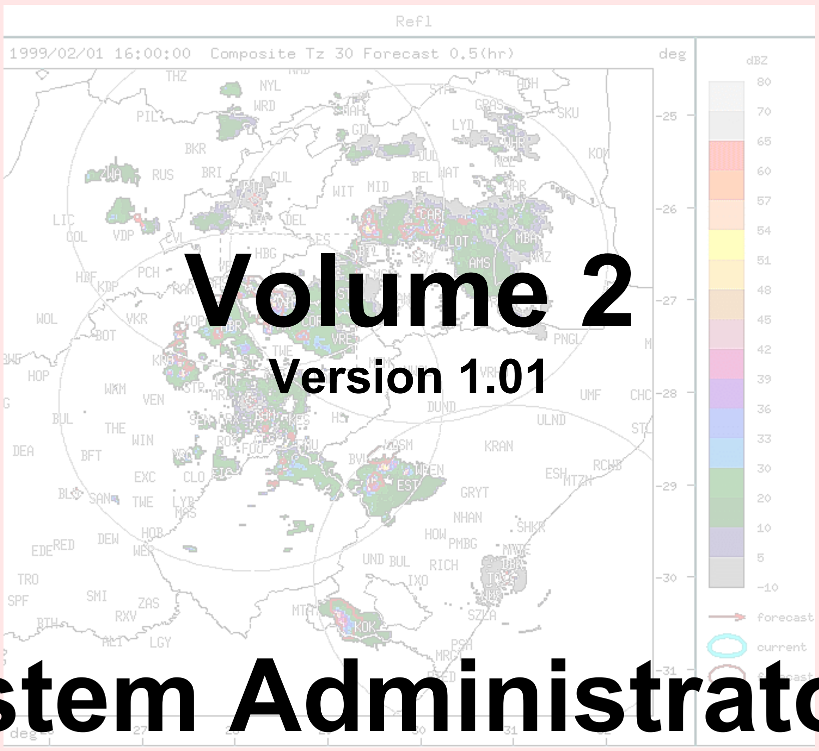
TITAN DOCUMENTATION



Volume 2

Version 1.01

System Administrator s Guide

March 2000

INDEX

Compiled by

Marion Mittermaier, Karel de Waal and Mike Dixon
©2000

METSYS
Private Bag X15, Bethlehem, 9700
Tel : +27 58 303 5571 Fax : +27 58 303 2352
e-mail : marion@metsys.ofs.gov.za , dewaal@metsys.ofs.gov.za, dixon@rap.ucar.edu

1. INTRODUCTION

This guide is meant to equip the system s administrator to perform routine operational tasks on the TITAN
system as a whole. Being a LINUX platform some basic UNIX background is desirable but not necessary. This guide is meant to enable the UNIX layman to also perform the necessary checks as well. With this in mind :

section 2 is devoted to providing a list of basic UNIX commands which are necessary to do the job;
section 3 discusses the new mdv directory structure;
section 4 will discuss the software system and the way in which the hardware is connected.

It must be stressed that this is not an installation guide. It is meant to provide :
" an overview of the system,
" how to make calibration related changes and
" how to fine tune the system for specific needs.

2. BASIC UNIX COMMANDS

This section is not intended to be a complete LINUX manual. For a complete reference to the LINUX operating system the book **LINUX in a nutshell** (2nd edition) from the O Reilly series can be highly recommended.

Below is a very short list of the commands needed to perform the necessary tasks :

`ls`	short list
`ls -l`	long list
`pwd`	print current directory
`date`	check the system time
`ps -x`	list of all the processes currently running on the system with their PIDs (process id s)
`df -k`	check the available/used disk space
`rm -rf`	remove [-r recursive][-f without asking for confirmation] file(s)/directories
`jobs`	check processes running in the background
`kill PID`	to kill the process with the id PID

For archiving data the following commands may come in handy :

`gzip -r dir or filelist`	gzip [-r recursive]
`tar cvzf tarfile.tgz filelist`	concatenate and compress files into a tgz file

To put processes in the background attach an `&` to the end of the command line.

Many of the directory names are very long and are therefore aliased and links are created. Typically there is a TITAN_HOME alias which accesses the TITAN root directory, i.e. `cd $TITAN_HOME` will take you there.

3. THE MDV DIRECTORY STRUCTURE

In this section a brief overview of the data directory structure will be given and an indication of what goes where. The alias *$RAP_DATA_DIR* accesses the *$TITAN_HOME/data* directory.

gure 1
The mdv directory structure with sub-directories and filename standards.

Figure 1 is a schematic of the new mdv directory structure. Typically this data directory tree will fall under a site s home directory, e.g. /hd/titan5/bethlehem/data/mdv/cdata for the Bethlehem radar. All the processed radar and aircraft data reside in this directory structure. For the mdv files, the directory names correspond to the date, and file names in a given day s directory will be stored using the time from midnight to midnight local time or GMT, depending on the set-up. Aircraft tracks are stored in spdb (symbolic product data base) format. Alternatively they can be stored as ascii files (older systems). The date acts as the file name. Each day s tracks have an index and a data file. The storm tracks file are also stored with the date assigned as the name. Four files are generated for each day, header and data files for the storms information (*.s*) and for the track information (*.t*).

4. THE SYSTEM

4.1 How does it all fit together ?

TITAN (Thunderstorm Identification Tracking Analysis and Nowcasting) is the collective name given to a whole list of processes running in the background. The list of processes may vary from place to place depending on the specific needs and uses of an installation. The descriptions given below are discussed from a multi-functional perspective as this reference does not cater for any specific group of users. Any software developed in this environment can be made to link into the interface and be displayed using *rview*. With radar data as the basic input many derived products, such as precipitation, storm severity structure (SSS), vertically integrated liquid water (VIL), bright band algorithms and hail kinetic energy (HKE) fields can be incorporated. In addition any latitude-longitude values can be superimposed to construct the background for the display. In this manner aircraft tracks can also be displayed.

For executables the following general rules apply :

1. Binaries/executables for specific processes can be found in the *$WORK/bin directory*. These executables are not normally accessed from here.
2. All **start** and **kill** scripts are found in the *$TITAN_HOME/bin* directory. These are used operationally. **Check that the paths are set correctly in all the scripts !!**

For process fine tuning :

1. All parameter files are found in the *$TITAN_HOME/params* directory. **Each process has a params file.** Changes in these can made without the need for recompilation of programs. More on this subject in section 4.3.
2. Color scales and background files can be added or modified in the *color_scales* and *maps* directories under *$TITAN_HOME*.

A word of warning - it is advisable to make back-ups of these files before fine tuning is done.

4.2 Data flow

The data flow that is described below is applicable to the RDAS-TITAN set-up. Figure 2 shows a data flow diagram of the system.

Figure 2
Data flow and process diagram.

Data transfer between the RDAS and TITAN PCs is accomplished using sockets. Raw volume scan data is cartesianized by one of two processes, either a continuous update mode or after a volume scan has been completed. At this point cartesianized mdv files have been created and stored and are ready for display. Depending on the processes running other derived products will be generated in mdv-format and stored, also ready for display. To be able to display any data a corresponding MDV-server must be active. These servers will ensure that the *rview* display will be updated with data. The *rview* display works with port numbers. All fields displayed by *rview* are also listed in the *rview* params file, right at the end. The port numbers in the *MDV-server* params file and the *rview* params file must correspond.

4.3 Real-time processes

Table 1 gives a list of processes that ought to be running for a typical installation. Instead of *dva_cart* and *dva_ingest* (for South Africa, updating end-of-scan mode) the ingest processes will be the triplet : *bprp2gate, polar_ingest* and *polar2mdv*, as well as *cart_slave* which provides the means for continuous updating, all started with the *start_titan_ingest* script in the *$TITAN_HOME/bin* directory. Furthermore *start_titan_tracking* will ensure that the tracking programs *storm_ident* and *storm_track* are running and that the *track_server* is also active, *start_titan_display* usually controls the *rview* and *time_hist* processes. When aircraft data are to be displayed the *start_ac_ingest* script must also be executed. All these scripts are usually encapsulated in an overall **START** script which will start all the above in one go.

Table 1

List of running processes as displayed using `ps -x`.

```
PID TTY STAT  TIME COMMAND
  580  p0 S    1:01 bprp2gate -params bprp2gate.ops
  598  p0 S    0:48 servmap -quiet
  608  p0 S    1:44 procmap -quiet
  657  p0 S   23:33 cart_slave -params cart_slave.ops
  758  p0 S  149:07 PrecipAccum -params PrecipAccum.1hr
 6594  p0 S    2:59 storm_ident bash /hd1 -params storm_ident.ops
 6597  p0 S    0:00 storm_track -params storm_track.ops
12549  ?  S    1:16 track_server -params track_server.ops
21672  ?  S   12:23 clutter_remove -params clutter_remove.ops
21684  ?  S   83:50 mdv2vil -params mdv2vil.ops
21769  ?  S  363:44 MDV_server -params MDV_server.ops
21782  ?  S    1:25 MDV_server -params MDV_server.precip
21791  ?  S    1:24 MDV_server -params MDV_server.vil
23647  ?  S   65:51 rview -params rview.ops
23650  ?  S    0:12 time_hist -params time_hist.ops
29083  ?  S  903:58 polar_ingest -params polar_ingest.ops
29095  ?  R  243:27 polar2mdv -params polar2mdv.ops
30000  ?  R    1:05 wmi_ac_ingest -params wmi_ac_ingest.ops
```

If any one of these processes are not responding, it may be necessary to kill that process and restart it. Only for extreme cases may it be necessary to run the complete **KILL** script and start up again.

servmap is the 'server mapper'. Every 2 minutes, each server registers with *servmap*, giving information such as host and port and updating information such as data times. The clients (such as displays) use servmap to determine the host and port for data servers.To view the current list of active servers the command `print_servmap` can be used.

On the other hand, *procmap* is the 'process mapper'. Every minute, each process registers with *procmap*, so that we know whether the process is running properly or not. The script *procmap_auto_restart* uses *procmap* to determine which processes requires restarting. All the active processes can be viewed using the `print_procmap` command. Below is an example of such a list.

```
PROCS REGISTERED - localhost - Wed Mar 15 14:22:30 2000
Uptime: 3333 secs

Name         Instance     Host       User       Pid
====         ========     ====       ====       ===
MDV_server   ops          fape_titan titan5     14123
MDV_server   precip       fape_titan titan5     14134
MDV_server   sss          fape_titan titan5     14156
MDV_server   vil          fape_titan titan5     14145
StormIdent   ops          fape_titan titan5     14567
clutter_remo ops          fape_titan titan5     694
dva_cart     ops          fape_titan titan5     682
dva_ingest   ops          fape_titan titan5     670
mdv2vil      ops          fape_titan titan5     718
precip_map   precip_accum fape_titan titan5     14602
rview        ops          fape_titan titan5     14330
servmap      servmap      fape_titan titan5     14087
storm_track  ops          fape_titan titan5     14572
time_hist    ops          fape_titan titan5     14341
track_server ops          fape_titan titan5     14173
```

4.4 Dealing with calibration-related changes

All the parameter files are found in the *$TITAN_HOME/params* directory. The params files are well documented internally but a number of pointers and helpful suggestions are supplied here. We start of this section by providing some golden rules.

Some golden rules !!!

1. After a calibration the **radar constant is not changed automatically** in TITAN. It must be changed manually in the params files (either *DVA* processes or *bprp2gate*).
2. **ANY** changes to the RDAS configuration (such as the skip, scan geometry, bin size etc) should be translated to the ingest program suite (either *DVA* processes or *bprp2gate, polar_ingest, polar2mdv, rc_table_generate*).

Where do I change the radar constant ?

```
/*
 * Radar constant.
 * Value of radar constant for Met type radar.
 *
 * Type: double
 * Default: -157
 */
radar_constant = -154.63654;
```

The radar constant features in the *bprp2gate* params file. A tip here is to do a `grep` of the term or expression you wish to change. This command will provide you with all the occurrences of the term in a given directory. For instance if the radar constant needs to be changed, `grep radar_constant *.ops` will provide a list of all the places where the radar constant is defined.

How do I compute a new clutter map ?

After calibrations it may be necessary to re-compute a clutter map. Here are a list of steps to follow :

a. Set parameters in *polar2mdv* so that no clutter will be removed during processing.

```
polar2mdv.remove_clutter: false
```

b. Set the signal-to-noise ratio rejection threshold 5 dB above the normal value. This makes sure you see all of the clutter.

```
polar2mdv.check_sn: true
polar2mdv.sn_threshold: ?? <--- normal value + 5
```

c. Collect clear-weather volumes. Run the system to collect 20 - 30 radar volumes in clear weather. A few moving echoes are OK, because the clutter procedure uses a median filter.

d. Copy these volumes into the *$RAP_DATA_DIR/mdv/clutter* sub-directory.

e. Now create the clutter tables by running the script *create_clutter_table*. The script triggers the processes *clutter_compute* (which calculates a median clutter map) and *clutter_table_generate* (which calculates a table used for the clutter removal on the fly option). This will load up the clutter table in the *$TITAN_HOME/tables* directory.

f. View the clutter data to make sure it is OK by running the script *view_clutter*. This will bring up a *rview* window showing the median clutter map.

g. Reset the parameters in *polar2mdv* back to performing clutter removal.

```
polar2mdv.remove_clutter: true
polar2mdv.check_sn: true
polar2mdv.sn_threshold: ?? <--- normal value
```

Now normal operations can continue.

4.5 Changing process parameters

Sometimes it is necessary to tune processes after installation. For weather modification programmes aircraft may change from season to season, i.e call signs need to be changed etc. Below are answers to some FAQs.

How do I change the properties of or add an aircraft ID to the display ?

In the *wmi_ac_ingest* params file all the call signs of aircraft in use need to be listed **in quotes**.

```
/*
 * Callsign array.
 * Array of valid callsigns.
 *
 * Type: string
 * Default: "none"
 * Array elem size: 4 bytes
 * Array has no max number of elements
 */
callsigns = {"JRA","JRB"};
```

To display all these aircraft these call signs have to be translated to the *rview* params file as well. All the aircraft IDs in use are listed sequentially. For instance :

```
rview.ac_posn_idents: 2
rview.ac_posn_idents : JRA JRB
rview.ac_posn_colors: white orange
```

Remember that if you had the *rview* running whilst you were making these changes, you need to quite *rview* and restart for the changes to take effect.

Remember also that the script *start_ac_ingest* must be run to begin data ingest from the base station.

How do I add a map/towns/boxes/borders to my display ?

All the map layers are specified in a *.conf file and this file is referred to in the *rview* params file. New layers are added by adding a line to the file. The header of the configuration file is given below. As well as an example map entry.

```
# maps configuration file for rview
#
# There is one line per map file.
#
# Fields per line are as follows:
#
#   1. Label (string)
#   2. X-color (string)
#   3. X-width (int)
#   4. X-font (string)
#   5. Ps-fontsize (int)
#   6. Ps-linestyle (width dash_length space_length) (int int int)
#   7. Limited (1 or 0) - plotted using middle button selection
#   8. All (1 or 0) - plotted using right button selection
#   9. File path (string)
#
#-----------------------------------------------------------------------------
#    1         2      3     4      5  6  7   8     9
#
Towns      gray80    1    Fixed   8  1  1   1   $(TITAN_HOME)/maps/merge_towns.map
```

Towns and points are declared as `type SIMPLELABEL lat lon identifier`.
Polygons are declared as type `POLYLINE name no_of_points+1`, followed by the lat-lon co-ordinates, ending with a -1000.0 -1000.0, to indicate a break or pen-up command. Remember that such a file must also have a `MAP_NAME name` in the first line of the file.

In this manner lat-lon boxes, borders, towns etc can be added to the background. Colors of these features are specified in the *.conf file.

5. CONCLUSION

This guide is by no means complete. We hope to make it as comprehensive as possible, given time and feedback from TITAN users. It is hoped that this small beginning will already help TITAN users in solving minor problems.

ACKNOWLEDGEMENTS

Many thanks to our Bethlehem colleagues who have also contributed to the compilation of the guide.