HF-Radar Network
Node Reference Guide

Mark Otero
July 31, 2009

Coastal Observing Research & Development Center
Marine Physical Laboratory
Scripps Institution of Oceanography
motero@mpl.ucsd.edu
www.cordc.ucsd.edu

Table of Contents

INTRODUCTION **1**

SYSTEM CONFIGURATION **3**

SuSE Linux 3
RedHat Enterprise Linux 4

SOFTWARE **7**

3dm2 7
Antelope 7
Documentation *7*
Startup and Shutdown *7*
Communication Requirements *8*
The Real-Time (rt) User Account *8*
Real-Time Directory *8*
Licensing and Upgrades *8*
MATLAB 9
Configuration *9*
Toolboxes *9*
Licensing *9*
Upgrades *9*
NetCDF 10
Installation *10*
Upgrades *10*

NEAR REAL-TIME VECTOR (RTV) PROCESSING **11**

New Site Additions 11
Site Decommissions 12

APPENDIX A: REAL-TIME DIRECTORY CONTENTS **13**

rtexec.pf 13
Processes *13*
Run *13*
Shutdown_order *13*
startup_shutdown_email *14*
status_email *14*
crontab *14*
email_incident_reports *14*
/bin 15
/db 15
/dbmaster 15
/logs 15
/orb 16
/pf 16
/rtsys 16
/state 16

APPENDIX B: COMMON SCRIPTS BY DIRECTORY **17**

/data/HFRadar/HFRNet/bin 17
MATLAB.pl *17*
hfrnet_newRTVs.pl *17*
/home/rt/matlab/Toolbox/HFRadar/drivers 17
tuv_driver_HFRNet.m *17*
currentAvg_driver_HFRNet.m *17*
/home/rt/matlab/Toolbox/HFRadar/data_defs 17

RealTimeParameters_*.m *17*

APPENDIX C: COMMON COMMAND REFERENCE **19**

RTEXEC 19
ORBSERVER 19
ORB2ORB 19
ORBSTAT 19
ORBHFRADAR2DB 20
DBE 20
DBDESCRIBE 21

INTRODUCTION

The backbone of the HF-Radar Network consists of Portals and Nodes. A Portal provides a method for HF-Radar data to enter into the network. A Node is a machine collecting data from any number of Portals (Figure 1).

Portals simply accept or acquire data and serve it through an orbserver. There are two methods currently in use for introducing data into the network, (1) by acquisition from local disk and (2) from remote hosts over SSH protocol. Local acquisition has been used for initial development efforts. However, data acquisition over SSH is designed to be the operational protocol for the network. Local data acquisition is retained as a back-up method for data access from sites that don't fulfill requirements for acquisition over SSH.

Nodes acquire data via orb2orb from Portals or other Nodes and build a database of radial files. Nodes may also do additional processing on the radials such as total vector production.

This documentation provides information on the configuration and operation of a HF-Radar Network Node.

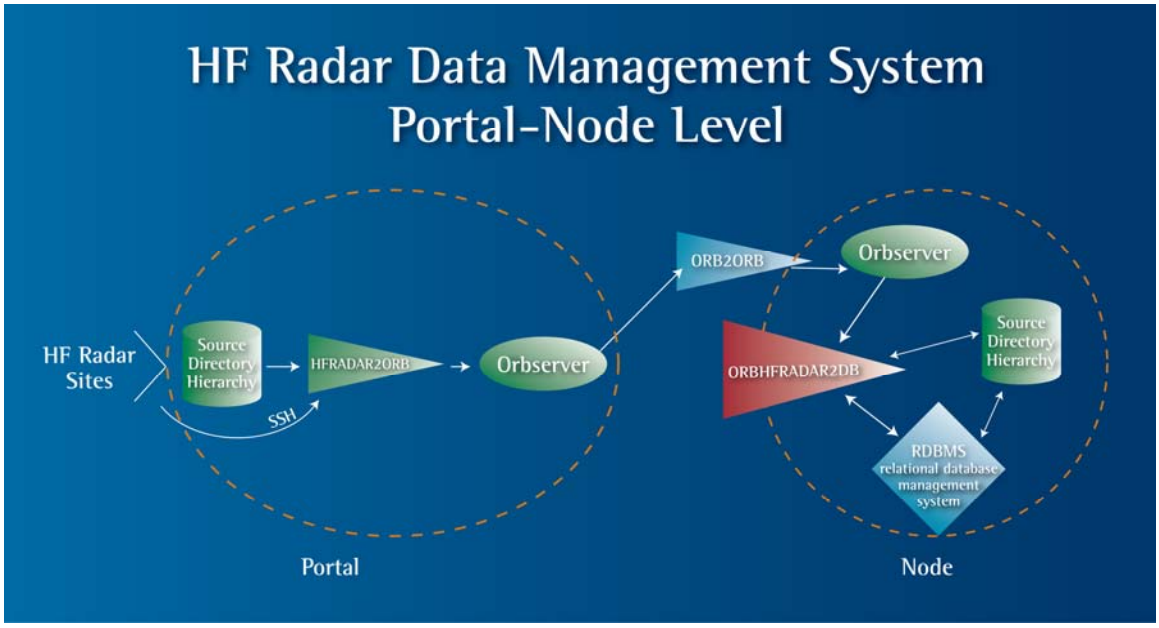

Figure 1. Data flow through the HF-Radar Network. Portals acquire radials through hfradar2orb, an executable that retrieves files over SSH from a remote host or through a local source directory. Once files are obtained they are placed in the ORB(server). The Portal's ORB makes data available to any allowed clients for data retrieval via orb2orb. Nodes concentrate data from any number of Portals by using orb2orb to copy packets from the Portal's ORB to their own local ORB. As packets arrive in the Node's local ORB, orbhfradar2db reads the packets and builds a database of radial files consisting of a Datascope database and a source directory hierarchy containing the radial files (as Binary Large OBjects or BLOBs).

SYSTEM CONFIGURATION

The HF Radar Network is built on a software application called Antelope which, for x86 architectures, is tested on the SuSE distribution of Linux. For this reason, SuSE was used in the original development of the network. However, the RedHat Enterprise Linux (RHEL) distribution has received wide support among system administrators and is now the preferred distribution for operational use in the network. Original SuSE distributions are being migrated to RHEL. System configurations for both operating systems are described below.

SuSE Linux

The following services are configured through YaST:

→ System
  → System Services (Runlevel)

Disable:	SUSEfirewaqll2_setup	nfs
	nfsboot	portmap
	cups	
Enable:	xntpd	

The above changes can be made in 'simple mode'. However, you should enter 'expert mode' to ensure that the SuSEfirewall2_setup is disabled at boot and any other run levels. *Please note: SuSEfirewall may be used by local administrators instead of iptables as discussed here.*

Edit `/etc/ssh/sshd_config` by changing the line `#PermitRootLogin yes` to `PermitRootLogin no`.

A packet filtering firewall using iptables should be used if no other firewall is in place. *NOTE: This is only one recommendation. Establishing the proper security policy for the machine is the responsibility of the local network administrator.* The basic policy for the firewall described here is to drop all forwarded packets. All incoming packets are also dropped by default with exceptions for connections that have been established, related to established connection or specifically allowed. All outgoing packets are allowed. As an example, two scripts used are `ipTablesRuleset.sh`:

```
#!/bin/bash

################################################################
# Ruleset to use for iptables on HF-Radar Network Data Portals #
################################################################

# Flush all rules (chains) from the (default = filter) table
iptables -F

# Set the policy for the chain (default action)
iptables -P INPUT DROP          # Set default behavior for INPUT chain
iptables -P OUTPUT ACCEPT       # Set default behavior for OUTPUT chain
iptables -P FORWARD DROP        # Set default behavior for FORWARD chain

# Allow Established & Related Connections for Input
iptables -A INPUT -p tcp  -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p udp  -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
iptables -A INPUT -p icmp -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allowed ICMPs
iptables -A INPUT -p icmp --icmp-type 0  -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3  -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 4  -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 5  -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8  -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT

# Allow Loopback
iptables -A INPUT -i lo -p ALL -d 127.0.0.1 -j ACCEPT

# Allow connections to ORB from Specific IPs
iptables -A INPUT -i eth0 -p tcp --dport 6580 -s 207.174.76.128 -j ACCEPT #brtt.com
iptables -A INPUT -i eth0 -p tcp --dport 6580 -s 207.174.76.144 -j ACCEPT #brtt.com
iptables -A INPUT -i eth0 -p tcp --dport 6580 -s 209.193.47.96  -j ACCEPT #Lindquist
iptables -A INPUT -i eth0 -p tcp --dport 6580 -s 132.239.4.58   -j ACCEPT #sccoos-hf-orb-1

# Allow SSH from specific IPs
iptables -A INPUT -i eth0 -p tcp --dport ssh -s 132.239.128.229 -j ACCEPT #sandbox
iptables -A INPUT -i eth0 -p tcp --dport ssh -s 132.239.127.145 -j ACCEPT #breather
```

and `ipTablesRestoreRuleset.sh`:

```
#!/bin/bash
iptables-restore < /etc/sysconfig/iptables
```

Installation of the firewall is as follows:

1) Copy `ipTablesRestoreRuleset.sh` and `ipTablesRuleset.sh` to `/root/bin`.
2) Run `ipTablesRuleset.sh`.
3) Export the ruleset using
   `# iptables-save > /etc/sysconfig/iptables`
4) Make sure both scripts have permissions 0700 or 0740.
5) Place a copy of `ipTablesRestoreRuleset.sh` in `/etc/init.d/`
6) Start YaST and go to System > System Services (Runlevel). Enter expert mode and, for `ipTablesRestoreRuleset.sh`, check runlevels B, 3 & 5.
7) Reboot & verify that the ruleset has been loaded by typing `iptables –L`.
8) Re-test that you can still SSH in from allowed IP's, that DNS & time-servers can be reached & that a port scan with nmap returns nothing.

To add additional source IP's temporarily, just issue commands through iptables. For example, to allow download from a specific site, type `iptables –A INPUT –i eth0 –p tcp –s xxx.xxx.xxx.xxx –j ACCEPT`. Then, to revert back to the original ruleset, run `ipTablesRestoreRuleset.sh`. To permenantly allow other IP's, edit `ipTablesRuleset.sh` and follow steps 2 – 3 above. Reboot to verify changes if possible.

RedHat Enterprise Linux

The following points cover the basics for RedHat Enterprise Linux configuration of Portals and Nodes

- Default run level should be set to 3
- Time zone should be set to GMT. To set the time zone, do the following as root
  1. Stop Antelope (if running)
  2. `cp /usr/share/zoneinfo/GMT /etc/localtime`

3. Edit `/etc/sysconfig/clock` to read:

```
ZONE="GMT"
UTC=true
ARC=false
```

4. `/sbin/hwclock --systohc`
5. Restart Antelope

- Limit services to a minimum. Use `chkconfig` to disable unnecessary services including:

  cups portmap xinetd nfsboot nfs

- Enable `ntpd` at runlevel 3 & 5 using `chkconfig`
- Disable remote root login over SSH by editing

  `/etc/ssh/sshd_config`

  And change to read

  `PermitRootLogin no`

SOFTWARE

3dm2

3dm2 is installed on Ashford Comupter Consulting servers. This software is used to provide disk health diagnostics and may be used as an aid in remote diagnostics. To enable remote access, edit

```
/etc/3dm2/3dm2.conf
```

To read

```
RemoteAccess 1
```

Then restart 3dm2 to commit the change with the command

```
/etc/init.d/tdm2 restart
```

Web access is then available through

https://*hostname*:888

Recommended settings:

- As long as the server is under contract, retain Peter Ashford on the email list at the info level (ashford@whisperpc.com)
- Make sure that under 'Management' > 'Scheduling' > 'Verify Tasks' that tests are run once a week and 'Follow Schedule' is enabled. Allow 4 hour duration for Controller ID 0 and 6 hours for Controller ID 1.
- Under the 'Self-test Tasks' drop-down, verify that both tasks 'Upgrade UDMA mode'& 'Check SMART Thresholds' are enabled for each night.

3dm2 is available through chkconfig

```
chkconfig --list tdm2
```

Antelope

Antelope software is used at the core of the HF-Radar Network. It is comprised of a system of software modules that implement data acquisition, buffering, distribution and archiving. Antelope is installed in `/opt/antelope/`*`ver`* (`$ANTELOPE`) and uses its own installations of some languages (e.g. Perl) for consistency.

Documentation

Antelope documentation is located in `$ANTELOPE/doc` and executables are documented in man pages. Additional documentation is available through BRTT's website (www.brtt.com).

Startup and Shutdown

Antelope is registered with `chkconfig` and is configured to start on boot. Antelope will also attempt to shut down cleanly upon receiving shutdown or reboot signals. However, because processes may not have enough time to exit cleanly, Antelope should be stopped manually whenever possible. As either user rt or root, stop the real-time system by issuing:

```
# /etc/init.d/antelope stop
```

You will be prompted for an explanation of why you're shutting the system down. Entering something will help re-trace steps in case problems occur. It may take several minutes for processes to complete. MATLAB jobs are not terminated when Antelope is shut down. If you're in a hurry to shut down, just wait until orbserver, orbhfradar2db and orb2orb processes are terminated before shutting down. Otherwise, wait until the MATLAB jobs run to completion. If you shutdown while MATLAB jobs are running, you can adjust the state files according to the logs to ensure that processing is completed. One method for monitoring processes is the `ps` command, for example `ps –ef` will show all processes with full format listing.

Unless Antelope is disabled through `chkconfig` or otherwise, it will come up on boot and no further action is needed. However, it is safest to check that the logs are clean and data is flowing when the system boots.

Communication Requirements

Inbound connections over SSH are required for remote maintenance and upgrades of code. Data communications via orb2orb are normally initiated from the Node over port 6580.

The Real-Time (rt) User Account

The user rt (UID 3710) is the primary account for managing the real-time system. The real-time user's .cshrc file sources `/opt/antelope/`*`ver`*`/setup.csh` which sets up the real-time environment. The real-time user is often given `sudo` access for editing init files and for performing software upgrades. Other accounts may be given access to Antelope for management of the HFRNet real-time system. All accounts using antelope should belong to the antelope group (GID 2020).

Real-Time Directory

The HF-Radar Network is contained in a real-time directory, `/data/HFRadar/HFRNet`. The real-time directory contains the ORB, the radial file database, logs and all of the files specific to the real-time system. For this reason, files and directories within the real-time directory should not be modified in any way without knowing the consequences of such actions. Additional data may be stored on the Node in `/data` or `/data/HFRadar` if appropriate, but not in the real-time directory. See Appendix A for more information on the real-time directory.

Licensing and Upgrades

Antelope upgrades are required to keep code consistent across the network, enable further development and ensure reliability. Licensing and upgrades will be performed by HF-Radar Network administrators (HFRNetAdm@mpl.ucsd.edu) and notification will be given prior to starting upgrade work. Please be aware that local customizations may require maintenance following any given upgrade.

MATLAB

MATLAB is currently used for processing radial vectors to total vectors and may be used for other applications as needed.

Configuration

MATLAB is installed in `/opt/matlab`*`ver`* and uses an individual license which is restricted to the user rt. Flexnet is used for license management for versions prior to 2008a (7.6) and has been configured to start on boot (registered with `chkconfig`). User rt has an alias to MATLAB through its `.cshrc` file which runs MATLAB without the desktop environment.

Toolboxes

The following toolboxes have been licensed through MATLAB: Mapping, Image Processing, Signal Processing, Symbolic Math and Statistics. In addition, the Antelope Toolbox for MATLAB (ATM) is installed with Antelope and is used for integrating Antelope's real-time system with total vector processing in MATLAB.

Additional third-party toolboxes are kept in individual users' home directories in `/home/`*`user`*`/matlab/Toolbox`. User rt has the HFRadar toolbox which is used for near-real-time processing of radial velocities to total vectors. The HFRadar toolbox was built off an early beta version (Fall 2005) of HFR_Progs and was modified for integration with Antelope and optimized for processing on large grids.

There are two toolboxes used for creating and interfacing with NetCDF files; MexNC and SNCTOOLS. MexNC is a mex-file interface to NetCDF files for MATLAB and has a roughly one-to-one equivalence with the C API for NetCDF. SNCTOOLS is a set of m-files that sits on top of MexNC and provides a user-friendly interface for reading and writing NetCDF files.

Licensing

MATLAB is normally installed with a single-user stand alone license with user rt as the (only) licensed user. As the license implies, only user rt is able to run MATLAB with this license.

Upgrades

MATLAB upgrades normally do not affect data flow through the network in any way. For this reason, local administrators may upgrade MATLAB on their own. However, knowledge of the way in which data is processed in MATLAB in conjunction with the real-time system is required. MATLAB upgrades typically involve:

- Obtaining a license (contact HFRNetAdm@mpl.ucsd.edu)
- Access to root for installation and init script editing
- Stop RTV processing
- Stop license manager (versions prior to 2008a only)
- Install MATLAB
- Update MATLAB.pl in `/data/HFRadar/HFRNet/bin`
- Remove old version of MATLAB once the new version is tested

NetCDF

Total vectors are saved in NetCDF format for data distribution and interoperability.

Installation

Installations to date have been successful following the 'Quick Instructions for Installing NetCDF on Unix' instructions posted on Unidata's NetCDF download site (http://www.unidata.ucar.edu/software/netcdf/) as well as in the INSTALL file included with the source. No customizations are performed during the installation.

Upgrades

Whether HF-Radar Network or local administrators will maintain NetCDF is uncertain because applications are still taking shape along with National file format standards. Please contact HF-Radar Network administrators if you're considering upgrading NetCDF or changing the file format on your Node.

Near Real-Time Vector (RTV) Processing

Radial files are obtained using Antelope through the HF-Radar Network's real-time system. Data are obtained from Portals and stored in a local Datascope database. Once the files are in the database, they are available for local processing. Total vector processing is done in MATLAB using the HFRadar toolbox.

Near real-time total vector processing is scheduled through the HF-Radar Network's real-time system cron, not the operating system's cron. The real-time system executive parameter file (`rtexec.pf`) contains parameterizations for the entire real-time system. MATLAB calls for RTV processing are located within the crontab array in `rtexec.pf` (see Appendix A or rtexec(1) for more information on `rtexec` and the `crontab` array).

New Site Additions

You may want to add new sites as they become available in the network for RTV processing. In order to do this, you'll need to edit parameter files for RTV processing. The only edit needed to add a site is to make an entry for it in the SitePrefs cell array for each appropriate RealTimeParameters file (Appendix B). Standard processing currently has ~25 MHz sites and higher contribute to the 1km processing, ~12 MHz sites and higher contribute to the 2km processing and all sites contribute to 6km processing. These guidelines may change and Node operators are free to change local processing as needed.

In addition to adding new sites to RTV processing, you may want to add a descriptive name to any new sites or networks. New sites and networks are automatically populated in the Datascope database. Local users may want to add descriptive names for sites and networks as they become available. The descriptive names are normally disseminated through announcements for new sites. Follow the steps below as user rt to add the descriptive name(s) to the site and network tables. See also Appendix A and C for descriptions of files and commands.

1) Stop orbhfradar2db
    a. Within the Run array of `rtexec.pf`, replacing the '1' (on) with a '0' (off) for orbhfradar2db.
    b. Once the file is saved, watch for orbhfradar2db to stop by monitoring processes or checking the log.
2) Edit the site and network tables
    a. With X11 enabled, change directory to `/data/HFRadar/HFRNet/db` and execute `dbe hfradar` to open up a GUI to the database.
    b. Click on the 'site' button to open the site table.
    c. Under 'Options' enable 'Allow Edits'
    d. Click on the 'staname' field for the site ('sta') row you want to edit. Type the entry for staname in the top left window and click 'ok'. The new entry should now appear in the staname field.
    e. Dismiss the table (bottom of window)
    f. Click on the 'network' button to open the network table.
    g. Under 'Options' enable 'Allow Edits'

h. Click on the netname field for the network (‘net’) row you want to edit. Type the entry for netname in the top left window and click ‘ok’. The new entry should now appear in the netname field.
i. Dismiss the table (bottom of window)
j. Press the ‘Quit’ button in the original window

3) Restart orbhfradar2db
   a. Within the Run array of `rtexec.pf`, replace the ‘0’ with a ‘1’ for orbhfradar2db.
   b. Once the file is saved, watch that orbhfradar2db starts again by monitoring processes or checking logs.

Site Decommissions

Decommissioned sites should be omitted from RTV processing by removing the site entries from the SitePrefs cell array of all relevant `RealTimeParameters_*.m`. files (see also Appendix B). You may also specify an endtime with an appropriate date (typically the current date or date timestamp of the last radial acquired) in the site table of the Datascope database using `dbe`. As outlined in the section describing new site additions, stop orbhfradar2db before manually editing the site table and remember to restart it when you’re finished.

Appendix A: Real-Time Directory Contents

The real-time directory (/data/HFRadar/HFRNet) contains all the settings and data specific to the HF-Radar Network. Contents of this directory that are most relevant to operational use of the data system are highlited below.

rtexec.pf

The real-time data acquisition system executive (rtexec) starts up, shuts down, and monitors the operation of the real-time data acquisition system. The execution of the real-time system is largely dictated by rtexes's parameter file (rtexec.pf) located in the real-time directory. Here we describe the most relevant sections of rtexec.pf in order of appearance. See rtexec(1) more information on rtexec and its parameter file.

Processes

This is a list of tasks which may be run by rtexec; they are started in the same order as they appear in this list. Each item in the list has a name, followed by the execution line as it would be typed on the command line. However, the execution line is interpreted by a shell, so special shell characters must be escaped or quoted to pass them on to the program. Relevant processes listed here are typically orbserver, orb2orb and orbhfradar2db.

Run

This is an array of flags indexed by task name. The corresponding task from the Process list is run only if the flag in Run is non-zero. Rtexec constantly monitors rtexec.pf for any changes and acts on the changes once they are committed (saved). Therefore, manipulating processes through the Run table is an effective way of stopping portions of the real time system without stopping the entire system.

For example, when the database needs manual editing for a new site, orbhfradar2db can be stopped by setting its flag in the Run table to '0'. Because rtexec constantly monitors the parameter file for changes, once the changes are commited rtexec will stop orbhfradar2db. When you're done editing the database, reset orbhfradar2db's run table value to '1'. Once the changes are saved rtexec will restart orbhfradar2db.

Shutdown_order

During shutdown, kill signals are sent to (running) tasks in the order named in this list. Note that the task name need not be related to the program run. The names in Shutdown_order are either task names (like orb2orb_SIO), or program names (like orb2orb). Each line can name multiple tasks, which are killed concurrently during a shutdown. All tasks which match entries on a particular line have either died or been sent kill -9 signals before any tasks from a later line are sent signals.

Processes not listed in the Shutdown table are the last to be sent signals; orbserver is often the last task to be killed.

Usually, the correct shutdown order is:

1) orb readers (like orbhfradar2db)
2) orb writers (like orb2orb)
3) orbserver

startup_shutdown_email
When the system is stopped or started, mail is sent to these email addresses. Please keep HFRNetAdm@mpl.ucsd.edu on this list.

status_email
These addresses receive email when:
1) rtexec declares a task failure as defined by the Failure_threshold parameter and gives up on restarting a task.
2) A task fails with a segmentation fault, bus error or other hardware failure
3) Some limit on resources is exceeded as defined by the Resource Limits parameter.

Please keep HFRNetAdm@mpl.ucsd.edu on this list.

crontab
It may be desirable to run certain jobs on a regular basis, but not continuously. This table is a list of jobs which are run by rtexec. Jobs typically run through the crontab include total vector processing and data export.

The format of each line is the similar to the system crontab, but with two additional leading parameters: a job name, and a timezone code. The log file for a cron job is named "cron:job_name". Another important difference between rtexec's crontab and the system crontab is that rtexec will not start a new cron job while the previous execution of the same name is still running.

The timezone code may be either UTC or LOCAL, and indicates whether the following parameters specify a UTC time or a local timezone time. crontab(1) jobs always use local time, but in a real time system, it's often more convenient to specify a UTC time.

These crontab-like jobs also differ from the system crontab lines in that they are run by rtexec, with the rtexec process environment, and from the same directory. This may simplify the problem of getting a crontab entry to work properly, as missing environment variables are a frequent problem in jobs run from the system crontab.

For a description of the remaining parameters see the system crontab (i.e. man 1 crontab).

email_incident_reports
When a program dies due to a segmentation violation or bus error, an incident report is generated by rtincident(1). A copy of this report is sent via email to the addresses specified in this list. Please keep HFRNetAdm@mpl.ucsd.edu and kent@lindquistconsulting.com on this list.

/bin

The bin directory contains scripts that are used by the real-time system directly. Typical contents include executables run from rtexec's crontab including total vector processing and data export scripts. Other executables written for non-real time applications should either be kept in ~rt/bin or elsewhere.

/db

The db directory contains the Datascope database for radial files and their metadata. All metadata is stored in the Datascope database while the radial files are stored external to the database in the files directory as regular radial files. The database tracks the location of the radial files and the files are formally a part of the database (as Binary Large OBjects or BLOBs) so they should not be moved or modified without making the equivalent modifications in the database.

The database name is typically hfradar. The file hfradar is the database descriptor file and defines the schema for the database (see dbdescribe for more information). hfradar.NAME are individual database tables.

All of the database files are plain ASCII files. It is permissible to view or grep for contents of these files for quick searches. However, you should not modify the files in any way while the real time system is running (or, more specifically, while orbhfradar2db is running) since the files may be written to at any time.

The files directory contains the radial files in a regular directory structure (as defined by orbhfradar2db's parameter file).

Other database backups may be present in this directory depending on the state of development and operations in the network.

/dbmaster

This directory is not currently used by the HF-Radar Network.

/logs

The logs directory contains all log files produced by the real time system. Log files most commonly checked during operations are the orb2orb_* logs for data telemetry, orbhfradar2db for database population from the orbserver and the cron:* and matlab:* logs for total vector processing.

Real time system log messages are prefixed with the type of message. Common message types include notify, complain and fatal. Normally we aren't concerned with notify messages and we're only interested in problems that may have occurred. A quick way to scan the logs for anything other than notify messages is using grep. For example, 'grep –

v notify orbhfradar2db' issued in the logs directory will return all lines not matching 'notify' which will be any complain or fatal messages.

If the real time system is started using rtexec –s then the log messages from the previous run are saved into a new subdirectory and compressed.

/orb

The orb directory contains the Object Ring Buffer (ORB) which is managed by orbserver. The contents of this directory should not be modified in any way since it is entirely managed by the real time system.

/pf

With the exception of rtexec.pf, all parameter files are kept in the pf directory. These files define parameters for executables run by rtexec. For example, orbhfradar2db.pf defines the directory structure for saving radial files. The orbserver parameter file is also typically found here.

/rtsys

This directory contains a Datascope database of rtexec statistics which can be viewed using dbe.

/state

State files are required for processes that are run intermittently (typically from the crontab). Normally a timestamp is stored in the state file which indicates where the last execution left off so that subsequent executions know where to pick up. Though rtexec does not launch new processes if existing processes are still running, lock files are still produced for historical reasons.

Appendix B: Common Scripts by Directory

/data/HFRadar/HFRNet/bin

MATLAB.pl
Batch processing in MATLAB from the real time system is done using MATLAB.pl. The script requires the MATLAB script to be run and a lockfile. Optional arguments may be provided to give a function-like utility to the script. However, the script must be expecting these arguments since they are retrieved from system environment variables. See tuv_driver_HFRNet.m as an example of how this is done. Further details on usage are given in the header.

hfrnet_newRTVs.pl
Since total vector data are not stored in a database, near-real time processing for products dependent on total vectors (e.g. 25-hour averages) must rely on the use of state files. hfrnet_newRTVs.pl returns a listing of new data timestamps produced since the (system) time specified in a given state file. The state file can optionally be updated with the current time upon execution which is the normal operational use in near-real time processing. Further details on usage are given in the header.

/home/rt/matlab/Toolbox/HFRadar/drivers

tuv_driver_HFRNet.m
The main MATLAB script that orchestrates near-real time total vector production is tuv_driver_HFRNet.m. Normally, this script is written to produce hourly total vector velocities in MATLAB, ASCII, and NetCDF format. The MATLAB file is the most complete record of processing parameters, input radials used and totals produced. ASCII files are typically for internal use only, such as web applications, and may be filtered depending on its usage. NetCDF format data are unfiltered and will be used for data dissemination and interoperability.

currentAvg_driver_HFRNet.m
This script typically produces 25-hour averaged total vector products in near-real time. It uses hfrnet_newRTVs.pl to determine which new total files have been produced since its last run and data are normally saved as MATLAB and ASCII files. As with total vector data, MATLAB files are slightly more complete datasets while ASCII files are for internal use only.

/home/rt/matlab/Toolbox/HFRadar/data_defs

RealTimeParameters_*.m
Near-real time processing parameters are specified for each region (e.g. USWC, USEGC) and resolution (e.g. 1km, 2km, 6km) in RealTimeParameters_REGION_RESOLUTION.m.

Of particular interest is the SitePrefs cell array. This array defines the radar sites that will be used in total vector processing for the given region and resolution. Below is a typical entry:

```
SitePrefs(5, 1:7) = cellstr( char('SFSU', 'BRKY', 'hfrss10lluv', 'm', 'z', 'i', 's') );
```

The character array has seven elements:

1) Site's network code (SFSU)
2) Site code (BRKY)
3) File format (hfrss10lluv)
4-7) Beampattern preference from highest to lowest (m, z, i, s)

The file format and beampattern preference rarely change. In the beampattern preference, 'm' and 'z' both refer to radials produced from measured beampatterns while 'i' and 's' both refer to radials produced from idealized beampatterns. There are a few instances where radials produced from idealized patterns are preferred but they are rare.

Addition of a new site to total vector processing simply involves adding a new row to the SitePrefs array, specifying the new site and its network.

You may want to be aware of some information in the parameter file that is Node specific. These entries are:

- RTgen.org
- RTgen.orgName
- RTtuv.nc_contactName
- RTtuv.nc_contactEmail
- RTtuv.nc_url

These pieces of information are stored as attributes in the NetCDF files and you should customize them to appropriate values.

Appendix C: Common Command Reference

rtexec

Instead of stopping and starting the real time system from the rc script (/etc/init.d/antelope), it can be stopped using rtexec from within the real time directory. Issue the –k switch with rtexec to stop the system. No switches are needed to start the system although the –s switch is handy for compressing and saving old logs before the system comes up. See rtexec(1) for a full description.

orbserver

All data in the HF-Radar Network is communicated through an object ring buffer (ORB) which is managed by orbserver. The orbserver is run on a specified port, typically 6580 which is also defined as roadnet. It is configured according to its parameter file located in the pf directory of the real time system. Points of interest in the parameter file are the valid_ip_addresses table which allows machines to access the ORB and the ringsize which is normally 1.4GB. These settings should not be changed.

orb2orb

Data flow through the HF-Radar Network is done using orb2orb which copies data from one orbserver to another. It is normally run on the same machine as the destination ORB to ensure robust and continuous transfers. Each Node independently connects to Portals via orb2orb.

Beyond specifying the machine and port of the remote ORB, statefiles and match expressions are commonly used. Statefiles are used to reduce the chance of a gap in data copied from the remote ORB to the local ORB in the case of a network outage. Match expressions are used to filter for specific packet types from the remote ORB.

orbstat

ORB status information is returned by orbstat, including information about the server, packets in the buffer, source names for the packets and any connected clients. Common applications for orbstat in the HF-Radar Network are for listing the source names of packets currently in the ORB and their latency. For example:

```
orbstat -s :roadnet
```

will return the source names of packets in the ORB. Source names for radial file packets always start with NET_SITE. The latency reported is derived from the data timestamp and the current time so latencies anywhere between 1 to 4 hours are normal depending on the averaging period for a given site. Since a large number of sites are typically present in a Node's ORB, filtering through grep and/or piping to less is often useful:

```
orbstat -s :roadnet | grep OSU | less
```

Clients connected to the orbserver can be shown using the –c switch. Typical clients include orb2orb and orbhfradar2db.

orbhfradar2db

Packets are taken out of the ORB and stored in the Datascope database by orbhfradar2db. Beyond specifying the ORB to read packets from and the directory to build the radial directory hierarchy, options commonly used are:

-o	Overwrite files that have identical pathnames to previous files. The default is not to overwrite previously created files.
-d dbname	Store file-tracking information in the specified database. This database is created if it does not already exist. Site and network information will be dynamically extracted into the database if the incoming packets are in LLUV format. Furthermore, new site rows will have the staname set to the most recently seen non-null value in the database for the relevant station.
-S statefile	This option specifies a filename in which to save the current state of the program (i.e. the current orb read position).

See orbhfradar2db(1) for a full description.

dbe

If you have X running you can launch dbe, a GUI interface for Datascope databases. In order to view the radial database, execute

```
dbe /data/HFRadar/HFRNet/db/hfradar &
```

and a GUI window will appear with buttons for each table in the database (Figure 2). Clicking on any button will open the table (Figure 3). You can subset, sort and find within any field by clicking on the field name at the top. All the fields for a given row can be displayed by ‘right’ clicking on the row of interest (Figure 4).

Figure 2. GUI view of the HF-Radar real-time database using dbe. Tables are shown as buttons

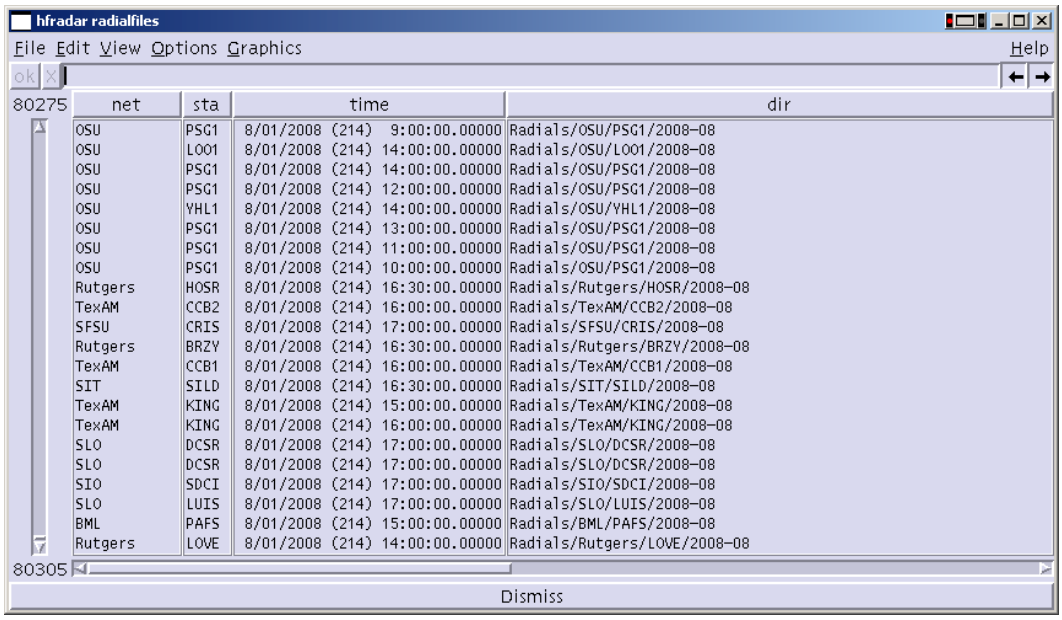

Figure 3. GUI view of the radialfiles table from the HF-Radar real-time database using dbe. Note that additional fields for the table are not shown in the image above because they scroll beyond the image.

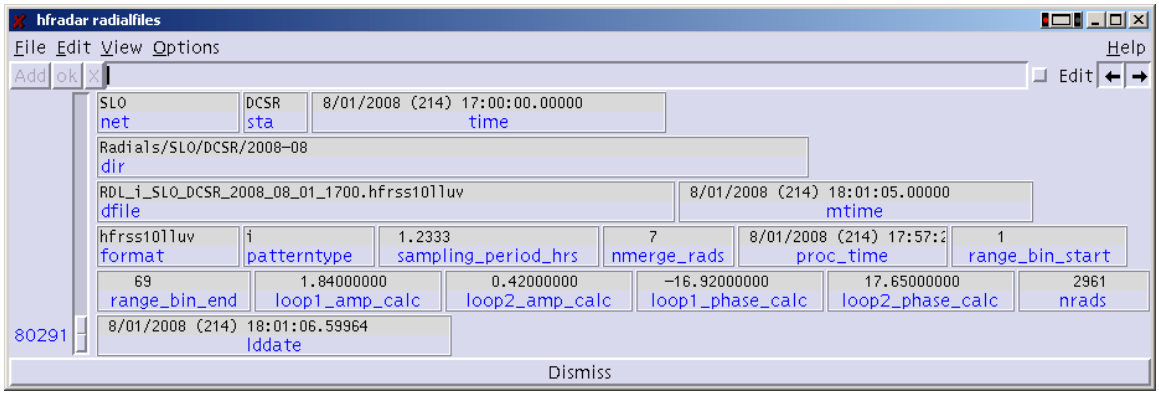

Figure 4. GUI view of a single record from the radialfiles table of the HF-Radar real-time database using dbe. This is often a useful way to view all the fields for a given record.

Be careful not to edit the database while orbhfradar2db is running.

dbdescribe

dbdescribe takes the name of a database or a schema and returns a description of the format of the database files. An optional table-name specified on the command line will limit dbdescribe's explanation to the one specified table. For example, to view the schema for the site table in Hfradar0.6 execute

```
dbdescribe Hfradar0.6 site | less
```