HF-Radar Network
Portal Reference Guide

Mark Otero
November 04, 2008

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

Table of Contents

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 Portal. *Organizations hosting the Portal normally only need to provide hardware and basic networking support since HF-Radar Network Administrators typically handle the maintenance and operation of the real-time system.*

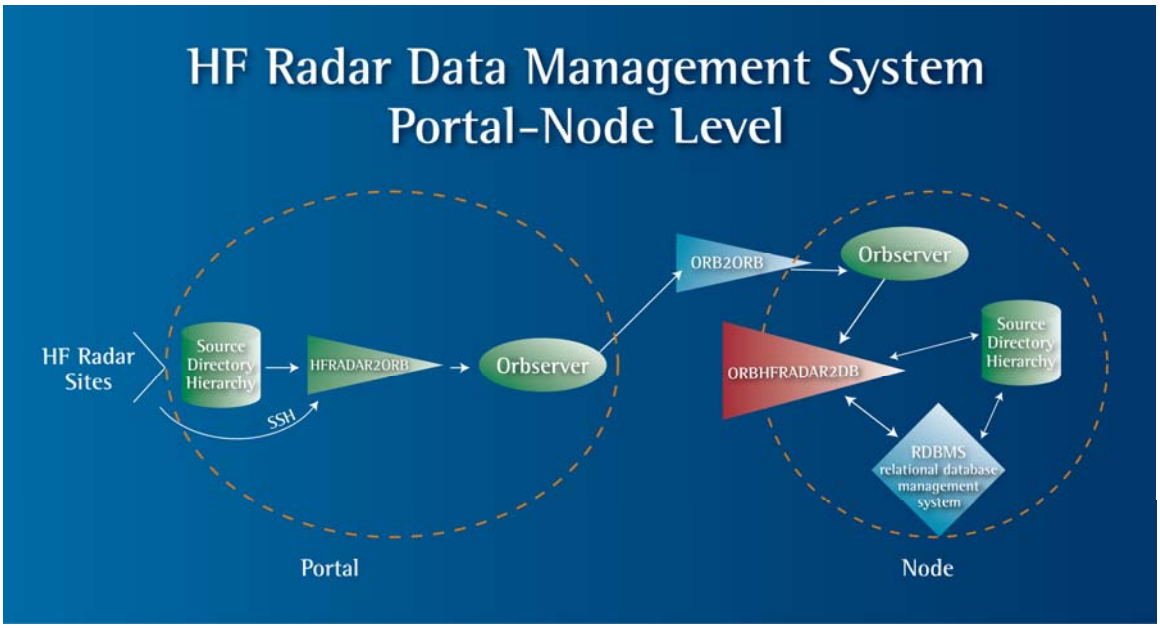

Figure 1. Data flow through the HF-Radar Network. Portals acquire radials through hfradar2orb, an executable that retrieves files over SSH from a remote host or through a local source directory. Once files are obtained they are placed in the ORB(server). The Portal's ORB makes data available to any allowed 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 the `chkconfig` command 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.
- 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 in addition to man pages.

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 a couple minutes for processes to complete. Before shutting down or rebooting, verify that hfradar2orb and orbserver processes have been terminated. 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. Inbound connections must also be allowed over port 6580 so that Nodes can acquire data from the Portal's ORB (using orb2orb).

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, 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. 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.

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 rtexec'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. Orbserver is the main process listed here.

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.

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, or program names. 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
2) orb writers (like hfradar2orb)
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. Other executables written for non-real time applications should either be kept in ~rt/bin or elsewhere.

/db
This directory is not currently used by Portals in the HF-Radar Network.

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

/localIngest
Radial files that are not obtained from a remote host over SSH may be placed in this directory for local acquisition. Files may be placed here by any number of methods including FTP.

/logs
The logs directory contains all log files produced by the real time system. Log files most commonly checked during operations are the cron:hfradar2orb logs for data acquisition.

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

/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, hfradar2orb_NET_SITE_PATT.pf defines regular expressions used for radial file acquisition. 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.

hfradar2orb keeps a state file which contains the data timestamp of the last file acquired. Subsequent executions of hfradar2orb define new data as files with data timestamps more recent than that stored in the state file.

Appendix B: 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.

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.

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

hfradar2orb

HF-Radar radial files are loaded into the ORB by hfradar2orb. It reads files from a specified directory (normally localIngest withing the real-time directory), encapsulates each file as an orb packet, and puts the packet on the specified orbserver. Files can also be obtained via ssh(1) and scp(1) commands to a remote server. This presumes that the ssh connection has been set up for password-less access [shared-key authentication]. If the intake-directory is a local pathname, by default hfradar2orb removes the copies from the intake directory after cloning them to the orbserver. If the intake directory is remote, i.e. accessed via ssh, no attempt is made to remove files once they are retrieved.

hfradar2orb searches for each filename pattern specified in its parameter file, looking for files that match the corresponding perl match expression. The site name, pattern type, and timestamp for each file is expected to be encoded somehow in the filename. If an input file is in CODAR Range-bin format, hfradar2orb attempts to convert it to CODAR LLUV format with the hfradartools::rb2lluv(3P) routine.

In an Antelope real-time system, hfradar2orb is intended to be run periodically out of the cron table of the rtexec(1) parameter file.

If the parameter too_new is specified, packets newer then the system clock plus the parameter value are skipped.