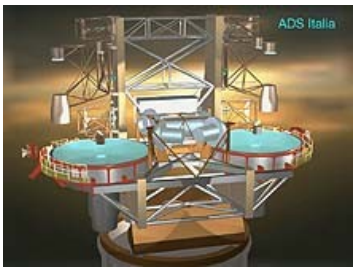
LBT PROJECT
2x8,4m TELESCOPE

Doc.No. : 228s010
Issue : b
Date : 10-May-2012



LBT PROJECT

2 X 8,4m OPTICAL TELESCOPE

UPS and Thermal Monitor

Client System Configuration

	Signature	Date
Prepared	Dan Cox	31-Oct-11
Reviewed	Norm Cushing	05-Jan-12
Approved	Joar Brynnel	05-Jan-12

1. Revision History

Issue	Date	Changes	Responsible
a	31-Oct-11	First draft	Dan Cox
b	10-May-12	References to “red button” replaced with “yellow button”; section added for testing shutdown applications.	Dan Cox

2. Table Of Contents

3.

4. List Of Abbreviations

BYB	Big Yellow Button
CRA	Computer Room A
CRB	Computer Room B
DD	TCS Data Dictionary
FSD	Fast Shutdown
LB	Low Battery
NUT	Network UPS Tools
OB	On Battery
OL	Online
SNMP	Simple Network Management Protocol
TCS	Telescope Control System
UPS	Uninterruptible Power Supply

5. About this document

1.1. Purpose

This document serves as a guide to configuring the client software tied to the LBT's UPS and thermal monitoring system. It provides an overview of the UPSes at the LBT, identifies scenarios in which client systems using the UPS and thermal monitoring software would be commanded to perform an orderly shutdown and explains how to address special shutdown-time requirements that client systems may have.

This document does not provide details of the underlying design and administration of the UPS and thermal monitoring system, such as the "Big Yellow Button" or thermal monitoring, other than as they apply to conditions client systems must accommodate.

1.2. Reference Documents

[**RD1**] *Network UPS Tools documentation,*
http://www.networkupstools.org/documentation.html

[**RD2**] *UPS Monitor Design and Administration*, LBT CAN 228sTBD

[**RD3**] *Computing Infrastructure Technical Specification,* LBT CAN 228s540

[**RD4**] *UPS Cleanout Maintenance Procedure*, LBT CAN 300s003

6. Overview

The LBT has three large UPSes for backup power, each of which, in general, serves different zones of the rotating and non-rotating building. These UPSes are of different sizes, capabilities and ages.

- The Exide Powerware 125 is a 125kVA UPS located on Level 1 in the passageway near the Instrument Lab. This UPS provides backup power to Level 2, with the primary usage being the control room and computer room A (CRA) and computer room B (CRB). The Exide - an older generation UPS - does not have built-in and native networking capabilities and was later modified to provide rudimentary information over a network.
- A 65kVA Liebert nPower Series UPS on Level 4 provides backup power to the telescope and all instruments on the telescope. The Liebert is a modern, fully-networked UPS and provides its status via SNMP.
- The 10kVA Savin 900 UPS, also located on Level 4, backs up the non-rotating parts of Levels 4 and 5. The Savin is also an older model UPS, yet does have a 300 baud UART which is used to connect to the network.

Details about the modifications to provide network capability to the older model UPS's are provided in [RD2].

The three UPS zones, however, do not cover the entire LBT observatory. None of the circuits on Level 1 is backed by any of the UPSs, including the clean room and instrument lab. A UPS is present in the LBT clean room, but it remains unmonitored and its use is not supported by the UPS monitor. Even in a UPS-backed zone, not all circuits are backed by a UPS. Requirements for such circuits should be made to LBT mountain personnel.

With a few exceptions, network switches throughout the LBT are on UPS circuits to maintain networking during power interruptions. However, network connections to and from the mountain cannot be guaranteed. The relay to the Internet is located in the VATT and, while it is on its own UPS, it may not be available for extended periods and should not be relied upon to be present.

7. Power Events

Historically, systems inside the LBT were unaware of the loss of commercial power. The UPSes at the LBT were not connected to the internal network or, if they did have a network connection, it was not monitored. Systems would continue to run until power was restored, either by the MGIO generator or the return of commercial power, or explicitly shut down.

When commercial power is interrupted, the MGIO generator is used to supply power to all telescopes on Mt. Graham. It typically requires three to five minutes for the generator to start and ramp up before it begins to provide power, during which time all systems and instruments run on UPS power.

This is the preferred scenario for when power is lost at the LBT. The UPS monitor and Network UPS Tools (NUT) is in place for when this scenario does not occur.

8. Thermal Events

Similar to the UPSes at the LBT, thermal conditions in the computer rooms at the LBT were unmonitored. The pump supplying cooling to the computer rooms monitored conditions, but did not perform any notification internally or externally of temperature, humidity or dew point beyond the established ranges. Again, much like the UPS monitor, a passive thermal monitor for the computer rooms was needed to watch temperatures and notify personnel both within the LBT and downtown.

Further information about the thermal monitoring for the computer rooms can be found in [**RD2**].

9. NUT Overview

NUT is a GPL'd software package based on a client-server model that monitors UPS states, run times, and vendor-specific capabilities, and, when necessary, halts the system. NUT scales well from single-system arrangements with locally attached UPSes to multi-system network-attached multi-UPS environments.

NUT is found in most major Linux distribution repositories or can be built from the source, available at [**RD1**]. A port to Windows (both XP and 7) has also been done and is available at the NUT website, http://www.networkupstools.org. It's still considered to be at "beta level", but has been successfully used in 32-bit mode. A determination of its stability in 64-bit mode has yet to be made.

NUT is comprised of the following:

- *upsd*, the NUT server, is the single daemon which monitors and alerts all connected clients of power events (online, on battery, low battery) and instructs clients to perform a shutdown when UPS batteries have depleted,
- *upsmon*, a NUT client daemon, that runs on each client system and performs a system shutdown when instructed,
- UPS "drivers" that provide the low-level and vendor-specific UPS information to the NUT server,

- a client library, *libupsclient.so*, that supplies routines used to interface with the NUT server,
- NUT applications built from the client library to provide status and logging of UPS information.

NUT is designed to operate in the background and, when on battery power, to continue running until either power is restored or the batteries are nearly depleted and the system is shut down. These design goals are met with the implementation of NUT at the LBT, even though the main UPS's batteries are not run to depletion.

The UPS to which all NUT clients monitor (given below) has a *virtual* battery runtime of 900 seconds. This runtime was chosen to allow sufficient time for the MGIO generator to begin supplying power when commercial power is interrupted.

NUT also provides a feature called "fast shutdown" (FSD) which, when sent to NUT clients, initiates a controlled shutdown regardless of the state of the UPS batteries. This feature is also used at the LBT [**RD2**].

1.3. UPS and Thermal Monitor

A 1U embedded Linux system in CRA runs the NUT server and NUT UPS drivers. It monitors the UPSes for power states as well as power inputs from the UPSes. The system hostname is *ups-master.mountain.lbto.org*.

One of the early design goals was to provide a physical switch that would provide personnel on site a way to quickly halt all computers from one location. This is handled by the "Big Yellow Button" (BYB) on the front panel of the 1U chassis and a UPS monitor daemon that also runs on ups-master. The UPS monitor runs independently of the NUT server and NUT UPS drivers, connecting to the NUT server as a client with the NUT client library. It monitors the UPSes for all states (on battery, online, bypass, etc) and handles the notification to an on- and off-site mail list.

Figure 1 below shows the relationship of the UPS monitor, the NUT server and NUT UPS drivers with the NUT clients.

A later requirement to the UPS monitor was added to monitor the temperatures in both computer rooms and, using the same underlying code that drives the BYB, halt all computers in both computer rooms due to high heat, humidity or dewpoint. The UPS (and thermal) monitor reads sensors in both computer rooms and the pump that provides cooling to both computer rooms and, if outside the threshold for those rooms, shuts down and powers off all computers in both CRA and CRB.

In both cases – a BYB-initiated halt or out of range thermal conditions – all NUT clients will be commanded to halt by the NUT server.

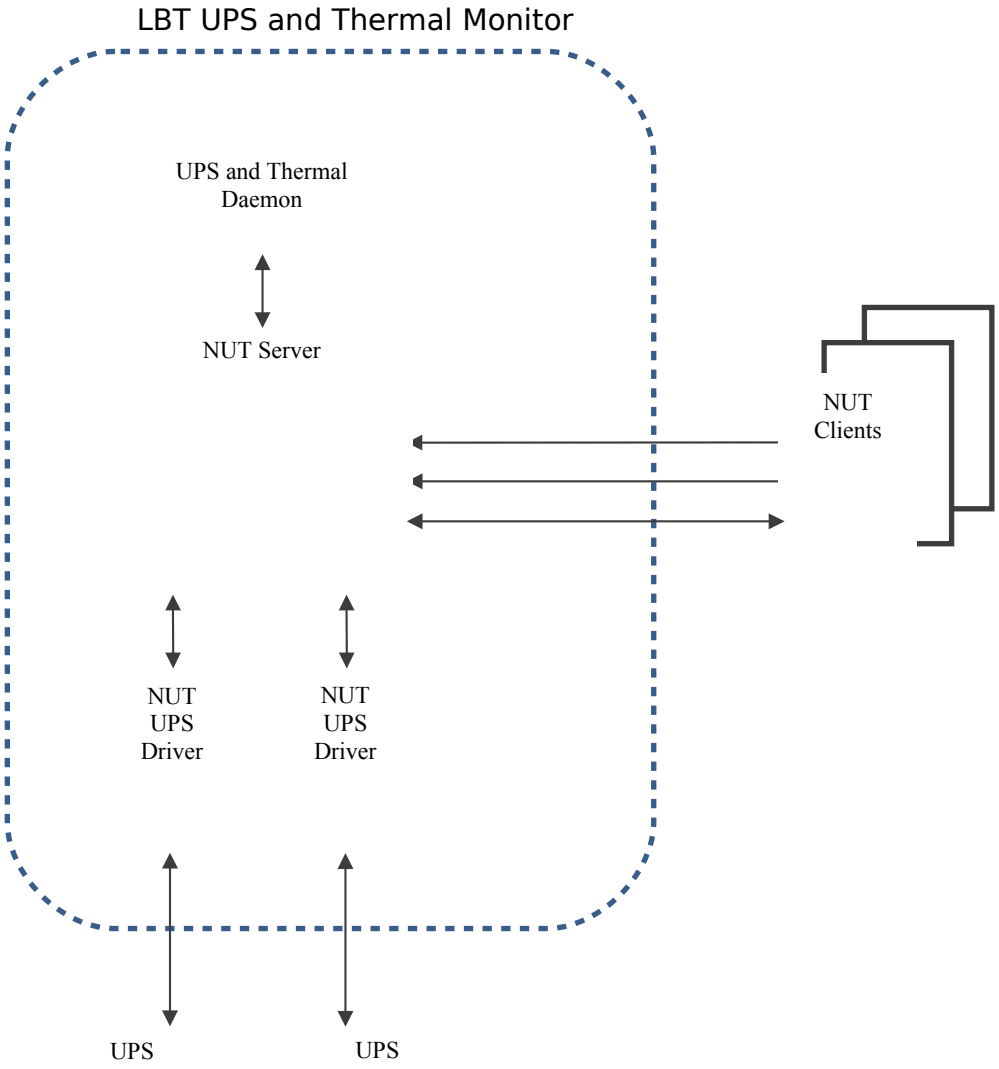


Figure 1 UPS and Thermal Monitor and NUT Server Environment

10. Installing NUT

NUT for Linux is available in most Linux distributions. For CentOS, it's available in binary format from the "epel" repository. Most of the major distributions ship a client-side package that contains all the necessary files to run on a NUT client. Again for CentOS, this is the "nut-client" RPM. The nut-client package contains all the necessary binaries, configuration files and initialization scripts.

The NUT server at the LBT runs NUT version 2.6.1. The protocol between NUT client and server has not changed in several revisions, so NUT clients can as old as 2.1 can be used. Most NUT clients at the LBT run at 2.2.1. This applies to 64-bit clients, as well.

Client systems that interface with the NUT server only need a few applications in order to run. The client-side application, *upsmon*, and its companion configuration file

1.4. Installing From Source

NUT may also be installed from the source package at the NUT website and built using the standard "configure, make, make install" GNU process. Although not required, both a user id and group id (i.e. "ups" or "nut") should be created prior to building the NUT package. These are used to set permissions of files and directories used by the different components of NUT.

```
./configure –prefix=/opt/nut –with-user=nut –with-group=nut
```

Permissions of directories and configuration files should be checked for the correct umask and file ownership.

1.5. Windows Installation

A NUT client port to Windows (XP & 7) is currently in use at the LBT, although its function is limited to UPS monitoring only. A full port of NUT (at revision 2.6.0) will be replacing the earlier limited client on all LBT machines, providing identical functionality as found on Linux.

At this time, the NUT package has only been tested on 32-bit Windows XP and 7 systems. Testing on 64-bit Windows systems is anticipated early in 2012.

11. Configuration of Client System

The client system is configured through the *upsmon.conf* file. The location of the file varies between distributions, though it will typically be in /etc/ups in CentOS packages. A sample file, *upsmon.conf.sample*, is included in the nut-client package.

If building from source, the same sample *upsmon* configuration file can be found in the install directory.

1.6. Suggested Configuration Values

Most configuration directives in the *upsmon.conf* sample configuration files can be accepted as they stand. Some apply only to *upsmon* master clients – clients responsible for halting slave clients – and are ignored by *upsmon* in slave mode. The directives that follow are those that are either mandatory in NUT or suggested settings that are currently in use by other systems at the LBT. A copy of the *upsmon.conf* of the LBT machines can also be provided, if necessary.

1.6.1. MONITOR

At the LBT, there is only one UPS to monitor: the Exide UPS. While possible in NUT, it is not recommended to monitor any of the other UPSes on site, as all defined UPSes would have to be in a low-battery condition for *upsmon* to halt the system.

The IP address of the NUT server is used on the *MONITOR* directive to avoid name resolution errors when the client attempts to connect (or reconnect) to the NUT server.

The required *MONITOR* directive is:

```
MONITOR exide@192.168.1.30 1 upsmon simpsons slave
```

"upsmon" in the *MONITOR* directive is the username registered with the NUT server and "simpsons" is the required password. "slave" indicates that this system is not a master for other client systems and should halt when commanded without delay.

1.6.2. MINSUPPLIES

As only one UPS is monitored, *MINSUPPLIES* should be set to "1".

```
MINSUPPLIES 1
```

1.6.3. DEADTIME

All three UPS's at the LBT are large enough to provide power longer than the configured 15 minutes of battery run time, so *DEADTIME* should be set high enough to avoid unnecessary shutdowns in the event the connection to the NUT server is lost after the Exide UPS has switched to battery power.

The recommended *DEADTIME* directive is:

DEADTIME 600

Note: A client system that halts due to an expired *DEADTIME* is generally an indication that a network switch – or intermediate piece of network equipment – is not on a UPS-protected circuit and re-cabling may be necessary.

1.6.4. SHUTDOWNCMD

The *SHUTDOWN* directive is a string used to halt the system. On Linux distributions, it need not be changed. If building from source, it should be changed to what is appropriate for that system:

SHUTDOWN <as appropriate to OS>

All commands listed below should be enclosed in double-quotes.

Operating System	*Command*
Linux	/sbin/shutdown –h +0
Solaris	shutdown –y –i5 –g0
Windows (XP & 7)	C:\\WINDOWS\\system32\\shutdown –s –f –t 0

1.6.5. POWERDOWNFLAG

The NUT developers recommend the following for the *POWERDOWNFLAG* directive only on Windows XP & 7:

POWERDOWNFLAG "C:\\killpower"

1.7. Notification Directives

The notification directives in *upsmon.conf* control how *upsmon* alerts users or administrators for a fixed set of NUT events. Through these directives a customized shutdown procedure for the client system can be performed.

1.7.1. NOTIFYFLAG

The *NOTIFYFLAG* directive indicates how the power events NUT recognizes are reported and where they are reported. All power-related events (and internal UPS events) are handled by the UPS monitor and reported to LBT personnel through a mailing list, so it is not necessary for client systems to perform any special notifications.

For machines at LBT, notifications are only posted to active terminals for "on-battery" and "online". Connection drops and reconnects to the NUT server are logged. All other conditions are handled by the UPS monitor.

```
NOTIFYFLAG ONLINE      WALL+SYSLOG
NOTIFYFLAG ONBATT      WALL+SYSLOG
NOTIFYFLAG COMMBAD     SYSLOG
NOTIFYFLAG COMMOK      SYSLOG
```

NOTIFYFLAG, used with the *NOTIFYCMD* directive, can be used to schedule events – such as a system shutdown – on a NUT client. A full description can be found at [**RD1**].

12. Client System Shutdown

Some client systems, prior to halting, may have other tasks to perform. A system with RAID, for instance, may need to halt the arrays or switch to read-only to prevent file system corruption. Similarly, instruments systems that require special handling of devices prior to a system halt need to provide a method by which their instruments can be made idle and safe at the time a shutdown command has been received by *upsmon*.

TCS and DD variables should not be considered authoritative at the time a halt has been commanded by the NUT server. All TCS systems will receive a halt command at the same time as instrument systems and may not be alive or in the process of halting, so DD variables, if present at all, may be stale and should not be trusted.

All LBT infrastructure machines – NFS servers, DNS, LDAP – continue to run for an additional 10 minutes after the NUT server has commanded a shutdown, in both the low-battery and FSD conditions. As these machines are using NUT's scheduling feature, and as the UPS batteries backing these systems are sufficiently large to continue to provide power, these times can be adjusted, if required.

Client systems which require special attention given to instruments or devices prior to shutdown have numerous options by which the software control systems can be made aware of a shutdown from the NUT server. Such methods could include a signal handler in the control software (catching SIGPWR, for example) or through a socket connection to the control system itself. In either case below, making the instrument or device safe

should be synchronous, i.e. only advance the shutdown process once the instrument or device has finished being made safe.

1.8. Shutdown scripts

A system with an instrument or device that requires little to no time to put into a safe state can use the normal system shutdown scripts (/etc/rc.d/rc0.d in CentOS, for example). Like syncing RAID devices, a process can run until the device is made safe while the system shutdown is already in progress. Note that the order in which the shutdown script is executed may effect what system resources (filesystems, networking, devices, etc) are still available when the shutdown script runs.

1.9. Shutdown Wrapper

An alternative to make a device safe during shutdown is to perform the operation prior to the start of a system shutdown using the NUT client, *upsmon*. A wrapper script is used to first safely stop and idle a device prior to invoking shutdown:

```
/#!/bin/bash
#
echo –n “Safely idling device”
/usr/local/bin/devIdle
echo –n “Starting shutdown”
/sbin/shutdown –h +0
exit
```

Replacing *SHUTDOWNCMD* in *upsmon.conf* with the name of a shell script like that above would begin halting the system *after* the “devIdle” has completed. All system resources remain available. “devIdle” is an application which be responsible for sending a signal or making a socket connection (or whatever is appropriate for an instrument control system) to the control system.

Note that *upsmon* runs the wrapper script pointed to by the *SHUTDOWNCMD* directive with root permission, so all system resources will be available.

13. TCS Interface

While a TCS interface to the UPS monitor is planned – in the TCS data dictionary – it should not be solely relied upon to make instruments or devices safe. This interface cannot protect those systems which do not access the TCS DD variables nor will it protect any system if the TCS is not running.

Additionally, systems should anticipate FSD commands due to high thermal conditions in either or both computer rooms or an immediate halt from pressing UPS monitor's BYB.

14. Testing Automatic Shutdown

NUT provides a method to test shutdown handling by simulating a UPS and manually initiating an on-battery, low-battery and fast-shutdown conditions. In addition to setting up a NUT client (as explained earlier in this document), you must also set up a NUT server to use the "dummy-ups" driver.

The dummy-ups driver creates and uses internal status and, with other NUT command-line tools, allows you to change the state of those variables.

With the "nut-client" RPM already installed, also install the "nut" RPM containing the NUT server, UPS drivers and server-side configuration files. Nothing extra is required if you built and installed NUT from the source.

Note: For this test, the NUT server and dummy UPS driver can be run remotely, if you choose. The instructions that follow presume you'll run both the server and client on the same machine.

1.10. Configuration

After installing the full NUT server package, the first step involves creating some rudimentary configuration files. Sample configuration files will be in the same location as the client configuration file from above. You'll need to configure a UPS driver configuration file (ups.conf), the NUT server (upsd.conf) and a permissions file (upsd.users) to provide a username and password with which to perform the simulated UPS state changes.

1.10.1. ups.conf

The *ups.conf* configuration file specifies one or more UPSes which will run UPS drivers. Drivers, as used by NUT, are not device drivers, but rather user-level daemons that exist solely to communicate with a UPS. In this case, the dummy-ups driver will create dummy data that is read by the NUT server.

To set up a dummy-ups driver, enter the following in *ups.conf*:

```
[dummy]
        driver = dummy-ups
        port = foo
        desc "Dummy UPS"
```

This creates a UPS named “dummy” and, when started by the UPS driver utility, will run a daemon called ‘dummy-ups’.

Additionally, in the same directory as this configuration file, create a zero-length file named “foo” to satisfy the “port =” line. This file could be used to have additional data transmitted by the dummy-ups driver.

A standard NUT server configuration file should be configured to listen for connection from the local host (LISTEN 127.0.0.1 3493) or from an Ethernet interface (LISTEN <ipaddr> 3493). No other configuration should be required.

1.10.2. upsd.conf

The NUT server configuration file, upsd.conf, only requires that the LISTEN directive be configured. To run the NUT server on the same test machine, remove the comment on the LISTEN directive so it specifies:

```
LISTEN 127.0.0.1 3493
```

The NUT server will listen on the localhost interface and accept connections to port 3493 once started.

1.10.3. upsd.users

A username and password for the privileged operations that will be necessary to test must be created in the same configuration directory. A username and password need to be included along with the permitted actions, as follows:

```
[admin]
        password = passwd
        actions = SET FSD
        instcmds = ALL

[upsmon]
        password = passwd
        upsmon slave
```

This allows the user ‘admin’ to modify UPS attributes and to initiate a fast-shutdown, both conditions required for testing. It also creates a user ‘upsmon’ which will be used by the NUT client.

1.10.4. upsmon.conf

You'll also need to point your NUT client to the new server by modifying the MONITOR directive in *upsmon.conf*, as follows:

```
MONITOR dummy@localhost 1 upsmon passwd slave
```

Note: Use the NUT client configuration file that you've configured from earlier in this document. If you're using a wrapper script, ensure the SHUTDOWNCMD directive points to this script.

1.11. Starting Your Test Environment

With the configuration files set up, start the NUT driver (with only the entry for the dummy driver):

```
upsdrvctl start dummy
```

Start the NUT server:

```
upsd
```

The NUT server can be stopped by entering:

```
upsd –c stop
```

To stop the dummy driver, enter:

```
upsdrvctl stop dummy
```

Finally, start the NUT client, using full debugging to standard output, allowing you to see the UPS state changes reported by the NUT server, including, if defined, running a wrapper script:

```
upsmon –DDDDD
```

To stop the NUT client, enter Control-C.

1.12. Toggling Simulated UPS Events

With the NUT driver, server and client started, you can test your shutdown scripts by changing the UPS status to “low-battery” or “fast-shutdown” using the *upsrw* command, as follows:

```
upsrw –s ups.status=value –u admin –p passwd dummy@localhost
```

where *value* is one of the following:

- OL – UPS is online, receiving site power
- OB – UPS is running from its batteries
- LB – UPS batteries are low and NUT clients will begin to shutdown (using SHUTDOWNCMD as specified in upsmon.conf)
- FSD – a fast-shutdown has been signaled and NUT clients will begin to shutdown (also using SHUTDOWNCMD)

--oOo--

Doc_info_start
Title: *Fill in title here*
Document Type: *Specification / Technical Report / Technical Manual*
Source: Steward Observatory
Issued by: *Author*
Date_of_Issue: *Fill in date here*
Revised by:
Date_of_Revision:
Checked by:
Date_of_Check:
Accepted by:
Date_of_Acceptance:
Released by:
Date_of_Release:
File Type: MS Word
Local Name: *Repeat title here*
Category: *Fill in category here (3 digits)*
Sub-Category: *Fill in sub category here (3 digits)*
Assembly: *Fill in Assembly name here*
Sub-Assembly:
Part Name:
CAN Designation: *Fill in full doc number here*
Revision: *Fill in revision letter here*
Doc_info_end