

Technical Tips and Tricks |

AlliedWare Plus™ Managed Layer 3 Switches

Introduction

This document contains useful technical tips and tricks for AlliedWare Plus Managed Layer 3 Switches.

These Tips and Tricks apply to:

SwitchBlade x908
 x600-24Ts
 x600-24Ts-POE
 x600-24Ts/XP
 x600-48Ts
 x600-48Ts/XP
 x900-12XT/S
 x900-24XT
 x900-24XT-N
 x900-24XS

Contents

This revision of Tips and Tricks contains the following items:

- 1. Management 2
 - Using Shell Scripts 2
 - Using Command Scripts 4
 - Large Configurations in the "conf t" Environment 6
 - Using SFTP to Transfer Files to/from an AlliedWare Plus Switch 7
- 2. Switching 10
 - How to View Switch Tables 10
 - How to View Port Counters 11
- 3. Resiliency 12
 - x600 Resiliency Link 12
- 4. Diagnostics 16
 - CPU Usage Spikes 16
 - MTR Switch Drops Packets 19
- 5. Hardware 22
 - Switch PSU Fault Analysis 22

Management

Using Shell Scripts

AlliedWare Plus supports shell scripts. You can use this powerful interface for information gathering and device configuration.

Important: shell scripts must have the file extension **.sh**.

This section describes a script that configures an IP interface, sets switch ports to trunk mode, executes **show** commands and returns output to the terminal.

Note that this script does not contain statically configured interface names and IP addresses. Instead, you enter these as command arguments when the script is executed. This allows you to re-use the script. You could develop a collection of scripts that allow you to perform frequent tasks quickly and efficiently.

When you run this script, you need to enter three parameters at the command line:

1. the VLAN ID to be created
2. the IP address to be assigned to the VLAN
3. the switch ports to be added to the VLAN

The script

The script is named **vlan-port-ip.sh** and contains:

```
# configure VLAN, add an IP
echo "Configuring VLAN and IP"
echo -e "
  enable\n
  configure terminal\n
  vlan database\n
  vlan $1\n
  exit\n
  interface vlan$1\n
  ip address $2\n
" | imish

# Assign switch ports to VLAN
echo "Configuring Switch Ports"
echo -e "
  enable\n
  configure terminal\n
  interface $3\n
  switchport access vlan $1
" | imish

# show ip interfaces
echo -e "
  show ip int brief\n
" | imish
```

Running the script

This example uses the script to create vlan120, assign it an IP address of 192.168.1.120/24, and put ports 1.0.10 and 1.0.11 into it. Enter **Privileged Exec** mode and use the command:

```
awplus#activate vlan-port-ip.sh 120 192.168.1.120/24
port1.0.10-port1.0.11
```

The script returns the following output to the console:

```
Configuring VLAN and IP

AlliedWare Plus (TM) 5.2.1 07/27/07 00:44:25

Enter configuration commands, one per line. End with CNTL/Z.

Configuring Switch Ports

AlliedWare Plus (TM) 5.2.1 07/27/07 00:44:25

Enter configuration commands, one per line. End with CNTL/Z.

AlliedWare Plus (TM) 5.2.1 07/27/07 00:44:25

Interface          IP-Address      Status          Protocol
eth0                172.28.8.220   admin up       running
vlan120            192.168.1.120  admin up       down
```

Verifying the configuration

You can verify the configuration by checking the running-config. The following figure shows the relevant parts of the resulting running-config:

```
awplus# show run

vlan database
  vlan 120 state enable
!
interface port1.0.10-1.0.11
  switchport mode access
  switchport access vlan 120
!
interface vlan120
  ip address 192.168.1.120/24
!
```

Using Command Scripts

Command scripts are also supported in AlliedWare Plus. These are similar to scripts in AlliedWare.

Command scripts must not have the file extension `.sh`. We recommend using `.scp`.

Note that command scripts are different to device configuration files.

This section describes a script that creates a VLAN with ID number 2, names it "video2", and assigns the IP address 192.168.2.1 with a class C mask. The script contains the same commands as you would enter at the command line.

The script

The script is named **vlan2.scp** and contains:

```
enable
conf t

vlan database
vlan 2 name video2

interface vlan2
ip address 192.168.2.1/24

end
```

Note that you have to include the commands **enable**, **conf t**, and **end** in the script.

Running the script

To run the script, enter **Privileged Exec** mode and use the command:

```
awplus#activate vlan2.scp
```

The script returns the following output to the console:

```
AlliedWare Plus (TM) 5.2.1 07/20/07 00:45:15

Enter configuration commands, one per line.  End with CNTL/Z.

awplus#
```

Verifying the configuration

You can verify the configuration by checking the running-config. The following figure shows the relevant parts of the resulting running-config:

```
awplus# show run

vlan database
  vlan 2 name video2
  vlan 2 state enable
!
interface vlan2
  ip address 192.168.2.1/24
```

Large Configurations in the "conf t" Environment

The problem

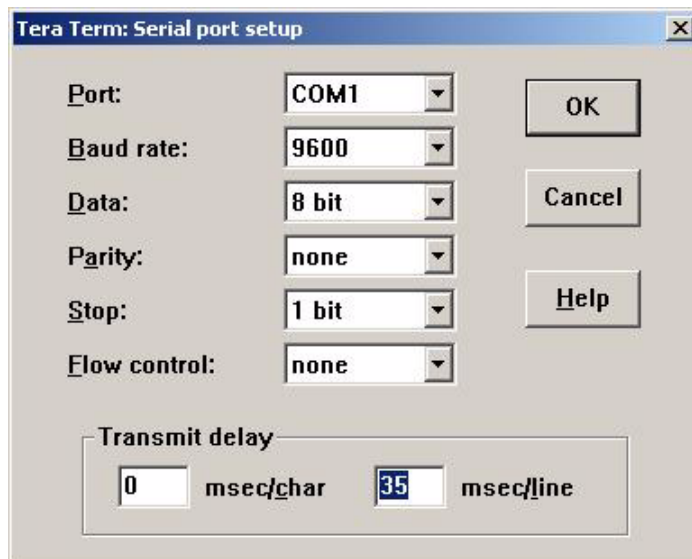
Pasting in very large configurations using the console at the "conf t" prompt can give unpredictable results. Consoles, as standard practice, do not have flow control. If too much text is pasted, it will exhaust the buffer size available for the console.

The solution

There are three possible options:

1. The best practice is to copy in as a file using TFTP.
2. If you do have to paste to conf t, you can void the issue by breaking the configuration down into smaller portions, and pasting in a portion at a time.
3. Another practical solution is to change your terminal program's setting to introduce an end line delay period. One example, using Linux minicom, involves setting a **Newline Delay** (using Ctrl-a, t, d) of at least 150ms to fix the issue.

Also, hyperterminal offers this setting on connection:



Using SFTP to Transfer Files to/from an AlliedWare Plus Switch

Introduction

Secure File Transfer Protocol (SFTP) is a file copy protocol that is supported by the Secure Shell (SSH) service in AlliedWare Plus. By default, when the SSH service is enabled on a switch running AlliedWare Plus, the SFTP service is also enabled.

You can see whether the service is enabled by using the **show ssh server** command:

```
awplus#show ssh server
Secure Shell Server Configuration
-----
SSH Server                : Enabled
Protocol                  : IPv4,IPv6
Port                      : 22
Version                   : 2,1
Services                  : scp, sftp <-----
User Authentication       : publickey, password
Resolve Hosts             : Disabled
Session Timeout           : 0 (Off)
Login Timeout              : 60 seconds
Maximum Startups          : 10
Debug                     : NONE
```

You can enable or disable the service using the command:

```
(no) ssh server sftp
```

The popular FTP client **Filezilla** can operate as an SFTP client. This provides a convenient graphical interface for transferring files to or from a switch running AlliedWare Plus.

Configuring the switch

There are three steps to enabling SSH server on the switch:

1. Create a hostkey:

```
awplus(config)#crypto key generate hostkey rsa
Generating host key (1024 bits rsa)
This may take a while. Please wait ... Done
WARNING: The SSH server must now be enabled with "service ssh"
```

2. Enable SSH Server:

```
awplus(config)#service ssh
WARNING: SSHv1 host key does not exist. SSH will not be available
for version 1.
```

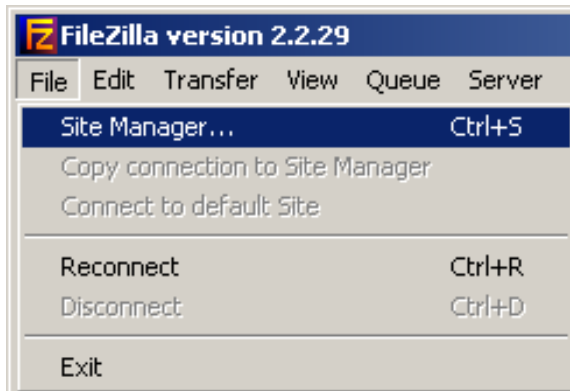
3. Enable one or more users to access SSH:

```
awplus(config)#ssh server allow-users manager
```

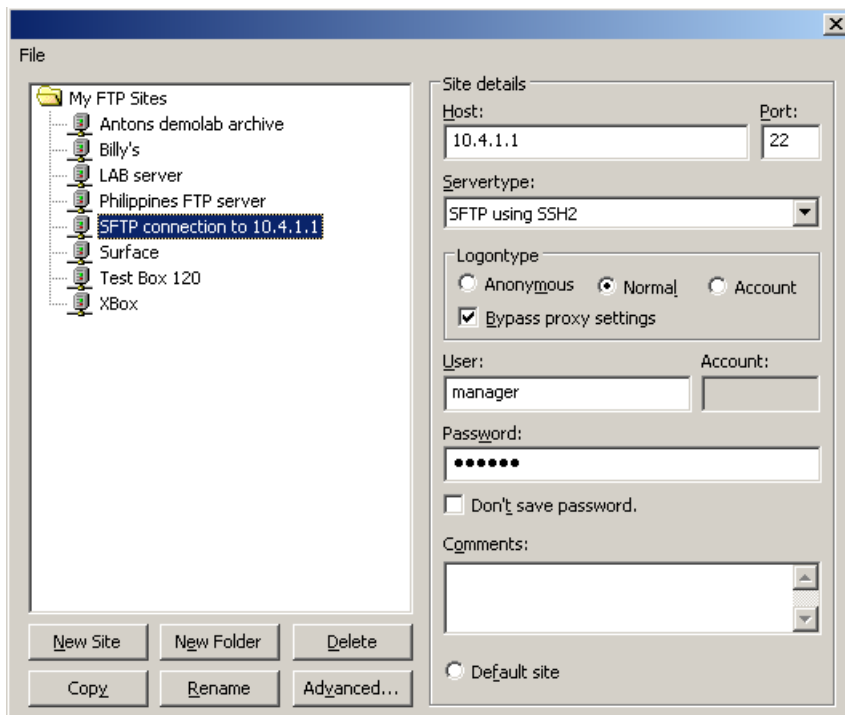
Configuring Filezilla

Within Filezilla, you need to create an FTP site definition that uses SFTP to connect to your switch.

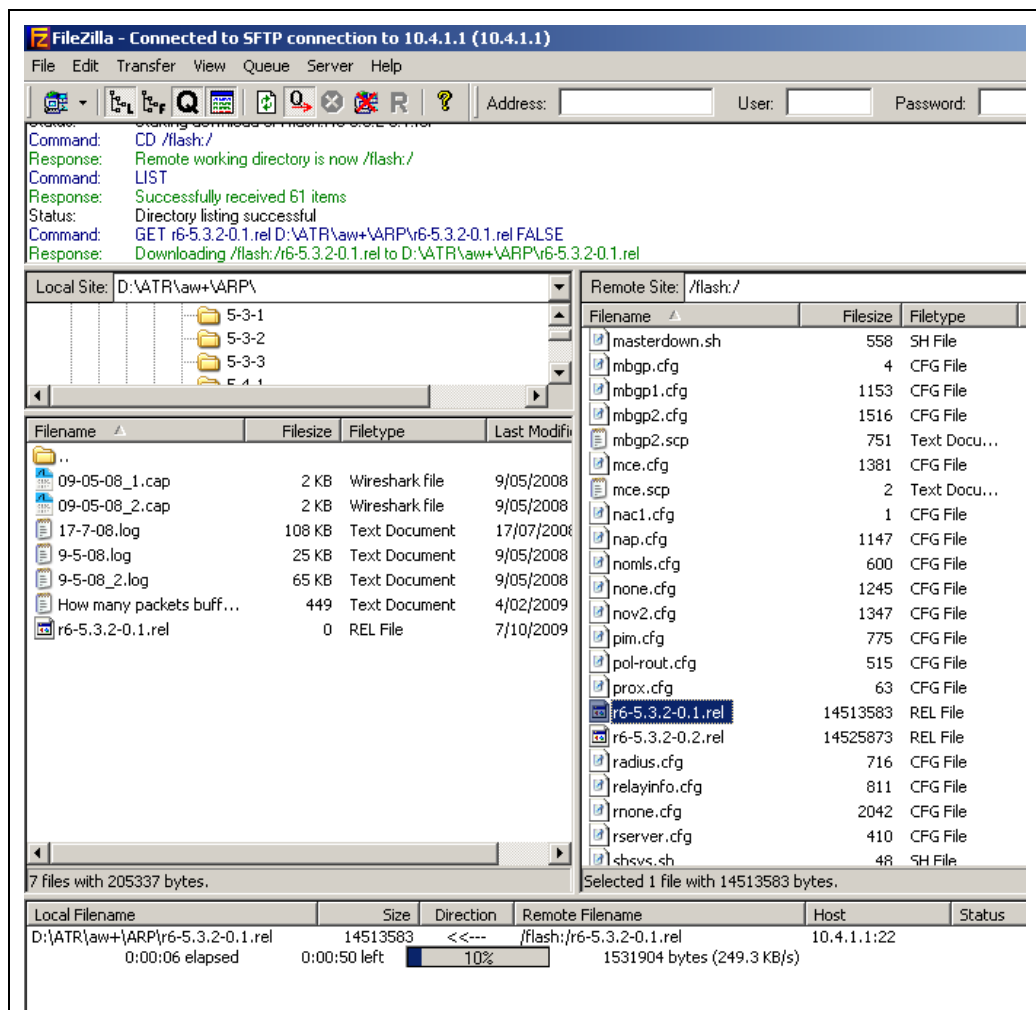
1. Select **File > Site Manager...**



2. Create an FTP site that uses the **servertype** "SFTP using SSH2". Filezilla automatically selects port 22 as the TCP port for this FTP site:



- Connect to the site. The contents of the file system on the switch are displayed in the **Remote Site** pane. Files can be transferred to and from the switch in the same way as they can be transferred to any FTP site by Filezilla:



Switching

How to View Switch Tables

You can view the contents of switch tables by using the command:

```
show platform table <table-name>
```

Commonly used tables are:

This keyword...	Displays...
fdb	the platform forwarding database table
ip	the platform IP table
ipmulti	the platform IP multicast table
l2mc	the platform L2 multicast table
macfull	the full platform MAC table—all MAC addresses that the switch has learned
port counters	counters from the platform port table—see the following section

How to View Port Counters

You can view port counters using the command:

```
awplus#show platform table port counters
```

The counters are similar to the AlliedWare port counters.

```
Switch Port Counters
-----

Port 1.0.1 Ethernet MAC counters:
Combined receive/transmit packets by size (octets) counters:
 64 0 512 - 1023 0
65 - 127 0 1024 - MaxPktSz 0
128 - 255 0
256 - 511 0

General Counters:
Receive Transmit
Octets 0 Octets 0
Pkts 0 Pkts 0
CRCErrors 0
MulticastPkts 0 MulticastPkts 0
BroadcastPkts 0 BroadcastPkts 0
FlowCtrlFrms 0 FlowCtrlFrms 0
OversizePkts 0
Fragments 0
Jabbers 0
UpsupportOpcode 0
UndersizePkts 0
Collisions 0
LateCollisions 0
ExcessivCollsns 0

Miscellaneous Counters:
MAC TxErr 0
MAC RxErr 0
Drop Events 0
```

Resiliency

x600 Resiliency Link

Note: This issue applies to the x600 Series Switches only.

Introduction

The resiliency link is an important component in the AlliedWare Plus Virtual Chassis Stacking (VCStack™) solution.

The resiliency link is an extra link between the stack members, which is independent of the stacking connections. It is used when switches lose contact with each other over the stacking connection. This link allows the Backup Member switch(es) to determine if the master is still present, and operational, via health-check messages sent by the master over the resiliency link interface.

Without a resiliency link: if communication is lost over the stacking connection, a Backup Member will automatically transition to Master status. So, if the Master switch was still operational, there would now be 2 active Masters in the stack.

With a resiliency link: the Backup members can see if the Master is still operational, so no Backup member transitions to Master unless it is required.

The problem

On the SwitchBlade™ x908, and the x900 family of switches, the out-of-band Ethernet port functions as the resiliency link interface. However, the x600 switches don't have an out-of-band Ethernet port. So a resiliency link within an x600 VCStack must use a switch port or ports. Because healthcheck messages need to be received by each stack Backup member unit, this means giving up one or more front-panel ports per switch, to be used solely for resiliency-link purposes.

The solution - a Resiliency VLAN

The x600 switch port(s) that will function as the resiliency link should be assigned to a dedicated VCStack **Resiliency VLAN**.

The resiliency VLAN should not be either:

- The Stack Management VLAN, or
- A VLAN that will carry any user traffic.

This VLAN must be used only for resiliency purposes, and should only carry data about VCStack healthcheck messages. This is achieved by **not** creating the resiliency VLAN in the switch's "VLAN Database" (like other user-defined VLANs).

There are two reasons for this:

- I. the resiliency link VLAN is handled internally in a very different way to other VLANs

2. users should not be able to change the resiliency link VLAN's configuration, apart from the **using resiliencylink** commands.

There are two commands required to configure the resiliency VLAN:

```
Stack resiliencylink
Switchport resiliencylink
```

Once these commands are executed, the resiliency link is active.

Configuring the x600 resiliency link

1. Once the x600 switches are stacked via the stacking cables, you can create the resiliency VLAN and add ports to it:

```
awplus#conf t
```

2. Enter configuration commands, one per line. End with Ctrl/Z.

```
awplus(config)#stack resiliencylink vlan4001
```

3. Configure two ports on each member in the stack as the resiliency link ports:

```
awplus(config)#int port1.0.1
awplus(config-if)#switchport resiliencylink
awplus(config-if)#exit
awplus(config)#int port1.0.2
awplus(config-if)#switchport resiliencylink
awplus(config-if)#exit
awplus(config)#int port2.0.1
awplus(config-if)#switchport resiliencylink
awplus(config-if)#exit
awplus(config)#int port2.0.2
awplus(config-if)#switchport resiliencylink
awplus(config-if)#exit
awplus(config)#int port3.0.1
awplus(config-if)#switchport resiliencylink
awplus(config-if)#exit
awplus(config)#int port3.0.2
awplus(config-if)#switchport resiliencylink
awplus(config-if)#exit
awplus(config)#int port4.0.1
awplus(config-if)#switchport resiliencylink
awplus(config-if)#exit
awplus(config)#int port4.0.2
awplus(config-if)#switchport resiliencylink
awplus(config-if)#exit
```

4. Check that this has been configured correctly using the command:

```
awplus#sh stack detail
Virtual Chassis Stacking detailed information

Stack Status:
-----
Normal operation
Operational Status           Enabled
Management VLAN ID           4094
Management VLAN subnet address 192.168.255.0

Stack member 1:
-----
ID                             1
Pending ID                     -
MAC address                    0015.77c2.4bb4
Last role change               Wed Sep 16 10:38:17 2009
Product type                   x600-24Ts
Role                           Backup Member
Priority                        128
Host name                      awplus-1
S/W version auto synchronization On
Fallback config                Not configured
Resiliency link                Successful
Port 1.0.1 status              Learnt neighbour 4
Port 1.0.2 status              Learnt neighbour 2

Stack member 2:
-----
ID                             2
Pending ID                     -
MAC address                    0015.7745.89d2
Last role change               Wed Sep 16 10:38:16 2009
Product type                   x600-24Ts
Role                           Backup Member
Priority                        128
Host name                      awplus
S/W version auto synchronization On
Fallback config                Not configured
Resiliency link                Successful
Port 2.0.1 status              Learnt neighbour 1
Port 2.0.2 status              Learnt neighbour 3

Stack member 3:
-----
ID                             3
Pending ID                     -
MAC address                    0015.77c2.4ba2
Last role change               Wed Sep 16 10:38:16 2009
Product type                   x600-24Ts
Role                           Active Master
Priority                        128
Host name                      awplus
S/W version auto synchronization On
Fallback config                Not configured
Resiliency link                Configured
Port 3.0.1 status              Learnt neighbour 2
Port 3.0.2 status              Learnt neighbour 4

Stack member 4:
-----
ID                             4
Pending ID                     -
MAC address                    0015.778e.62fa
Last role change               Wed Sep 16 10:38:16 2009
Product type                   x600-24Ts
Role                           Backup Member
Priority                        128
Host name                      awplus
S/W version auto synchronization On
Fallback config                Not configured
Resiliency link                Successful
Port 4.0.1 status              Learnt neighbour 2
Port 4.0.2 status              Learnt neighbour 1
```

Note: *There are no counters that can be viewed, because the resiliency link is only used when a Backup Member loses connectivity with the Master via the stacking cables.*

How VCS failover operates

When Backup Members lose XEM-STK connectivity with the Master, the resiliency-link determines whether the Master is still online. If no VCS healthcheck messages are received over the resiliency link within 2 seconds of failover, Backup Members assume that the Master is offline.

If the resiliency link is configured and active, but the interface is down, it is assumed that the Master is offline.

Failover situations in which the Backup Members know the master is rebooting always result in a Backup Member transitioning to Active Master. This occurs when the master is rebooted via the CLI, or when a node failover occurs due to processes on the master locking up or crashing.

If the Backup Member knows the Master is definitely online, then that Backup Member should become a **Fallback Master** or **Disabled Master**. These possible failover states are essentially the same as the Active Master (i.e. the master is running the active processes), but with differences in network configuration:

- **Fallback Master**
The stack operates as usual, but is running an alternative configuration file called the fallback configuration (fallback-config) to avoid network conflicts with the master. This provides a back-up IP address for members that become isolated from the Master; although the fallback-config can also potentially contain the complete configuration for an alternative stack setup.
- **Disabled Master**
The stack has disabled all its switch ports to avoid network conflicts, and is basically inactive. The stack is still assigned the Active process workload so the user can log in and reboot or reconfigure it. In this state, a separated slave won't disrupt the network because its trunked ports are still up. This is the default if neither the resiliency link nor the fallback-config is configured.

If the Backup Member has to leave the stack due to incompatible software, it should not cause network conflicts with the existing Master.

Health-check messages

Health-check messages are received if the Master is still online, but the stack will now split into two different 'stubs':

- The stub containing the existing Master continues operating as normal.
- The members of the other (Master-less) stub now use the fallback-config to form a second temporary stack. This utilizes the remaining stack members' resources without conflicting directly with the Master's configuration. If no fallback-config is specified for the stack, then the Master-less stub members disable their switch ports.

If no health-check messages are received, then the Master is assumed to be completely offline, and the other stack members can safely take over the Master's configuration.

Diagnostics

CPU Usage Spikes

Note: This issue occurred in AlliedWare Plus Release 5.2.2, and was resolved in AlliedWare Plus Release 5.3.1.

The problem

The CPU usage has the potential to spike to 40% every 15 seconds, as shown below:

```
Stack member 1:

Per second CPU load history

100
 90
 80
 70
 60
 50
 40      *          *          *          *
 30
 20
 10 *****
|...|...|...|...|...|...|...|...|...|...|...|...
Oldest Newest
CPU load% per second (last 60 seconds)
* = average CPU load%
```

These spikes in CPU usage are caused by the SNMP protocol. However, this occurs even when the SNMP process is disabled on the switch:

```
Dong_Bu_Ring#sh snmp-server
SNMP enable ..... No
SNMPv3 engine ID (configured) ..... Not set
SNMPv3 engine ID (actual)..... Not set
```

If you turn on Terminal Monitor, you will see that the following SNMP log messages occur every 15 seconds. This shows that SNMP is still polling the software protocol modules, even though it is disabled:

```
15:19:29 Dong_Bu_Ring LACP[1966]: AgentX: ping, Operational state, fail 0
15:19:29 Dong_Bu_Ring LACP[1966]: AgentX: pinging:
15:19:29 Dong_Bu_Ring LACP[1966]: AgentX: build Ping-PDU
15:19:29 Dong_Bu_Ring LACP[1966]: -> AgentX Header:
15:19:29 Dong_Bu_Ring LACP[1966]:     Version: 1
15:19:29 Dong_Bu_Ring LACP[1966]:     Type: 13 (Ping)
15:19:29 Dong_Bu_Ring LACP[1966]:     Flags: 00
15:19:29 Dong_Bu_Ring LACP[1966]:     <reserved>: 0
15:19:29 Dong_Bu_Ring LACP[1966]:     Session ID: 13 (0x0D)
15:19:29 Dong_Bu_Ring LACP[1966]: ->     Integer: 13 (0x0D)
15:19:29 Dong_Bu_Ring IMI[1928]: AgentX: ping, Operational state, fail 0
15:19:29 Dong_Bu_Ring IMI[1928]: AgentX: pinging:
15:19:29 Dong_Bu_Ring IMI[1928]: AgentX: build Ping-PDU
15:19:29 Dong_Bu_Ring IMI[1928]: -> AgentX Header:
15:19:29 Dong_Bu_Ring IMI[1928]:     Version: 1
15:19:29 Dong_Bu_Ring IMI[1928]:     Type: 13 (Ping)
15:19:29 Dong_Bu_Ring IMI[1928]:     Flags: 00
15:19:29 Dong_Bu_Ring IMI[1928]:     <reserved>: 0
15:19:29 Dong_Bu_Ring IMI[1928]:     Session ID: 14 (0x0E)
15:19:29 Dong_Bu_Ring IMI[1928]: ->     Integer: 14 (0x0E)
15:19:29 Dong_Bu_Ring LACP[1966]:     Transaction ID: 0 (0x00)
15:19:29 Dong_Bu_Ring LACP[1966]: ->     Integer: 0 (0x00)
15:19:29 Dong_Bu_Ring LACP[1966]:     Packet ID: 621120 (0x97A40)
15:19:29 Dong_Bu_Ring LACP[1966]: ->     Integer: 621120 (0x97A40)
15:19:29 Dong_Bu_Ring LACP[1966]:     Dummy Length: -(
15:19:29 Dong_Bu_Ring LACP[1966]: ->     Integer: 0 (0x00)
15:19:29 Dong_Bu_Ring LACP[1966]:     Payload
15:19:29 Dong_Bu_Ring LACP[1966]: ->     Integer (length of PDU) : 0 (0x00)
15:19:29 Dong_Bu_Ring LACP[1966]: AgentX: built packet okay
15:19:29 Dong_Bu_Ring LACP[1966]: AgentX: sending PDU-XDUMP:
15:19:29 Dong_Bu_Ring NSM[2005]: AgentX: ping, Operational state, fail 0
15:19:29 Dong_Bu_Ring NSM[2005]: AgentX: pinging:
15:19:29 Dong_Bu_Ring NSM[2005]: AgentX: build Ping-PDU
15:19:29 Dong_Bu_Ring NSM[2005]: -> AgentX Header:
15:19:29 Dong_Bu_Ring NSM[2005]:     Version: 1
15:19:29 Dong_Bu_Ring NSM[2005]:     Type: 13 (Ping)
15:19:29 Dong_Bu_Ring NSM[2005]:     Flags: 00
15:19:29 Dong_Bu_Ring NSM[2005]:     <reserved>: 0
15:19:29 Dong_Bu_Ring NSM[2005]:     Session ID: 12 (0x0C)
15:19:29 Dong_Bu_Ring NSM[2005]: ->     Integer: 12 (0x0C)
15:19:29 Dong_Bu_Ring 802.1X[1809]: AgentX: ping, Operational state, fail 0
15:19:29 Dong_Bu_Ring 802.1X[1809]: AgentX: pinging:
15:19:29 Dong_Bu_Ring 802.1X[1809]: AgentX: build Ping-PDU
15:19:29 Dong_Bu_Ring 802.1X[1809]: -> AgentX Header:
15:19:29 Dong_Bu_Ring 802.1X[1809]:     Version: 1
15:19:29 Dong_Bu_Ring 802.1X[1809]:     Type: 13 (Ping)
15:19:29 Dong_Bu_Ring 802.1X[1809]:     Flags: 00
15:19:29 Dong_Bu_Ring 802.1X[1809]:     <reserved>: 0
15:19:29 Dong_Bu_Ring 802.1X[1809]:     Session ID: 16 (0x10)
15:19:29 Dong_Bu_Ring 802.1X[1809]: ->     Integer: 16 (0x10)
15:19:29 Dong_Bu_Ring BGP[1846]: AgentX: ping, Operational state, fail 0
15:19:29 Dong_Bu_Ring BGP[1846]: AgentX: pinging:
15:19:29 Dong_Bu_Ring BGP[1846]: AgentX: build Ping-PDU
15:19:29 Dong_Bu_Ring BGP[1846]: -> AgentX Header:
15:19:29 Dong_Bu_Ring BGP[1846]:     Version: 1
15:19:29 Dong_Bu_Ring BGP[1846]:     Type: 13 (Ping)
15:19:29 Dong_Bu_Ring BGP[1846]:     Flags: 00
15:19:29 Dong_Bu_Ring BGP[1846]:     <reserved>: 0
15:19:29 Dong_Bu_Ring BGP[1846]:     Session ID: 15 (0x0F)
15:19:29 Dong_Bu_Ring BGP[1846]: ->     Integer: 15 (0x0F)
```

Why this occurs

In release 5.2.2: the `no snmp-server` command does not actually disable SNMP, it just de-configures SNMP so it is not available via the network. The SNMP software still continues to run and gather information from the protocol modules in the software. Even if SNMP appears to be disabled, the AgentX polling [as shown above] continues every 15 seconds.

The solution

In 5.3.1 and later releases: when SNMP is disabled, the connections between subagents and the master are broken. The lack of connections prevents the AgentX polling that would otherwise cause the CPU spikes.

In summary, this is expected behaviour in 5.2.2, and was fixed in 5.3.1.

MTR Switch Drops Packets

Introduction

My Trace Route (MTR) combines the functionality of the **traceroute** and **ping** programs into a single network diagnostic tool. This tool investigates the network connection between the host on which MTR is running, and a user-specified destination host.

The problem

MTR does not report packet loss when directed to an AlliedWare Plus switch. However, it reports very high packet loss when directed to a device beyond the switch.

Why this occurs

To understand why this occurs, it is important to understand how MTR works. The MTR website is misleading. It states “it sends a sequence ICMP ECHO requests to each one”, which is not strictly true.

What we have observed is that MTR sends two frames per 100ms. These two frames are ICMP echo requests, destined for 192.168.1.2. One has a Time-To-Live (TTL) of 1 and the other has a TTL of 2.

So, MTR is sending ICMP echo requests, destined for the final hop, but with decreasing TTL values. This means that routers along the path will respond with ICMP Time-To-Live Exceeded messages, instead of ICMP echo replies.

This is significant because many network equipment vendors limit the rate of ICMP messages that are generated.

Further information about ICMP rate limiting in the Linux Kernel is available at:

<http://www.kernel.org/doc/man-pages/online/pages/man7/icmp.7.html>

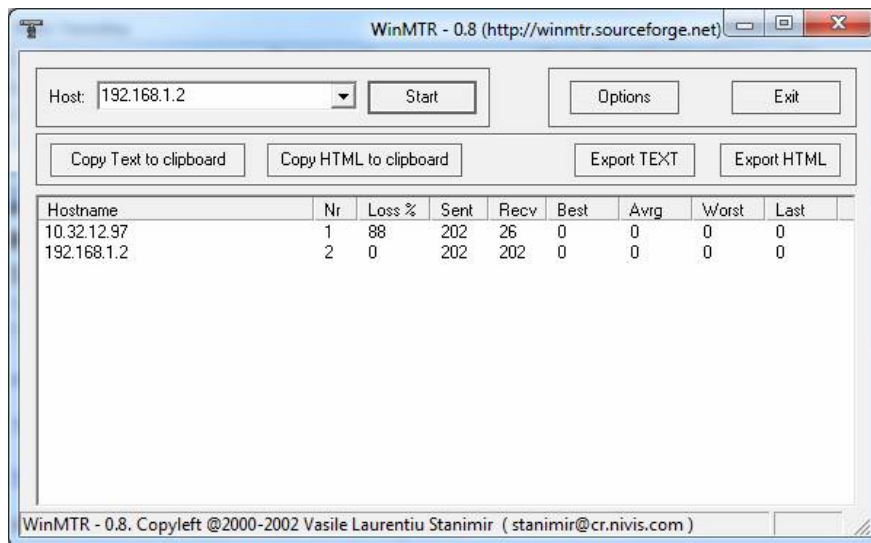
Example

1. The network is like this:

```
Pc1 (10.32.12.99) ----- (port1.1.1, vlan1, 10.32.12.97) x908
  (port1.1.2, vlan2, 192.168.1.1) ----- (192.168.1.2, port49)
  8648/2SP
```

2. When we run this test (ICMP packets at 100ms interval), MTR to 192.168.1.2 shows

enormous packet loss:



3. MTR sends two frames every 100ms. These are ICMP echo requests, destined for 192.168.1.2. One has a TTL of 1 and the other has a TTL of 2.
4. The packet with a TTL of 2 reaches its destination and an ICMP echo reply is sent. This is correct.
5. The packet with a TTL of 1 reaches the AlliedWare Plus switch (10.32.12.97) and the TTL expires, so the switch sends back an ICMP Time-to-live Exceeded message. This is also correct.

However, the Linux kernel employs ICMP rate limiting for certain ICMP packets. This means that only around 1 in 10 TTL expired packets will actually result in an ICMP Time-to-live Exceeded message in this scenario. This explains why MTR reports about 88% loss to the SwitchBlade x908.

6. In AlliedWare Plus, ICMP Echo Replies are not subject to the same rate limiting, which explains why when MTR is directed to the AlliedWare Plus switch, the rate of response is high. This is expected behaviour and is designed to prevent ICMP DoS attacks.

In summary

ICMP Time-To-Live Exceeded messages are rate-limited from an AlliedWare Plus switch, and ICMP Echoes are not rate limited. This explains the differences in behaviour.

The output example below shows the network traffic generated by MTR and the associated responses from the devices in the network. In this example, you can see that MTR sends two ICMP echo requests to 192.168.1.2. One has a TTL of 1, and one has a TTL of 2.

You can also see the ICMP time exceeded in-transit, with messages occurring approximately every 1 in 10 requests. This is the AlliedWare Plus Kernel rate-limiting the ICMP responses, and is the reason for the loss reported by MTR.

Note: This behaviour differs from the explanation on the MTR website. The website states that ICMP echo requests are sent to each host. This is not strictly true as shown by the capture below, where all ICMP Echoes are directed at the far end host, but with differing TTLs that will expire in transit and trigger a response from all L3 devices on route to the destination.

```
16:20:16.892634 IP (tos 0x0, ttl 1, id 45630, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 32514, length 44
16:20:16.943286 IP (tos 0x0, ttl 2, id 45631, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 32770, length 44
16:20:16.943975 IP (tos 0x0, ttl 63, id 17755, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 32770, length 44
16:20:16.996321 IP (tos 0x0, ttl 1, id 45632, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 33026, length 44
16:20:16.997926 IP (tos 0xc0, ttl 64, id 36459, offset 0, flags [none], proto ICMP (1), length 92) 10.32.12.97 > 10.32.12.99:
  ICMP time exceeded in-transit, length 72
  IP (tos 0x0, ttl 1, id 45632, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2: ICMP echo
  request, id 60967, seq 33026, length 44
16:20:17.047216 IP (tos 0x0, ttl 2, id 45633, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 33282, length 44
16:20:17.047976 IP (tos 0x0, ttl 63, id 17756, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 33282, length 44
16:20:17.100293 IP (tos 0x0, ttl 1, id 45634, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 33538, length 44
16:20:17.151302 IP (tos 0x0, ttl 2, id 45635, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 33794, length 44
16:20:17.151976 IP (tos 0x0, ttl 63, id 17757, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 33794, length 44
16:20:17.204249 IP (tos 0x0, ttl 1, id 45636, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 34050, length 44
16:20:17.255502 IP (tos 0x0, ttl 2, id 45637, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 34306, length 44
16:20:17.257006 IP (tos 0x0, ttl 63, id 17758, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 34306, length 44
16:20:17.308258 IP (tos 0x0, ttl 1, id 45638, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 34562, length 44
16:20:17.359319 IP (tos 0x0, ttl 2, id 45639, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 34818, length 44
16:20:17.360990 IP (tos 0x0, ttl 63, id 17759, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 34818, length 44
16:20:17.412669 IP (tos 0x0, ttl 1, id 45640, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 35074, length 44
16:20:17.463376 IP (tos 0x0, ttl 2, id 45641, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 35330, length 44
16:20:17.463989 IP (tos 0x0, ttl 63, id 17760, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 35330, length 44
16:20:17.516289 IP (tos 0x0, ttl 1, id 45642, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 35586, length 44
16:20:17.567295 IP (tos 0x0, ttl 2, id 45643, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 35842, length 44
16:20:17.567988 IP (tos 0x0, ttl 63, id 17761, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 35842, length 44
16:20:17.620316 IP (tos 0x0, ttl 1, id 45644, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 36098, length 44
16:20:17.671299 IP (tos 0x0, ttl 2, id 45645, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 36354, length 44
16:20:17.671989 IP (tos 0x0, ttl 63, id 17762, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 36354, length 44
16:20:17.724297 IP (tos 0x0, ttl 1, id 45646, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 36610, length 44
16:20:17.775392 IP (tos 0x0, ttl 2, id 45647, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 36866, length 44
16:20:17.775987 IP (tos 0x0, ttl 63, id 17763, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 36866, length 44
16:20:17.828298 IP (tos 0x0, ttl 1, id 45648, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 37122, length 44
16:20:17.879354 IP (tos 0x0, ttl 2, id 45649, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 37378, length 44
16:20:17.879989 IP (tos 0x0, ttl 63, id 17764, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 37378, length 44
16:20:17.932369 IP (tos 0x0, ttl 1, id 45650, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 37634, length 44
16:20:17.983342 IP (tos 0x0, ttl 2, id 45651, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 37890, length 44
16:20:17.984007 IP (tos 0x0, ttl 63, id 17765, offset 0, flags [none], proto ICMP (1), length 64) 192.168.1.2 > 10.32.12.99:
  ICMP echo reply, id 60967, seq 37890, length 44
16:20:18.036551 IP (tos 0x0, ttl 1, id 45652, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 38146, length 44
16:20:18.037920 IP (tos 0xc0, ttl 64, id 36460, offset 0, flags [none], proto ICMP (1), length 92) 10.32.12.97 > 10.32.12.99:
  ICMP time exceeded in-transit, length 72
  IP (tos 0x0, ttl 1, id 45652, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2: ICMP echo
  request, id 60967, seq 38146, length 44
16:20:18.087043 IP (tos 0x0, ttl 2, id 45653, offset 0, flags [none], proto ICMP (1), length 64) 10.32.12.99 > 192.168.1.2:
  ICMP echo request, id 60967, seq 38402, length 44
```

Hardware

Switch PSU Fault Analysis

Note: *This issue applies to the SwitchBlade x908 switch only.*

Introduction

The SwitchBlade x908 Switch provides AC and DC Power Supply options, and allows for Power Supply redundancy by providing two PSU Bays in the chassis. The PSUs are designed to provide long life and reliability. The units also provide good status and alarm communication for both monitoring and abnormal state information.

The AlliedWare Plus Operating System can provide PSU status information via either the CLI or SNMP MIBs. It has also been designed to extract as much information as possible when an alarm signal interrupt is received.

If a PSU becomes faulty, the micro-controller on the PSU may quickly decide to shut down, which means that information about what initiated failure can get lost. For this reason, AlliedWare Plus takes a snapshot of the information available as quickly as possible, once it receives an interrupt signal from the PSU. However, in the case of rapid shutdown, it cannot guarantee to capture the initial cause of the fault. Even so, the correct cause condition is usually stated, or can be deduced, as explained later in this article.

This Tips and Tricks item will aid in analysing and understanding any SwitchBlade x908 PWR05 PSU failures. The following sections will explain the types and meaning of information available from the PSU units, and explain about the variable results that can occur for given cause conditions.

Feature requirements

The ability to interrogate the I2C bus, to find an error code for logging after a PSU Indication Pin interrupt event, was introduced in Software Release 5.3.3-03. The error logging facility is important for PSU troubleshooting, therefore upgrading to this release or later is recommended.

PSU models this document applies to

There are two main PWR05 variants, an AC and a DC version. The table below indicates the names used:

Version	Allied Telesis Model Names	Manufacturer's Model Reference
AC Version	AT-PWR05-AC	FNP600-12S153G
DC Version	AT-PWR05-DC	FND850-12DRG or FND850-12DRS101G

Information types and their meanings

Indication pins

There are a set of indication pins that the PSU uses to communicate:

- Device Present
- PSU Fan/Temperature Fault
- PSU Power Output
- PSU Power Input

Note: These indication pin values are also visible when viewing **show system environment**.

Interrogation of PSU I²C device

The switch CPU can seek data from the PSU via the I²C bus. The data is sought in response to the **show system psu** command, or in response to interrupts due to state changes on the PSU's Fan /Temperature Fault indication pin. In the case of an interrupt, the information is presented as an Error Code in the switch's system log.

How information is presented to the user

After a PSU interrupt event, the **show log** will log a single octet Error code. This code is in fact the first (most significant) octet of the Fault Bytes.

Here is an example of the **show log** output:

```
awplus#01:15:24 awplus HPI: SENSOR PSU slot 2 - PSU Power Output: BAD
01:15:24 awplus HPI: SENSOR PSU slot 2 - PSU Fan/Temperature Fault: BAD -
Error code 0x10
01:15:24 awplus HPI: SENSOR PSU slot 2 - PSU Power Output: BAD
```

The **show system psu** command quotes a two octet Fault Bytes figure in the **Dynamic Data** section, as shown below:

```
x908#sh sys psu
System PSU Information

Resource ID: 7 Name: AT-PWR05-AC Bay: 2
  Part Number      : FNP600-12S153G
  Serial Number    : 080732-004PN
  Revision         : AA
  Mfg. date       : 2008-03-17
  Manufacturer     : POWER-ONE
  Mfg. location   : 2

Device Ratings:
Output rail 1    : 12000 mV, 51000 mA
Output rail 2    : 12000 mV, 500 mA
Output Power     : 606 W
Min AC input     : 90 V
Max AC input     : 264 V

Dynamic Data:
Fault Bytes      : 21 01
Time in service  : 3946 hours
Measured rail 1  : 0 mV, 0 mA
x908#
```

SNMP Traps

PSU Temperature and Fan Alarms also produce SNMP Trap events based on the AT-ENVMONv2-MIB. Information about this MIB is available in the SNMP MIBs chapter of the SwitchBlade x908 and x900 Series Switches AlliedWare Plus Operating System Software Reference:

http://www.alliedtelesis.com/media/datasheets/reference/x900_alliedware_plus_ref_a_v532.pdf

About these examples:

- The Error Code shows 0x10, meaning **Temperature-Prewarning**. However, we know that the PSU only actually alarms (causes interrupt) when **Over-Temperature** threshold is reached, therefore the code should have indicated 0x20. In this case, on interrupt the CPU has actually probed the PSU I2C device before the Fault Byte bits were changed.
- If a redundant PSU is still operational, after this PSU thermal failure the Fault Bytes will show a realistic end-result figure of 0x 21 01- this means **Over Temperature, Power Supply NOT OK, and Output I Voltage Not OK**.

Meaning of the show system PSU Fault Bytes

The PSU's I²C device expresses alarm states by setting individual bits within the Fault Bytes.

The following example shows the make-up of the two octets, and defines the bit positions of the significant alarms:

```
Two Bytes Position Numbers:  
<< MSB           LSB >>  
76543210  76543210
```

Fault Byte 1

(The Error Code Octet)

Bit Position /Meaning:

- 7 -
- 6 - Fan Not OK
- 5 - Over Temperature
- 4 - Temperature Pre-warning
- 3 -
- 2 -
- 1 - AC Not In range
- 0 - Power Supply NOT OK

Fault Byte 2

(This Octet is NOT quoted as part of Error code. It is miscellaneous information).

Bit Position /Meaning:

- 7 -
- 6 -
- 5 -
- 4 - Output I Current NOT OK
- 3 -
- 2 -
- 1 -
- 0 - Output I Voltage Not OK

Because alarm states are expressed by setting individual bits within the error byte, several alarms can be enabled simultaneously. A fault condition only has a distinct hex value if it is the only alarm active. If there are other faults, then the hex value is the sum of both fault values.

The original cause value is often only available for inspection for no longer than 1 second. This is why the alarm code quoted in the log is not always the cause code, and why **show system psu** often only shows a PSU shut-down status, rather than the cause condition.

Meaning of the show log error codes

As previously mentioned, when the PSU Fan /Temperature Fault indication pin changes state, this causes an interrupt to the switch's CPU, which then in turn interrogates the PSU's I²C for further information.

This information is displayed in a log message, and quotes a single octet error code. This single octet is in fact the first, or most significant octet of the Fault Bytes discussed above.

When translated to Hex values, the initial distinct error code values of fault conditions are:

0x10 - Temperature Pre-Warning
0x20 - Over Temperature
0x40 - Fan Fail

Note that these are not necessarily the values that will be logged.

For example: For **Over Temperature**, the binary value of the first octet of the Fault Bytes will be - 00100000 - and this translates to a hex value of 0x20. However, for **Power Supply NOT OK** the Fault Bytes may be 00100001, which is a hex value of 0x21.

Understanding the variable data results

Both versions of the PWR05 PSU were tested to show the typical Error Code and Fault Bytes values that are logged in failure conditions. Because the values dynamically change at the moment of failure, the captured value is not always the expected initial value.

This can be because interrogation has happened too quickly, before bits have been set; or too late, after bits have been reset.

Here are the tested typical values:

Model	Cause Fault Condition	Error code should be	Typical Final Value quoted in log error code	Tested Final show system PSU Fault Bytes value
PWR05 AC	Fan Fail	0x40	0x40	0x 01 01
PWR05 AC	Thermal Failure	0x20	0x10	0x 21 01
PWR05 DC	Fan Fail	0x40	0x00	0x 41 00
PWR05 DC	Thermal Failure	0x20	0x20	0x 21 01

Notes:

- The tested final **show system psu** Fault Bytes value is the expected value assuming there is a redundant PSU still operational. To enter the command **show system psu** on the switch after a power supply shut-down, the switch must have a redundant PSU still operational. If the PSU that shut down was the only operational PSU in the switch, then shut down of the PSU would have shut down the whole switch.
- Thermal failure should indicate 0x20. For the AC model, you typically see the Temp Pre-warning value 0x10 instead.
- The Fan Fail should indicate 0x40. For the DC version, you typically see 0x00 instead, because the bit is not set in time.

How to determine a PSU failure cause when the log is inconclusive

As previously mentioned, the AlliedWare Plus Operating System does not always capture the initial cause of the fault. If no cause issue is shown, then you need to figure out if the failure was due to Fan Failure, or to Over Temperature.

If the failure was due to Over Temperature, then the temperature would have been climbing prior to the shutdown event. As the temperature climbed, other sensors in the switch would have indicated some temperature events.

Therefore:

- If there were prior temperature alarms elsewhere in the switch, then it was caused by over-temperature. This is often caused by high ambient /room temperature.
- If not, it was caused by fan failure.

The SwitchBlade x908 directs air through the chassis first, and then through the PSU. Therefore in the case of high ambient temperatures, any over-temperature failure would be pre-warned by switch chassis or module temperature alarms.

While the PSU has a temperature pre-warning fault code, this state does not initiate an alarm state on the indication pins, therefore the PSU pre-warnings are not logged.

Temperature operating range

Allied Telesis Lab testing has shown that the PWR05 AC version can tolerate ambient air temperatures of up to 72 - 84 degrees C before tripping, for an AC supply of either 110v or 230v.

Official manufacturer documentation indicates a more conservative trip point as follows:

Over-temp set point: 71.5degrees C

Recovery temp: 65.5degrees C

Fault sequences

The PSU micro-controller fault sequence

1. PSU detects a fan fail or over-temperature condition.
2. PSU changes the indication pin for PSU Fan/Temperature Fault (causing an interrupt to the SBx908), and lights the PSU O/T LED.
3. PSU shuts the PSU output down, changes the indication pin for output power and extinguishes the PSU Power Out LED.

The SwitchBlade x908 CPU fault sequence

1. CPU receives an interrupt indicating that a PSU indication pin has changed state.
2. CPU retrieves the PSU indication pin states from the SBx908 PSU monitor.
3. CPU interrogates the PSU's I²C device to get the Fault Bytes.
4. CPU takes appropriate action to indicate the fault.

During a fault condition, the PSU Micro-Controller first commences its 3-step fault sequence. When PSU event #2 occurs, the SwitchBlade x908 begins its fault sequence.

The timing of PSU event #3 may fall at any point during the switch's fault sequence. If we are lucky, then PSU event #3 does not occur until the end of the SwitchBlade x908 fault sequence. But if PSU event #3 occurs somewhere in the middle of the SwitchBlade x908 fault sequence, the amount of information that the SwitchBlade x908 can present to the user about the cause of the error is unpredictable.

Addendum: **Information about upcoming POE supply**

At the date of publication, the Power over Ethernet (PoE) version of the PSU had not been released. Early indications are that this PSU will not have Fault Byte information available. It will only have a simple alarm supplied via Indication Pins.