White Rabbit Switch: Failures and Diagnostics

Grzegorz Daniluk
Adam Wujek

CERN BE-CO-HT
wr-switch-sw-v4.2

October 1, 2015
## Contents

1 Introduction .......................................................... 3

2 Possible Errors ......................................................... 4
   2.1 Timing error ...................................................... 4
   2.2 Data error ......................................................... 10
   2.3 Other errors ...................................................... 13

3 SNMP exports .......................................................... 21
   3.1 Operator/basic objects .......................................... 21
   3.2 Expert/extended status .......................................... 24
1 Introduction

This document tries to list all possible ways the White Rabbit Switch can break and describes the information exported from our device to help diagnose the problems.

The document is organized in two parts. First one (section 2) tries to list all the possible failures that may disturb synchronization and Ethernet switching. The structure of each failure description is the following:

Mode: for timing failures, it says which modes are affected. Possible values are:

- **Slave** - WR Switch has at least one Slave port synchronized to another WR device higher in the timing hierarchy (though it may be also Master to other WR/PTP devices lower in the timing hierarchy).
- **Grand Master** - WR Switch at the top of the synchronization hierarchy. It is synchronized to an external clock (e.g. GPSDO, Cesium) and provides timing to other WR/PTP devices.
- **Free-Running Master** - WR Switch at the top of the synchronization hierarchy. It provides timing to other WR/PTP devices but runs from a local oscillator (not synchronized to external atomic clock).

Description: What the problem is about, how important it is and what bad may happen if it occurs.

**SNMP objects:** Which SNMP objects should be monitored to detect the failure. These may be objects from [WR-SWITCH-MIB](#) or one of the standard MIBs used by the [net-snmp](#).

**Notes:** Optional comment for SNMP implementation. It may describe current implementation of ideas how to implement it in the future.

Section 3 is a documentation for people integrating WR switch into a control system, operators and WR experts. It describes all essential SNMP objects exported by the device divided into two groups: **Operator/basic objects, Expert objects**
2 Possible Errors

2.1 Timing error

As a timing error we define WR Switch not being able to provide its slave nodes/switches with correct timing information consistent with the rest of the WR network. This section contains the list of faults leading to a timing error.

2.1.1 PTP/PPSi went out of TRACK_PHASE

- **Status**: DONE
- **Severity**: ERROR
- **Mode**: Slave
- **Description**: If the PTP/PPSi WR servo goes out of the TRACK_PHASE state, this means something bad has happened and switch lost the synchronization to its Master.

**SNMP objects:**
- \texttt{WR-SWITCH-MIB::wrsPtpServoState.<n> – PTP servo state as string}
- \texttt{WR-SWITCH-MIB::wrsPtpServoStateN.<n> – PTP servo state as number}
- \texttt{WR-SWITCH-MIB::wrsPtpServoStateErrCnt}
- \texttt{WR-SWITCH-MIB::wrsPTPStatus}

**Note**: PTP servo state is exported as a string and a number.

2.1.2 Offset jump not compensated by Slave

- **Status**: DONE
- **Severity**: ERROR
- **Mode**: Slave
- **Description**: This may happen if Master resets its WR time counters (e.g. because it lost the link to its Master higher in the hierarchy or to external clock), but Slave switch does not follow the jump.

**SNMP objects:**
- \texttt{WR-SWITCH-MIB::wrsPtpClockOffsetPs.<n> – value of the offset in ps}
- \texttt{WR-SWITCH-MIB::wrsPtpClockOffsetPsHR.<n> – 32-bit signed value of the offset in ps; with saturation on overflow and underflow}
- \texttt{WR-SWITCH-MIB::wrsPtpClockOffsetErrCnt}
- \texttt{WR-SWITCH-MIB::wrsPTPStatus}

2.1.3 Detected jump in the RTT value calculated by PTP/PPSi

- **Status**: DONE
- **Severity**: ERROR
- **Mode**: Slave
- **Description**: Once WR link is established round-trip delay (RTT) can change smoothly due to the temperature variations. If a sudden jump is detected, that means erroneous timestamp was generated.
either on Master or Slave side. One cause of that could be the wrong value of the t24p transition point.

SNMP objects:
WR-SWITCH-MIB::wrsPtpRTT.<n>
WR-SWITCH-MIB::wrsPtpRTTErrCnt
WR-SWITCH-MIB::wrsPTPStatus

2.1.4 Wrong $\Delta_{TXM}$, $\Delta_{RXM}$, $\Delta_{TXS}$, $\Delta_{RXS}$ values are reported to the PTP/PPSi daemon

Status: DONE
Severity: ERROR
Mode: all
Description:
If $PTP/PPSi$ doesn’t get the correct values of fixed hardware delays, it won’t be able to calculate a proper Master-to-Slave delay. Although the estimated offset in $PTP/PPSi$ is close to 0, WRS won’t be synchronized to Master with the sub-nanosecond accuracy.

SNMP objects:
WR-SWITCH-MIB::wrsPtpDeltaTxM.<n>
WR-SWITCH-MIB::wrsPtpDeltaRxM.<n>
WR-SWITCH-MIB::wrsPtpDeltaTxS.<n>
WR-SWITCH-MIB::wrsPtpDeltaRxS.<n>
WR-SWITCH-MIB::wrsPTPStatus

2.1.5 SoftPLL became unlocked

Status: DONE
Severity: ERROR
Mode: all
Description:
If $SoftPLL$ loses lock, for any reason, Slave or Grand Master switch can no longer be syntonized and phase aligned with its time source. WRS in Free-running mode without properly locked Helper PLL is not able to perform reliable phase measurements for enhancing Rx timestamps resolution. For Grand Master the reason of $SoftPLL$ going out of lock might be disconnected 1-PPS/10MHz signals or external clock down. In that case, the switch goes into Free-running mode and resets WR time. Later we will have a holdover to keep the Grand Master switch disciplined in case it loses external reference.

SNMP objects:
WR-SWITCH-MIB::wrsSpllMode
WR-SWITCH-MIB::wrsSpllSeqState
WR-SWITCH-MIB::wrsSpllAlignState
WR-SWITCH-MIB::wrsSpllHlock
WR-SWITCH-MIB::wrsSpllMlock
WR-SWITCH-MIB::wrsSpllDelCnt
WR-SWITCH-MIB::wrsSoftPLLStatus
2.1.6  *SoftPLL* has crashed/restarted

**Status:** TODO (depends on *SoftPLL* mem read), (require changes in *lm32* software)

**Severity:** ERROR

**Mode:** all

**Description:**
If LM32 software crashes or restarts for some reason, its state may be either reseted or random (if for some reason variables were overwritten with junk values). In such case PLL becomes unlocked and switch is not able to provide synchronization to other devices.

**SNMP objects:**
- WR-SWITCH-MIB::wrsSpllIrqCnt
- WR-SWITCH-MIB::wrsStartCntSPLL *(not yet implemented)*

**Note:** We have a similar mechanism as in the *wrpc-sw* to detect if the LM32 program has restarted because of the CPU following a NULL pointer. However, LM32 program hangs on re-initialization phase. In addition to that, we can detect if *SoftPLL* is hanging (but not restarted) based on irq counter.

2.1.7  **Link to WR Master is down for slave**

**Status:** DONE

**Severity:** ERROR (will become WARNING with the switch-over)

**Mode:** Slave

**Description:**
In that case, WR Switch loses timing reference, resets counters responsible for keeping the WR time, and starts operating in a Free-Running Master mode.

**SNMP objects:**
- WR-SWITCH-MIB::wrsPortStatusLink.<n>
- WR-SWITCH-MIB::wrsPortStatusConfiguredMode.<n>
- WR-SWITCH-MIB::wrsSlaveLinksStatus

2.1.8  **Link to WR Master is up for master**

**Status:** DONE

**Severity:** ERROR

**Mode:** Grand Master, Free-Running Master

**Description:**
In that case there is probably wrong configuration. Neither the Grand Master nor the Free-Running Master should be connected to another WR Master.

**SNMP objects:**
- WR-SWITCH-MIB::wrsPortStatusLink.<n>
- WR-SWITCH-MIB::wrsPortStatusConfiguredMode.<n>
- WR-SWITCH-MIB::wrsSlaveLinksStatus

2.1.9  **PTP frames don’t reach ARM**

**Status:** DONE
Severity: ERROR
Mode: all
Description:
In this case, PTP/PPSi will fail to stay synchronized and provide synchronization. Even if WR servo is in the TRACK_PHASE state, it calculates new phase shift based on the Master-to-Slave delay variations. To calculate these variations, it still needs timestamped PTP frames flowing. There could be several causes of such fault:

- HDL problem (e.g. SwCore or Endpoint hanging)
- wr_nic.ko driver crash
- wrong VLANs configuration

SNMP objects:
WR-SWITCH-MIB::wrsPortStatusPtpTxFrames.<n>
WR-SWITCH-MIB::wrsPortStatusPtpRxFrames.<n>
WR-SWITCH-MIB::wrsPortStatusLink.<n>
WR-SWITCH-MIB::wrsPortStatusConfiguredMode.<n>
WR-SWITCH-MIB::wrsPTPFramesFlowing

Note: If the kernel driver crashes, there is not much we can do. We end up with either our system frozen or a reboot. For wrong VLAN configuration and HDL problems we can monitor if PTP frames are flowing on Slave port(s) of WRS and raise an alarm (change status word) if they don’t flow anymore. We should combine this with the link status (up/down). If VLANs are mis configured, we don’t receive PTP frames, but the link is still up. This could let us distinguish from a lack of frames due to the link down (which is a separate issue).

2.1.10 Detected SFP not supported for WR timing

Status: DONE
Severity: ERROR
Mode: all
Description:
By not supported SFP for WR timing we mean a transceiver that doesn’t have the alpha parameter and fixed hardware delays defined in the SFP database (CONFIG_SFPXX_PARAMS parameters in dot-config). The consequence is PTP/PPSi not having the right values to estimate link asymmetry. Despite PTP/PPSi offset being close to 0 ps, the device won’t be properly synchronized.

SNMP objects:
WR-SWITCH-MIB::wrsPortStatusConfiguredMode.<n>
WR-SWITCH-MIB::wrsPortStatusSfpVN.<n>
WR-SWITCH-MIB::wrsPortStatusSfpPN.<n>
WR-SWITCH-MIB::wrsPortStatusSfpVS.<n>
WR-SWITCH-MIB::wrsPortStatusSfpInDB.<n>
WR-SWITCH-MIB::wrsPortStatusSfpGbE.<n>
WR-SWITCH-MIB::wrsPortStatusSfpError.<n>
WR-SWITCH-MIB::wrsSFPsStatus

Note: WRS configuration allow to disable this check on some ports. That is because ports may be used for regular (non-WR) PTP synchronization or for data transfer only (no timing). In
that case any Gigabit SFP can be used (also copper). Detecting if a non-Gigabit Ethernet SFP is plugged into the cage is covered in a separate issue [2.3.9].

2.1.11  **PTP/PPSi process has crashed/restarted**

**Status:** DONE  
**Severity:** ERROR  
**Mode:** all  
**Description:**  
If the PTP/PPSi daemon crashes we lose any synchronization capabilities. Then Monit restarts the missing process. The number of process starts is stored in a corresponding object.  
**SNMP objects:**  
WR-SWITCH-MIB::wrsStartCntPTP  
WR-SWITCH-MIB::wrsBootUserspaceDaemonsMissing  
HOST-RESOURCES-MIB::hrSWRunName.<n>

2.1.12  **HAL process has crashed/restarted**

**Status:** DONE  
**Severity:** WARNING  
**Mode:** all  
**Description:**  
If HAL crashes, PTP/PPSi is not able to communicate with the hardware i.e. read phase shift, get timestamps, phase shift the clock etc. When HAL crashes, Monit will restart it.  
**SNMP objects:**  
WR-SWITCH-MIB::wrsStartCntHAL  
WR-SWITCH-MIB::wrsBootUserspaceDaemonsMissing  
HOST-RESOURCES-MIB::hrSWRunName.<n>

2.1.13  **Wrong configuration applied**

**Status:** TODO (to be done later)  
**Severity:** WARNING  
**Mode:** all  
**Description:**  
If there is a wrong configuration applied to the PTP/PPSi or HAL (i.e. wrong fixed delays, mode of operation etc.) there is not much we can do. The responsibility of WR experts (or person deploying the system) is to make sure that all the devices have a correct configuration. Later we can only generate warnings, if the key configuration options are changed remotely (e.g. Grand Master mode to Free-running Master or updated fixed hardware delays values). For misconfigured VLANs, we can monitor if PTP frames are flowing on Slave port(s) of the switch.  
**SNMP objects:** (not yet implemented)  
**Note:** monitor remote updates of key configuration options (PTP/WR mode, fixed hardware delays)
2.1.14 Switchover failed

- **Status:** for later
- **Severity:** ERROR
- **Mode:** Slave, Grand Master
- **Description:** (not yet implemented)

In case the primary timing link breaks, switchover is responsible for seamless switching to the backup one to keep the device in sync. If WRS operates in a Slave mode, switchover is about switching between two (or more) WR links to one or multiple WR Masters. If it operates in a Grand Master mode, it is about broken/lost connection to an external reference and switching to a backup WR link (another WR Master). Regardless of the configuration, if we fail to switch-over to a backup link (e.g. because it is down), WRS resets the time counters and continue the operation as a Free-Running Master.

**SNMP objects:** (not yet implemented)

**Note:** we should probably use parameters reported by the backup channel(s) of the SoftPLL and the backup PTP servo to be able to detect and report that something went wrong.

2.1.15 Holdover for too long

- **Status:** for later
- **Severity:** WARNING
- **Mode:** Grand Master
- **Description:** (not yet implemented)

Signaling active holdover is one thing, but if a Grand Master switch is kept in holdover for too long, it may drift away from the ideal external reference too much. All devices in a WR network will be still synchronized, but no longer in sync with the external reference.

**SNMP objects:** (not yet implemented)
2.2 Data error

As a data error we define WR Switch not being able to forward Ethernet traffic between devices connected to the ports.

This section contains the list of faults leading to a data error.

2.2.1 Link down

Status: DONE (to be changed later for switchover)
Severity: ERROR (will be WARNING with the switch-over)
Description:
This obviously stops the flow of frames on an Ethernet port and there is not much we can do besides reporting an error. Topology redundancy is a cure for that (if backup link is fine, and reconfiguration does not fail). There might be several causes of a link down:
- unplugged fiber
- broken fiber
- broken SFP
- wrong(non-complementary) pair of WDM SPF's used

However, we are not able to distinguish between them inside the switch.
SNMP objects:
IF-MIB::ifOperStatus.<n>
WR-SWITCH-MIB::wrsPortStatusLink.<n>

2.2.2 Fault in the Endpoint’s transmission/reception path

Status: DONE
Severity: ERROR
Description:
This fault covers various errors reported by the Endpoint, e.g. FIFO underrun in the Tx PCS or FIFO overrun in the Rx PCS, receiving invalid 8b10b code, CRC error etc.

SNMP objects:
WR-SWITCH-MIB::wrsPstatsTXUnderrun.<n>
WR-SWITCH-MIB::wrsPstatsRXOverrun.<n>
WR-SWITCH-MIB::wrsPstatsRXInvalidCode.<n>
WR-SWITCH-MIB::wrsPstatsRXPfilterDropped.<n>
WR-SWITCH-MIB::wrsPstatsRXPCSErrors.<n>
WR-SWITCH-MIB::wrsPstatsRXCRCErrors.<n>

2.2.3 Problem with the SwCore or Endpoint HDL module

Status: TODO (add monitoring of the Endpoint hangs, depend on HDL)
Severity: ERROR
Description:
If the SwCore is hanging, then the Ethernet forwarding is not performed on one or multiple ports. We have a HDL watchdog module which constantly monitors if the SwCore is not stuck. If such a situation is detected the whole SwCore is reset, all the frames enqueued in the Endpoints are acknowledged and lost. After this the switch can continue its operation and the watchdog triggers counter is incremented.

SNMP objects:
- WR-SWITCH-MIB::wrsGwWatchdogTimeouts
- WR-SWITCH-MIB::wrsPstatsTXFrames.<n>
- WR-SWITCH-MIB::wrsPstatsForwarded.<n>

Note: For Endpoint monitoring we could compare per-port RTUfwd counter with the Tx Endpoint counter for each port. RTUfwd counts all forwarding decisions from RTU to the port <n> (excluding PTP frames from NIC). If the sum of this number and RTU decisions generated from NIC is equal to the number of frames actually transmitted by the Endpoint, then everything works fine.

2.2.4 RTU is full and cannot accept more requests

Status: DONE
Severity: ERROR
Description:
If RTU is full for a given port, it’s not able to accept more requests and generate new responses. In such case frames are dropped in the Rx path of the Endpoint.

SNMP objects:
- WR-SWITCH-MIB::wrsPstatsRXDropRTUFull.<n>

2.2.5 Too much HP traffic / Per-priority queue full

Status: TODO (depends on HDL)
Severity: ERROR
Description:
If we get too much High Priority traffic, then SwCore will be busy all the time forwarding HP frames. This way regular/best effort traffic won’t be flowing through the switch. In the extreme case, HP traffic queue may become full and we start losing HP frames, which is unacceptable.

SNMP objects:
- WR-SWITCH-MIB::wrsPstatsFastMatchPriority.<n> - HP frames on a port
- WR-SWITCH-MIB::wrsPstatsRXFrames<n> - Total number of Rx frames on the port
- WR-SWITCH-MIB::wrsPstatsRXFrio0.<n> - Rx priorities 0-7
- [..]
- WR-SWITCH-MIB::wrsPstatsRXFrio7.<n>

Note: we need to get from SwCore the information about per-priority queue utilization, or at least an event when it’s full.

2.2.6 RTUd has crashed

Status: DONE
Severity: WARNING
Description:
If RTUd crashed, traffic would be still routed between the WRS ports, but only based on the already existing static and dynamic rules. There would be no learning or aging functionality. This means, MAC addresses wouldn’t be removed from the RTU table if a device is disconnected from a port. Without learning, each frame with yet unknown destination MAC would be broadcast to all ports (within a VLAN). When RTUd crashes, Monit will restart it.

SNMP objects:
WR-SWITCH-MIB::wrsStartCntRTUd
WR-SWITCH-MIB::wrsBootUserspaceDaemonsMissing
HOST-RESOURCES-MIB::hrSWRunName.<n>

2.2.7 Network loop - two or more identical MACs on two or more ports

Status: TODO (to be done later)
Severity: ERROR
Description:
In such case we have a ping-pong situation. If two ports receive frames with the same source MAC, it is learned on one of these ports. Then if it comes on a second port, it is learned on a second port, and removed from the first one. Later, MAC is learned again on the first port, and removed from the MAC table for the second port, and so on. This situation is a network configuration problem or eRSTP failure.

SNMP objects: (not yet implemented)
Note: we need to monitor the rtu_stat to detect ping-pong in the RTU table.

2.2.8 Wrong configuration applied (e.g. wrong VLAN config)

Status: TODO (to be done later)
Severity: WARNING
Description:
The same problem as described in the timing fault 2.1.9

2.2.9 Topology Redundancy failure

Status: for later
Severity: ERROR
Description: (not yet implemented)
Topology redundancy let’s us prevent from losing data when the primary uplink is down for some reason. However, if a backup link is also down or reconfiguration to backup link fails, we start losing data and an alarm should be raised.

SNMP objects: (not yet implemented)
Note: One thing we need to report is a backup link(s) going down, but we should also think about how to determine if there is some problem with eRSTP and if it may fail/has failed if the primary link is down.
2.3 Other errors

2.3.1 WR Switch did not boot correctly

Status: TODO (add rebooting system when boot is not successful, add stop restarting system after defined number of restarts)
Severity: ERROR
Description:
Every time the switch boots, we verify that all the services have started and are running correctly. If any of them fails, an alarm is raised. The SNMP object wrsBootSuccessful says if a WRS has booted correctly, FPGA is programmed, all kernel drivers are loaded and all daemons are up and running. If it’s not the case, we report what went wrong:

– status of reading HW information from dataflash
– status of programming FPGA and LM32
– status of loading kernel modules
– status of starting userspace daemons

SNMP objects:
WR-SWITCH-MIB::wrsBootSuccessful – status word informing whether switch booted correctly
WR-SWITCH-MIB::wrsRestartReason
WR-SWITCH-MIB::wrsRestartReasonMonit
WR-SWITCH-MIB::wrsConfigSource
WR-SWITCH-MIB::wrsConfigSourceUrl
WR-SWITCH-MIB::wrsBootHwinfoReadout
WR-SWITCH-MIB::wrsBootLoadFPGA
WR-SWITCH-MIB::wrsBootLoadLM32
WR-SWITCH-MIB::wrsBootKernelModulesMissing
WR-SWITCH-MIB::wrsBootUserspaceDaemonsMissing

Note: The idea is to reboot the system if it was not able to boot correctly. Then we use the scratchpad registers of the processor to keep the boot count. If the value of this counter is more than X we stop rebooting and try to have a system running with at least dropbear for SSH and net-snmp to allow remote diagnostics. If on the other hand the switch has booted correctly, we set the boot count to 0.

2.3.2 Dot-config error

Status: DONE
Severity: ERROR
Description:
Dot-config file used to configure the switch can be stored locally or retrieved from a central server. Additionally URL to the remote dot-config can be retrieved via DHCP request. When dot-config is fetch from the server it has to be verified before being applied. If downloading or verification has failed an alarm is raised.
SNMP objects:
WR-SWITCH-MIB::wrsBootSuccessful – status word informing whether switch booted correctly
WR-SWITCH-MIB::wrsConfigSource – source of a dot-config, local, remote or get URL to the dot-config via DHCP. When wrsConfigSource is set to the tryDhcp, then failure of getting dot-config’s URL via DHCP does not rise an error in wrsBootSuccessful
WR-SWITCH-MIB::wrsConfigSourceUrl – path to the dot-config on a server (if not local)
WR-SWITCH-MIB::wrsBootConfigStatus – result of the dot-config verification

2.3.3 Any userspace daemon has crashed/restarted

Status: TODO (depends on monit)
Severity: ERROR / WARNING (depending on the process)
Description:
Running processes are monitored by Monit. When any of them crashes, Monit restarts a missing process and increments a corresponding start counter. If a process is restarted 5 times within 100 seconds, then the entire switch is restarted.

SNMP objects:
HOST-RESOURCES-MIB::hrSWRunName.<n> - list of processes in standard MIB
WR-SWITCH-MIB::wrsStartCntHAL
WR-SWITCH-MIB::wrsStartCntPTP
WR-SWITCH-MIB::wrsStartCntRTUd
WR-SWITCH-MIB::wrsStartCntSshd
WR-SWITCH-MIB::wrsStartCntHttpd
WR-SWITCH-MIB::wrsStartCntSnmpd
WR-SWITCH-MIB::wrsStartCntSyslogd
WR-SWITCH-MIB::wrsStartCntWrsWatchdog
WR-SWITCH-MIB::wrsStartCntSPLL (not implemented)
WR-SWITCH-MIB::wrsBootUserspaceDaemonsMissing - number of missing processes
WR-SWITCH-MIB::wrsBootSuccessful - status word informing whether switch booted correctly

Note: We shall distinguish between crucial processes - error should be reported if one of them crashes; and less important processes (warning should be reported if they crash). If any of the processes has crashed, we need to restart it and increment a per-process counter reported through the SNMP.

Crucial processes (Error report if any of them crashes):
- PTP/PPSi
- wrsw_rtdud - after adding configuration preserving code on restart, RTUd could be crossed out from this list
- wrsw_hal

Less critical processes (Restarting them and Warning generation is enough):
- dropbear
- udhcpc
- rsyslogd
- snmpd
- lighttpd
- TRUd/eRSTPd - not yet implemented

wrsw rtud - we need to set the flag informing the process has crashed so that when it runs again it knows that HDL is already configured. It should not erase static entries in the RTU table (e.g. multicastrs for PTP), the static entries set by-hand as well as VLANs. Dynamic entries are not a problem. RTUd can learn all MACs after restarting. The only consequence will be increased network traffic due to frames broadcast until all the MACs are learned. In general, the source code has to be checked to make sure what is cleared on the startup and modified to preserve the configuration.

TRUd/eRSTPd - topology reconfiguration is done in hardware if needed, the daemon is used only to configure the TRU/RTU HDL module. However, the story is similar as with the RTUd. If eRSTPd crashes, we need to store this information so that when it runs again, it does not erase the whole configuration. Also if a topology reconfiguration happens while eRSTPd is down, HDL should keep the flag for the eRSTPd so that it’s aware the backup link is active.

2.3.4 Kernel crash

Status: DONE
Severity: ERROR
Description:
If the Linux kernel has crashed, system reboots. Until the next boot we have no synchronization, no SNMP to report the status, FPGA may be still forwarding Ethernet traffic, but based on dynamic and static routing rules from before the crash. Based on the SNMP objects below it is possible to figure out that reboot took place and what was the reason of the last reboot.

SNMP objects:
WR-SWITCH-MIB::wrsBootCnt
WR-SWITCH-MIB::wrsRebootCnt
WR-SWITCH-MIB::wrsRestartReason
WR-SWITCH-MIB::wrsFaultIP (not implemented)
WR-SWITCH-MIB::wrsFaultLR (not implemented)

Note: Unfortunately, right now it is not possible to distinguish whether the reboot was caused by the kernel panic function or the reboot command. Preserving the state of IP and LR registers has to be implemented.

2.3.5 System nearly out of memory

Status: DONE
Severity: WARNING
Description:
We need to monitor the amount of free memory, report it through SNMP and raise an alarm if it’s extremely low (but still enough to keep the system running).
SNMP objects:
WR-SWITCH-MIB::wrsMemoryTotal
WR-SWITCH-MIB::wrsMemoryUsed
WR-SWITCH-MIB::wrsMemoryUsedPerc - percentage of used memory
WR-SWITCH-MIB::wrsMemoryFree
WR-SWITCH-MIB::wrsMemoryFreeLow - warning or error on low memory

2.3.6 Disk space low

Status: DONE
Severity: WARNING
Description:
We need to monitor the amount of free disk space, report it through SNMP and raise an alarm if it's extremely low (but still enough to keep the system running).

SNMP objects:
WR-SWITCH-MIB::wrsDiskMountPath.<n>
WR-SWITCH-MIB::wrsDiskSize.<n>
WR-SWITCH-MIB::wrsDiskUsed.<n>
WR-SWITCH-MIB::wrsDiskFree.<n>
WR-SWITCH-MIB::wrsDiskUseRate.<n>
WR-SWITCH-MIB::wrsDiskFilesystem.<n>
WR-SWITCH-MIB::wrsDiskSpaceLow - warning or error on low disk space
HOST-RESOURCES-MIB::hrStorageDescr.<n>
HOST-RESOURCES-MIB::hrStorageSize.<n>
HOST-RESOURCES-MIB::hrStorageUsed.<n>

Note: Objects like HOST-RESOURCES-MIB::hrStorage*.<n> are available via standard MIB. The same functionality is implemented in WR-SWITCH-MIB objects wrsDisk*.<n> (to ease the implementation of wrsDiskSpaceLow).

2.3.7 CPU load too high

Status: DONE
Severity: WARNING
Description:
On a healthy switch the average CPU load should be below 0.1. Some actions like SNMP queries or web interface activity may increase the average system load. The system load averages for the past 1, 5 and 15 minutes are exported via SNMP objects. Additionally wrsCpuLoadHigh alerts when the load is too high.

SNMP objects:
WR-SWITCH-MIB::wrsCPULoadAvg1min
WR-SWITCH-MIB::wrsCPULoadAvg5min
WR-SWITCH-MIB::wrsCPULoadAvg15min
WR-SWITCH-MIB::wrsCpuLoadHigh - warning or error when CPU load too high
2.3.8 Temperature inside the box too high

Status: DONE
Severity: WARNING
Description:
If the temperature raises too high we might break our electronics inside the box. It also means that most probably one or both of the fans inside the box are broken and should be replaced. There are 4 temperature sensors monitored:

- \textit{IC19} - temperature below the FPGA
- \textit{IC20, IC17} - temperature near the SCB power supply circuit
- \textit{IC18} - temperature near the VCXO and PLLs (AD9516, CDCM6100)

\texttt{wrsTemperatureWarning} is raised when the temperature read from any of these sensors exceeds a threshold configured in the \textit{dot-config}. When at least one threshold temperature is not set \texttt{wrsTemperatureWarning} is set to \texttt{Threshold-not-set}.

SNMP objects:
WR-SWITCH-MIB::wrsTempFPGA
WR-SWITCH-MIB::wrsTempPLL
WR-SWITCH-MIB::wrsTempPSL
WR-SWITCH-MIB::wrsTempPSR
WR-SWITCH-MIB::wrsTempThresholdFPGA
WR-SWITCH-MIB::wrsTempThresholdPLL
WR-SWITCH-MIB::wrsTempThresholdPSL
WR-SWITCH-MIB::wrsTempThresholdPSR
WR-SWITCH-MIB::wrsTemperatureWarning

2.3.9 Not supported SFP plugged into the cage (especially non 1-Gb SFP)

Status: DONE
Severity: WARNING
Description:
If a not supported Gigabit optical SFP is plugged into the cage, then it’s a timing issue 2.1.10. However, if a non 1-Gb SFP is used, then no Ethernet traffic would be flowing on that port. It’s due to the fact, that we don’t have 10/100Mbit Ethernet implemented inside the WRS.

SNMP objects:
WR-SWITCH-MIB::wrsPortStatusSfpVN.<n>
WR-SWITCH-MIB::wrsPortStatusSfpPN.<n>
WR-SWITCH-MIB::wrsPortStatusSfpVS.<n>
WR-SWITCH-MIB::wrsPortStatusSfpGbE.<n>
WR-SWITCH-MIB::wrsPortStatusSfpError.<n>
WR-SWITCH-MIB::wrsSFPsStatus - status word for SFPs’ status

2.3.10 File system / Memory corruption

Description:
SNMP objects: (none)

Note: how shall we detect this? Based on the dmesg errors reported by UBI and system in general? This is bad, crazy things may happen, we can’t do much about it.

2.3.11 Kernel freeze

Description:
If kernel freezes we can do nothing. It can freeze e.g. due to some infinite loop in the irq handler. It’s like with the power failure, somebody has to go to the place where WRS is installed and investigate/restart the device.
SNMP objects: (none)
Note: If we have watchdog in our CPU it should be used.

2.3.12 Power failure

Description:
Power failure may be either a WRS problem (i.e. broken power supply inside the switch) or an external problem (i.e. providing voltage to the device). There is not much reporting we can do in such case. It’s up to the Network Management Station to raise an alarm if the SNMP Agent does not respond to the SNMP requests.
SNMP objects: (none)

2.3.13 Hardware problem

Description:
If any crucial hardware part breaks we’ll most probably notice it as one (or multiple) timing / data errors described previously. Besides that, we don’t have any self-diagnostics on-board.
Few examples:
– DAC / VCO - problems with synchronization
– cooling fans - rise of the temperature inside the WRS box (failure 2.3.8)
– power supply, ARM, FPGA - booting problem (failure 2.3.1)
– memory chip - data corruption (failure 2.3.10)
SNMP objects: (none)

2.3.14 Management link down

Description:
For obvious reasons we are not able to report through SNMP that the management link is down. This should be detected and reported by the NMS if it does not receive SNMP and ICMP responses from the WRS.
SNMP objects: (none)
2.3.15  **No static IP on the management port & failed to DHCP**

Description:
From operator’s point of view it is similar to the issue 2.3.14 WRS is not accessible through the management port, so its status cannot be reported. This should be detected and reported by the NMS if it does not receive SNMP and ICMP responses from the WRS. In such case WR expert should make a physical connection to the management USB port of the WRS to diagnose the problem.

SNMP objects: *(none)*

2.3.16  **IP address on the management port has changed**

**Status**: TODO  
**Severity**: WARNING  
**Description**:  
I’m not yet sure how we should report this. Probably SNMP is not the best choice because if the IP changes we’re no longer able to poll SNMP objects (until IP is updated also in the Network Management Station). We should either generate SNMP trap to NMS or send Syslog message to a central server.

SNMP objects: *(not yet implemented)*

2.3.17  **Multiple unauthorized access attempts**

**Status**: for later  
**Severity**: WARNING  
**Description**:  
If we observe many attempts to gain a root access through the ssh (or the web interface) this might be somebody trying to do something nasty. We should report such situation as a Warning.

SNMP objects: *(not yet implemented)*  

*Note*: Bad password event is reported by Syslog as a warning. We should probably use this information to add an SNMP object.

2.3.18  **Network reconfiguration (RSTP)**

**Status**: for later  
**Severity**: WARNING  
**Description**: *(not yet implemented)*  

If topology reconfiguration occurs because of the primary link failure, this fact should be reported through SNMP as a warning. It’s not critical situation, WR network still works. However, further investigation should be performed to repair the broken link.

SNMP objects: *(not yet implemented)*

2.3.19  **Backup link down**

**Status**: for later  
**Severity**: WARNING
Description: (not yet implemented)
This is related to the issue 2.3.18. If the WRS uses primary uplink, but the backup one fails, it’s not a critical fault. WR Network still works, but the link should be diagnosed and repaired to have the backup link operational in case the primary one fails.

SNMP objects: (not yet implemented)
3 SNMP exports

This section describes SNMP objects exported by the WR Switch. Objects within the WR-SWITCH-MIB are divided into two categories:

- **operator/basic objects (section 3.1)** - providing basic status of the switch. It should be used by a control system operators and people without a deep knowledge of the White Rabbit internals. These values report a general status of the device and high level errors.

- **expert/extended status objects (section 3.2)** - can be used by White Rabbit experts for the in-depth diagnosis of the switch failures. These values are verbose and should not be used by the operators.

3.1 Operator/basic objects

This section describes the general status MIB objects that are calculated based on the other SNMP (detailed) exports. Most of the status objects described in this section can have one of the following values:

- **NA** – status value was not calculated at all (returned value is 0). Something bad has happened.
- **OK** – status of the particular object is correct.
- **Warning** – objects used to calculate this value are outside the proper values, but problem in not critical enough to report Error.
- **WarningNA** – at least one of the objects used to calculate the status has a value NA or WarningNA.
- **Error** – error in values used to calculate the particular object.
- **FirstRead** – the value of the object cannot be calculated because at least one condition uses deltas between the current and previous value. This value should appear only at first SNMP read. Threated as a correct value.
- **Bug** – Something wrong has happened while calculating the object. If you see this please report to WR developers.

General Status objects:

- **wrsGeneralStatusGroup** – Group containing collective statuses of various subsys-

  - **wrsMainSystemStatus** – WRS general status of a switch can be OK, Warning or Error. When there is an error or warning please check the values of wrsOSStatus, wrsTimingStatus and wrsNetworkingStatus to find out which subsys-

  - **wrsOSStatus** – Collective status of the wrsOSStatusGroup. For details please check the group’s content.
- wrsTimingStatus – Collective status of the wrsTimingStatusGroup. For details please check the group’s content.

- wrsNetworkingStatus – Collective status of the wrsNetworkingStatusGroup. For details please check the group’s content.

- wrsDetailedStatusesGroup – Branch with collective statuses of various switch subsystems.

- wrsOSStatusGroup – Group with collective statuses of the embedded operating system running on the switch.
  * wrsBootSuccessful – Grouped status of wrsBootStatusGroup, indicating whether boot was successful. Error when dot-config source is wrong, unable to get the dot-config, unable to get URL to the dot-config, dot-config contains errors, unable to read the hwinfo, unable to load the FPGA bitstream, unable to load the LM32 software, any kernel modules or userspace daemons are missing (issue 2.3.1 2.3.2).
  * wrsTemperatureWarning – Report whether the temperature thresholds are not set or are exceeded (issue 2.3.8).
  * wrsMemoryFreeLow – Warning when 50% of the memory is used, error when more than 80% of the memory is used (issue 2.3.5).
  * wrsCpuLoadHigh – Warning when the average CPU load is more than 2 for the past 1min, 1.5 for 5min or 1 for 15min. Error when the average CPU load is more than 3 for the past 1min, 2 for 5min or 1.5 for 15min (issue 2.3.7).
  * wrsDiskSpaceLow – Warning when more than 80% of any disk partition is used. Error when more than 90% of any disk partition is used (issue 2.3.6).

- wrsTimingStatusGroup – Group with collective statuses of the timing subsystem.
  * wrsPTPStatus – Error when any of PTP error counters in wrsPtpDataTable (wrsPtpServoStateErrCnt, wrsPtpClockOffsetErrCnt or wrsPtpRTTErrCnt) has increased since the last scan (issue 2.1.1 2.1.2 2.1.3), at least one of the $\Delta T_X M$, $\Delta R_X M$, $\Delta T_X S$, $\Delta R_X S$ is 0 (issue 2.1.4) or PTP servo update counter is not increasing.
  * wrsSoftPLLStatus – Error when wrsSp11SeqState is not Ready, or wrsSp11AlignState is not Locked (for Grand Master mode), or any of wrsSp11Hlock, wrsSp11Mlock equals to 0 (for Slave mode) (issue 2.1.5).
    Warning when wrsSp11DelCnt > 0 (for Grand Master mode) or wrsSp11DelCnt has changed (for all other modes).
  * wrsSlaveLinksStatus – Error when link to Master is down for a switch in the Slave mode (issue 2.1.7). Additionally, Error when the link to Master is up for a switch in the Free-running Master or Grand Master mode (issue 2.1.8).
  * wrsPTPFramesFlowing – Error when PTP Tx/Rx frame counters on active links (Master / Slave ports) are not being incremented. (issue 2.1.9). Report the first run.
- wrsNetworkingStatusGroup – Group with collective statuses of the networking subsystem.
  * wrsSFPsStatus – Error when any of the SFPs reports an error. To find out which SFP caused the problem check wrsPortStatusSfpError.<n> (issue [2.1.10][2.3.9])
  * wrsEndpointStatus – Error when there is a fault in the Endpoint’s transmission/reception path (issue [2.2.2]).
  * wrsSwcoreStatus – Not used in the current release. Always reports OK.
  * wrsRTUStatus – Error when RTU is full and cannot accept more requests (issue [2.2.4]).

- wrsVersionGroup – Hardware, gateware and software versions. Additionally the serial number and other hardware information for the WRS.
  - wrsVersionSwVersion – software version (as returned from the git describe at build time).
  - wrsVersionSwBuildBy – software build-by (as returned from the git config --get-all user.name at build time)
  - wrsVersionSwBuildDate – software build date (_DATE_ at build time)
  - wrsVersionBackplaneVersion – hardware version of the minbackplane PCB
  - wrsVersionFpgaType – FPGA model inside the switch
  - wrsVersionManufacturer – name of the manufacturing company
  - wrsVersionSwitchSerialNumber – serial number (or string) of the switch
  - wrsVersionScbVersion – version of the SCB (the motherboard)
  - wrsVersionGwVersion – version of the gateware (FPGA bitstream)
  - wrsVersionGwBuild – build ID of the gateware (FPGA bitstream)
  - wrsVersionSwitchHdlCommitId – gateware version: commit ID from the wr_switch_hdl repository
  - wrsVersionGeneralCoresCommitId – gateware version: commit ID from the general-cores repository
  - wrsVersionWrCoresCommitId – gateware version: commit ID from the wr-cores repository
  - wrsVersionLastUpdateDate – date and time of last firmware update, this information may not be accurate, due to hard restarts or lack of the proper time at update.
3.2 Expert/extended status

Expert Status:

- wrsOperationStatus
  - wrsCurrentTimeGroup
    * wrsDateTAI
    * wrsDateTAIString
  - wrsBootStatusGroup
    * wrsBootCnt
    * wrsRebootCnt
    * wrsRestartReason
    * wrsFaultIP – Not implemented
    * wrsFaultLR – Not implemented
    * wrsConfigSource
    * wrsConfigSourceUrl
    * wrsRestartReasonMonit – Process that caused monit to trigger a restart.
    * wrsBootConfigStatus
    * wrsBootHwinfocReadout
    * wrsBootLoadFPGA
    * wrsBootLoadLM32
    * wrsBootKernelModulesMissing – List of kernel modules is defined in the source code.
    * wrsBootUserspaceDaemonsMissing – List of daemons is defined in the source code.
    * wrsGwWatchdogTimeouts – Number of times the watchdog has restarted the HDL module responsible for the Ethernet switching process (issue 2.2.3).
  - wrsTemperatureGroup
    * wrsTempFPGA
    * wrsTempPLL
    * wrsTempPSL
    * wrsTempPSR
    * wrsTempThresholdFPGA
    * wrsTempThresholdPLL
    * wrsTempThresholdPSL
    * wrsTempThresholdPSR
  - wrsMemoryGroup
    * wrsMemoryTotal
    * wrsMemoryUsed
    * wrsMemoryUsedPerc
* wrsMemoryFree
  - wrsCpuLoadGroup
    * wrsCPUloadAvg1min
    * wrsCPUloadAvg5min
    * wrsCPUloadAvg15min
  - wrsDiskTable – Table with a row for every partition.
    * wrsDiskIndex
    * wrsDiskMountPath
    * wrsDiskSize
    * wrsDiskUsed
    * wrsDiskFree
    * wrsDiskUseRate
    * wrsDiskFilesystem
  • wrsStartCntGroup
    - wrsStartCntHAL – issue 2.1.12, 2.3.3
    - wrsStartCntPTP – issue 2.1.11, 2.3.3
    - wrsStartCntRTUd – issue 2.2.6, 2.3.3
    - wrsStartCntSshd
    - wrsStartCntHttpd
    - wrsStartCntSnmpd
    - wrsStartCntSyslogd
    - wrsStartCntWrsWatchdog
  • wrsSpllState
    - wrsSpllVersionGroup
      * wrsSpllVersion
      * wrsSpllBuildDate
    - wrsSpllStatusGroup
      * wrsSpllMode
      * wrsSpllIrqCnt
      * wrsSpllSeqState
      * wrsSpllAlignState
      * wrsSpllHlock
      * wrsSpllMlock
      * wrsSpllHY
      * wrsSpllMY
      * wrsSpllDelCnt

25
• wrsPstatsTable – Table with pstats values, one row per port.

  - wrsPstatsIndex
  - wrsPstatsPortName
  - wrsPstatsTXUnderrun
  - wrsPstatsRXOverrun
  - wrsPstatsRXInvalidCode
  - wrsPstatsRXSyncLost
  - wrsPstatsRXPauseFrames
  - wrsPstatsRXPfilterDropped
  - wrsPstatsRXPcSErrors
  - wrsPstatsRXGiantFrames
  - wrsPstatsRXRuntFrames
  - wrsPstatsRXCRCErrors
  - wrsPstatsRXPclass0
  - wrsPstatsRXPclass1
  - wrsPstatsRXPclass2
  - wrsPstatsRXPclass3
  - wrsPstatsRXPclass4
  - wrsPstatsRXPclass5
  - wrsPstatsRXPclass6
  - wrsPstatsRXPclass7
  - wrsPstatsTXFrames
  - wrsPstatsRXFrames
  - wrsPstatsRXDropRTUFull
  - wrsPstatsRXPrio0
  - wrsPstatsRXPrio1
  - wrsPstatsRXPrio2
  - wrsPstatsRXPrio3
  - wrsPstatsRXPrio4
  - wrsPstatsRXPrio5
  - wrsPstatsRXPrio6
  - wrsPstatsRXPrio7
  - wrsPstatsRTUValid
  - wrsPstatsRTUResponses
  - wrsPstatsRTUDropped
- wrsPstatsFastMatchPriority
- wrsPstatsFastMatchFastForward
- wrsPstatsFastMatchNonForward
- wrsPstatsFastMatchRespValid
- wrsPstatsFullMatchRespValid
- wrsPstatsForwarded
- wrsPstatsTRURespValid

• wrsPtpDataTable – Table with a row per PTP servo instance.
  - wrsPtpIndex
  - wrsPtpPortName – The port on which the instance is running.
  - wrsPtpGrandmasterID – Not implemented.
  - wrsPtpOwnID – Not implemented.
  - wrsPtpMode
  - wrsPtpServoState
  - wrsPtpServoStateN
  - wrsPtpPhaseTracking
  - wrsPtpSyncSource
  - wrsPtpClockOffsetPs
  - wrsPtpClockOffsetPsHR
  - wrsPtpSkew
  - wrsPtpRTT
  - wrsPtpLinkLength
  - wrsPtpServoUpdates
  - wrsPtpDeltaTxM
  - wrsPtpDeltaRxM
  - wrsPtpDeltaTxS
  - wrsPtpDeltaRxS
  - wrsPtpServoStateErrCnt – Number of the servo updates when servo is out of the TRACK_PHASE (issue 2.1.1).
  - wrsPtpClockOffsetErrCnt – Number of servo updates when offset is larger than 500ps or smaller than -500ps (issue 2.1.2).
  - wrsPtpRTTErrCnt – Number of servo updates when RTT delta between subsequent updates is larger than 1000ps or smaller than -1000ps (issue 2.1.3).

• wrsPortStatusTable – Table with a row per port.
  - wrsPortStatusIndex
- wrsPortStatusPortName
- wrsPortStatusLink
- wrsPortStatusConfiguredMode
- wrsPortStatusLocked
- wrsPortStatusPeer
- wrsPortStatusSfpVN
- wrsPortStatusSfpPN
- wrsPortStatusSfpVS
- wrsPortStatusSfpInDB
- wrsPortStatusSfpGbE
- wrsPortStatusSfpError
- wrsPortStatusPtpTxFrames
- wrsPortStatusPtpRxFrames