White Rabbit Switch: Failures and Diagnostics

Grzegorz Daniluk
Adam Wujek

CERN BE-CO-HT

wr-switch-sw-v5.0

December 16, 2016

Contents

1 Introduction

This document provides information about the diagnostics of White Rabbit switches. It is a complementary documentation to the official *White Rabbit Switch: User's Manual* published with every stable firmware release. Please refer to this user manual for a basic information about the switch and guidelines on its configuration.

White Rabbit Switch firmware starting from *v4.2* provides diagnostic mechanisms in the form of SNMP objects and Syslog messages. This document is organized in two parts. It starts with a description of the SNMP objects and procedures to be followed if various errors are reported (section 2). This first part is meant for the operators and people integrating a WR switch into a control system, without the deep knowledge about the White Rabbit internals. These people usually have to perform a quick diagnostics and decide on actions to restore a WR network. Second part of the document tries to list all the possible failures that may disturb synchronization and Ethernet switching (section 3). It is meant for the WR experts to help them with in-depth diagnosis of the problems reported by SNMP.

This document has many internal hyperlinks that associate general SNMP status objects and expert SNMP objects with related problems' description and the other way round. These links can be easily used when reading the document on a computer.

2 SNMP diagnostics and solving problems

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

- General status objects for operators (section 2.1) - provide a summary about the status of a switch and several main subsystems (like timing, networking, OS). These should be used by control system operators and users without a comprehensive knowledge of the White Rabbit internals. These exports provide a general status of the device and high level errors which is enough in most cases to perform a quick repair.
- Expert objects (section 2.2) - can be used by White Rabbit experts for the in-depth diagnosis of the switch failures. These values are verbose and normally should not be used by the operators.

Description of the general status objects in section 2.1 includes also a list of actions to follow if a particular object reports an error. These repair procedures don't require any in-depth knowledge about White Rabbit. Independently of an error reported, there are some common remarks that apply to all situations:

- If a procedure given for a specific SNMP object does not solve the problem, please contact WR experts to perform a more in-depth analysis of the network. For this, you should provide a complete dump of the WRS status generated in the first step of each procedure.
- The first action in most of the procedures below named *Dump state* requires simply calling a tool provided by WR developers that reads all the detailed information from the switch and writes it to a single file that can be later analyzed by the experts.
  TODO: point to the tool once it's done
- If a problem solving procedure requires restarting or replacing a broken WR Switch, please make sure that after the repair, all other WR devices connected to the affected switch are synchronized and do not report any problems.
- If a procedure requires replacing a switch with a new unit, the broken one should be handled to WR experts or the switch manufacturer to investigate the problem.
- Until version 5.0 of the switch firmware Linux inside the WR Switch enumerated WR interfaces starting from 0. This means we have to use internally port indexes 0..17. However, the port numbers printed on the front panel were 1..18. Syslog messages generated from the switch used the Linux port numbering. The consequence was that every time Syslog says there is a problem on port X, this refered to port index X+1 on the front panel of the switch. From the v5.0 ports are numbered wri1 to wri18.

2.1 General status objects for operators

This section describes the general status MIB objects that represent the overall status of a device and its subsystems. They are organized in a tree structure (fig.1) where each object reports a problem based on the status of its child objects. SNMP objects in the third layer of this tree are calculated based on the SNMP expert objects. Most of the status objects described in this section can have one of the following values:

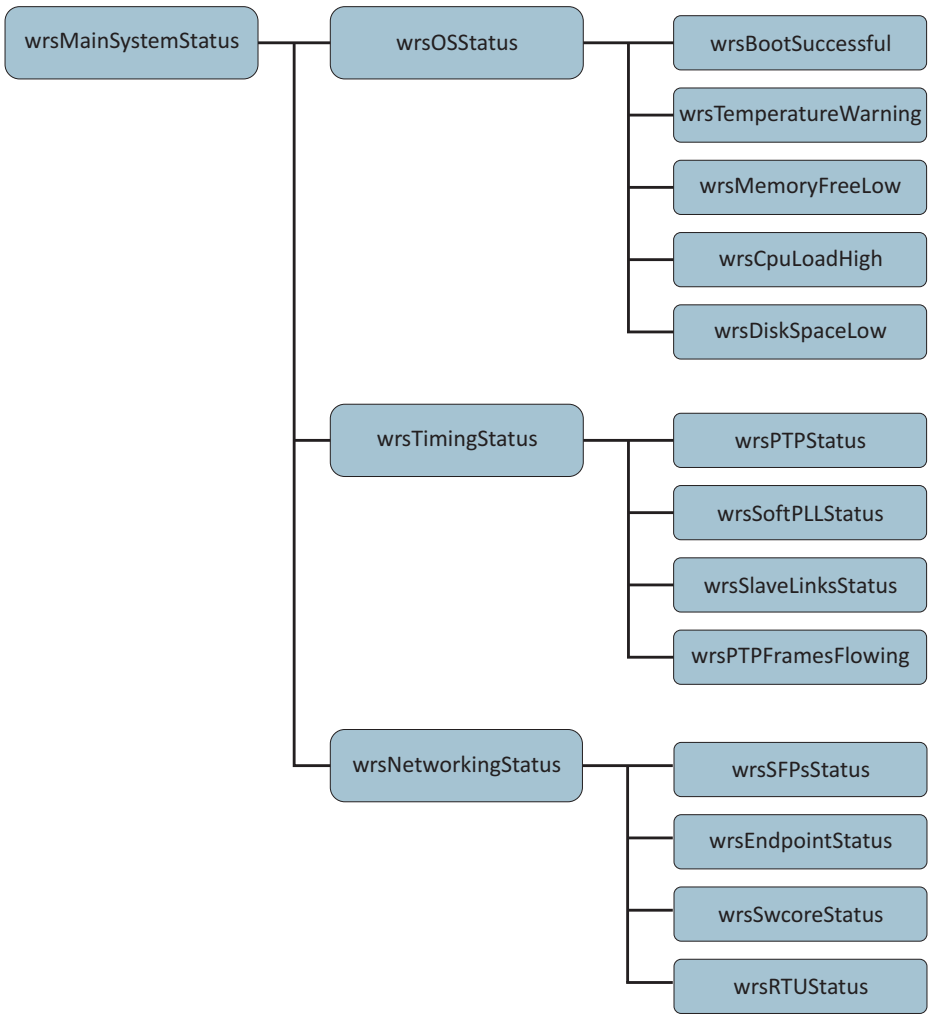

Figure 1: The structure of general status objects for operators

- `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. To be treated as a correct value.

- `Bug` – Something wrong has happened while calculating the object. If you see this please report to WR developers.

SNMP objects:

- `WR-SWITCH-MIB::wrsGeneralStatusGroup`
  Group containing collective status of the switch and its various subsystems.

  - `wrsMainSystemStatus`
    Description: WRS general status of a switch can be `OK`, `Warning` or `Error`. In case of an error or warning, please check the values of `wrsOSStatus`, `wrsTimingStatus` and `wrsNetworkingStatus` to find out which subsystem causes the problem.
    Related problems: 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.5, 3.1.7, 3.1.8, 3.1.9, 3.1.10, 3.1.11, 3.1.12, 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.5, 3.2.6, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.6, 3.3.7, 3.3.8, 3.3.9

  - `wrsOSStatus`
    Description: Collective status of the operating system running on WR switch. In case of an error or warning, please check status objects in the `wrsOSStatusGroup`.
    Related problems: 3.1.11, 3.1.12, 3.2.6, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.6, 3.3.7, 3.3.8

  - `wrsTimingStatus`
    Description: Collective status of the synchronization subsystem. In case of an error or warning, please check status objects in the `wrsTimingStatusGroup`.
    Related problems: 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.5, 3.1.7, 3.1.8, 3.1.9, 3.2.1

  - `wrsNetworkingStatus`
    Description: Collective status of the Ethernet switching subsystem. In case of an error or warning, please check status objects in the `wrsNetworkingStatusGroup`.
    Related problems: 3.1.10, 3.2.2, 3.2.3, 3.2.4, 3.2.5, 3.3.9

- `WR-SWITCH-MIB::wrsDetailedStatusesGroup`
  Branch with collective statuses of various switch subsystems.

  - `wrsOSStatusGroup`
    Description: Group with collective statuses of the embedded operating system running on the switch. 3.1.9

    * `wrsBootSuccessful`
      Description: 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, failed to update firmware due to checksum error, unable to download custom boot script, custom boot script fails during execution, fails setting up auxclk on connector clk2, fails setting up a limit on the Rx bandwidth of the traffic that goes from WR ports to Linux (throttling), fails

setting up VLANs.

On error:

1. Dump state
2. Check `wrsBootConfigStatus`, `wrsCustomBootScriptStatus` if it reports an error, please verify your WRS configuration.
3. Restart the switch
4. Please consult WR experts if the problem persists.

Related problems: 3.1.9, 3.1.11, 3.1.12, 3.2.6, 3.3.1, 3.3.2, 3.3.3, 3.3.4

* `wrsTemperatureWarning`

  Description: Reports whether the temperature thresholds are not set or are exceeded.

  On error:

  1. Dump state
  2. Verify if your switch configuration contains valid temperature thresholds. By default, they are all set to 80 °C.
  3. Verify if cooling of the rack where WR Switch is installed works properly.
  4. Verify if both cooling fans in the back of the WR Switch case are working.
  5. Replace the switch with a new unit and consult the WR Switch manufacturer for a repair.

  Related problems: 3.3.8

* `wrsMemoryFreeLow`

  Description: Reports `Warning` when more than 50%, or `Error` when more than 80% of the memory is used.

  On error:

  1. Dump state
  2. Restart the switch
  3. Send the dumped state of the switch to WR experts for analysis as this might mean there is some internal problem in the WRS firmware.

  Related problems: 3.3.5

* `wrsCpuLoadHigh`

  Description: Reports `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.

  On error:

  1. Dump state
  2. Restart the switch
  3. Send the dumped state of the switch to WR experts for analysis as this might mean there is some internal problem in the WRS firmware.

  Related problems: 3.3.7

* `wrsDiskSpaceLow`

  Description: `Warning` when more than 80% of any disk partition is used. `Error` when more than 90% of any disk partition is used.

On error:

1. Dump state
2. Check the values of *CONFIG_WRS_LOG_** configuration options on the switch. These are the parameters describing where log messages should be sent from various processes in the switch. Normally users don't need to modify them, but if any of them is set to a file in the WRS filesystem (e.g. /tmp/snmp.log) this may reduce the free space after some time of operation.
3. Restart the switch
4. Send the dumped state of the switch to WR experts for analysis as this might mean there is some internal problem in the WRS firmware.

Related problems: 3.3.6

- `wrsTimingStatusGroup`

  Description: Group with collective statuses of the timing subsystem.

  - `wrsPTPStatus`

    Description: Reports the status of PTP daemon running on the switch.
    `Error` when any of PTP error counters in `wrsPtpDataTable` (`wrsPtpServoStateErrCnt.<n>`, `wrsPtpClockOffsetErrCnt.<n>` or `wrsPtpRTTErrCnt.<n>`) has increased since the last scan (issue 3.1.1, 3.1.2, 3.1.3), at least one of the Δ_{TXM}, Δ_{RXM}, Δ_{TXS}, Δ_{RXS} is 0 (issue 3.1.4) or PTP servo update counter is not increasing.
    On error:

    1. Dump state
    2. Check `wrsSoftPLLStatus` on the Master (WR device one step higher in a timing hierarchy). Eventually proceed to investigate the problem on the Master switch. Otherwise, continue with the primary WRS.
    3. Verify if the link to WR Master was not lost by checking the object `wrsSlaveLinksStatus`.
    4. If this is not the case, restart the switch.
    5. If the problem persists replace the switch with a new unit.

    Related problems: 3.1.1, 3.1.2, 3.1.3, 3.1.4

  - `wrsSoftPLLStatus`

    Description: Reports the status of the PLLs inside the switch.
    `Error` when `wrsSpllSeqState` is not *Ready*, or `wrsSpllAlignState` is not *Locked* (for Grand Master mode), or any of `wrsSpllHlock`, `wrsSpllMlock` equals to 0 (for Boundary Clock mode).
    `Warning` when `wrsSpllDelCnt` > 0 (for Grand Master mode) or `wrsSpllDelCnt` has changed (for all other modes).
    On error:
    For GrandMaster WRS:

    1. Dump state
    2. Check 1-PPS and 10 MHz signals coming from an external source. Verify if they are properly connected and, in case of a GPS receiver, check if it is synchronized

and locked.

3. Restart the GrandMaster switch.
4. If the problem persists, replace the switch with a new unit.

For Boundary Clock WRS:

1. Dump state
2. Check `wrsSoftPLLStatus` on the Master. Eventually proceed to investigate the problem on the Master switch.
3. Verify if the link to WR Master was not lost by checking the object `wrsSlaveLinksStatus`.
4. Restart the switch.
5. If the problem persists, replace the switch with a new unit.

Related problems: 3.1.5

* `wrsSlaveLinksStatus`

Description: Reports the status of the link on WR ports configured to slave role. `Error` when link to master is down or softpll mode is not slave for a switch in the Boundary Clock mode. Additionally, `Error` is generated when the link to master is up for a switch in the Free-running Master or Grand Master mode.

On error:

For Master/GrandMaster WRS:

1. Check the configuration of the switch. Especially if the *Timing Mode* is correctly set (i.e. if it was not accidentally set to *Boundary Clock*).
2. Check the role of each port timing configuration. They should be all set to *master*. If any of them is set to *slave* you should verify if there is no WR Master connected to it.

For Boundary Clock WRS:

1. Check the fiber connection on the slave port of the WRS.
2. Check the configuration of the switch. Especially if the *Timing Mode* is correctly set (i.e. if it was not accidentally set to *Grand-Master* or *Free-Running Master*).
3. Check the VLANs configuration. Whether both switches are configured to use the same VLAN.
4. Check the status of the WR Master connected to the slave port of the WRS.
5. Replace the faulty switch with a new unit, if this does not solve the problem, make sure your fiber link is not broken.

Related problems: 3.1.7, 3.1.8, 3.2.1

* `wrsPTPFramesFlowing`

Description: Reports `Error` when PTP frames are not being sent and received on active WR ports - Tx/Rx frame counters on active links (master / slave ports) are not being incremented. Reports also `FirstRead` value.

On error:

1. Check Syslog message to determine the WR port on which the problem is reported. You should see a message similar to this one:
   `SNMP: wrsPTPFramesFlowing failed for port 1`

2. Check your network layout and the WR Switch configuration. If you have some non-WR devices connected to ports of the WR Switch (e.g. computer sending/receiving only data, without the need of synchronization), these ports should have their role in the timing configuration set to *non-wr* to avoid PTP traffic or *none* to disable PTP traffic at all.
3. Check the status of a WR device connected to the reported port.
4. Restart the switch.
5. If the problem persists, please contact WR experts for in-depth investigation.

Related problems: 3.1.9

- `wrsNetworkingStatusGroup`

  Description: Group with collective statuses of the networking subsystem.

  - `wrsSFPsStatus`

    Description: Reports the status of SFP transceivers inserted to the switch.
    `Error` when any of the SFPs reports an error. To find out which SFP caused the problem check `wrsPortStatusSfpError.<n>`.
    On error:

    1. Check `wrsPortStatusSfpError.<n>` SNMP objects or Syslog messages to determine the WR port on which the problem is reported. In case of Syslog, you should see a message similar to this one:
       `Unknown SFP vn="AVAGO" pn="ABCU-5710RZ" vs="AN1151PD8A" on port wri2`
    2. If the reported port is intended to be used with WR not compatible equipment (e.g. using a copper SFP module), to avoid SNMP errors set this port to *non-wr*. To disable PTP traffic on this port set it to *none*.
    3. Otherwise, you should use a WR-supported SFP module and make sure it is declared together with calibration values in the WRS configuration.

    Related problems: 3.1.10, 3.3.9

  - `wrsEndpointStatus`

    Description: Reports the status of Ethernet MAC endpoints on WR ports
    `Error` when there is a fault in the Endpoint's transmission/reception path.
    On error:

    1. Make several state dumps.
    2. Restart the switch.
    3. Check Syslog messages to determine the WR port on which the problem is reported. You should see a message similar to this one:
       `SNMP: wrsEndpointStatus failed for port 1`
    4. Check the fiber link on a reported port, i.e. try replacing SFP transceivers on both sides of the link, try using another fiber.
    5. If the problem persists, please contact WR experts for in-depth investigation.

    Related problems: 3.2.2

  - `wrsSwcoreStatus`

Description: Reports the status of the Ethernet switching module.
Status object not implemented in the current firmware release. Always reports `OK`.
On error:

1. Dump state.
2. Restart the switch.
3. Please contact WR experts since this might mean that either there is too much high priority traffic in your network, or there is some internal problem in the WRS firmware.

Related problems: 3.2.3, 3.2.5

* `wrsRTUStatus`
  Description: Reports the status of the routing module responsible for deciding where (to which port) incoming Ethernet frames should be forwarded.
  `Error` when RTU is overloaded and cannot accept more requests.
  On error:
  1. Dump state
  2. Restart the switch.
  3. If possible, try reducing the load of small Ethernet frames flowing through your switch. If possible in your application, try using larger Ethernet frames with lower load to transfer information.

  Related problems: 3.2.4

- `wrsVersionGroup`
  Description: Hardware, gateware and software versions. Additionally the serial number and other hardware information for the WRS.

  * `wrsVersionSwVersion`
    Description: Software version in the form of release version and eventually git commit from the repository (information provided from *git describe* command at build time).

  * `wrsVersionSwBuildBy`
    Description: Information who has built the firmware running on the switch (provided from `git config --get-all user.name` command at build time).

  * `wrsVersionSwBuildDate`
    Description: Firmware build date (`__DATE__` at build time).

  * `wrsVersionBackplaneVersion`
    Description: Hardware version of the Minibackplane board.

  * `wrsVersionFpgaType`
    Description: FPGA model inside the switch.

  * `wrsVersionManufacturer`
    Description: Name of the manufacturing company.

* `wrsVersionSwitchSerialNumber`
  Description: Serial number of the switch.

* `wrsVersionScbVersion`
  Description: Hardware version of the main SCB board.

* `wrsVersionGwVersion`
  Description: Version of the FPGA bitstream (Gateware).

* `wrsVersionGwBuild`
  Description: Build ID of the FGPA bitstream - the synthesis date

* `wrsVersionSwitchHdlCommitId`
  Description: FPGA bitstream commit ID from the main `wr_switch_hdl` repository.

* `wrsVersionGeneralCoresCommitId`
  Description: FPGA bitstream commit ID from the `general-cores` repository.

* `wrsVersionWrCoresCommitId`
  Description: FPGA bitstream commit ID from the `wr-cores` repository.

* `wrsVersionLastUpdateDate`
  Description: Date and time of the last firmware update. This information may not be accurate, due to hard restarts or lack of the proper time during the upgrade.

2.2 Expert objects

SNMP objects:

- `WR-SWITCH-MIB::wrsOperationStatus`
  - `wrsCurrentTimeGroup`
    - `wrsDateTAI`
    - `wrsDateTAIString`
  - `wrsBootStatusGroup`
    - `wrsBootCnt` 3.3.4
    - `wrsRebootCnt` 3.3.4
    - `wrsRestartReason` 3.3.1, 3.3.4
    - `wrsFaultIP`
      Not implemented 3.3.4
    - `wrsFaultLR`
      Not implemented 3.3.4
    - `wrsConfigSource`
      Source of a configuration file. When it is set to `tryDhcp`, then a failure of getting the URL via DHCP does not rise an error in `wrsBootSuccessful` 3.3.1, 3.3.2
    - `wrsConfigSourceUrl`
      Path to the dot-config on a remote server (if local file is not used). 3.3.1, 3.3.2
    - `wrsRestartReasonMonit`
      Process that caused `monit` to trigger a restart. 3.3.1
    - `wrsBootConfigStatus`
      Result of the dot-config verification. 3.3.2
    - `wrsBootHwinfoReadout` 3.3.1
    - `wrsBootLoadFPGA` 3.3.1
    - `wrsBootLoadLM32` 3.3.1
    - `wrsBootKernelModulesMissing`
      List of kernel modules is defined in the source code. 3.3.1
    - `wrsBootUserspaceDaemonsMissing`
      Number of missing (not running while they should) processes in embedded Linux.

3.1.11, 3.1.12, 3.2.6, 3.3.1, 3.3.3

  * wrsGwWatchdogTimeouts
    Number of times the watchdog has restarted the HDL module responsible for the Ethernet switching process 3.2.3

  * wrsFwUpdateStatus
    Status of the last firmware update 3.3.1

  * wrsCustomBootScriptSource
    Source of the custom script that can be executed once at boot time. It can be used to setup a switch in a way not supported by dot-config 3.3.1

  * wrsCustomBootScriptSourceUrl
    Path to the custom boot script on a remote server (if local scripts is not used) 3.3.1

  * wrsCustomBootScriptStatus
    Result of custom boot script execution 3.3.1

  * wrsAuxClkSetStatus
    Result of setting up auxclk on connector clk2 3.3.1

  * wrsThrottlingSetStatus
    Result of setting up a limit on the Rx bandwidth of the traffic that goes from WR ports to Linux (throttling) 3.3.1

  * wrsVlansSetStatus
    Result of setting up auxclk on connector clk2 3.1.9, 3.3.1

- wrsTemperatureGroup

  * wrsTempFPGA 3.3.8

  * wrsTempPLL 3.3.8

  * wrsTempPSL 3.3.8

  * wrsTempPSR 3.3.8

  * wrsTempThresholdFPGA 3.3.8

  * wrsTempThresholdPLL 3.3.8

  * wrsTempThresholdPSL 3.3.8

  * wrsTempThresholdPSR 3.3.8

- wrsMemoryGroup

    * wrsMemoryTotal 3.3.5
    * wrsMemoryUsed 3.3.5
    * wrsMemoryUsedPerc
      Percentage of used memory. 3.3.5
    * wrsMemoryFree 3.3.5
  ◦ wrsCpuLoadGroup
    * wrsCPULoadAvg1min 3.3.7
    * wrsCPULoadAvg5min 3.3.7
    * wrsCPULoadAvg15min 3.3.7
  ◦ wrsDiskTable
    Table with a row for every partition.
    * wrsDiskIndex.<n>
    * wrsDiskMountPath.<n> 3.3.6
    * wrsDiskSize.<n> 3.3.6
    * wrsDiskUsed.<n> 3.3.6
    * wrsDiskFree.<n> 3.3.6
    * wrsDiskUseRate.<n> 3.3.6
    * wrsDiskFilesystem.<n> 3.3.6
• WR-SWITCH-MIB::wrsStartCntGroup
  ◦ wrsStartCntHAL
    issue 3.1.12, 3.3.3 3.1.12, 3.3.3
  ◦ wrsStartCntPTP
    issue 3.1.11, 3.3.3 3.1.11, 3.3.3
  ◦ wrsStartCntRTUd
    issue 3.2.6, 3.3.3 3.2.6, 3.3.3
  ◦ wrsStartCntSshd 3.3.3
  ◦ wrsStartCntHttpd 3.3.3
  ◦ wrsStartCntSnmpd 3.3.3

- wrsStartCntSyslogd 3.3.3
- wrsStartCntWrsWatchdog 3.3.3
- wrsStartCntSPLL
  Not implemented 3.1.6, 3.3.3

- WR-SWITCH-MIB::wrsSpllState
  - wrsSpllVersionGroup
    - wrsSpllVersion
    - wrsSpllBuildDate
    - wrsSpllBuildBy
  - wrsSpllStatusGroup
    - wrsSpllMode 3.1.5
    - wrsSpllIrqCnt 3.1.6
    - wrsSpllSeqState 3.1.5
    - wrsSpllAlignState 3.1.5
    - wrsSpllHlock 3.1.5
    - wrsSpllMlock 3.1.5
    - wrsSpllHY
    - wrsSpllMY
    - wrsSpllDelCnt 3.1.5

- WR-SWITCH-MIB::wrsPstatsHCTable
  Table with pstats values, one row per port.
  - wrsPstatsHCIndex.<n>
  - wrsPstatsHCPortName.<n>
  - wrsPstatsHCTXUnderrun.<n> 3.2.2
  - wrsPstatsHCRXOverrun.<n> 3.2.2
  - wrsPstatsHCRXInvalidCode.<n> 3.2.2

- wrsPstatsHCRXSyncLost.<n> 3.2.2
- wrsPstatsHCRXPauseFrames.<n>
- wrsPstatsHCRXPfilterDropped.<n> 3.2.2
- wrsPstatsHCRXPCSErrors.<n> 3.2.2
- wrsPstatsHCRXGiantFrames.<n>
- wrsPstatsHCRXRuntFrames.<n>
- wrsPstatsHCRXCRCErrors.<n> 3.2.2
- wrsPstatsHCRXPclass0.<n>
- wrsPstatsHCRXPclass1.<n>
- wrsPstatsHCRXPclass2.<n>
- wrsPstatsHCRXPclass3.<n>
- wrsPstatsHCRXPclass4.<n>
- wrsPstatsHCRXPclass5.<n>
- wrsPstatsHCRXPclass6.<n>
- wrsPstatsHCRXPclass7.<n>
- wrsPstatsHCTXFrames.<n> 3.2.3
- wrsPstatsHCRXFrames.<n>
  Total number of Rx frames on port *n* 3.2.5
- wrsPstatsHCRXDropRTUFull.<n> 3.2.4
- wrsPstatsHCRXPrio0.<n>
  Rx priority 0 3.2.5
- wrsPstatsHCRXPrio1.<n>
  Rx priority 1 3.2.5
- wrsPstatsHCRXPrio2.<n>
  Rx priority 2 3.2.5
- wrsPstatsHCRXPrio3.<n>
  Rx priority 3 3.2.5
- wrsPstatsHCRXPrio4.<n>

Rx priority 4 3.2.5

- wrsPstatsHCRXPrio5.<n>
  Rx priority 5 3.2.5
- wrsPstatsHCRXPrio6.<n>
  Rx priority 6 3.2.5
- wrsPstatsHCRXPrio7.<n>
  Rx priority 7 3.2.5
- wrsPstatsHCRTUValid.<n>
- wrsPstatsHCRTUResponses.<n>
- wrsPstatsHCRTUDropped.<n>
- wrsPstatsHCFastMatchPriority.<n>
  HP frames on port *n* 3.2.5
- wrsPstatsHCFastMatchFastForward.<n>
- wrsPstatsHCFastMatchNonForward.<n>
- wrsPstatsHCFastMatchRespValid.<n>
- wrsPstatsHCFullMatchRespValid.<n>
- wrsPstatsHCForwarded.<n> 3.2.3
- wrsPstatsHCTRURespValid.<n>

- WR-SWITCH-MIB::wrsPtpDataTable
  Table with a row per PTP servo instance.
  - wrsPtpIndex.<n>
  - wrsPtpPortName.<n>
    The port on which the instance is running.
  - wrsPtpGrandmasterID.<n>
    Not implemented.
  - wrsPtpOwnID.<n>
    Not implemented.
  - wrsPtpMode.<n>
  - wrsPtpServoState.<n>

PTP servo state as string 3.1.1

- wrsPtpServoStateN.<n>
  PTP servo state as number 3.1.1
- wrsPtpPhaseTracking.<n>
- wrsPtpSyncSource.<n>
- wrsPtpClockOffsetPs.<n>
  value of the offset in ps 3.1.2
- wrsPtpClockOffsetPsHR.<n>
  32-bit signed value of the offset in picoseconds, with saturation 3.1.2
- wrsPtpSkew.<n>
- wrsPtpRTT.<n> 3.1.3
- wrsPtpLinkLength.<n>
- wrsPtpServoUpdates.<n>
- wrsPtpDeltaTxM.<n> 3.1.4
- wrsPtpDeltaRxM.<n> 3.1.4
- wrsPtpDeltaTxS.<n> 3.1.4
- wrsPtpDeltaRxS.<n> 3.1.4
- wrsPtpServoStateErrCnt.<n>
  Number of the servo updates when servo is out of the TRACK_PHASE. 3.1.1
- wrsPtpClockOffsetErrCnt.<n>
  Number of servo updates when offset is larger than 500ps or smaller than -500ps. 3.1.2
- wrsPtpRTTErrCnt.<n>
  Number of servo updates when RTT delta between subsequent updates is larger than 1000ps or smaller than -1000ps. 3.1.3
- wrsPtpServoUpdateTime.<n>
  TAI Nanosecond of the last servo's update.

- WR-SWITCH-MIB::wrsPortStatusTable
  Table with a row per port.
  - wrsPortStatusIndex.<n>

- wrsPortStatusPortName.<n>
- wrsPortStatusLink.<n> 3.1.7, 3.1.8, 3.1.9, 3.2.1
- wrsPortStatusConfiguredMode.<n> 3.1.7, 3.1.8, 3.1.9, 3.1.10
- wrsPortStatusLocked.<n>
- wrsPortStatusPeer.<n>
- wrsPortStatusSfpVN.<n> 3.1.10, 3.3.9
- wrsPortStatusSfpPN.<n> 3.1.10, 3.3.9
- wrsPortStatusSfpVS.<n> 3.1.10, 3.3.9
- wrsPortStatusSfpInDB.<n> 3.1.10
- wrsPortStatusSfpGbE.<n> 3.1.10, 3.3.9
- wrsPortStatusSfpError.<n> 3.1.10, 3.3.9
- wrsPortStatusPtpTxFrames.<n> 3.1.9
- wrsPortStatusPtpRxFrames.<n> 3.1.9

Objects from other MIBs:

- HOST-RESOURCES-MIB::hrSWRunName.<n>
  List of running processes. 3.1.11, 3.1.12, 3.2.6, 3.3.3
- HOST-RESOURCES-MIB::hrStorageDescr.<n> 3.3.6
- HOST-RESOURCES-MIB::hrStorageSize.<n> 3.3.6
- HOST-RESOURCES-MIB::hrStorageUsed.<n> 3.3.6
- IF-MIB::ifOperStatus.<n> 3.2.1

3 Possible Errors

This section tries to identify all the possible ways the White Rabbit Switch can fail. The structure of each error description is the following:

Status: describes the implementation status of the WRS diagnostics detecting the fault. Can be one of the following:
- DONE: all the SNMP objects are implemented and the problem is reported by a switch
- TODO: not all of the SNMP objects are already implemented, the problem is either reported only in some situations or not reported at all
- *for later*: the problem concerns functionality that is not yet present in the stable release of the WR switch firmware i.e. it will never happen with the current stable firmware release.

Severity: describes how critical is the fault. Currently we distinguish two severity levels:
- WARNING - means that despite the fault the synchronization and Ethernet switching functionality were not affected so the switch behaves correctly in the WR network.
- ERROR - means that the fault is critical and most probably a WR switch misbehaves in a WR network, possibly causing also problems to other WR devices connected to this switch.

Mode: for timing failures, it describes which modes are affected. Possible values are:
- *Boundary Clock* - the 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* - the WR Switch at the top of the synchronization hierarchy. It is synchronized to an external clock (e.g. GPS, Cesium) and provides timing to other WR/PTP devices.
- *Free-Running Master* - the 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 an external clock).
- *all* - any WR switch can be affected regardless the timing mode.

Description: What the problem is about, how important it is and what are the effects 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 the SNMP implementation. It may describe the current implementation of ideas or how to implement it in the future.

3.1 Timing error

As a timing error we define the 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.

3.1.1 *PTP/PPSi* went out of `TRACK_PHASE`

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

```
WR-SWITCH-MIB::wrsPtpServoState.<n>
WR-SWITCH-MIB::wrsPtpServoStateN.<n>
WR-SWITCH-MIB::wrsPtpServoStateErrCnt.<n>
WR-SWITCH-MIB::wrsPTPStatus
WR-SWITCH-MIB::wrsTimingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

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

3.1.2 Offset jump not compensated by Slave

Status: DONE
Severity: ERROR
Mode: *Boundary Clock*
Description:
This may happen if the 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 the WR Slave does not follow the jump.
SNMP objects:

```
WR-SWITCH-MIB::wrsPtpClockOffsetPs.<n>
WR-SWITCH-MIB::wrsPtpClockOffsetPsHR.<n>
WR-SWITCH-MIB::wrsPtpClockOffsetErrCnt.<n>
WR-SWITCH-MIB::wrsPTPStatus
WR-SWITCH-MIB::wrsTimingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

3.1.3 Detected jump in the RTT value calculated by *PTP/PPSi*

Status: DONE
Severity: ERROR
Mode: *Boundary Clock*
Description:
Once a WR link is established the round-trip delay (RTT) can change smoothly due to the temperature variations. However, if a sudden jump is detected, that means that an erroneous timestamp was generated either on the Master or the 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.<n>
```

```
WR-SWITCH-MIB::wrsPTPStatus
WR-SWITCH-MIB::wrsTimingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

3.1.4 Wrong Δ_{TXM}, Δ_{RXM}, Δ_{TXS}, Δ_{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, the WRS won't be synchronized to the 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
WR-SWITCH-MIB::wrsTimingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

3.1.5 *SoftPLL* became unlocked

Status: DONE (to be improved with holdover)
Severity: ERROR
Mode: *all*
Description:
If the *SoftPLL* loses lock, for any reason, Boundary Clock 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 a Grand Master the reason of *SoftPLL* going out of lock might be disconnected 1-PPS/10MHz signals or that the external clock is down. In that case, the switch goes into Free-running mode and resets the 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
WR-SWITCH-MIB::wrsTimingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

3.1.6 *SoftPLL* has crashed/restarted

Status: TODO *(depends on SoftPLL mem read), (requires changes in lm32 software)*
Severity: ERROR
Mode: *all*
Description:
If the LM32 software crashes or restarts for some reason, its state may be either reset 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`
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.

3.1.7 Link to WR Master is down for slave

Status: DONE
Severity: ERROR (will become WARNING with the switch-over)
Mode: *Boundary Clock*
Description:
If a Boundary Clock switch loses the link on its Slave port, the timing reference is lost. The switch 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`
`WR-SWITCH-MIB::wrsTimingStatus`
`WR-SWITCH-MIB::wrsMainSystemStatus`

3.1.8 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 a 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`

WR-SWITCH-MIB::wrsTimingStatus
WR-SWITCH-MIB::wrsMainSystemStatus

3.1.9 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 the WR servo is in the TRACK_PHASE state, it calculates a 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
WR-SWITCH-MIB::wrsTimingStatus
WR-SWITCH-MIB::wrsVlansSetStatus
WR-SWITCH-MIB::wrsBootSuccessful
WR-SWITCH-MIB::wrsOSStatusGroup
WR-SWITCH-MIB::wrsMainSystemStatus
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).

3.1.10 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
WR-SWITCH-MIB::wrsNetworkingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
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 issue 3.3.9.

3.1.11 *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>
WR-SWITCH-MIB::wrsBootSuccessful
WR-SWITCH-MIB::wrsOSStatus
WR-SWITCH-MIB::wrsMainSystemStatus

3.1.12 *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>
WR-SWITCH-MIB::wrsBootSuccessful

`WR-SWITCH-MIB::wrsOSStatus`
`WR-SWITCH-MIB::wrsMainSystemStatus`

3.1.13 Wrong configuration applied

Status: TODO
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: When a new configuration file is fetched on boot time, compare it with a previously used config (the whole file, but especially timing-critical fields like PTP/WR mode, fixed hardware delays). Report using the Syslog (*info*/*warning*) if the configuration has changed.

3.1.14 Switchover failed

Status: for later
Severity: ERROR
Mode: *Boundary Clock*, *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 *Boundary Clock* 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.

3.1.15 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)*

3.2 Data error

When the WR switch is not able to forward Ethernet traffic between devices connected to the ports, we consider this a data error. This section contains the list of faults leading to a data error.

3.2.1 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 a 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 SPFs used

However, we are not able to distinguish between them inside the switch.
SNMP objects:

```
IF-MIB::ifOperStatus.<n>
WR-SWITCH-MIB::wrsPortStatusLink.<n>
WR-SWITCH-MIB::wrsSlaveLinksStatus
WR-SWITCH-MIB::wrsTimingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

3.2.2 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::wrsPstatsHCTXUnderrun.<n>
WR-SWITCH-MIB::wrsPstatsHCRXOverrun.<n>
WR-SWITCH-MIB::wrsPstatsHCRXInvalidCode.<n>
WR-SWITCH-MIB::wrsPstatsHCRXSyncLost.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPfilterDropped.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPCSErrors.<n>
WR-SWITCH-MIB::wrsPstatsHCRXCRCErrors.<n>
WR-SWITCH-MIB::wrsEndpointStatus
WR-SWITCH-MIB::wrsNetworkingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

3.2.3 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 queued 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::wrsPstatsHCTXFrames.<n>
WR-SWITCH-MIB::wrsPstatsHCForwarded.<n>
WR-SWITCH-MIB::wrsSwcoreStatus
WR-SWITCH-MIB::wrsNetworkingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

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.

3.2.4 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::wrsPstatsHCRXDropRTUFull.<n>
WR-SWITCH-MIB::wrsRTUStatus
WR-SWITCH-MIB::wrsNetworkingStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

3.2.5 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::wrsPstatsHCFastMatchPriority.<n>
```

WR-SWITCH-MIB::wrsPstatsHCRXFrames.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPrio0.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPrio1.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPrio2.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPrio3.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPrio4.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPrio5.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPrio6.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPrio7.<n>
WR-SWITCH-MIB::wrsSwcoreStatus
WR-SWITCH-MIB::wrsNetworkingStatus
WR-SWITCH-MIB::wrsMainSystemStatus

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

3.2.6 *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>
WR-SWITCH-MIB::wrsBootSuccessful
WR-SWITCH-MIB::wrsOSStatus
WR-SWITCH-MIB::wrsMainSystemStatus

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

Status: TODO
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.

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

Status: TODO
Severity: WARNING
Description:
The same problem as described in the timing fault 3.1.9

3.2.9 Topology Redundancy failure

Status: for later
Severity: ERROR
Description: *(not yet implemented)*
Topology redundancy lets us prevent from losing data when the primary uplink is down for some reason. However, if a backup link is also down or if the 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.

3.3 Other errors

3.3.1 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
- status of execution of a custom boot script
- status of setting up auxclk on connector clk2
- status of setting up a limit on the Rx bandwidth of the traffic that goes from WR ports to Linux (throttling)
- status of setting up VLANs

SNMP objects:

```
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
WR-SWITCH-MIB::wrsFwUpdateStatus
WR-SWITCH-MIB::wrsCustomBootScriptSource
WR-SWITCH-MIB::wrsCustomBootScriptSourceUrl
WR-SWITCH-MIB::wrsCustomBootScriptStatus
WR-SWITCH-MIB::wrsAuxClkSetStatus
WR-SWITCH-MIB::wrsThrottlingSetStatus
WR-SWITCH-MIB::wrsVlansSetStatus
WR-SWITCH-MIB::wrsBootSuccessful
WR-SWITCH-MIB::wrsOSStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

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.

3.3.2 Dot-config error

Status: DONE
Severity: ERROR
Description:
A dot-config file used to configure the switch can be stored locally or retrieved from a central server. Additionally a URL to the remote dot-config can be retrieved via DHCP request. When the dot-config is fetched 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::wrsConfigSource`
`WR-SWITCH-MIB::wrsConfigSourceUrl`
`WR-SWITCH-MIB::wrsBootConfigStatus`
`WR-SWITCH-MIB::wrsBootSuccessful`
`WR-SWITCH-MIB::wrsOSStatus`
`WR-SWITCH-MIB::wrsMainSystemStatus`

3.3.3 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>`
`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`
`WR-SWITCH-MIB::wrsBootUserspaceDaemonsMissing`
`WR-SWITCH-MIB::wrsBootSuccessful`
`WR-SWITCH-MIB::wrsOSStatus`
`WR-SWITCH-MIB::wrsMainSystemStatus`
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. Dot-config should also let us define which processes are not that important and the switch should not restart even if such a process fails to start (e.g. *lighttpd*).
Crucial processes (Error report if any of them crashes):

- *PTP/PPSi*
- *wrsw_rtud* – 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. multicasts 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.

3.3.4 Kernel crash

Status: TODO (preserving stats of IP/LR registers)
Severity: ERROR
Description:
If the Linux kernel has crashed, the system reboots. Until the next boot we have no synchronization, no SNMP to report the status, and the 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
WR-SWITCH-MIB::wrsFaultLR
WR-SWITCH-MIB::wrsBootSuccessful
WR-SWITCH-MIB::wrsOSStatus
WR-SWITCH-MIB::wrsMainSystemStatus

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.

3.3.5 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
WR-SWITCH-MIB::wrsMemoryFree
WR-SWITCH-MIB::wrsMemoryFreeLow
WR-SWITCH-MIB::wrsOSStatus
WR-SWITCH-MIB::wrsMainSystemStatus

3.3.6 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
WR-SWITCH-MIB::wrsOSStatus
WR-SWITCH-MIB::wrsMainSystemStatus
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).

3.3.7 CPU load too high

Status: DONE
Severity: WARNING
Description:
On a healthy switch the average CPU load should be below *0.1* (10%). 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
WR-SWITCH-MIB::wrsOSStatus
WR-SWITCH-MIB::wrsMainSystemStatus
```

3.3.8 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:

- *IC19* – temperature below the FPGA
- *IC20*, *IC17* – temperature near the SCB power supply circuit
- *IC18* – temperature near the VCXO and PLLs (AD9516, CDCM6100)

wrsTemperatureWarning is raised when the temperature read from any of these sensors exceeds a threshold configured in the *dot-config* (80 degrees by default). When at least one threshold temperature is not set wrsTemperatureWarning is set to *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
```

WR-SWITCH-MIB::wrsOSStatus
WR-SWITCH-MIB::wrsMainSystemStatus

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

Status: DONE
Severity: WARNING
Description:
If a not supported Gigabit optical SFP (or an SFP that couldn't have been matched with the CONFIG_SFP<XX>_PARAMS entries in the configuration file) is plugged into the cage, then it's a timing issue 3.1.10. 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
WR-SWITCH-MIB::wrsNetworkingStatus
WR-SWITCH-MIB::wrsMainSystemStatus

3.3.10 IP address on the management port has changed

Status: TODO
Severity: WARNING
Description:
The change of an IP address on the management port might be a normal situation or a result of an accidental modification of a DHCP server or the WR Switch configuration. Notifying about such a situation is not done through SNMP, since the IP address of a switch has to be known to the SNMP manager prior querying the switch. Therefore, the switch only generates a Syslog warning message if setting a new IP address is detected.
SNMP objects: *(none)*, Syslog message is generated

3.3.11 Multiple unauthorized access attempts

Status: TODO
Severity: WARNING
Description:
Many attempts to gain a root access through the ssh (or the web interface), might mean that somebody tries to do something nasty. Every unsuccessful attempt to login is reported as a Syslog warning message.
SNMP objects: *(none)*, Syslog message is generated

3.3.12 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)*

3.3.13 Backup link down

Status: for later
Severity: WARNING
Description: *(not yet implemented)*
This is related to the issue 3.3.12. 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)*

3.4 Undetectable errors

Beside the various errors already listed in previous sections, there are some situations when reporting a problem to the SNMP manager or Syslog server is not possible. This section lists some of them and proposes alternative ways of diagnostics.

3.4.1 File system / Memory corruption

Description:
Memory or file system corruption can produce unpredictable results. It may cause a failure of any of the processes running on the switch.
SNMP objects: *(none)*

3.4.2 Kernel freeze

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

3.4.3 Power failure

Description:
Power failure may be either a WRS problem (i.e. broken power supply inside the switch) or an external voltage problem. 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)*

3.4.4 Hardware problem

Description:
If any crucial hardware part breaks, it will be most probably noticed as one (or multiple) timing / data errors described in the previous sections. Besides that, there is no self-diagnostics built-in on the switch hardware boards. A few examples of hardware failures and problems it may cause:

- DAC / VCO – problems with synchronization (failures in 3.1)
- cooling fans – rise of the temperature inside the WRS box (failure 3.3.8)
- power supply, ARM, FPGA – booting problem (failure 3.3.1)
- memory chip – data corruption (failure 3.4.1)

SNMP objects: *(none)*

3.4.5 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)*

3.4.6 No static IP on the management port & failed to DHCP

Description:
From the operator's point of view it is similar to the issue 3.4.5. 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 the configuration of the switch and management network should be verified.
SNMP objects: *(none)*

A Operator's diagnostic example

This section presents an example how a problem could be diagnosed and appropriate procedure applied by the operator of a WR Switch based on the general status objects described in section 2.1. The screenshots included in this example were made from *Diamon* tool used at CERN for diagnostics. Any other SNMP manager (like *Nagios* or *Icinga*) can be used to fetch the value of these status objects.

1. Operator gets an e-mail/sms alarm or notices in the SNMP manager that the status of a WR switch has changed to `Error` (fig.2)

2. By checking the general status objects (`glshyperlinkWR-SWITCH-MIB::wrsOSStatus`, `wrsTimingStatus`, `wrsNetworkingStatus`) one can realize that the problem is reported by the synchronization subsystem. The value of `wrsTimingStatus` is *2*= error (fig.3).

3. Following the tree structure of status objects from figure 1, if `wrsTimingStatus` reports an error, then status objects: `wrsPTPStatus`, `wrsSoftPLLStatus`, `wrsSlaveLinksStatus`, `wrsPTPFramesFlowing` should be checked. In this example `wrsSlaveLinksStatus` reports an error (fig.4).

4. The operator should search section 2.1 for procedure to follow when `wrsSlaveLinksStatus` reports an error.

5. In this example, the WR Switch that reports a problem works in the Boundary Clock mode, which means that the first step according to the procedure should be checking a fiber connection on the slave port.

6. Plugging the fiber to the slave port fixes the problem and WR Switch does not report more errors (fig.5).

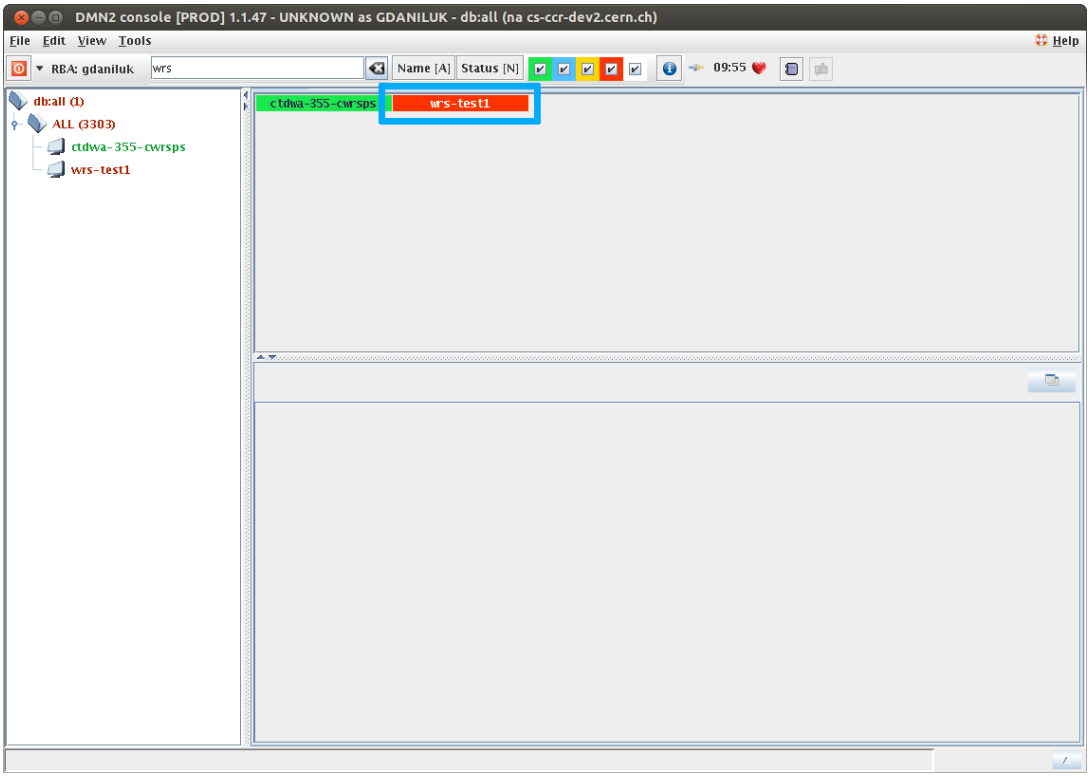

Figure 2: SNMP manager reports an error on a WR Switch

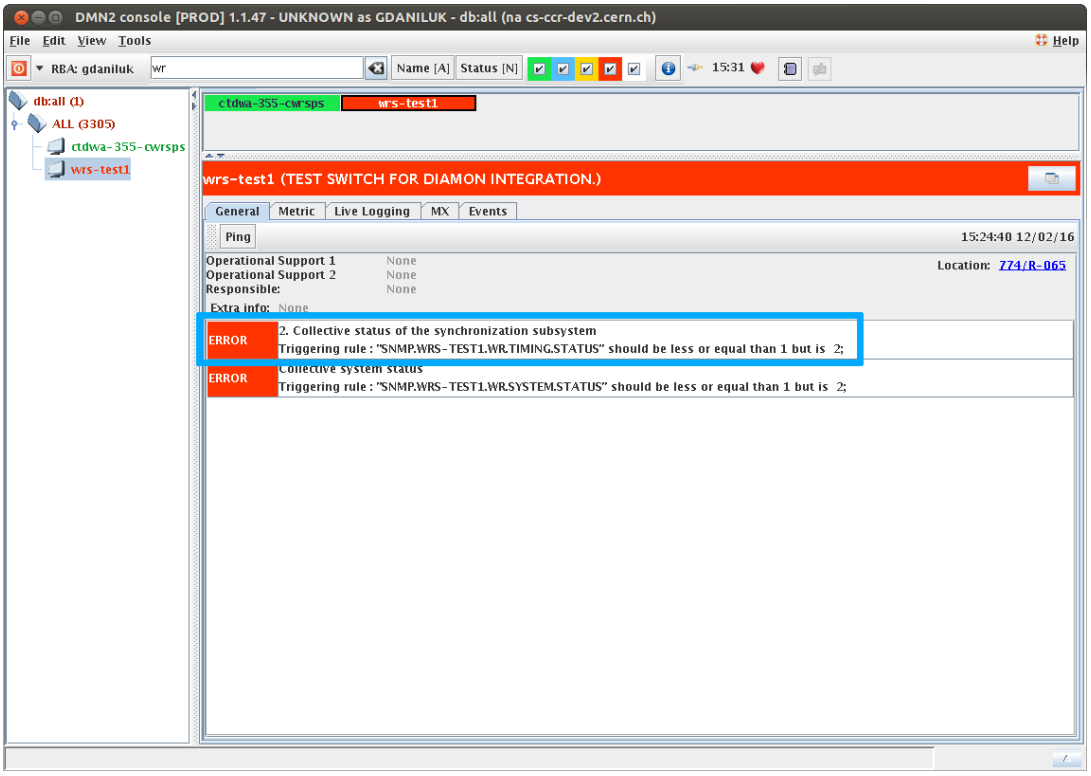

Figure 3: WR Switch has problem with the synchronization subsystem

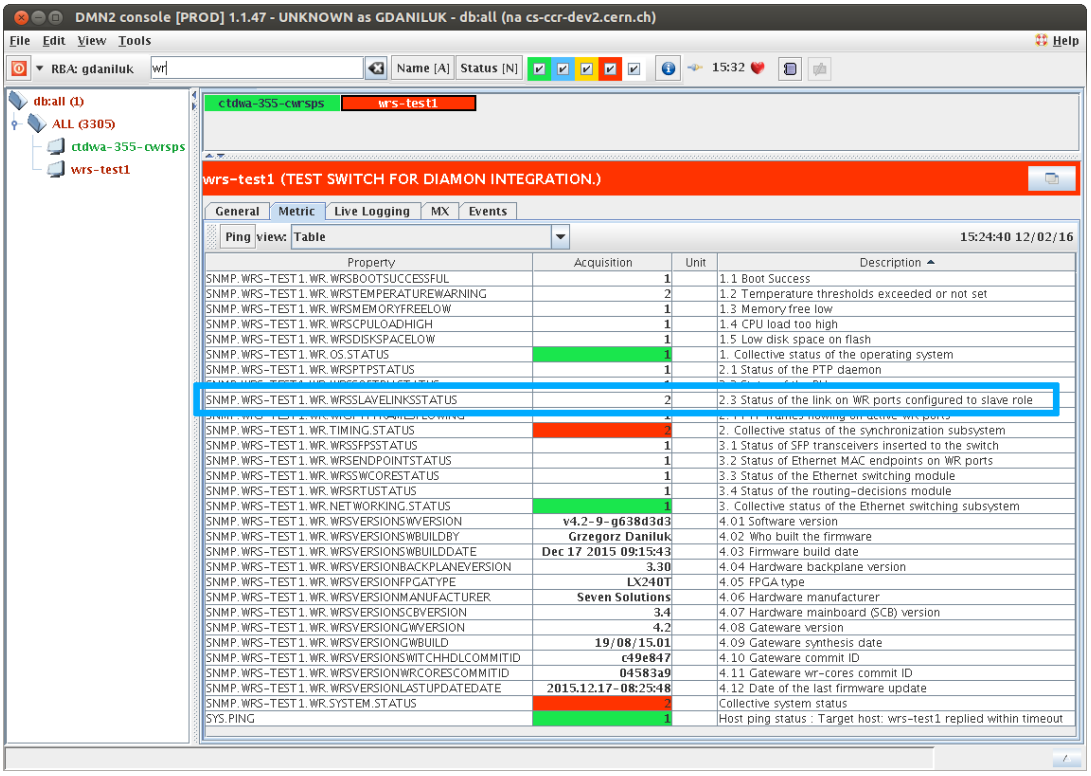

Figure 4: `wrsSlaveLinksStatus` object reports an error

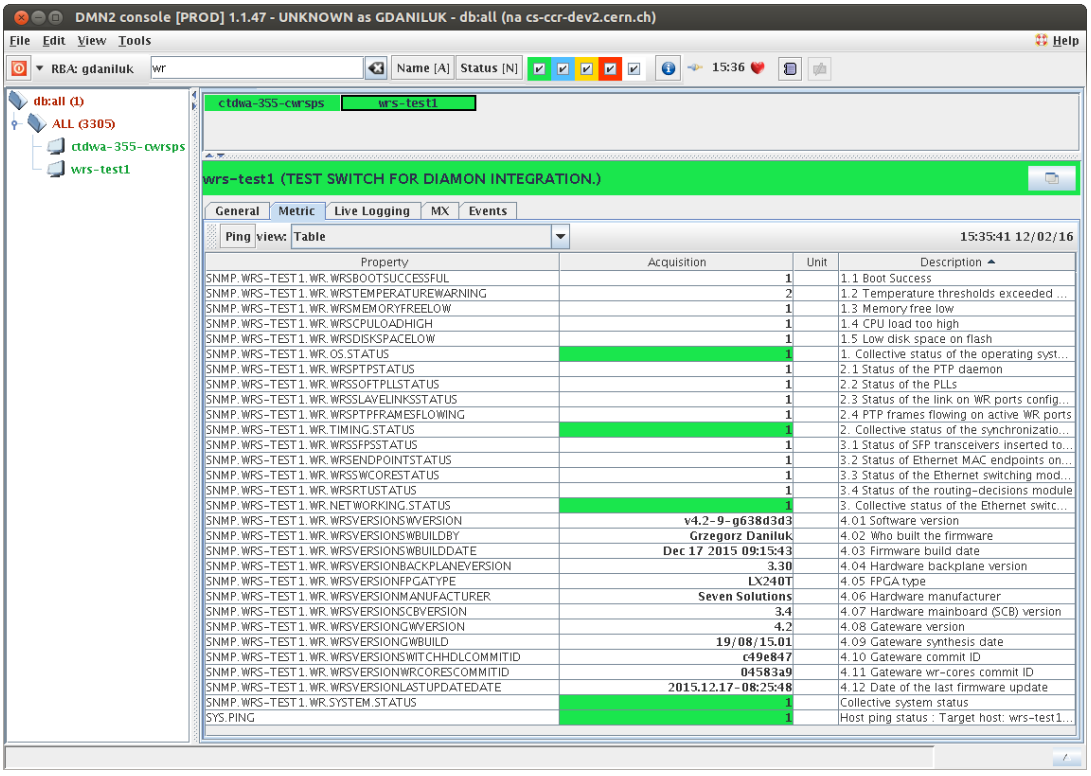

Figure 5: WR Switch does not report any errors

B Sorted list of all MIB objects

WR-SWITCH-MIB::wrsGeneralStatusGroup
WR-SWITCH-MIB::wrsGwWatchdogTimeouts 3.2.3,
WR-SWITCH-MIB::wrsMainSystemStatus 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.5, 3.1.7, 3.1.8, 3.1.9, 3.1.10, 3.1.11, 3.1.12, 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.5, 3.2.6, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.6, 3.3.7, 3.3.8, 3.3.9,
WR-SWITCH-MIB::wrsMemoryFree 3.3.5,
WR-SWITCH-MIB::wrsMemoryFreeLow 3.3.5,
WR-SWITCH-MIB::wrsMemoryGroup
WR-SWITCH-MIB::wrsMemoryTotal 3.3.5,
WR-SWITCH-MIB::wrsMemoryUsed 3.3.5,
WR-SWITCH-MIB::wrsMemoryUsedPerc 3.3.5,
WR-SWITCH-MIB::wrsNetworkingStatus 3.1.10, 3.2.2, 3.2.3, 3.2.4, 3.2.5, 3.3.9,
WR-SWITCH-MIB::wrsNetworkingStatusGroup
WR-SWITCH-MIB::wrsOperationStatus
WR-SWITCH-MIB::wrsOSStatus 3.1.11, 3.1.12, 3.2.6, 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.6, 3.3.7, 3.3.8,
WR-SWITCH-MIB::wrsOSStatusGroup 3.1.9,
WR-SWITCH-MIB::wrsPortStatusConfiguredMode.<n> 3.1.7, 3.1.8, 3.1.9, 3.1.10,
WR-SWITCH-MIB::wrsPortStatusIndex.<n>
WR-SWITCH-MIB::wrsPortStatusLink.<n> 3.1.7, 3.1.8, 3.1.9, 3.2.1,
WR-SWITCH-MIB::wrsPortStatusLocked.<n>
WR-SWITCH-MIB::wrsPortStatusPeer.<n>
WR-SWITCH-MIB::wrsPortStatusPortName.<n>
WR-SWITCH-MIB::wrsPortStatusPtpRxFrames.<n> 3.1.9,
WR-SWITCH-MIB::wrsPortStatusPtpTxFrames.<n> 3.1.9,
WR-SWITCH-MIB::wrsPortStatusSfpError.<n> 3.1.10, 3.3.9,
WR-SWITCH-MIB::wrsPortStatusSfpGbE.<n> 3.1.10, 3.3.9,
WR-SWITCH-MIB::wrsPortStatusSfpInDB.<n> 3.1.10,
WR-SWITCH-MIB::wrsPortStatusSfpPN.<n> 3.1.10, 3.3.9,
WR-SWITCH-MIB::wrsPortStatusSfpVN.<n> 3.1.10, 3.3.9,
WR-SWITCH-MIB::wrsPortStatusSfpVS.<n> 3.1.10, 3.3.9,
WR-SWITCH-MIB::wrsPortStatusTable
WR-SWITCH-MIB::wrsPstatsHCFastMatchFastForward.<n>
WR-SWITCH-MIB::wrsPstatsHCFastMatchNonForward.<n>
WR-SWITCH-MIB::wrsPstatsHCFastMatchPriority.<n> 3.2.5,
WR-SWITCH-MIB::wrsPstatsHCFastMatchRespValid.<n>
WR-SWITCH-MIB::wrsPstatsHCForwarded.<n> 3.2.3,
WR-SWITCH-MIB::wrsPstatsHCFullMatchRespValid.<n>
WR-SWITCH-MIB::wrsPstatsHCIndex.<n>
WR-SWITCH-MIB::wrsPstatsHCPortName.<n>
WR-SWITCH-MIB::wrsPstatsHCRTUDropped.<n>
WR-SWITCH-MIB::wrsPstatsHCRTUResponses.<n>
WR-SWITCH-MIB::wrsPstatsHCRTUValid.<n>
WR-SWITCH-MIB::wrsPstatsHCRXCRCErrors.<n> 3.2.2,
WR-SWITCH-MIB::wrsPstatsHCRXDropRTUFull.<n> 3.2.4,
WR-SWITCH-MIB::wrsPstatsHCRXFrames.<n> 3.2.5,

WR-SWITCH-MIB::wrsPstatsHCRXGiantFrames.<n>
WR-SWITCH-MIB::wrsPstatsHCRXInvalidCode.<n> 3.2.2,
WR-SWITCH-MIB::wrsPstatsHCRXOverrun.<n> 3.2.2,
WR-SWITCH-MIB::wrsPstatsHCRXPauseFrames.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPclass0.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPclass1.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPclass2.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPclass3.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPclass4.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPclass5.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPclass6.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPclass7.<n>
WR-SWITCH-MIB::wrsPstatsHCRXPCSErrors.<n> 3.2.2,
WR-SWITCH-MIB::wrsPstatsHCRXPfilterDropped.<n> 3.2.2,
WR-SWITCH-MIB::wrsPstatsHCRXPrio0.<n> 3.2.5,
WR-SWITCH-MIB::wrsPstatsHCRXPrio1.<n> 3.2.5,
WR-SWITCH-MIB::wrsPstatsHCRXPrio2.<n> 3.2.5,
WR-SWITCH-MIB::wrsPstatsHCRXPrio3.<n> 3.2.5,
WR-SWITCH-MIB::wrsPstatsHCRXPrio4.<n> 3.2.5,
WR-SWITCH-MIB::wrsPstatsHCRXPrio5.<n> 3.2.5,
WR-SWITCH-MIB::wrsPstatsHCRXPrio6.<n> 3.2.5,
WR-SWITCH-MIB::wrsPstatsHCRXPrio7.<n> 3.2.5,
WR-SWITCH-MIB::wrsPstatsHCRXRuntFrames.<n>
WR-SWITCH-MIB::wrsPstatsHCRXSyncLost.<n> 3.2.2,
WR-SWITCH-MIB::wrsPstatsHCTable
WR-SWITCH-MIB::wrsPstatsHCTRURespValid.<n>
WR-SWITCH-MIB::wrsPstatsHCTXFrames.<n> 3.2.3,
WR-SWITCH-MIB::wrsPstatsHCTXUnderrun.<n> 3.2.2,
WR-SWITCH-MIB::wrsPtpClockOffsetErrCnt.<n> 3.1.2,
WR-SWITCH-MIB::wrsPtpClockOffsetPs.<n> 3.1.2,
WR-SWITCH-MIB::wrsPtpClockOffsetPsHR.<n> 3.1.2,
WR-SWITCH-MIB::wrsPtpDataTable
WR-SWITCH-MIB::wrsPtpDeltaRxM.<n> 3.1.4,
WR-SWITCH-MIB::wrsPtpDeltaRxS.<n> 3.1.4,
WR-SWITCH-MIB::wrsPtpDeltaTxM.<n> 3.1.4,
WR-SWITCH-MIB::wrsPtpDeltaTxS.<n> 3.1.4,
WR-SWITCH-MIB::wrsPTPFramesFlowing 3.1.9,
WR-SWITCH-MIB::wrsPtpGrandmasterID.<n>
WR-SWITCH-MIB::wrsPtpIndex.<n>
WR-SWITCH-MIB::wrsPtpLinkLength.<n>
WR-SWITCH-MIB::wrsPtpMode.<n>
WR-SWITCH-MIB::wrsPtpOwnID.<n>
WR-SWITCH-MIB::wrsPtpPhaseTracking.<n>
WR-SWITCH-MIB::wrsPtpPortName.<n>
WR-SWITCH-MIB::wrsPtpRTT.<n> 3.1.3,
WR-SWITCH-MIB::wrsPtpRTTErrCnt.<n> 3.1.3,

WR-SWITCH-MIB::wrsPtpServoState.<n> 3.1.1,
WR-SWITCH-MIB::wrsPtpServoStateErrCnt.<n> 3.1.1,
WR-SWITCH-MIB::wrsPtpServoStateN.<n> 3.1.1,
WR-SWITCH-MIB::wrsPtpServoUpdates.<n>
WR-SWITCH-MIB::wrsPtpServoUpdateTime.<n>
WR-SWITCH-MIB::wrsPtpSkew.<n>
WR-SWITCH-MIB::wrsPTPStatus 3.1.1, 3.1.2, 3.1.3, 3.1.4,
WR-SWITCH-MIB::wrsPtpSyncSource.<n>
WR-SWITCH-MIB::wrsRebootCnt 3.3.4,
WR-SWITCH-MIB::wrsRestartReason 3.3.1, 3.3.4,
WR-SWITCH-MIB::wrsRestartReasonMonit 3.3.1,
WR-SWITCH-MIB::wrsRTUStatus 3.2.4,
WR-SWITCH-MIB::wrsSFPsStatus 3.1.10, 3.3.9,
WR-SWITCH-MIB::wrsSlaveLinksStatus 3.1.7, 3.1.8, 3.2.1,
WR-SWITCH-MIB::wrsSoftPLLStatus 3.1.5,
WR-SWITCH-MIB::wrsSpllAlignState 3.1.5,
WR-SWITCH-MIB::wrsSpllBuildBy
WR-SWITCH-MIB::wrsSpllBuildDate
WR-SWITCH-MIB::wrsSpllDelCnt 3.1.5,
WR-SWITCH-MIB::wrsSpllHlock 3.1.5,
WR-SWITCH-MIB::wrsSpllHY
WR-SWITCH-MIB::wrsSpllIrqCnt 3.1.6,
WR-SWITCH-MIB::wrsSpllMlock 3.1.5,
WR-SWITCH-MIB::wrsSpllMode 3.1.5,
WR-SWITCH-MIB::wrsSpllMY
WR-SWITCH-MIB::wrsSpllSeqState 3.1.5,
WR-SWITCH-MIB::wrsSpllState
WR-SWITCH-MIB::wrsSpllStatusGroup
WR-SWITCH-MIB::wrsSpllVersion
WR-SWITCH-MIB::wrsSpllVersionGroup
WR-SWITCH-MIB::wrsStartCntGroup
WR-SWITCH-MIB::wrsStartCntHAL 3.1.12, 3.3.3,
WR-SWITCH-MIB::wrsStartCntHttpd 3.3.3,
WR-SWITCH-MIB::wrsStartCntPTP 3.1.11, 3.3.3,
WR-SWITCH-MIB::wrsStartCntRTUd 3.2.6, 3.3.3,
WR-SWITCH-MIB::wrsStartCntSnmpd 3.3.3,
WR-SWITCH-MIB::wrsStartCntSPLL 3.1.6, 3.3.3,
WR-SWITCH-MIB::wrsStartCntSshd 3.3.3,
WR-SWITCH-MIB::wrsStartCntSyslogd 3.3.3,
WR-SWITCH-MIB::wrsStartCntWrsWatchdog 3.3.3,
WR-SWITCH-MIB::wrsSwcoreStatus 3.2.3, 3.2.5,
WR-SWITCH-MIB::wrsTemperatureGroup
WR-SWITCH-MIB::wrsTemperatureWarning 3.3.8,
WR-SWITCH-MIB::wrsTempFPGA 3.3.8,
WR-SWITCH-MIB::wrsTempPLL 3.3.8,
WR-SWITCH-MIB::wrsTempPSL 3.3.8,