

Technical Support DATA (EMEA)

**System Functional and Design Specification**

**Preventive Maintenance**

Author(s): Raphael Voyer, Jonathan Trebaol

Revision: 0.1

Date: 10/06/2021

**Copyright © 1995-2016 Alcatel-Lucent, Incorporated**

**ALL RIGHTS RESERVED WORLDWIDE**

**Alcatel-Lucent DSD**

**26801 West Agoura Road, Calabasas, CA 91301**

**(818) 880-3500**

**TABLE OF CONTENTS**

[1 INTRODUCTION 6](#_Toc76555045)

[1.1 Purpose 6](#_Toc76555046)

[1.2 Scope 6](#_Toc76555047)

[1.3 Intended Audience 6](#_Toc76555048)

[2 FUNCTIONAL DESCRIPTION 7](#_Toc76555049)

[2.1 Basic Overview 7](#_Toc76555050)

[2.2 Platform Supported 7](#_Toc76555051)

[2.3 Design Constraints. 8](#_Toc76555052)

[2.3.1 Software Limitations 8](#_Toc76555053)

[2.3.2 Hardware Limitations 8](#_Toc76555054)

[2.4 Assumptions and Dependencies 8](#_Toc76555059)

[3 SYSTEM REQUIREMENT SPECIFICATIONS 9](#_Toc76555061)

[3.1 Configuration Requirements 9](#_Toc76555062)

[3.1.1 Python Requirements 9](#_Toc76555063)

[3.1.2 TFTP Requirements 9](#_Toc76555071)

[4 DEBIAN SERVER INITIALIZATION 9](#_Toc76555072)

[4.1 Introduction 9](#_Toc76555073)

[4.2 Flowchart 9](#_Toc76555074)

[4.3 Services 9](#_Toc76555075)

[4.3.1 TFTP 9](#_Toc76555076)

[4.3.2 Rsyslog 9](#_Toc76555077)

[4.3.3 Logrotate 10](#_Toc76555078)

[4.3.4 Iptables 10](#_Toc76555079)

[5 UTILS PYTHON SCRIPTS 11](#_Toc76555080)

[5.1 Introduction 11](#_Toc76555081)

[5.2 Support Tools Script 11](#_Toc76555082)

[5.3 Active Output Socket 11](#_Toc76555083)

[5.4 Send Notification 11](#_Toc76555084)

[5.5 Requests Handler 11](#_Toc76555085)

[5.6 Web Receiver 12](#_Toc76555086)

[6 USE CASE PYTHON SCRIPTS 12](#_Toc76555087)

[6.1 L2 Loop Script 12](#_Toc76555088)

[6.2 Port Flapping Script 12](#_Toc76555089)

[6.3 Port Scanning (DDOS) Script 12](#_Toc76555090)

[6.4 Collection Logs (Switch) Script 12](#_Toc76555091)

[6.5 Collection Logs (Stallar AP) Script 12](#_Toc76555092)

**REVISION HISTORY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Revision | Date | Agile Revision | Update By | Revision Description | Status |
| 0.1 | 16/06/2021 |  | Raphael Voyer | Initial document, | Draft |

# INTRODUCTION

## Purpose

The document describes the Software Requirements Specification and Functional Specification of Preventive Maintenance for AOS 8.x .

## Scope

The scope of this document is to present the complete set of requirements and functional specifications for Preventive Maintenance.

## Intended Audience

This document is intended for the following:

* Technical support data Team
* Engineering Design Team
* Product Testing Team
* Technical Writing Team responsible for developing the user documentation
* Support organization providing end-user support.







# FUNCTIONAL DESCRIPTION

## Basic Overview

Preventive maintenance allows the automation of troubleshooting on OmniSwitch AOS 8.x, Stellar APs equipment, and a notification function either by email or by Rainbow can be configured. It consists of a Debian server that will receive logs of the various devices on the network. All logs will be filtered by Rsyslog. Rsyslog will both detect one or more keywords within the logs, and then store them and run a script based on the received log. All scripts for resolutions are developed in Python

Here is the example of the process of a case:

1. reception of a log containing "Buffer list is empty"

2. Rsyslog runs the Log Debug script debug2

3. Put switches to debug mode

4. Receive a log containing 'slnhwlrncbkhandler', "port" and "bcmd'

5. Run the L2 Loop script

6 The script detects a loop by parsing the logs

7. The script sends a beech request to the administrator via Rainbow and/or email

8. The administrator answers yes to the request

9. The script sends an ssh command to fix problem

10. The script disables the switch debug mode

To be able to use the Preventive Maintenance, the folder must contain the following files:  
• **Setup.sh**: Allows configuration of the Debian server.  
• **Devices.csv**: Contains the IP addresses of the switches to configure active output socket.  
• **support\_active\_output\_socket.py**: Configure out socket on switches contained in Device.csv.  
• **support\_tools.py**: Allows all other support scripts to work.  
• **support\_send\_notification.py**: allows the sending of different notifications  
• **support\_response\_handler.py**: Orchestrates the sending and receiving of requests by notifications  
• **support\_web\_receiver\_class.py**: Allows the configuration of the web server on the fly to receive replies sent by the user.

For the first release of Preventive Maintenance, in order to fully configure the Debian server, bash script under the name of Setup.sh has been developed. This allows you to configure the following:

• Configuration of notifications (Rainbow/Email)

• Pattern for collecting switchs logs

• Credentials of Switchs

• Credentials of Stellar access points

• Pattern for collecting AP logs

• Subnets authorised to send logs

• Installation of Python3 and outbuildings

• Rsyslog configuration

• Logrotate configuration

• Configuration of iptables (not yet implemented)

• TFTP Server Configuration

• Active socket output on switches

• Create /opt/ALE\_Script directory

## Platform Supported

The proposed Preventive Maintenance functionality shall be supported on Omniswitch AOS 8.X

## Design Constraints.

### Software Limitations

* Unknown

### Hardware Limitations

* Unknown

## Assumptions and Dependencies

* None

# SYSTEM REQUIREMENT SPECIFICATIONS

This chapter captures the requirements for Preventive Maintenance.

## Configuration Requirements

### Python Requirements

Python Version : Python 3.5.3 or higher

### TFTP Requirements

# DEBIAN SERVER INITIALIZATION

## Introduction

The purpose of the Setup.sh script is to initialize all the services that will be useful for the proper operation of the Preventive Maintenance functionality. To run Preventive Maintenance requires service that are not present on a Ubuntu server at installation. It is also necessary to be able to configure these different services. Security is also added to prevent bad entries, checking the form of the mail or an IP address for example.

## Services

All services described below are fully installed and configured by the Setup.sh script. It is not necessary to modify the configuration files of these various services under penalty of malfunctions.

### TFTP

A TFTP server is installed on the Ubuntu server, the default directory is set to /tftpboot. It is accessible by SFTP on port 22. Its role is to receive the logs of switchs (show tech-support eng complete) or AP Stellar(take\_snapshot.sh start).

### Rsyslog

Rsyslog is the service that will allow you to retrieve logs from different equipment. It is configured to detect the keywords that the devices are likely to send according to the different cases. Once a keyword is detected, the rsyslog will store the different logs received in the path/var/log/devices/ with the name of this form: ip\_address\_date\_history.json

Example: 10.130.7.248\_2021-06-24\_history.json

It will also create another temporary file depending on the case, in order to be able to handle more easily the latest logs received:

• lastlog.json

• lastlog\_ddos.json

• lastlog\_ddos\_ip.json

• lastlog\_flapping.json

• lastlog\_loop.json

These various files will be described in the relevant parts of the report.

Rsyslog will also allow depending on the keywords detected in the logs to run the corresponding python scripts.

## Flowchart

Diagram

Description automatically generated

### Logrotate

Logrotate limits the size of log files in/var/log.

For each log file, logrotate performs 2 simultaneous operations:

• Rotation: It archives the log file under a different name and deletes the oldest archive

• compression: it may compress the log file before archiving it

### Iptables

Not yet implemented

# UTILS PYTHON SCRIPTS

## Introduction

The use of utils scripts allows multiple scripts having the same action to do not multiply lines of code. This allows you to reuse the same code on different scripts. (send notification ,support tools,)

It also makes it easier to find the code by splitting it and classifying it into several files. (requests handler, support tools)

## Support Tools Script

Name : support\_tools.py

Rather all the tools allowing the automation of network support tasks.

There are the functions that allow you to perform actions on the switches.

Il y a aussi toutes les fonctions permettant la détection des incidents.

Ce script permet la factorisation du code afin de réutiliser les fonction s qui peuvent être identique sur les différents scripts.

## Active Output Socket

Name : support\_active\_output\_socket.py

This script is executed only once when Setup.sh.

It allows the configuration of «swlog output socket ip\_address\_debian\_server», this command allows to send the logs of the Alcatel switch to the debian server for use by Rsyslog.

## Send Notification

Name : support\_send\_notification.py

Rather all functions to send email or rainbow message.

Email functions can be a request to act or not on the issue. Feature handle attachments files.

Rainbow function Rainbow functions can be a request to act or not on the issue. Feature handle text attachments files.

## Web Receiver

Name : support\_web\_receiver\_class.py

Permit the setup of the web server, with goods keys and values in url parameters. For example : when the web server is started , there is only 3 pages available(yes, no, and save). To ensure security, the other parameters to access at the good page is the client id and case id. When the server receive an answer by access on the url , the webserver stop. If there is no access on the web server after 60 seconds the answer will be automatically yes.

## Requests Handler

Name: support\_response\_handler.py

Conductor of the request feature. To work the script needs the save\_decision.conf file, which will register the cases, that the administrator already fix and wanted to save or the case that the administrator don’t want to fix.

The save\_decision.conf structure file :

10.130.7.247,1/1/33,flapping,never

192.168.80.27,0,ddos,always

192.168.80.27,1/1/19,flapping,always

192.168.80.25,1/1/19,flapping,always

192.168.80.27,1/1/4,loop,always

192.168.80.27,1/1/35,loop,never

# USE CASE PYTHON SCRIPTS

The keywords in the logs received from Alcatel equipment alone are not enough to detect whether this is a real network issue.

For this purpose, functions allowing the analysis of the latest logs with the same received keyword is done (on the frequency of received messages, the quantity of received messages and thanks to Timestamps).

The scripts also allow the sending of commands to the equipment, which will allow the resolution of incidents.

## Log Debug debug2

Name : support\_switch\_debugging.py

Rsyslog Trigger: ‘Buffer list is empty’

Last log file : lastlog.json

Log example:

*{"@timestamp":"2021-06-07T10:03:49+02:00","type":"syslog\_json","relayip":"10.130.7.251","hostname":"os6860e-core1","message":"<131>Jun 7 10:03*:*49 OS6860E-Core1 swlogd bcmd rpcs ERR: rpcUtilBufListGet:155* *Buffer list is empty","end\_msg":""}*

The purpose of this script is to enable debugging. Indeed, if the debug mode is set on ‘info’. All log received on the server log will be Buffer is not empty. To get the log with more detail on our server log we need to enter the switch in debug2 mode.

To do that we use the enable\_debugging function which uses sshpass to enter the command on the switch.

When the script is executed, a ssh command “*swlog appid bcmd subapp 3 level debug2*” is send to the device which sent the log.

This commends permit the reception of more details logs, then the rsyslog can detect which case are responsible of these logs.

After the switch is in debug mode, the script clears the log file lastlog.json.

## Log Debug debug3

Name : support\_switch\_debugging\_ddos.py

Rsyslog Trigger: ‘Denial of Service attack detected: <port-scan>’

Last log file : lastlog\_ddos.json

Log example:

When the script is executed, a ssh command “*swlog appid ipv4 subapp all level debug3*” is send to the device which sent the log.

This commends permit the reception of more details logs, then the rsyslog can detect which case are responsible of these logs.

## L2 Loop Script

Name: support\_switch\_port\_disable.py

Rsyslog Trigger: ‘slnHwlrnCbkHandler' ;'port';'bcmd'

Last log file : lastlog\_loop.json

Log example :

*{"@timestamp":"2021-05-26T11:41:25+02:00","type":"syslog\_json","relayip":"192.168.80.27","hostname":"os6860e-2-gartner","message":"<135>May 26 11:41:25 OS6860E-2-Gartner swlogd bcmd rpcs DBG2: slnHwlrnCbkHandler:648 port 19 mod 0 auth 0 group 0","end\_msg":""}*

The purpose of this script is to detect a L2 loop in a network by analyzed the log file lastlog\_loop.json, shut down the port which responsible of the loop and return back the switch in debug mode info.

First step, the script gets all informations to connect to the switch send rainbow message or email.

Next, the script extracts the switch IP and the port responsible of the loop which have been sent in the logs.

Script Steps:

**Condition one :** no loop has been detected since 10 seconds.

To do that, the script uses check\_timestamp() function. In this function, we use 2 files, lastlog\_loop and a new file logtemp. In this new file, we will register the lastlog that we use to stop the previous loop.

* If logtemp doesn’t exist or there is not only 1 line in logtemp . We copy the first lastlog\_loop log in it.
* We extract the timestamps from the first line of lastlog\_loop and the timestamp from logtemp.
* We return the difference of the two timestamps then we replace the log in logtemp by the new log .

**Condition two :** 10 loop logs received in less than 2 seconds.

* Check if there is more than 10 lines in lastlog\_loop. Else we return 0
* Extract the timestamps of the first line and the 10th line of lastlog\_loop
* Change time to decimal to get a continuous number ( else, there is an error due to second changes 60 to 0)
* If there is less than 2 seconds between the two timestamps, we return 1 else we return 0
* the script send a request by Mail and by Rainbow to fix or not the issue:
  + If the answer is yes, the script send a ssh packet to disable, and enable the port(s) sent in the logs.
  + If the answer Is yes and save my decision sends a ssh packet to disable the port(s) sent in the logs. Then save the answer in the decisions\_save.conf file
  + If the answer is no, the script does nothing, and save the decision in the decisions\_save.conf file.
* Send the log file and a message by the rainbow bot (the bot will be explained in a other part).
* Send the log in attachment and a email to the email address in ALE\_script.conf
* Then we disable the debugging on the switch.
* If the condition one is not respected the script clear the file last log to get in first position a most recent logs.
* If the condition 2 is not respected the script does nothing.

Diagram

Description automatically generated

## Port Flapping Script

Name : support\_switch\_port\_flapping.py

RsyslogTrigger: 'pmnHALLinkStatusCallback:206'

Last log file: lastlog\_flapping.json

Log example:

*{"@timestamp":"2021-07-08T10:46:33+02:00","type":"syslog\_json","relayip":"192.168.80.27","hostname":"os6860e-2-gartner","message":"<134>Jul* *8 10:46:33 OS6860E-2-Gartner swlogd portMgrNi main INFO: : [pmnHALLinkStatusCallback:206] LINKSTS 1\/1\/3 DOWN (gport 0x2) Speed 0 Duplex HALF","end\_msg":""}*

The purpose of this script is to detect a port flapping in a network by analyzed the log file lastlog-flapping.json, restart the port which responsible of the flapping and return back the switch in debug mode info.

Script steps:

* Get all informations to connect to the switch send rainbow message or email.
* Extract the switches IP and the ports responsible of the flapping which have been sent in the logs. For that we use the function detect\_port\_flapping(). This function will do the most part of Port Flapping:
* Check if there is more than 30 lines in lastlog\_flapping, if there is the case we delete the log file. And we return NULL IP Addresses and NULL port numbers
* For each line in the log file we do:
  + We extract the timestamp and the IP address
  + If there is no First\_IP we put the IP address in the First\_IP, and we put the timestamp in memory for the log of the first ip (last\_time\_first)
  + If there is no Second\_IP and the current IP is not the first IP address, we put the IP address in the Second\_IP and we put the timestamp in memory for the log of the second ip (last\_time\_second)
  + If there is a third ip address, the script clears the log file.
  + To get the port associate to one of the IP Address, the script check if there is LINKSTS in the log, the port number will be the next element in the log.
  + If there is the first port we copy the port in the variable first\_port, if the first port is already set et the second port is empty we put the port in second\_port.
  + We check if there is DOWN in the log , we don’t take UP log in consideration to don’t make the accounting two times.
  + Now we check if the current ip in the log equals the first or the second IP.
  + Then we calculate the time difference between the current log and the previous log for this IP. If there is less than 10 seconds( for the demo , we can down to 1), the script increments a counter associate to the current IP.
  + When one of the two counters reaches at 5, the script returns IP addresses and numbers ports. If there is only one IP address the other ip address return is set to 0 and the port to 1/1/0
  + if counters are less than 5 the script returns NULL IP addresses and NULL port numbers

The main script port\_flapping:

* If both port numbers not equal to 0 , the script continue else the script stops.
* There are now 3 different scenarios: 2 IP addresses, 1st IP address or 2nd IP address, even if the third case couldn’t be reach.
* the script send a request by Mail and by Rainbow to fix or not the issue:
  + If the answer is yes, the script send a ssh packet to disable ,and enable the port(s) sent in the logs.
  + If the answer Is yes and save my decision send a ssh packet to disable the port(s) sent in the logs. Then save the answer in the decisions\_save.conf file
  + If the answer is no, the script does nothing, and save the decision in the decisions\_save.conf file.
* Send the log file and a message by the rainbow bot (the bot will be explained in a other part).
* Send the log in attachment and a email to the email address in ALE\_script.conf
* Then we disable the debugging on the switch(es).

Diagram

Description automatically generated

## Port Scanning (DDOS) Script

Name: support\_switch\_enable\_qos.py

Rsyslog Trigger: 'ALV4 event: PSCAN'

Last log file : lastlog\_ddos\_ip.json

Log example:

*{"@timestamp":"2021-06-04T00:09:18+02:00","type":"syslog\_json","relayip":"10.130.7.251","hostname":"os6860e-core1","message":"<135>Jun 4 00:09:18 OS6860E-Core1 swlogd ipv4 alv4ni DBG1: ALV4 event: PSCAN vrf 0 8.8.4.4","end\_msg":""}*

*The purpose of this script is to detect a port scanning* in a network by analyzed the log file lastlog\_ddos\_ip.json.json, apply a QOS policy to block data access on the IP address responsible of the scan and return back the switch in debug info mode.

Script steps:

* Get all information to connect to the switch send rainbow message or email.
* Extract the switch IP responsible of the scan which have been sent in the logs.
* Extract the attacker IP address with the function extract\_ip\_ddos():
* Check if there is less than 10 seconds between the two last logs
  + If there is less than 10 seconds , the lastlog\_ddos\_ip.json is clearing.
  + If there is less than 2 logs, we do nothing
* If the condition is met, we return the Attacker IP address.
* the script send a request by Mail and by Rainbow to fix or not the issue:
  + If the answer is yes, the script send a ssh packet to disable ,and enable the port(s) sent in the logs.
  + If the answer Is yes and save my decision send a ssh packet to disable the port(s) sent in the logs. Then save the answer in the decisions\_save.conf file
  + If the answer is no, the script does nothing, and save the decision in the decisions\_save.conf file.
* Send the log file and a message by the rainbow bot (the bot will be explained in a other part).
* Send the log in attachment and a email to the email address in ALE\_script.conf
* Then we disable the debugging on the switch(es).

Diagram

Description automatically generated

## Collection Logs (Switch) Script

Name: support\_switch\_get\_log.py

Rsyslog Trigger: Patterns chosen by user during setup.sh

## Collection Logs (Stallar AP) Script

Name : support\_AP\_get\_log.py

Rsyslog Trigger: Patterns chosen by user during setup.sh







































# Improvements:

OS6