

Technical Support DATA (EMEA)

**System Functional and Design Specification**

**Preventive Maintenance**

Author(s): Raphael Voyer, Jonathan Treboal

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)

**LIST OF FIGURES**

[Figure 1:Typical Cluster IP address 13](#_Toc436661300)

[Figure 2: Typical Cluster Deployment 14](#_Toc436661301)

[Figure 3: Switch connected to an L2 cluster through 3 ports (1/3,1/4,1/5) 40](#_Toc436661302)

[Figure 4: Switch connected to an L3 cluster through 3 ports (1/3,1/4,1/5) 41](#_Toc436661303)

[Figure 5: Switch connected to an L3 cluster(IGMP) through 3 ports (1/3,1/4,1/5) on VRF 43](#_Toc436661304)

[Figure 6: Switch connected to an L3 cluster(IGMP) through 3 ports (1/3,1/4,1/5) 44](#_Toc436661305)

[Figure 7: HAVLAN CMM in AOS 46](#_Toc436661306)

[Figure 8: Boot Sequence 54](#_Toc436661307)

[Figure 9: NI Down 54](#_Toc436661308)

[Figure 10: CMM Takeover 55](#_Toc436661309)

[Figure 11: Cluster creation 56](#_Toc436661310)

[Figure 12: L2 Cluster 57](#_Toc436661311)

[Figure 13: L3 Cluster 59](#_Toc436661312)

[Figure 14: L3 cluster(IGMP) 61](#_Toc436661313)

[Figure 15 Cluster Modification 62](#_Toc436661314)

[Figure 16: Cluster Deletion 63](#_Toc436661315)

[Figure 17: Cluster Disable 65](#_Toc436661316)

[Figure 18: Cluster Enable 66](#_Toc436661317)

[Figure 19: PM Interface 67](#_Toc436661318)

[Figure 20: VM Interface 67](#_Toc436661319)

[Figure 21: IP CMM interface. 68](#_Toc436661320)

[Figure 22 : Interface with IPMS CMM 69](#_Toc436661321)

VRF

**LIST OF TABLES**

[Table 1: Document Organization 10](#_Toc270435668)

[Table 2: CLI Syntax Conventions 30](#_Toc270435669)

[Table 3: System Flows 77](#_Toc270435670)

[Table 4: CMM Critical Computing Resources 122](#_Toc270435671)

**REVISION HISTORY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Revision | Date | Agile Revision | Update By | Revision Description | Status |
| 0.1 | 16/06/2021 |  | Raphael Voyer | Initial document, capturing the HA Vlan functional specifications and design. | 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 permet l’automatisation des résolutions de problèmes sur les équipements OmniSwitch AOS 8.x, Stellar APs, de plus une fonction de notification soit par email soit par Rainbow peut être configurable. Elle est constituée d’un serveur Debian qui permettra de recevoir les logs des différents équipements sur le réseau. Tous les logs seront filtrés par Rsyslog. Rsyslog permettra à la fois de détecter un ou des mots clés au sein des logs, pour ensuite les stocker et exécuter un script en fonction du log reçu. Tous les scripts permettant les résolutions sont développés en Python.

Voici l’exemple du déroulement d’un case :

1. Reception d’un log contenant « Buffer list is empty »
2. Rsyslog lance l’execution du script [Log Debug debug2](#_Log_Debug_debug2)
3. Switch passe en mode debug
4. Reception d’un log contenant ‘slnHwlrnCbkHandler', 'port' et 'bcmd'
5. Exécution du script [L2 Loop](#_L2_Loop_Script)
6. Le script detecte une loop en analysant les logs
7. Le script envoie d’une demande de fix à l’administrateur par Rainbow et/ou email
8. L’administrateur réponds oui à la demande
9. Le script envoie une commande ssh afin de réparer le probleme
10. Le script désactive le mode debug du switch

Pour pouvoir utiliser la Préventive Maintenance, le dossier devra impérativement contenir les fichiers suivants :

* **Setup.sh** : permet la configuration du serveur Debian.
* **Devices.csv** : Contient les IP address des switches afin de configurer active output socket.
* **support\_active\_output\_socket.py** : Configure out socket sur les switches contenu dans Device.csv.
* **support\_tools.py** : Permet le fonctionnement de tous les autres scripts support.
* **support\_send\_notification.py** : permet l’envoie des différentes notifications
* **support\_response\_handler.py** : Permet d’orchestrer l’émission et la réception des requests par notifications
* **support\_web\_receiver\_class.py**: Permet la configuration du serveur web on the fly pour recevoir les réponses envoyées par l’utilsateur.

Pour la première version de Preventive Maintenance, afin de configurer entièrement le serveur Debian, script bash sous le nom de Setup.sh à été développé. Celui-ci permet de configurer les éléments suivants :

* Configuration des notifications (Rainbow/Email)
* Pattern pour la collecte des logs des switchs
* Credentials des switchs
* Credentials des Stellar access points
* Pattern pour la collecte des logs des AP
* Les sous réseaux autorisés à envoyer les logs
* Installation de Python3 et des dépendances
* Configuration du Rsyslog
* Configuration du Logrotate
* Configuration de Iptables (pas encore implémenté)
* Configuration du serveur TFTP
* Création répertoire /opt/ALE\_Script

Active socket output on switches

## 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

L’objectif du script Setup.sh est d’initialisé tous les services qui seront utiles au bon fonctionnement de la fonctionnalité Preventive Maintenance. Pour fonctionner Préventive Maintenance a besoin de service qui ne sont pas présent sur un serveur Ubuntu a l’installation. Il faut également pouvoir configurer ces différents services. Une sécurité est également ajoutée afin de prévenir des mauvaises entrées, vérification de la forme du mail par example ou d’une adresse IP.

## Flowchart

## Services

Tous les services décrit ci-dessous sont entièrement installés et configurés par le script Setup.sh. Il n’est nullement nécessaire de modifier les fichiers de configuration de ces différents services sous peines de disfonctionnements.

### TFTP

Un serveur TFTP est installer sur le serveur Ubuntu, sont répertoire par default est configuré au chemin /tftpboot. Il est accessible en SFTP sur le port 22. Son rôle est de pouvoir recevoir les logs des switchs (show tech-support eng complete) ou des AP Stellar(take\_snapshot.sh start).

### Rsyslog

Rsyslog est le service qui permettra de récupérer les logs provenant des différents équipements. Il est configurer pour détecter les mots clé que les équipements sont susceptibles d’envoyer en fonction des différents cases. Une fois qu’un mot clé est détecté , le rsyslog va stocker les différents logs reçu dans le chemin /var/log/devices/ avec le nom de cette forme : ip\_address\_date\_history.json

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

Il va également crée un autre fichier temporaire en fonction du case, afin de pourvoir manipuler plus facilement les derniers logs reçu :

* lastlog.json
* lastlog\_ddos.json
* lastlog\_ddos\_ip.json
* lastlog\_flapping.json
* lastlog\_loop.json

Ces différents fichiers seront décrits dans les parties du rapport qui les concernes.

Rsyslog permettra également en fonction des mot clés détectés dans les logs d’exécuter le scripts python correspondant.

Flowchart :

Diagram

Description automatically generated

### Logrotate

Logrotate permet de limiter la taille des fichiers journaux présents dans /var/log.

Pour chaque fichier journal, logrotate réalise 2 opérations simultanées :

* la rotation : il archive le fichier journal sous un autre nom et supprime la plus ancienne archive
* la compression : il compresse éventuellement le fichier journal avant de l'archiver

### Iptables

Not yet implemented

# UTILS PYTHON SCRIPTS

## Introduction

L’utilisation de scripts utils permet a plusieurs script ayant la même action a faire de pas multiplier les lignes de code. En effet, cela permet de réutiliser le même code sur différents scripts. (send notification ,support tools,)

Cela permet aussi de retrouver le code plus facilement en le fractionnant et en les classant en plusieurs fichiers. (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

Ce script est exécuté une seule fois lors de l’exécution de Setup.sh.

Il permet la configuration de « swlog output socket ip\_address\_debian\_server », cette commande permet d’envoyer les logs du switch Alcatel vers le serveur debian afin qu’ils soient utilisés par 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

Les mots clés présent dans les logs reçu de la part de équipements Alcatel ne suffissent pas a eux seuls de détecter si il s’agit bien d’un véritable problème.

Pour cela des fonctions permettant l’analyse des derniers logs avec le même mot clé reçu est fait (sur la fréquence des messages reçus, la quantité de messages reçus et grâce aux Timestamps).

Les scripts permettent également l’envoie de commandes vers les équipements, ce qui permettra la résolution des 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 detects 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 thing , 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.

**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

When both conditions are met 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 the port sent in the logs.
* If the answer Is yes and same my decision send a ssh packet to disable the port 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 detects 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.

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

Next, the script extracts 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 counter 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 number equals 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.

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":""}*

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