|  |
| --- |
| ambicotech GmbH |
| Integration eines Faxservers |
| Installation eines Faxservers auf Basis von Debian Linux und HylaFAX über Ansible |

|  |
| --- |
| Christian Hann  Azubi-Nr.: 3600417  Prüflings-Nr.: 90997  25.5.2016  Ambicotech Projects - Ambicotec GmbH  Alexanderstr. 7  10178 Berlin Germany |

Inhalt

[Abbildungsverzeichnis III](#_Toc451504589)

[Abkürzungsverzeichnis III](#_Toc451504590)

[1. Einleitung 1](#_Toc451504591)

[1.1 Projektbeschreibung 1](#_Toc451504592)

[1.2 Projektziele 1](#_Toc451504593)

[1.3 Projektumfeld 1](#_Toc451504594)

[1.4 Projektbegründung 1](#_Toc451504595)

[1.4.1 Projektschnittstellen 2](#_Toc451504596)

[2 Projektplanung 2](#_Toc451504597)

[2.1 Projektphasen 2](#_Toc451504598)

[2.2 Ressourcenplanung 2](#_Toc451504599)

[2.3 Entwicklungsprozesse 2](#_Toc451504600)

[2.3.1 Ansible 2](#_Toc451504601)

[2.3.2 Shellscript Sendebericht 3](#_Toc451504602)

[3. Analyse 3](#_Toc451504603)

[3.1 Ist-Analyse 3](#_Toc451504604)

[3.2 Wirtschaftlichkeitsanalyse 3](#_Toc451504605)

[3.2.1 Make or Buy? 3](#_Toc451504606)

[3.2.2 Projektkosten 4](#_Toc451504607)

[4. Projektdurchführung 4](#_Toc451504608)

[4.1 Erstellung der Virtuellen Maschine 4](#_Toc451504609)

[4.2 Installation des Grundsystems 4](#_Toc451504610)

[4.2.1 Konfiguration Debian 8.4.0 5](#_Toc451504611)

[4.3 Erstellung Ansible-Playbook 5](#_Toc451504612)

[4.3.1 Definition Task 5](#_Toc451504613)

[4.3.2 Definition Hosts 5](#_Toc451504614)

[4.3.3 Definition ansible.cfg 6](#_Toc451504615)

[4.3.4 Definition Playbook 6](#_Toc451504616)

[4.4 Softwarekonfiguration 6](#_Toc451504617)

[4.4.1 IAXModem-Konfiguration 6](#_Toc451504618)

[4.4.2 Asterisk-Konfiguration 6](#_Toc451504619)

[4.4.3 HylaFAX-Konfiguration 7](#_Toc451504620)

[4.4.4 Konfiguration Monitoring 8](#_Toc451504621)

[5. Testphase 8](#_Toc451504622)

[5.1. HylaFAX, Asterisk, IAXModem 8](#_Toc451504623)

[5.1.1 Kontrolle des Faxversands 8](#_Toc451504624)

[5.2 Kontrolle des Monitorings 8](#_Toc451504625)

[6. Soll / Ist-Vergleich 9](#_Toc451504626)

[7. Fazit 9](#_Toc451504627)

[7.1 9](#_Toc451504628)

[Anhang i](#_Toc451504629)

[A.1 – Detaillierte Zeitplanung i](#_Toc451504630)

[A.2 – Script-Code ii](#_Toc451504631)

[A.2.1 – Ansible-Playbook ii](#_Toc451504632)

[A.2.2 – Shellscript vollqualifizierter Sendebericht iii](#_Toc451504633)

[A.3 – Systemeinstellungen v](#_Toc451504634)

[A.3.1 - VMware ESXi 6.0 v](#_Toc451504635)

[A.3.2 – Debian 8.4.0 Systemeinstellungen v](#_Toc451504636)

[A.3.2.1 – Debian Netzwerkkonfiguration v](#_Toc451504637)

[A.3.2.2 – SSH v](#_Toc451504638)

[A.4.0 - Softwarekonfiguration vi](#_Toc451504639)

[A.4.1 - IAXModem-Konfiguration vi](#_Toc451504640)

[A.4.2 - Asterisk vi](#_Toc451504641)

[A.4.2.1 - extensions.conf vi](#_Toc451504642)

[A.4.2.2 - iax.conf vi](#_Toc451504643)

[A.4.2.3 - sip.conf vii](#_Toc451504644)

[A.4.2.4 - check\_faxq viii](#_Toc451504645)

[A.4.2.5 - check\_hylafax xi](#_Toc451504646)

[A.4.2.6 - commands.conf xiii](#_Toc451504647)

# Abbildungsverzeichnis

[Abb. 1 - Vollqualifizierter Sendebericht 3](#_Toc451503680)

[Abb. 2 - Graphenansicht Faxversand 9](#_Toc451503681)

# Abkürzungsverzeichnis

* **CT:** Computed tomography (Computertomographie)
* **DVT:** Deep vein thrombosis (Tiefe Venenthrombose)
* **DXA:** Dual-energy X-ray absorptiometry (Dual-Röntgen-Absorptiometrie, Bestimmung der Knochendichte)
* **MRT:** Magnetic Resonance Tomography (Magnetresonanztomographie)
* **PACS:** Picture Archiving and Communication System (Bildablage- und Kommunikationssystem)
* **RIS:** Radiologieinformationssystem
* **YAML:** YAML Ain’t Markup Language (Vereinfachte Auszeichnungssprache nach RFC2822 für Perl, Python und C)
* **SSH:** Secure Shell (sichere Kommandozeile über Remoteverbindung)
* **VM:** Virtuelle Maschine / Computer
* **SCP:** Secure Copy (Gegenstück zu SSH, wird zum Kopieren von Dateien über eine SSH-Verbindung genutzt)
* **IAXModem:** Faxmodem auf Softwarebasis

# 1. Einleitung

In der vorliegenden Projektdokumentation wird der Aufbau und die Durchführung des IHK-Projekts, was vom Author Christian Hann im Rahmen der Umschulung zum Fachinformatiker Systemintegration, erarbeitet und erstellt wurde. Praktikumsbetrieb ist die Ambicotech GmbH mit Hauptsitz in Österreich und einer Zweigstelle in Berlin.

Zu den Hauptaufgaben zählt, für privatärztliche Praxen die IT-Infrastruktur für bildgebende Untersuchungsformen ([CT](#CT), [MRT](#MRT), [DXA](#DXA), [DVT](#DVT)) zu planen, organisieren und aufzubauen.

## 1.1 Projektbeschreibung

Der Faxserver wird auf Basis von Debian 8.4.0, Asterisk und HylaFAX2 6.0.6 für die Standorte Radiologie Zentrum Freiburg und der Privatärztliche radiologische Praxis Dr. Jäger (Saarbrücken) aufgesetzt. Das Monitoring soll durch Icinga2 realisiert werden. Bei der Installation müssen die Pflichtziele eingehalten und die optionalen Ziele je nach verbleibender Zeit realisiert werden.

## 1.2 Projektziele

Als Projektziele werden hier die Funktionsfähigkeit des Faxservers für die Protokolle G.711 und T.38, ein vollqualifizierter Sendebericht und das einbinden des Faxservers in das bestehende Monitoring gesetzt. Zusätzlich soll noch die Integration des Faxservers an die bestehende Infrastruktur von [PACS](#PACS) / [RIS](#RIS) geschehen.

Als optionale Ziele gelten hier die Benachrichtigungen bei eingehender Faxe und Monitoringereignissen.

## 1.3 Projektumfeld

Auftraggeber ist hier die Radiologische Praxis Dr. Jäger (Saarbrücken) und der Standort Radiologie Zentrum Freiburg, welche beide Dr. Jäger gehören.

Dr. Jäger betriebt mehrere privatärztliche Praxen für radiologische Untersuchung und arbeitet dabei mit bildgebenden Untersuchungsmethoden. Geräte wie das [CT](#CT) oder [MRT](#MRT) von Philips sind hierbei in der Lage die Untersuchungsergebnisse und relevanten Bilder direkt an die zuweisenden Ärzte zu verschicken.

Um den Versand der Untersuchungsberichte und Bilder gesetzeskonform zu ermöglicht wird hier auf einen Faxbetrieb zurückgegriffen, da dieses in Deutschland als einzige rechtlich anerkannte Methode für medizinische Daten gilt.

## 1.4 Projektbegründung

Aktuell wird in den Praxen ein ISDN-Anschluss der Telekom genutzt, welcher in Zuge der Modernisierung auf einen IP-Anschluss umgestellt wird. Da IP-Anschlüsse keinen Support für die Protokolle G.711 und T.38 bieten, soll mit diesem Fax-Server eine entsprechende funktionale und praxistaugliche Lösung angeboten werden. Zusätzlich soll hier die Installation der Software automatisch mit Ansible geschehen, um spätere Installation einfach "per Knopfdruck" auszurollen.

Die Einbindung ist das Monitoring geschieht für eintretende Supportfälle oder auch vorbeugende Maßnahmen (Neustart des Servers, Kontrolle der Fax-Queue bei Übertragungsfehlern oder abgebrochenen Übertragungen, usw.)

## 1.4.1 Projektschnittstellen

Für umfangreiche Scripte steht hierzu ein Anwendungsentwickler zur Verfügung, welcher unterstützend bei diesem Projekt tätig ist.

# 2 Projektplanung

|  |  |
| --- | --- |
| Projektphase | Zeit |
| Projektplanung | 4 Stunden |
| Erstellung Ansible-Playbook | 8 Stunden |
| Serverinstallation | 3 Stunden |
| Installation Software | 1,5 Stunden |
| Konfiguration Software | 7,75 Stunden |
| Testphase | 2 Testphase |
| Dokumentation & Abnahme | 8 Stunden |

## 2.1 Projektphasen

Tab - Zeitliche Planung Projektphasen

Für die Umsetzung des Projektes standen insgesamt 35 Stunden zur Verfügung. Die Anzahl der Stunden wurde zu Beginn auf das gesamte Projekt verteilt um einen Workflow zu erhalten, der beim Kunden jederzeit nachvollzogen werden kann. Eine grobe Planung der Phasen lässt sich der untenstehenden Tabelle entnehmen. Eine detaillierte Planung lässt sich im Anhang unter [A.1 - Detaillierter Zeitplanung](#_A.1_–_Detaillierte) einsehen, sowie unter [A.5 Projektplanung](#_A.5.0_-_Projektplanung) ist das Gant-Diagramm zu finden.

## 2.2 Ressourcenplanung

Bei der Planung der Ressourcen wird hauptsächlich auf bereits vorhandene Hard- und Software aufgesetzt und darauf geachtet das freie Quellen (Open Source) genutzt werden.

Für die Hardware wird ein bereits existierender Dell PowerEdge R410 genutzt. Auf diesem Server ist der Hypervisor VMware ESXi 6.0 installiert, der bereits virtuelle Computer (Monitoring, Management-Client) hostet.

Debian, Asterisk, HylaFAX und Ansible sind jeweils Betriebssystem und Softwarepakete welche es als Open Source gibt.

Bei der Implementierung des Monitorings werden als Basis Scripte verwendet, die bereits so im Gebrauch sind und auf den Faxserver mit seinen individuellen Merkmalen zugeschnitten wird.

## 2.3 Entwicklungsprozesse

### 2.3.1 Ansible

Für das Ansible-Playbook werden zuerst die Rahmenbedingungen für das Playbook festgelegt und im Anschluss in [YAML](#YAML) geschrieben. Als Rahmenbedingung gilt hier welche Tasks zusammengefasst werden können und was bei der Installation als Task gewertet werden kann.

Nach der Definition der Rahmenbedingungen wird das Playbook in einer ersten Version verfasst und getestet, sowie eventuell gefundene Fehler direkt angepasst. In der zweiten Version wird das Playbook einem Refactoring unterzogen um dieses zu optimieren und übersichtlicher zu gestalten

### 2.3.2 Shellscript Sendebericht

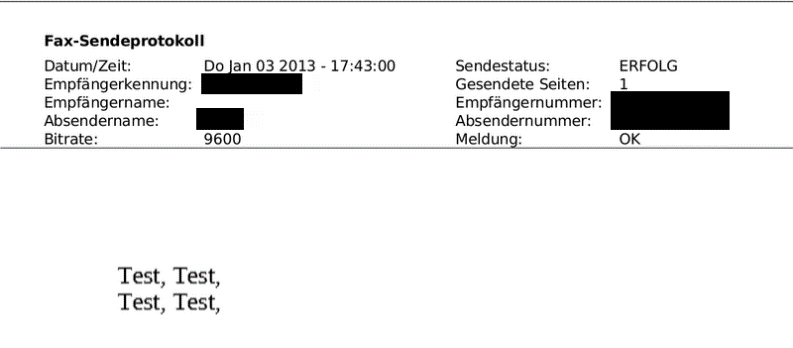
Das Shellscript für den vollqualifizierten Sendebericht wird anhand eines vorhandenen Scripts aus HylaFAX so angepasst das es dem geforderten Layout entspricht (siehe Abb. 1).

Abb. - Vollqualifizierter Sendebericht

Alle Scripte die benutzt werden sind im Anhang unter [A.2 – Scripte](#_A.2_–_Script-Code) zu finden. Der Script-Code ist hierbei auch soweit kommentiert, das man sehen kann an welcher Stelle was gemacht wurde um diesen einfach nachvollziehen zu können.

# 3. Analyse

## 3.1 Ist-Analyse

Derzeitig liegt in der Arztpraxis ein ISDN-Anschluss, mit bestehendem physikalischen Brother FAX-2820 Laserfaxgerät, der im Zuge der Modernisierung seitens der Telekom zu einem Anschluss auf IP-Basis ausgebaut werden soll. Da IP-basierte Anschlüsse keinen Support für die Protokolle T.38 / G.711 bieten, muss das bestehende Faxgerät gegen einen Faxserver ausgetauscht und ins Monitoring eingebunden werden.

In der bestehenden Infrastruktur befindet sich ein Dell PowerEdge R310-Server mit VMware ESXi 6.0 der für die Virtualisierung genutzt werden kann.

Zusätzlich erfordern die medizinischen bildgebenden Geräte einen direkten Faxserver an welchen sie die Untersuchungsergebnisse (Befunde) und Bilder (Röntgenaufnahmen) weiterreichen können. Dies ist dabei eine grundlegende Anforderung seitens Phillips und der verarbeitenden Software meddixRIS und meddixPACS.

## 3.2 Wirtschaftlichkeitsanalyse

Aufgrund der in [1.4](#_1.4_Projektbegründung) ([Projektbegründung](#_1.4_Projektbegründung)) und in [3.1](#_3.1_Ist-Analyse) ([Ist-Analyse](#_3.1_Ist-Analyse)) genannten Gründen, ist es zwingend das dieses Projekt umgesetzt wird. Ob es gerechtfertigt ist dieses Projekt aufgrund der Kosten nicht durchzuführen stellt sich hierbei nicht, da es für den reibungslosen Betrieb des Geräteparks und der Praxen essentiell ist.

### 3.2.1 Make or Buy?

Schon während der Projektplanung kam die Frage auf ob man eine entsprechende Lösung bei einem Hersteller hinzukauft oder ob man diese selbst zusammenstellt. Alle Angebote hatten den Nachteil, dass sie eine ISDN-Adapterkarte verwendeten um die Faxfunktionalität zu gewährleisten.

Da ISDN mit der Zeit seitens der Telekom durch All-IP ersetzt wird, fiel die Entscheidung zugunsten des selbst gebauten Servers, welcher ausschließlich via TCP/IP die Faxe versendet und damit sicherstellt auch in Zukunft mit IPv6 zusammenarbeitet.

Die untenstehende Tabelle dient hierbei als kaufmännische Betrachtung die anhand technischer Merkmale durchgeführt. Finanzelle Mittel spielten hierbei keine Rolle, da es sich um einen essentiellen Systembestandteil handelt.

|  |  |  |
| --- | --- | --- |
| Anforderung | Angebot | Eigenlösung |
| Fax-Versand via TCP/IP | Nur per ISDN-Karte | Ja |
| Konfiguration | Wird vom Hersteller vorgenommen | Wird anhand definierter Parameter selbst vorgenommen |
| IPv4 | Nur Management | Ja |
| IPv6 | Nicht implementiert / aktiviert | Ja |
| Wartung | Hersteller | Administrator |
| Erweiterbarkeit | Nicht vorgesehen | Ja |
| Sicherheit | Nur durch Hersteller | Administrator |

Tab - Kaufmännische Betrachtung anhand technischer Merkmale

### 3.2.2 Projektkosten

Die anfallenden Projektkosten sollen im Folgenden kalkuliert werden. Neben den Personalkosten, müssen hierbei die Kosten für die Realisierung und Ressourcen (Hardware, Software, Arbeitsplatz) beachtet werden. Die genauen Personalkosten können in der Berechnung aus datenschutzrechtlichen Gründen nicht genutzt werden. Aus diesem Grund werden die Marktüblichen Preise genutzt. Hierbei beträgt der Stundensatz für einen Auszubildenden / Praktikant 15€, der eines festangestellten Mitarbeiters 35€. Die Ressourcennutzung wird mit einem pauschalen Stundensatz von 20€ berechnet.

Die Kosten für jeglichen Vorgang, sowie die gesamten Projektkosten, lassen sich aus der nachfolgenden Tabelle entnehmen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vorgang | Mitarbeiter | Zeit | Personal | Ressourcen | Gesamt |
| Planung | 1x Praktikant | 4h | 60€ | 80€ | 140€ |
| Installation | 1x Praktikant | 4,5h | 67,50€ | 90€ | 157,50€ |
| Konfiguration | 1x Praktikant | 7,75h | 116,25€ | 155€ | 271,25€ |
| Testphase | 1x Praktikant | 2h | 30€ | 40€ | 70€ |
| Dokumentation | 1x Praktikant | 5h | 75€ | 100€ | 175€ |
| Abnahme | 1x Mitarbeiter | 3h | 105€ | 60€ | 165€ |
| Projektkosten gesamt: | | | | | **978,75€** |

Tab 3 - Übersicht der Projektkosten

# 4. Projektdurchführung

## 4.1 Erstellung der Virtuellen Maschine

Die virtuelle Maschine wird auf einem Dell PowerEdge R410 mit VMware ESXi 6.0 erstellt.

Hierbei wurde von mir mit dem Verwaltungstool vSphere-Client die virtuelle Maschine mit den Parametern aus [A.3.1 – VMware ESXi 6.0](#_A.3.1_-_VMware) erstellt und gestartet.

## 4.2 Installation des Grundsystems

Im Anschluss zur Erstellung der virtuellen Maschine wurde als erstes Debian als Minimal-Version installiert. Hierbei wurde von mir zuerst eine dynamisch IP-Adresse gewählt und in der Paketverwaltung die Pakete [*SSH-Server*](#SSH) und *Systemwerkzeuge* markiert. Eine Desktopumgebung wird nicht benötigt, da der Server später via [SSH](#SSH) administriert wird. Nach der Paketauswahl beginnt die Installation des eigentlichen Kernsystems mit den Standardeinstellungen des Debian-Installers.

### 4.2.1 Konfiguration Debian 8.4.0

Die Grundkonfiguration des Installers wurde von mir den folgenden Punkten angepasst:

* *VMware-Tools*: Diese werden für den späteren Betrieb benötigt um eine reibungslose Kommunikation auf der Netzwerkschnittstelle zu ermöglichen. Dabei habe ich das Installationspaket gewählt was direkt von VMware geliefert wird. Diese Pakete wurden auf eine korrekte Arbeitsweise von VMware geprüft und freigegeben.
* *IP-Adressierung*: Hier habe ich der [VM](#VM) eine feste IP-Adresse aus dem bereits bestehenden Adressraum gegeben um die VM später eindeutig zu identifizieren, das Monitoring zu vereinfachen und die Einbindung in die bestehende Infrastruktur zu erleichtern.  
  Eine Bespielhafte Netzwerkkonfiguration ist unter [A.3.2.1 Debian Netzwerkkonfiguration](#_A.3.2.1_–_Debian) einsehbar. Die tatsächliche Netzwerkkonfiguration wird aus Sicherheits- und datenschutzrechtlichen Gründen hier nicht wiedergegeben.
* *sshd*: Der SSH-Dienst wird von mir so konfiguriert das ich von dem vorhandenen Service-PC jederzeit auf den Faxserver zugreifen kann. Hierzu wird das root-Login über SSH erlaubt und der Zugriff auf eine einzige feste IP-Adresse beschränkt (siehe [A.3.2.2 – SSH](#_A.3.2.2_–_SSH)).  
  Zusätzlich wird von mir noch ein privater und öffentlicher Schlüssel generiert um die Kommunikation zwischen dem Client und Server zu automatisieren. Dieser Schritt wird später für die Installation über Ansible benötigt.

Nachdem die Grundkonfiguration des Debian-Servers abgeschlossen wurde, wurde von mir eine doppelte Sicherung der VM durchgeführt. Dazu wurde einmal ein Snapshot der VM erstellt und diese zusätzlich über [SCP](#SCP) auf einem Netzwerkspeicherort gesichert.

## 4.3 Erstellung Ansible-Playbook

Bei Ansible handelt es sich um eine Deployment-Software, die anhand sogenannter Playbooks Installationen automatisiert durchführen kann. Die Wahl Ansible zu nutzen liegt darin begründet das man später weitere Instanzen des Fax-Servers mit entsprechend angepasster Config-Dateien individualisiert ausrollen kann.

Um ein funktionierendes Playbook zu erhalten musste ich jeden einzelnen Installationsschritt in Tasks aufschlüsseln. Die Tasks werden innerhalb des Playbooks auch als Einzelschritte ausgeführt. Dabei wird jeder Task nacheinander so ausgeführt wie es im Playbook erfasst ist.

### 4.3.1 Definition Task

Als ein Task wird definiert was eine eigenständige Einheit ist wie z.B. die Installation eines Softwarepakets. Das Playbook gliedert sich dabei in drei Haupt-Tasks (Paketinstallationen) und fünf Neben-Tasks (Konfiguration).

### 4.3.2 Definition Hosts

Als nächstes werden die Hosts bestimmt auf die Ansible für die Installation zugreifen muss. In diesem Fall sind es zwei Hosts die von mir identifiziert wurden. Dies sind der Monitoring-Host *icinga2* und der Fax-Host *hylafax2*. Damit ist die Erstellung der host-Datei bereits abgeschlossen (siehe unten).

[servers]

icinga2 ansible\_ssh\_host=192.168.1.201 ansible\_ssh\_port=22

hylafax2 ansible\_ssh\_host=192.168.1.200 ansible\_ssh\_port=22

### 4.3.3 Definition ansible.cfg

Im nächsten Schritt wurde von mir die Ansible-Konfigurationsdatei so bearbeitet das die vorbereitete Host-Datei verarbeitet wird und Ansible auch den privaten Schlüssel für die SSH-Verbindung findet. Dazu werden die folgenden Werte entsprechend in der Datei ansible.cfg eingefügt.

[defaults]

hostfile = hosts

remote\_user = root

private\_key\_file = ~/.ssh/id\_rsa

host\_key\_checking = true

Damit ist die Grundkonfiguration für Ansible und das Playbook abgeschlossen und wodurch ich nun mit der eigentlichen Erstellung des Playbooks beginnen konnte.

### 4.3.4 Definition Playbook

Im nächsten Schritt wurde von mir eine erste Version des Playbooks entworfen und getestet (Script siehe [A.2.1 - Ansible](#_A.2.1_–_Ansible)). Dabei verlief der erste Test ohne Fehler, wodurch der Fax-Server bereits einsatzfähig ist.

Zusätzliche wurde die Konfiguration des Monitorings, Asterisk, HylaFAX und IAXModem über Ansible realisiert. Die Konfiguration der weiteren Bestandteile erfolgt vor der Ansible-Konfiguration und die entsprechenden Dateien werden dann automatisch durch Ansible in die entsprechenden Ordner kopiert.

## 4.4 Softwarekonfiguration

### 4.4.1 IAXModem-Konfiguration

Das [IAXModem](#IAX) ist die Softwarekomponete welche die eigentliche Schnittstelle zwischen Faxserver und IP-Netzwerk darstellt. Hierzu habe ich die Version aus den Paketquellen von Debian 8.4.0 benutzt, welche über das Ansible-Playbook installiert wird.

Für die Konfiguration ist es nötig das man der Software mitteilt welche Gerätedatei genutzt werden soll. Diese muss zuvor erstellt werden und ist im [A.4.1 - IAXModem](#_A.4.1_-_IAXModem-Konfiguration) zu finden.

### 4.4.2 Asterisk-Konfiguration

Für die softwarebasierte Telefonanlage Asterisk, welche die Kommunikation des Faxmodems in die IP-Netze weiterleitet erfolgt die Konfiguration in drei Schritt. Hierzu habe ich die drei Dateien *extensions.conf, iax.conf* und *sip.conf* bearbeitet. Diese drei Dateien werden ebenfalls zu Anfang erstellt und via Ansible in das korrekte Verzeichnis kopiert. Nachfolgend eine Erläuterung der Bedeutung der einzelnen Dateien.

* [*extensions.conf*](#_A.4.2.1_-_extensions.conf)*:* Stellt die Wählregeln für ein und ausgehende Anrufe / Datenverbindung her. Nach diesem Regelsatz werden die Verbindungen auch den einzelnen Geräten zugewiesen.
* [*iax.conf*](#_A.4.2.2_-_iax.conf)*:* Legt fest wie Asterisk mit dem Faxmodem kommuniziert.
* [*sip.conf*](#_A.4.2.3_-_sip.conf)*:* Legt fest welches Protokoll und wie die Kommunikation nach außen geregelt wird.

Den Inhalt der Konfigurationsdateien kann im Anhang [A.4.2 - Asterisk](#_A.4.2_-_Asterisk) eingesehen werden. Nach einspielen der Konfiguration muss Asterisk neu gestartet werden um die Änderungen aktiv zu schalten.

### 4.4.3 HylaFAX-Konfiguration

#### 4.4.3.1 Allgemeine Konfiguration

Als letzten Schritt wird HylaFAX an sich konfiguriert. Dies wurde von mir per Hand gemacht, da die Konfiguration nicht per Ansible durchgeführt werden kann. Eine zusätzliche Schwierigkeit dabei war, dass es zu viele unterschiedliche Konfigurationsdateien bearbeitet werden müssen.

Für eine korrekte Konfiguration werden zwei Kommandozeilen-Tools benötigt die von HylaFAX bei in der Installation mitgeliefert werden. Bei diesen beiden Tools handelt es sich um *faxaddmodem* und *faxsetup*.

* *faxaddmodem*: Mit diesem Kommandozeilen-Tool werden die einzelnen Parameter des Faxmodems abgefragt und getestet. Das Tool legt hierbei auch die optimale Übertragungsgeschwindigkeit für die Faxe fest.
* *faxsetup*: Mit diesem Kommandozeilen können die Parameter des Faxservers direkt konfiguriert werden. Dazu zählen die wichtigen Parameter:
  + *Absender*
  + *Faxnummer*
  + *Datum & Uhrzeit*
  + *Vorwahl*
  + *Länderkennung für Internationale Faxe*

#### 4.4.3.2 Konfiguration Sendebericht

Als nächste wurde von mir der Sendebericht konfiguriert um beim Versenden eines Faxes auch eine entsprechende Nachweismöglichkeit zu bieten. Dieser Sendebericht wird im Normalfall von Hylafax nicht unterstützt und muss händisch implementiert werden. Da der Sendebericht zusätzlich nicht via Drucker ausgegeben werden soll, muss hierzu noch der exim4 zum Mailversand konfiguriert werden, was allerdings in einem vorgehenden Projekt bereits geschehen ist. Hier wird exim 4 nur eingebunden.

Um den Sendebericht zu konfigurieren kann man in der die Funktion *BuildAttachArgs()* und *ConvertFile()* aus den custom-functions editieren. Die vollständigen Funktionen befinden sich im Anhang [A.2.2 - Shellscript vollqualifizierter Sendebericht](#_A.2.2_–_Shellscript). Dabei wurden die Codezeilen die normalerweise im Script vorhanden sind und von der Änderung nicht betroffen sind entfernt um nur das wesentliche darzustellen.

Im Anschluss werden die beiden Standardscripte *notify* und *faxrcvd* von HylaFAX so angepasst das die Funktion BuildAttachArgsMod() anstatt der Funktion BuildAttachArgs() aufgerufen wird. Zusätzlich wird der Funktionsaufruf in dem Shellscript notify noch mit dem Parameter notify versehen. Um einen reibungslosen Aufruf beider geänderten Funktionen zu ermöglichen wird am Anfang jeder Datei noch die Zeile

. bin/custom-functions

eingefügt. Diese Zeile sorgt dafür, dass das Shell-Script *custom-functions* auch aufgerufen wird. Ist dies erledigt hat man direkt die Möglichkeit mit den Keywords *REPORT* die erste Seite des Fax an den Sendebericht zu hängen und mit *REPORT+* eine vollständige Kopie des Faxes anzuhängen.

Zum Abschluß muss noch das Paket *ImageMagick* installiert werden um sicherzustellen das die Bilder auf dem Sendebericht korrekt skaliert werden. Das Paket wird dabei von mir mit einem

apt-get install imagemagick -y

installiert. ImageMagick benötigt hierbei keine weitere Konfiguration.

### 4.4.4 Konfiguration Monitoring

Um das Monitoring zu gewährleisten wird via Ansible die entsprechenden check-Scripts auf den Monitoring-Server übertragen. Zusätzlich wird dazu die *commands.conf* bearbeitet um die dazugehörigen Kommandos einzubinden.

Das bearbeiten der *commands.conf* passiert vor der Installation mit Ansible, da hier von mir die Richtigkeit der Syntax besser sichergestellt werden kann. Den entsprechenden Scriptcode, sowie die Änderungen in der *commands.conf* können in den Anhängen [A.4.2.4 - check\_faxq](#_A.4.2.4_-_check_faxq), [A.4.2.5 - check\_hylafax](#_A.4.2.5_-_check_hylafax) und [A.4.2.6 - commands.conf](#_A.4.2.6_-_commands.conf) nachgelesen werden.

Die check-Dateien wurden hierbei von einem Anwendungsentwickler geschrieben, der anhand benötigter Spezifikationen die Scripte entwickelt hat.

# 5. Testphase

Zur Testphase wird der ordnungsgemäße Betrieb der Installation sichergestellt und auf eventuelle Fehler geprüft.

## 5.1. HylaFAX, Asterisk, IAXModem

Um den Faxserver zu testen werden hierzu von mir mehrere Faxe verschickt und von einer Gegenstelle werden Faxe auf die allgemein bekannte Faxnummer geschickt um den korrekten Empfang sicherzustellen.

Den Faxversand zu testen erfolgt hierbei in zwei unterschiedlichen Varianten. Für die erste Variante habe ich den normalen Weg über einen Faxdrucker gewählt, der unter einem entsprechenden Windows-Client eingebunden wurde und darüber Faxe mit einer unterschiedlichen Anzahl von Seiten verschickt habe.

Bei dem zweiten Test wurden die Faxe über die Kommandozeile mit dem Befehl *sendfax* verschickt. Hierzu wurden auch wieder mehrere Faxdateien mit unterschiedlicher Seitenanzahl erstellt, welche dann per Kommandozeile verschickt wurden.

### 5.1.1 Kontrolle des Faxversands

Um den Ordnungsgemäßen Versand der Faxe zu verifizieren wurde zum einen der entstandene Sendebericht überprüft und zusätzlich die Logfiles des Linux-Servers. In den Logfiles wurde dabei auch auf die nahtlose Kommunikation der einzelnen Komponenten geachtet, um sicherzustellen das es hier bei keinem der Komponenten Fehler auftreten.

Kriterien für den Test waren eine reibungslose Kommunikation der Komponenten, eine einwandfreie Generierung des Sendeberichts und einwandfreie Darstellung beim Empfänger. Hierbei ließen sich keine Fehler ermitteln. Der Sendebericht wurde dabei korrekt erstellt und auch das Failsafe-System der Komponenten griff bei absichtlich herbeigeführten Fehlern. Die Qualität der versendeten Faxe war auch dicht am Original (hier war mit Verlusten aufgrund der Komprimierung im Faxsignal zu rechnen).

## 5.2 Kontrolle des Monitorings

Um das Monitoring zu kontrollieren wird zuerst überprüft ob der Monitoring-Server den Faxserver per IP finden und ansprechen kann. Ist dies der Fall wird als nächstes Kontrolliert ob die entsprechenden Auslastungsgraphen dargestellt werden.

Die Auslastungsgraphen sind dann bei den Faxtests auch die Kontrollinstanz ob das Monitoring richtig arbeitet (siehe Abb. 2 - Graphenansicht Faxversand).

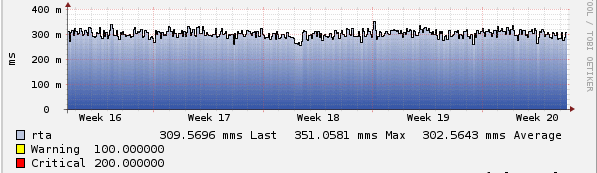


Abb. - Graphenansicht Faxversand

Die Graphik zeigt hierbei die Sendedauer in *ms* an.

Um ebenfalls späteren Support anhand des Monitorings zu gewährleisten wurde für den Fall das die Warn-Schwellen erreicht werden eine Notification per E-Mail meinerseits eingerichtet.

# 6. Soll- / Ist-Vergleich

## 6.1 Erfüllung der Projektziele

Hierbei wurden von mir alle Projektziele, welche unter Punkt [1.2 Projektziele](#_1.2_Projektziele) definiert sind, erreicht. Dabei wurde die zuvor unter [2.1 Projektphasen](#_2.1_Projektphasen) ermittelten Zeiten eingehalten.

|  |  |  |  |
| --- | --- | --- | --- |
| Projektphase | Soll | Ist | Differenz |
| Projektplanung | 4 h | 4 h | 0 h |
| Erstellung Ansible-Playbook | 8 h | 7,5 h | -1 h |
| Serverinstallation | 3 h | 2,75 h | - 0,25 h |
| Installation Software | 1,5 h | 1,5 h | 0 h |
| Konfiguration Software | 7,75 h | 7,5 h | - 0,25 h |
| Testphase | 2 h | 2 h | 0 h |
| Dokumentation & Abnahme | 8 h | 8,5h | + 0,5h |

Tab - Soll- / Ist-Vergleich

# 7. Fazit

Als Fazit kann ich festhalten das dieses Projekt jederzeit wieder durchführbar wäre und auch eine sehr gute Ergänzung und eine starke Verbesserung gegenüber dem aktuellen Faxstandard darstellt, welcher nicht mehr dem Stand der Technik der entspricht.

Auch das automatische Deployment über Ansible würde ich wieder so durchführen, allerdings eine bis zwei Änderungen einfließen lassen, welche sich anhand der der zugrundeliegenden Infrastruktur entscheidet.

Sollte die Infrastruktur auf reinen virtuellen Maschinen aufbauen, könnte man hier das Ansible-Script so erweitern das die virtuelle Maschine anhand eines Templates erstellt und installiert wird. Mit den entsprechenden Ansible-Plugins, welche verfügbar sind, ist dies durchaus realisierbar und automatisiert den Prozess noch stärker. Wenn dann zusätzlich noch Clients über den Faxserver arbeiten sollen, könnte man hier über Ansible dies ebenfalls realisieren, indem man die Installation des virtuellen Faxdruckers auf den Clients automatisiert

# Anhang

# A.1 – Detaillierte Zeitplanung

|  |  |
| --- | --- |
| Projektphasen | Zeit |
| Projektplanung   * Planung des Projekts mit Meilensteinen und Arbeitspakete | *4 Stunden gesamt*  4 Stunden |
| Erstellung Ansible Playbook   * Festlegen Rahmenbedingungen * Schreiben des Playbooks | *8 Stunden gesamt*  3 Stunden  5 Stunden |
| Serverinstallation   * Generierung VM * Installation Debian * Aktualisierung Debian | *3 Stunden gesamt*  0,5 Stunden  2 Stunden  0,5 Stunden |
| Installation Software   * Installation IAXModem * Installation Asterisk * Installation HylaFAX | *1,5 Stunden gesamt*  0,5 Stunden  0,5 Stunden  0,5 Stunden |
| Konfiguration Software   * Konfiguration IAXModem * Konfiguration Asterisk * Konfiguration HylaFAX * Konfiguration Sendebericht * Konfiguration Monitoring | *7,75 Stunden gesamt*  0,25 Stunden  0,5 Stunden  2 Stunden  2 Stunden  4 Stunden |
| Testphase   * Test HylaFAX-Server * Test Monitoring | *2 Stunden gesamt*  1,5 Stunden  0,5 Stunden |
| Dokumentation & Abnahme   * Dokumentation & Übergabegespräch * Abnahme & Übergabe | *8 Stunden gesamt*  5 Stunden  3 Stunden |

# A.2 – Script-Code

## A.2.1 – Ansible-Playbook

## A.2.2 – Shellscript vollqualifizierter Sendebericht

BuildAttachArgsMod ()

{

REPORT|report|REPORT+|report+)

a\_type="application/pdf"

a\_desc="Fax-Report (PDF)"

;;

CustomFileName $a\_name $2 $1 $QFILE

if [ -f "$a\_file" ] && [ "${FILEMERGING,,}" == "yes" ] ; then

MERGEFILES="$MERGEFILES"" "$a\_file

MERGEPATH=$(dirname $a\_file)

MERGEFILE=$MERGEPATH/merge.tmp.$1

MERGENAME=$a\_name

FILECOUNT=$((FILECOUNT+1))

elif [ -f "$a\_file" ] && [ "${FILEMERGING,,}" != "yes" ] ; then

ARGS="$ARGS"" \"$a\_file\" \"$a\_type\" \"$a\_name\" \"$a\_desc\""

fi

fi

fi

done

if [ "$MERGEFILES" != "" ] && [ "${FILEMERGING,,}" == "yes" ] && [ "$FILECOUNT" -gt 1 ] ; then

touch $MERGEFILE

case $1 in

PostScript|Postscript|PS|ps)

TraceLog "Using ghostscript to merge multiple postscript files"

gs \

-q -dNOPAUSE -dBATCH -sDEVICE=pswrite -dSAFER \

-dColorConversionStrategy=/LeaveColorUnchanged \

-dSubsetFonts=true -dEmbedAllFonts=true \

-sOutputFile="$MERGEFILE" \

${MERGEFILES:1}

ARGS="$ARGS"" \"$MERGEFILE\" \"application/postscript\" \"$MERGENAME\" \"FAX Dokument (Postscript)\""

;;

PDF|pdf|REPORT|report|REPORT+|report+)

TraceLog "Using ghostscript to merge multiple PDF files"

gs \

-q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -dSAFER -dPDFSETTINGS="/printer" \

-dColorConversionStrategy=/LeaveColorUnchanged \

-dSubsetFonts=true -dEmbedAllFonts=true \

-sOutputFile="$MERGEFILE" \

${MERGEFILES:1}

if [ "${1,,}" == "pdf" ] ; then

ARGS="$ARGS"" \"$MERGEFILE\" \"application/pdf\" \"$MERGENAME\" \"FAX-Dokument (PDF)\""

else

ARGS="$ARGS"" \"$MERGEFILE\" \"application/pdf\" \"$MERGENAME\" \"FAX-Report (PDF)\""

fi

;;

TIFF|tiff|TIF|tif)

TraceLog "Using tiffcp to merge multiple TIFF files"

tiffcp ${MERGEFILES:1} $MERGEFILE

ARGS="$ARGS"" \"$MERGEFILE\" \"image/tiff\" \"$MERGENAME\" \"FAX-Dokument (TIFF)\""

;;

esac

elif [ -f "$a\_file" ] && [ "$1" != "raw" ] && [ "$1" != "original" ] ; then

TraceLog "No merging of files required; returning single file"

ARGS="$ARGS"" \"$a\_file\" \"$a\_type\" \"$a\_name\" \"$a\_desc\""

fi

printf '%s' "$ARGS"

}

ConvertFileMod ()

{

REPORT|report|REPORT+|report+)

TraceLog "Deciding whether or not to create a fax report"

case $4 in

notify)

TraceLog "Converting $1 to PDF fax report"

case $2 in

PostScript|Postscript|PS|ps)

TraceLog "Using ps2pdf"

NAME="`basename $1 .ps`"

if [ "${3,,}" == "report+" ] ; then

PDFIN=$TMPDIR/$NAME$.report+.pdf

else

PDFIN=$TMPDIR/$NAME$.report.pdf

fi

$PS2PDF $1 $PDFIN >/dev/null 2>&1

TraceLog "Using CreateFaxReport"

CreateFaxReport $PDFIN $QFILE $WHY $3

;;

PDF|pdf)

TraceLog "Original file in pdf format"

NAME="`basename $1 .pdf`"

if [ "${3,,}" == "report+" ] ; then

PDFIN=$TMPDIR/$NAME.report+.pdf

else

PDFIN=$TMPDIR/$NAME.report.pdf

fi

cp $1 $PDFIN

TraceLog "Using CreateFaxReport"

CreateFaxReport $PDFIN $QFILE $WHY $3

;;

TIFF|tiff|TIF|tif)

TraceLog "Using tiff2pdf"

NAME="`basename $1 .tif`"

if [ "${3,,}" == "report+" ] ; then

PDFIN=$TMPDIR/$NAME.report+.pdf

else

PDFIN=$TMPDIR/$NAME.report.pdf

fi

$TIFF2PDF -o $PDFIN $1

TraceLog "Using CreateFaxReport"

CreateFaxReport $PDFIN $QFILE $WHY $3

;;

esac

;;

\*)

TraceLog "Not invoked by 'notify', hence no fax report"

;;

esac

;;

esac

}

# A.3 – Systemeinstellungen

## A.3.1 - VMware ESXi 6.0

* *VMware-Version*: 11
* *Zugewiesener Arbeitsspeicher*: 4GB
* *Anzahl CPU-Kerne*: 4
* *Zugewiesene Festplatte*: 20GB (als einzelne Datei)
* *Netzwerk*: VMXNET-Switch des ESXI 6.0-Hosts
* *CDROM-Image*: Debian-8.4.0-x86\_64-minimal.iso (Beim Start verbunden)
* *Disketten*-*Laufwerk*: Aus Performancegründen entfernt

## A.3.2 – Debian 8.4.0 Systemeinstellungen

### A.3.2.1 – Debian Netzwerkkonfiguration

# The loopback network interface

auto lo

iface lo inet loopback

# The primary network interface

allow-hotplug eth0

iface eth0 inet static

address 192.168.1.200

netmask 255.255.255.0

gateway 192.168.1.1

dns-nameservers 8.8.8.8 8.8.4.4

### A.3.2.2 – SSH

*sshd\_config*: PermitRootLogin yes

*iptables:*

iptables - A INPUT -p tcp -m state --state NEW --source 192.168.1.61 --dport 22 -j ACCEPT

iptables -A INPUT -p tcp --dport 22 -j drop

# A.4 - Softwarekonfiguration

## A.4.1 - IAXModem-Konfiguration

device /dev/ttyIAX0 # devicefilename

owner uucp:uucp # user & group with access to the modem

mode 660 # filemode

port 4570 # communicationport

refresh 50 # seconds for refresh the modem

server 127.0.0.1 # server-ip

peername iaxmodem # name of the peer for asterisk

secret Pa§§w0rd # password for connection

codec alaw # used codec

## A.4.2 - Asterisk

### A.4.2.1 - extensions.conf

[fax-in]

exten => \_X.,1,Dial(IAX2/iaxmodem)

[fax-out]

exten => \_X.,1,Dial(SIP/620/${EXTEN})

[unauthenticated]

### A.4.2.2 - iax.conf

[general]

bindport=4569

bindaddr=0.0.0.0

disallow=all

allow=alaw

allow=ulaw

calltokenoptional=127.0.0.1/255.255.255.0

[iaxmodem]

type=friend

secret=Pa§§w0rd

port=4570

host=dynamic

context=fax-out

disallow=all

allow=alaw

requirecalltoken=no

jitterbuffer=no

trunk=no

### A.4.2.3 - sip.conf

[general]

context=unauthenticated

bindport=5070

bindaddr=0.0.0.0

localnet=192.168.1.200/255.255.255.0

nat=comedia

srvlookup=yes

allowguest=no

alwaysauthreject=yes

register => 620:Pa§§w0rd@192.168.1.11/620

[620]

type=friend

insecure=port,invite

nat=comedia

username=620

fromuser=620

fromdomain=192.168.1.11

secret=Pa§§w0rd

host=192.168.1.11

qualify=yes

context=fax-in

## A.4.3 - Monitoring

### A.4.3.1 - check\_faxq

#!/bin/sh

###### ATTENTION ATTENTION ATTENTION ATTENTION ATTENTION

# To run this script you have to set hfaxd.conf JobFormat like this

# JobFmt: "%-3j %3i %1a %6.6o %-12.12e %5P %5D %7z %Y %.25s"

###### ATTENTION ATTENTION ATTENTION ATTENTION ATTENTION

#

#

# Another Plugin for Nagios to monitor the hylafax-queue

# Written by T. Bruentjen (alte-schlonte at gmx dot com)

# Date: 2007/12/06

#

# Usage: ./check\_faxq -<warn> -<crit>

#

# Output:

# This plugin gives a number of faxes with errors

#

# Exit Codes:

# 0 OK The number of faxes in the queue are under the warning

# or under the critical threshold

# 1 Warning Number of faxes with errors in the queue above "warning" threshold

# 2 Critical Number of faxes with errors in the queue above "critical" threshold

# 3 Unknown One of the needed services are not running e.g. hfaxd,faxgetty,faxq

#

#CUSTOM VARIABLES

TEMPDIR=/tmp

#PARTIAL CUSTOM PART - CHECK IF HYLAFAX RUNS (depens on using Modem or CAPI)

#Unset the checks for modems or customize to fit your needs

#(Personally i prefer modems or cards which emulate them)

modems=8

modem\_count=0

daemon\_count=0

spool\_count=0

for i in `ps -e|grep fax`

do

case "$i" in

"faxgetty")

modem\_count=$((modem\_count+1));;

"hfaxd")

daemon\_count=$((daemon\_count+1));;

"faxq")

spool\_count=$((spool\_count+1));;

esac

done

if [ $modem\_count -ne $modems ]

then

echo $modem\_count" faxgettys running instead of "$modems

exit 3 #Unknown

fi

if [ $daemon\_count -ne 1 ]

then

echo $daemon\_count" hfaxd running instead of 1"

exit 3 #Unknown

fi

if [ $spool\_count -ne 1 ]

then

echo $spool\_count" Queue daemons running instead of 1"

exit 3 #Unknown

fi

#NON CUSTOM PART - check for Parameters

if [ "X$1" == "X" ] || [ "X$2" == "X" ]

then

echo "Usage: check\_faxq W C"

echo " W=Warning Level - Number of failed Faxes"

echo " C=Critical Level - Number of failed Faxes"

exit 128

fi

warn\_level=$1

crit\_level=$2

#NON CUSTUM NAGIOS PART

OK=0

WARNING=1

CRITICAL=2

UNKNOWN=3

#NON CUSTOM CHECK PART

local\_day=`date +%d`

local\_month=`date +%m`

file=`tempfile -d $TEMPDIR`

/usr/bin/faxstat -s -d > $file

filelength=`wc -l $file|cut -d' ' -f1`

counter=1

qcount=1

ecount=0

#Initial Setting of Queue Name

queue="SENDQ"

while [ $counter -le $filelength ]

do

sed\_arg='sed -n '$counter'p '$file

line=`$sed\_arg`

row1=`echo $line|cut -d' ' -f1`

#Check what the line looks like, either System or Message

case "$row1" in

"HylaFAX")

;;

"Modem")

;;

"JID")

;;

"")

qcount=$((qcount+1))

if [ $qcount -gt 2 ]

then

#Change the Queue

queue="DONEQ"

fi;;

\*)

line=`echo $line | tr -s [:space:]`

jobid=`echo $line|cut -d' ' -f1`

priority=`echo $line|cut -d' ' -f2`

jobstat=`echo $line|cut -d' ' -f3`

owner=`echo $line|cut -d' ' -f4`

number=`echo $line|cut -d' ' -f5`

pages=`echo $line|cut -d' ' -f6`

tries=`echo $line|cut -d' ' -f7`

date=`echo $line|cut -d' ' -f8`

time=`echo $line|cut -d' ' -f9`

error=`echo $line|cut -d' ' -f10`

error=`echo $error|tr -d [:space:]`

if [ "X$error" != "X" ]

then

old\_error=$error

error=${line##\* $error}

error\_day=`echo $date|cut -d'/' -f3`

error\_month=`echo $date|cut -d'/' -f2`

if [ $error\_day -eq $local\_day ] && [ $error\_month -eq $local\_month ]

then

echo -n $jobid" "$number" "$old\_error$error" "

#Count Errors for checking

ecount=$((ecount+1))

fi

fi;;

esac

#++

counter=$((counter+1))

done

#Delete the temp file

rm $file

#Now get the state

if [ $ecount -ge $crit\_level ]

then

local\_state=$CRITICAL

else

if [ $ecount -ge $warn\_level ]

then

local\_state=$WARNING

else

echo "Everything looks allright"

local\_state=$OK

fi

fi

#Get the status back to nagios

exit $local\_state

### A.4.3.2 - check\_hylafax

#! /usr/bin/perl -wT

use strict;

use Getopt::Long;

use vars qw($opt\_V $opt\_h $opt\_H $opt\_P $opt\_w $opt\_c $opt\_l $opt\_p $PROGNAME);

use lib "/usr/lib/nagios/plugins" ;

use utils qw(%ERRORS &print\_revision &support &usage);

use Net::FTP;

$PROGNAME = "check\_hylafax";

sub print\_help ();

sub print\_usage ();

$ENV{'PATH'}='';

$ENV{'BASH\_ENV'}='';

$ENV{'ENV'}='';

Getopt::Long::Configure('bundling');

GetOptions

("V" => \$opt\_V, "version" => \$opt\_V,

"h" => \$opt\_h, "help" => \$opt\_h,

"w=s" => \$opt\_w, "warning=s" => \$opt\_w,

"c=s" => \$opt\_c, "critical=s" => \$opt\_c,

"H=s" => \$opt\_H, "hostname=s" => \$opt\_H,

"P=i" => \$opt\_P, "port=i" => \$opt\_P,

"l=s" => \$opt\_l, "login=s" => \$opt\_l,

"p=s" => \$opt\_p, "password=s" => \$opt\_p,

);

if ($opt\_V) {

print\_revision($PROGNAME,'$Revision: 1.0 $');

exit $ERRORS{'OK'};

}

if ($opt\_h) {print\_help(); exit $ERRORS{'OK'};}

my $host = ($opt\_H ? $opt\_H : "localhost");

usage("Invalid host: $opt\_H\n") unless $host =~ /^[-.A-Za-z0-9]+$/;

my $port = ($opt\_P ? $opt\_P : "4559");

usage("Invalid port: $opt\_P\n") unless ($port > 1 and $port < 32767);

usage("Warning threshold not specified\n") if not $opt\_w;

my $warning = $opt\_w;

usage("Invalid warning threshold: $opt\_w\n") unless $warning =~ /^\d+$/;

usage("Critical threshold not specified\n") if not $opt\_c;

my $critical = $opt\_c;

usage("Invalid critical threshold: $opt\_c\n") unless $critical =~ /^\d+$/;

#($opt\_c) || usage("Critical threshold not specified\n");

#my $critical = $1 if ($opt\_c =~ /([0-9]{1,2}|100)/);

#($critical) || usage("Invalid critical threshold: $opt\_c\n");

my $sig=0;

my $ftp = Net::FTP->new($host, Port => $port);

if (not $ftp) {

print "ERROR: Can't connect to $opt\_h: $@\n";

exit $ERRORS{'UNKNOWN'};

}

if (not $ftp->login($opt\_l, $opt\_p)) {

print "ERROR: Can't login to $opt\_h: $@\n";

exit $ERRORS{'UNKNOWN'};

}

my @jobs = $ftp->ls("sendq");

$ftp->quit;

$sig = scalar(@jobs);

my $err = "OK";

if ($sig >= $critical) {

$err = "CRITICAL";

} elsif ($sig >= $warning) {

$err = "WARNING";

}

print "$err: $sig jobs in queue\n";

exit $ERRORS{$err};

sub print\_usage () {

print "Usage: $PROGNAME [-H host] [-P port] [-l login] [-p password] -w <warn> -c <crit>\n";

print "\n";

print "Required Arguments:\n";

print " -w, --warning==INTEGER\n";

print " generate warning state if job count is greater than this number\n";

print " -c, --critical==INTEGER\n";

print " generate critical state if job count is greater than this number\n";

print "\nOptional Arguments:\n";

print " -H, --hostname=STRING\n";

print " connect to this host. Default is localhost.\n";

print " -P, --port=INTEGER\n";

print " connect to this port. Default is 4559.\n";

print " -l, --login=STRING\n";

print " connect using this login name.\n";

print " -p, --password=STRING\n";

print " connect using this password.\n";

print "\n";

}

sub print\_help () {

print\_revision($PROGNAME,'$Revision: 1.0 $');

print "Created by Adam Fox, November 2004

This plugin reports how many jobs are in a HylaFAX fax queue.

";

print\_usage();

support();

}

### A.4.3.3 - commands.conf

object CheckCommand "check\_hylafax\_queue" {

import "plugin-check-command"

command = [ CustomPluginDir + "/check\_hylafax" ]

arguments = {

"-H" = "$address$"

"-w" = "$warning$"

"-c" = "$critical$"

"-l" = "$login$"

"-p" = "$password$"

}

}

object CheckCommand "check\_faxmodem" {

import "by\_ssh"

arguments = {

"-l" = "$by\_ssh\_logname$"

"-H" = "$address$"

"-C" = "$by\_ssh\_command$"

"-o" = "StrictHostKeyChecking=no"

"-E" = ""

}

vars.by\_ssh\_command = "/usr/lib/nagios/custom-plugins/check\_faxq -w $warning$ -c $critical$"

vars.warning = 5

vars.critical = 10

}

# A.5 - Projektplanung

