Inhaltsverzeichnis

1 Automatisierung

1.1 Einführung

- Arbeit handelt um Automatisierung
- Warum wurde dieses Thema für die Arbeit gewählt?
- Automatisierung von Programminstallationen
- Betreuung und Analyse eines gesamten zu Automatisierenden Prozesses
- Daraufsicht auf den Prozess der Automatisierung managen und arbeiten nach ISO bzw. ITIL Konformität Praxis 4.

Häufig gibt es gewisse Aufgaben oder Prozesse, welche keine großen Unterschiede in der Abwicklung aufweisen. Für jene kann sich mit der Anwendung von Automatisierung beschäftigt werden. Die generelle Anforderung an Automatisierung ist es Arbeits- oder Produktionsprozesse für den Menschen so durchzuführen, dass dieser nicht unmittelbar tätig werden muss, um Aufgaben zu verrichten. Spezifischer auf das Themenfeld der Projektarbeit bezogen, stellt sich die Aufgabe dar, mit Hilfe von Skripten Installationsprozesse auf ein Minimum von Zeitaufwand zu beschränken. https://m.bpb.de/nachschlagen/lexika/lexikon-der-wirtschaft/18743/automatisierung Das Skript soll also insoweit die Prozesse automatisieren, dass vom Nutzer lediglich ein paar Parameter angepasst werden müssen, die sich bei jeder Installation unterscheiden

Das Ziel der Arbeit ist es also neben der klassischen Dokumentierung, was in den Praxisphasen erarbeitet wurde, einen guten Überblick über die Herangehensweise, Analyse und das Potential von Automatisierung zu verschaffen.

2 Task Automation mit Powershell

Der Schwerpunkt dieser Praxisarbeit wurde auf die Implementierung von zu automatisierenden Prozessen in PowerShell gelegt. Es wird beginnend bei der Ideenfindung für einen solchen Prozess, bis zur abschließenden Einbindung, ein umfassender Überblick gegeben, wie Projekt und Aufgabe abgelaufen sind.

2.1 Aufgabenstellung

Als äußerste Frage, welche es zunächst zu beantworten galt, wurde die Folgende formuliert:

Wie können Prozesse mit Hilfe von Powershell weitestgehend automatisiert werden?

Aus jener Frage lassen sich mehrere Aufgaben ableiten. Einerseits kristallisiert sich daraus das Recherchieren und Verstehen von Prozessen, sowohl im übergeordneten Zusammenhang mit Anderen, als auch der Eigentliche an sich. Geht man beispielsweise von einer Programm Installation aus ist dies der Kernprozess, er hängt aber übergeordnet mit den Berechtigungen die auf dem Betriebssystem gelten zusammen. Andererseits spielt die Aufgabe des strukturierten Dokumentieren und Vorgehen bei der Umsetzung eine große Rolle.

Nicht zuletzt muss sich mit der Verwendung von Powershell, vielmehr der Implementierung in der Powershell ISE, befasst werden. Auf die Aufgabe des Implementierens wurde in diesem Teil der Arbeit das Hauptaugenmerk gelegt.

Ziel der Aufgabe ist es also den gewählten Prozess weitestgehend zu automatisieren. Durch die implementierten Skripte soll dabei eine schnellere und qualitativ gleichbleibende Installation, sowie Konfiguration der Software ermöglicht werden.

2.2 Einordnung der Aufgabenstellung in übergeordnete Prozesse

Ausgewählt wurde diese Aufgabe in der Abteilung Virtualisierung, da bei Digitalen Arbeitsplätzen die Automatisierung solcher Prozesse eine generell recht übliche Prozedur ist. Zudem ist das Aufsetzen und die Verwaltung, insbesondere von virtuellen Desktops, meist nur minimal Abweichend, wenn nicht sogar nahezu identisch in der Durchführung.

Demnach kann das Unternehmen durch die Automatisierung kosten sparen und bietet zudem die Verlagerung von Ressourcen. Ganzheitlich gesehen können durch die gewonnene zeitliche Ersparnis, die Arbeitskräfte in die Umsetzung komplexerer Aufgaben gesetzt werden.

Arbeiten im AD soll vereinfacht werden, dass der Nutzer nicht mehr auf einzelne Schritte zurückgreifen muss.

2.3 Praktische Umsetzung

Zunächst soll im Analyseteil der Aufgabe ein grundlegendes Verständnis zu den ablaufenden Prozessen aufgebaut werden. Hierfür wurde eine kurze Liste mit den in der Abteilung verwendeten Produkten ausgehändigt:

- Citrix Cloud Connector (CC)
- Powershell
- PowerShell ISE

Festgelegt wurde sich auf die Installations Automatisierung des Citrix Cloud Connectors.

In den folgenden Unterkapiteln soll ein genereller Überblick über die in dem Projekt verwendete Software geschaffen werden. Dabei soll dem Leser ein

klarer Nutzen und Mehrwert des Produkts, sowie eine kurze Heranführung über die verwendeten Funktionen, in dem Projekt, erläutert werden.

2.3.1 Citrix Cloud Connector

Der Citrix Cloud Connector ist ein Produkt, das die Verbindung zu einem Citrix Cloud-Server ermöglicht. Dieser kann wiederum einen virtuellen Desktop hosten und gewährt den Zugriff von anderen Computern auf diese Virtuelle Maschine. Dabei sei angemerkt, dass von dem Nutzer auch eigene Ressourcenstandorte auf Serverseite integriert werden können. Der Vorteil dabei ist, das von überall mit nahezu jeder Art von Hardware ein solches System genutzt werden kann. Die Erreichbarkeit durch den Faktor Cloud ist immer gegeben, sobald der Nutzer eine Internetverbindung zur Verfügung hat. Außerdem kann durch die Netzwerkstruktur der Zugriff von nahezu jedem Gerät, welches vom Cloud Connector unterstützt wird, stattfinden. Dieses Produkt wird demnach von Atos als Drittanbieter vertrieben. Es wird sich sowohl um die Einbindung als auch den Erhalt nach Launch, in From von SLAs, angeboten. [1]

Allgemein ist das Netzwerk so aufgebaut, dass es Virtuelle Apps und Desktop-Services gibt, die einem Nutzer zur Verfügung gestellt werden sollen. Diese Apps und Services werden von einem Ressourcenblock, der je nach Bedarf extern von Drittanbietern oder auch vom Anbieter selbst direkt bezogen werden kann, bereitgestellt. Der Cloud Connector bietet dabei die Schnittstelle zwischen Nutzer und Ressourcen. Er entnimmt also die benötigten Ressourcen und stellt diese dem Nutzer auf seinen Endgeräten zur Verfügung.[2]

Funktionen des Cloud Connectors sind dabei wie bereits genannt Apps und Virtuelle Maschinen, aber auch die Verwaltung und Verwendung des Active Directories, Store Front und Endpoint Management. Er kann also sehr flexibel genutzt werden, um einzelne Funktionen in die Cloud zu verlagern.

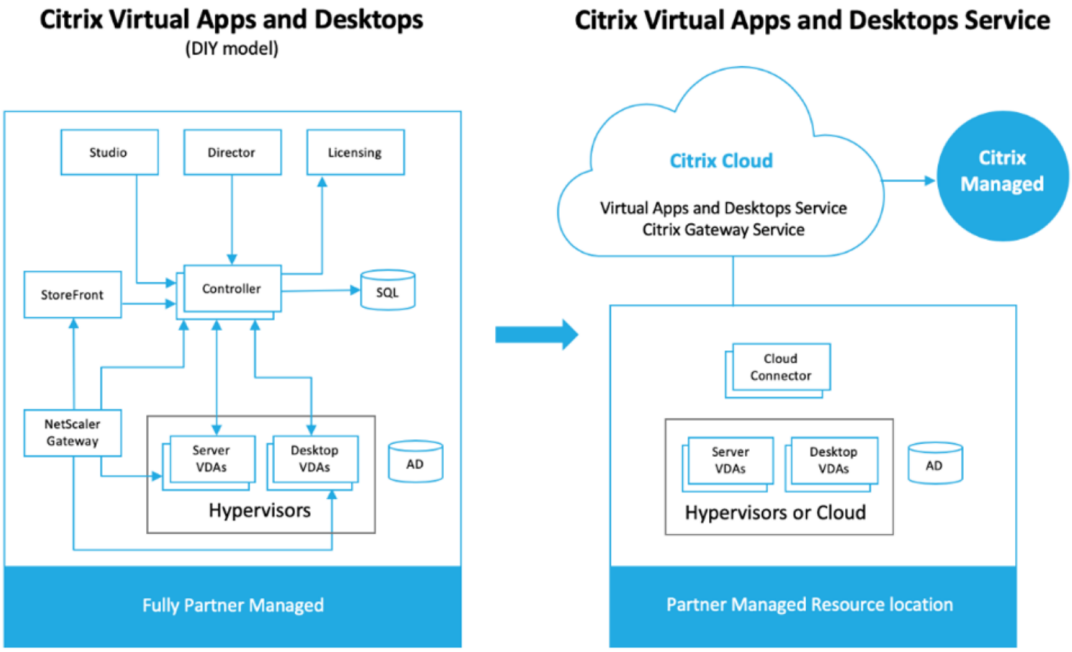


Figure 1: Citrix Cloud Connector Möglichkeiten

Der Grafik zu entnehmen sind hierbei die Unterschiede zwischen einer komplett Bereitstellung des Produkts links und rechts das Modell mit eigenen Ressourcen. Letzteres ist das Modell welches von der Abteilung genutzt wird.

2.3.2 Store Front

Store Front gehört zu den Citrix Produkten und arbeitet in Verbindung mit dem Citrix Cloud Connector. Dieses Produkt ist ein Service, der die Verwaltung von virtuellen Desktops und virtuellen Workstations ermöglicht. Es bietet Single-Sign-On-Zugriff (SSO) auf Anwendungen an und Desktops auf Basis des Citrix NetScaler Gateways. Dementsprechend enthält es weitere Sicherheitsmerkmale wie die Smartcard-Authentifizierung. Im Software Development Kit (SDK) wird die Anpassung von Benutzeroberflächen und dem App-Deployment durchgeführt. Als Beispiel dafür wäre das Laden von geschäftskritischen Anwendungen nach der Anmeldung zu nennen. [3]

2.3.3 PowerShell

Zur Umsetzung des Skripts wurde das plattformübergreifende Framework PowerShell verwendet. Es ist eine moderne Befehlsshell, die aus verschiedenen Vorteilen anderer Shells entwickelt wurde. Ein Vorteil von PowerShell ist das Akzeptieren und Zurückgeben von .NET-Objekten. [4]

In Bezug auf Skriptsprachen wird sie gerne als Standard zur Konfiguration und automatisierten Verwaltung von jeglichen Systemen genutzt. Dadurch ist sie ideal für die Aufgabe der automatisierten Installationskonfigurationen geeignet.

Für die spätere praktische Anwendung sei zu beachten, dass der Nutzer des Skripts auf dem Zielsystem Administrationsrechte benötigt, um im Vorhinein jeglichen Berechtigungsproblemen präventiv aus dem Weg zu gehen. Zudem sollte das System unter einem Betriebssystem der Firma Microsoft laufen. [5] Sind diese Kernaspekte eingehalten kann das Ausführen von Skripten gewährleistet werden.

2.3.4 PowerShell ISE

Die ISE (Integrated Scripting Environment) ist eine Hostanwendung für PowerShell. Ausgeführt wird diese Anwendung durch den cmdlet ise. Sie bietet eine grafische Oberfläche und dient der Skript Bearbeitung. Durch beispielsweise Syntaxhighlighting sorgt die ISE für eine übersichtlichere Einsicht in das Skript. Funktional bietet sie einen Debugger, Skriptvervollständigung und eine selektive Skriptausführung. [6]

Diese funktionalen Vorteile können jedoch durch Erweiterungen noch weiter verbessert werden. Als nächstes galt es also eine Erweiterung, mit dem Schwerpunkt Debugger, zu finden. Diese Erweiterung wird im folgenden Kapitel erläutert.

2.3.4.1 ISE Steroids

Bei der Recherche zu einer passenden Erweiterung im Aspekt Debugger kam Power GUI und ISE Steroids am häufigsten auf. Leider wurde ersteres nicht mehr aktiv unterstützt.

Um das Erstellen des Skripts weitestgehend zu vereinfachen, wurde somit auf das Plugin ISESteroids zurückgegriffen. Diese Erweiterung kann durch einen cmdlet, in PowerShell, heruntergeladen und installiert werden. Sobald die ISE gestartet wurde, kann über den Befehl Start-Steroids das Plugin ausgeführt werden. Dies erweitert die ISE um viele verschiedene Funktionen und Einstellungsmöglichkeiten. Beispielsweise können Funktionen einfacher erzeugt, Skriptteile können durch automatische Vervollständigung gebildet und vorgefertigte Funktionen können direkt erzeugt werden. Die jedoch wichtigste Funktion, die für eine gute und schnelle Implementierung geboten wird, ist der vollwertige Debugger. Mit ihm kann das Skript an gewissen Stellen angehalten und Variablenwerte überprüft werden. Einerseits wird dadurch ein klareres Verständnis über das Skript ermöglicht und andererseits lassen sich logische Zusammenhänge schneller Nachvollziehen. [7]

2.3.5 Vorbereitungen

Zur besseren Vorbereitung, der späteren Implementierung, wurde sich mit einem Entwickler aus der Abteilung vernetzt. Dieser zeigte mir einige praktisch bereits umgesetzte Projekte und händigte Skript Beispiele aus, anhand derer sich orientiert, sowie weitergebildet werden konnte. Eines der vorgegebenen Skriptteile konnte spezifisch für die eigene Aufgabe mit einbezogen werden. Die Inhalte des vorgegebenen Skripts waren allgemeine Installation und Download cmdlets des Cloud Connectors.

Nach dem Austausch setze ich mich mit meinem Betreuer in Verbindung, um die Implementierung und weiteres Vorgehen zu planen. Wie bereits gesagt lag

das eigentliche Installationsskript bereits vor, daran mussten nur kleine spezifische Optimierung für die Unternehmensumgebung vorgenommen werden. Diese Aufgabe wurde jedoch zurückgestellt, da sie zunächst nicht elementar war.

Es wurde gemeinsam eine Excel Tabelle mit den zu erledigenden Aufgaben erstellt. Zu den einzelnen Aufgaben sollten klare Termine festgelegt werden. Um diese zu eindeutigen Zeitpunkten erledigt zu haben. Dabei ging es zunächst nicht um die endgültige Fertigstellung und Abgabe der Aufgabe, sondern viel mehr um das gewähren einer besseren Übersicht und klareren Strukturierung des weiteren Vorgehen. Bezogen auf die Ebene des Projektmanagements kann man dabei von festgelegten Sprints reden. Ebenfalls wurde sich zu Daily Huddles verabredet, um sich stetig mit dem Betreuer auszutauschen.

Die Tabelle sieht wie folgt aus:

Aufgabe	Wer	Status	Zieldatum
Erstellung Generalisierung Script zur Übergabe von Parametern.	Löhr	[onGoing]	<24.01>
Erstellung / Überarbeitung Script Basisinstallation Cloud Connectoren	Löhr	[offen]	<04.02>
Erstellung Script Cloud Connector Preferences Check	Löhr	[offen]	<18.02>
Erstellung Script Storefront Preferences Check	Löhr	[offen]	<18.02>
Erstellung Script Storefront Basis Installation	Löhr	[offen]	<25.02>
Erstellung Script Storefront Konfiguration	Löhr	[offen]	<25.02>
Erarbeitung Prozessablauf Diagramm	Löhr	[onGoing]	
Erarbeitung Dokumentation	Löhr	[onGoing]	

2.4 Logischer Aufbau

Wie der Tabelle zu entnehmen ist wurde sich zuerst mit der Übergabe der Installationsparameter beschäftigt. Diese Aufgabe beanspruchte in der Praxisphase am meisten Zeit. Die einzige Vorgabe hierfür war es eine Datei mit den Installationsparametern zu erstellen. Diese Datei sollte entweder das Format XML oder JSON haben. Die Parameterdatei darf später für den Endnutzer die einzige zu editierende Datei sein. Ausgelesen werden diese Parameter aus der Datei in dem Hauptskript, welches die Installation dann durchführt. Je nach Art der Installation gibt es zudem einen Source Ordner mit den Installationsdateien. Im Idealfall soll zudem noch eine log-Datei erstellt werden um den gesamten Prozess zu Dokumentieren. Des Überblicks halber wurde ein Technisches Ablauf Diagramm erstellt.

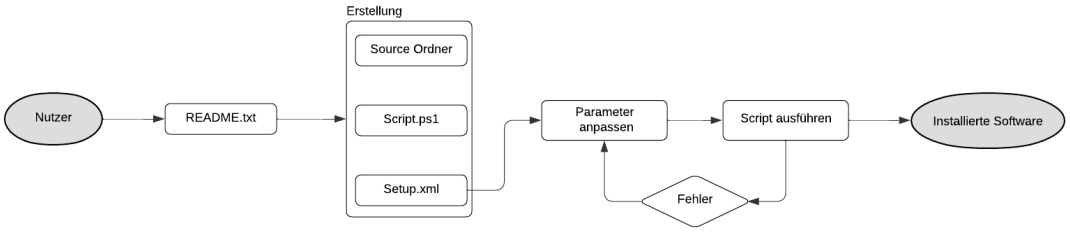


Figure 2: Technisches Ablauf Diagramm

Dieses Ablaufdiagramm soll final eine maximale Übersicht des implementierten Projekts ermöglichen und dem Nutzer dazu dienen die später gegebene Ordnerstruktur besser zu verstehen. Es ist im Verlauf des Projekts stetig ergänzt und angepasst worden.

Zum besseren Verständnis wurden ebenfalls eine ReadMe Datei hinterlegt. Ergänzend ist noch zu Sagen, dass dem Technischen Diagramm die Idee des Parameter Eintragens visualisiert wurde. Fehler die der Nutzer beim Eintragen der Parameter in die Datei macht sollen abgefangen werden und zum korrigieren der Parameter forcieren.

2.5 Automatisierung Citrix Cloud Connector

Dieser Abschnitt thematisiert die gesamte Implementierung des Skripts. Durchweg ist der Aufbau der einzelnen folgenden Abschnitte mit der angeführten Idee und Erklärung, warum sich genau für diese Art der Umsetzung entschieden wurde, strukturiert.

Nachdem ein grundlegendes Verständnis mit Powershell aufgebaut wurde, ging dies über in die Auswahl des Dateityps der Parameterdatei. Festgelegt wurde sich auf das Format XML. Durch die hierarchische Strukturierung der Daten in einer XML kann eine klare Übersicht geschaffen werden. Zudem lassen sich Daten zugänglich und individuell, durch die simple Markup Language, bearbeiten. [8]

2.5.1 Setup.xml

Die Bezeichnung Setup.xml wurde gewählt, um dem Nutzer zu vermitteln, dass dies der Ort zur Anpassung der wesentlichen Parameter für die Installation ist. Der Aufbau der ersten XML Version wurde zuerst wie folgt gewählt:

```
<Setup>
    <CloudConnector>
    <CTXCloudCustomerID>AtosInforma2</CTXCloudCustomerID>
    <CTXCloudClientId>afbe8291-581f-4ff8-z465-e260e062f32e</CTXCloudClientId>
    <CTXCloudClientSecret>Si6XTSBsCJAQbQjweQlyYKyQFuw==</CTXCloudClientSecret>
    <CTXCloudResourceID>e749bfbc-7777-4bd4-8712-fe318ba90974</CTXCloudResourceID>
    <Vendor>Citrix</Vendor>
    <Product>Cloud Connector</Product>
    <PackageName>cwcconnector</PackageName>
    <InstallerType>exe</InstallerType>
    </CloudConnector>
</Setup>
```

Setup bildet den Wurzelknoten, auf den die einzelnen Installationsverzeichnisse folgen. Über diesen Knoten soll später größer skaliert werden (Install, Config, Setup, Remove). Geht man nun in das Element CloudConnector lassen sich acht Attribute finden. In diesen Attributen werden die Werte / Parameter, welche individuell für jede Installation angepasst werden müssen, hinterlegt.

Bei einer größeren Skalierung muss lediglich ein neuer Hüllen Elternknoten erstellt werden.

Dieses Schema langte zunächst und bestätigte sich als praktikabel für die Tests, bei der Parameterübergabe an das CCC Skript.

Während dem Testen wurde eine dynamische Anpassung für die Dateien in der XML integriert. Diese Dateien waren die Store Front Installations exe und ein Backup mit dem Ablageort (URL).

```
<Setup>
        <PackageName>CitrixStoreFront-x64.exe</PackageName>
        <ConfigZip>backup</ConfigZip>
        <HostBaseURL>https://serverB.example.com</HostBaseURL>

        <CloudConnector>
            <CTXCloudCustomerID>AtosInforma2</CTXCloudCustomerID>

        ...
</Setup>
```

Im späteren Verlauf, als sich Gedanken mit der Skalierung gemacht werden konnte, ist man auf zwei Probleme gestoßen. Zum Einen kam auf, dass das Parsen mit Powershell sich als etwas komplizierter für das anfangs gewählte Schema darstellte. Die Problematik hierbei war, dass das Ausgeben der Knotennamen wesentlich umständlicher war, als das der Werte selbst, die in dem Knoten hinterlegt waren.

Zum Anderen kam die Idee auf, das Skript später so zu generalisieren, dass aus einer Setup.xml nicht nur bereits genannte einzelne Konfigurationen hinterlegt werden sollen, sondern auch Installationen mit Backups und ganze Powershell Skripte.

So wurde sich darauf geeinigt, dass mit generalisierten Blöcken gearbeitet werden soll. Dabei bestehen die einzelnen Blöcke aus eine der zwei Kategorien **< Script >** oder **< Parameter >**. In dem jeweiligen Parameter Knoten ist dann als erstes die Art **< Type >** hinterlegt. Hierbei kann zwischen den Bereits vorher erläuterten Kategorien (Install, Config, Setup, Remove) gewählt werden. Der

Weitere Aufbau besteht dann aus dem bereits bekannten Schema.

Die XML wurde also final zusammengeführt und wie folgt angepasst:

```
<Parameter>
    <Type>Install/Config</Type>

    <ConfigZip>backup</ConfigZip>
    <HostBaseURL>https://serverB.example.com</HostBaseURL>

    <PackageName>CitrixStoreFront-x64.exe</PackageName>
    <PackageParameter>-silent -q -quiet</PackageParameter>

    <CTXCloudCustomerID>AtosInforma2</CTXCloudCustomerID>

    ...
</Parameter>
```

Die Idee mit dem Integrieren von verschiedenen Arten von Skripten in dieser XML ist somit umsetzbar, jedoch langte leider die Zeit nicht eine Routine zu schreiben, um ein solches aus der XML in Powershell zu laden.

2.5.2 Main Skript

Im Main Skript vorhanden sind Funktionen zur XML-Zerlegung, dem Downloaden der Installationsdatei und Anweisungen zur Installation im Grundsätzlichen. Der genaue Aufbau wird demnach in den folgenden drei Unterkapiteln erläutert.

2.5.2.1 XML-Parser Zuvor wurden in der Setup.xml alle Attribute mit Werten festgelegt. Die Aufgabe des XML-Parsers ist es nun, sowohl die Attributnamen, als auch deren Werte auszulesen. Der Fokus lag also zuerst auf dem Auslesen der XML-Daten im Allgemeinen. Gelöst wurde dies durch eine foreach-Schleife, die über XML-Pfadbeschreibungen durch die gesamte XML iteriert und die gesammelten Knoten in einem Array speichert. Dieses Array wird dann über eine for-Schleife ausgelesen, um die einzelnen Werte der Knoten in einer

neuen Liste zu speichern. Diese Liste wird dann später bei der Installation weiterverwendet.

Nach der Kernimplementierung wurde sich damit beschäftigt das Skript zu optimieren. So wurde sich um die Fehlerbehandlung und Datenprüfung gekümmert. Einen genaueren Einblick hierfür kann sich im Kapitel Datenprüfung und Fehlerbehandlung eingeholt werden. Dasselbe gilt für das Schreiben und Erzeugen der log-Datei. Fortgefahren wurde dann damit, dass Skriptteile, die eine spezifische Teilaufgabe verfolgen, in einzelnen Funktionen gebündelt wurden. Dies soll die Erstellung von späteren (anderen) Installationen erleichtern, da man gewisse Funktionen universell anwenden kann.

Begonnen wird mit der Funktion LesenderEingabe{}. Zuerst wird die einzulesende XML-Datei ausgewählt. Abgespeichert muss diese dafür in dem aktuellen Dateiverzeichnis sein. Also jenes in dem der Nutzer das Skript abspeichert und ausführt. Sofern eingehalten wird der gesamte XML-Inhalt in der Variable $xmlFile gespeichert. Über diese Variable wird dann mit Hilfe einer Pfadbeschreibung auf den Installationsknoten verwiesen. Der Knoten beinhaltet den spezifischen Input mit den Attributwerten, die für die Installation wichtig sind. Die ausgelesenen Attribute werden mit ihren Werten als Liste dann in $xmlElements abgespeichert und von der Funktion ausgegeben.

Als nächstes werden von der Funktion Get-Nodes{} die in $xmlElements gelisteten Attribute Ausgelesen und in dem Array $returnElemente abgespeichert. Dies dient dazu um in der darauf folgenden Funktion Ausgebenvon-Nullwerten{} die einzelnen Attribute dahingehend zu überprüfen ob diese mit Werten festgelegt sind. Ist dies nicht der Fall so bricht das Skript ab und fordert den Nutzer dazu auf die Attribute mit fehlenden Werten zu belegen.

Sind alle Attribute mit Werten belegt geht es in die letzte Funktion des Parsers Get-NodeData{}. Get-NodeData{} ist von der Struktur ähnlich zu Get-Nodes{}. Es werden hier jedoch nun die einzelnen Werte der Attribute in $returnValue in

Form von einer Liste #text abgespeichert. Dabei ist zu beachten, dass die reinen Werte nur durch den Befehl .'#text' aus dem Array gefiltert werden. So können durch den einfachen Aufruf von Get-Nodes{} überall im Skript die Werte der Attribute genutzt werden.

2.5.2.2 Start-Main

Der zentrale Teil des Skripts ist ein individuell anzupassende Teil. Dieser besteht lediglich aus der Start-Main Funktion. Jener Abschnitt muss je nach Art der Software, die Installiert werden soll, für die Software angepasst werden. Entnommen werden können die vorgefertigten Blöcke der einzelnen Installationen aus der Start Datei.

Über den Abschnitt Start-Main wird das Skript im Ganzen ausgeführt. Der Aufbau basiert auf mehreren if-Anweisungen, bei denen jeweils überprüft wird, ob jede Funktion des XML-Parsers Werte aufweist. Sind die Abfragen durchweg erfüllt, werden die zur Installation relevanten Variablen benannt und mit den geparsten Werten belegt.

2.5.2.3 Download / Installation

Sowohl der Abschnitt Download als auch der darauffolgende Teil der Installation (im Skript) beinhalten Code, welcher bereits existierte und nur minimal zur Zusammenführung mit dem Parser angepasst werden musste. Dieser Skriptblock muss je nach Software, die heruntergeladen werden soll mit der Installation angepasst werden. Durch ihn soll dafür gesorgt werden, dass die Software heruntergeladen wird und die Installation stattfinden kann.

Elementar beginnt die Downloadsektion mit der Deklarierung der Variable $DownloadLocCloudConnector. Sie enthält den spezifischen Downloadlink mit bezogenen Daten aus der XML. Darauf wird $TargetLocCloudConnector mit dem Verzeichnis, in dem die Installations .exe geladen werden soll, erzeugt. Überprüft wird dann, ob die Webanfrage erfolgreich ist. Ist dies der Fall, so

wird der Download gestartet. Andernfalls wird der Download übersprungen.

Abschließend wird dann die Installation mit der Deklarierung von $Arguments eingeleitet. Dazu wird der Start-Process cmdlet von PowerShell verwendet. Ist der Code 0 war die Installation erfolgreich. Ansonsten wird der Nutzer durch eine Terminalausgabe über die fehlerhafte Installation und den Statuscode hingewiesen.

2.5.2.4 URL Check Der in der Download Sektion definierte Downloadlink soll zuvor überprüft werden. Bei der Überprüfung soll die Erreichbarkeit und Dauer zurückgegeben werden. Es handelt sich also im Prinzip um einen klassischen Ping. Implementiert werden sollte dieser Check um sicher zu gehen, dass sich das Skript im Falle eines nicht erreichbaren oder fehlerhaften Links nicht aufhängt, sondern dem Nutzer eine Ausgabe liefert.

Umgesetzt wurde diese Ping Routine mit einer do until-Schleife. Diese Schleife wird solange ausgeführt, bis der Ping erfolgreich ist oder nach 5 Sekunden bei keiner Antwort abgebrochen wird.

Um eine klarere Aussage über die Erreichbarkeit der Website zu erhalten wird zudem noch ein Statuscode abgefragt. Dieser Statuscode sollte immer 200 sein, wenn die Website erreichbar ist. Anhand dieses Statuscodes soll dann später der Download gestartet oder abgebrochen werden.

Die Funktion des URL Checkers wurde letztendlich jedoch nicht in das Finale Skript integriert, da sich im Verlauf des Projektes herausstellte, dass davon auszugehen war, dass lediglich ein paar wenige Links genutzt werden und diese generell immer erreichbar seien. Außerdem würde das einbinden dieser Fehlerüberprüfung in das Hauptskript zu einer größeren Unübersichtlichkeit führen und dadurch die Verständlichkeit des Skripts verschlechtern.

Da aber dennoch Arbeitszeit in diese Aufgabe geflossen ist wollte ich dieses Teilskript erwähnt haben. Zu finden ist es ebenfalls im Github-Repository unter

dem Ordner ‘Project’.

2.6 Formale Komponenten

Dieses Kapitel dient der Skriptvollständigkeit und beschäftigt sich mit dem Schreiben einer Log Datei, dem Fehlerabfangen und die Erläuterung der Datenüberprüfung.

2.6.1 Log Datei

Durch die Dokumentierung gewisser Prozesse eines Programmablaufs kann eine Log-Datei erzeugt werden. Dabei sollte darauf geachtet werden, wie sinnvoll die gesammelten Daten sind und welchen Nutzen sie für beispielsweise spätere Analysen bieten.

Diese späteren Analysen können sowohl während der Entwicklung, als auch nach Fertigstellung des Skripts erfolgen. Während des Entstehungsprozesses dienen sie der Identifikation und Rückverfolgung von Fehlern (Debugging). Nach der Fertigstellung wiederum eher dem Monitoring und Nachvollziehen was exakt passiert. [9]

Eine Einbindung einer Schnittstelle, zur zentralen Analyse der Log Datei, wie zum Beispiel bei einem SIEM in der Netzwerk Security, war nicht nötig. Aus dem Grund da es bei einer solch überschaubaren Skriptgröße nicht sonderbar viel Information zu sammeln gibt. Zudem wäre dies aus zeitlichen Gründen und dem Aspekt der Ressourceneffizienz nicht sinnvoll.

Zum Erstellen der Logfile wurde mit dem cmdlet Start-Transcript, der in der PowerShell Bibliothek enthalten ist, gearbeitet. Beim Verwenden der cmdlets muss zunächst ein Verzeichnis angegeben werden, in dem die log-Datei gespeichert werden soll. Bei Beendigung des Skripts sollte das Schreiben der log-Datei abgeschlossen werden. Mit Hilfe von unterschiedlichen Parametern kön-

nen bestimmte Ereignisse spezifisch in die Datei mit eingetragen werden.[10]

2.6.2 Datenprüfung / Fehlerbehandlung

Als erstes wurde der Dateipfad auf Lese- und Schreibrechte geprüft. Umgesetzt wurde dies mit Hilfe von try-catch Blöcken. Es wird versucht (try) das aktuelle Verzeichnis, in dem das Skript gestartet wurde zu lesen und zum Anderen die log-Datei in jenes Verzeichnis zu schreiben. Ist dies nicht möglich (catch), wird der Nutzer über das Terminal auf fehlende Rechte hingewiesen. Weitere try-catch Blöcke wurden beim Abrufen der XML-Datei und Daten verwendet.

Ist der Abruf der Datei oder der Werte in der XML nicht möglich, wird dies ebenfalls über das Terminal mitgeteilt. Zudem wird bei einer fehlerhaften XML-Dateiabfrage dem Nutzer der spezifische Exception Code ausgegeben. Dies wurde lediglich an dieser Stelle im Skript ausprobiert. Da jedoch davon auszugehen ist, das der Exitcode nicht für jeden Nutzer verständlich ist, wurde im Gesamten mit den bereits genannten Terminal Text ausgaben, die über den cmdlet Write-Host ausgegeben werden, gearbeitet.

Die Überprüfung auf Nullwerte, der entnommenen Daten, findet in der Funktion AusgabeVonNullwerten{} statt. Umgesetzt wird dies mit einer if-Abfrage auf Nullwerte der Daten, welche in Form eines Arrays abgespeichert wurden.

2.6.3 README.txt

In der Readme.txt werden die Installationsanleitungen zu den einzelnen Schritten des Skripts ausgegeben. Der Nutzer soll sich damit durch den gesamten Prozess durcharbeiten können. Orientiert wurde sich beim Erstellen der Readme.txt an die Anleitungen aus der Dokumentation von Great Learning.

Dieses Skript wird dazu verwendet die Installation des CitrixCloudConnec-

tors weitgehend zu automatisieren.

Dafür müssen einzelne Parameter zur Installation konfiguriert werden.

Vor der Ausführung: - Ordner Source anlegen (enthält Installations exe) - Setup.xml nach Schema erstellen - Parameter mit personalisierten Werten festlegen

```
<Setup>
    <CloudConnector>
    <CTXCloudCustomerID>AtosInforma2</CTXCloudCustomerID>
    <CTXCloudClientId>afbe8291-581f-4ff8-z465-e260e062f32e</CTXCloudClientId>
    <CTXCloudClientSecret>Si6XTSBsCJAQbQjweQlyYKyQFuw==</CTXCloudClientSecret>
    <CTXCloudResourceID>e749bfbc-7777-4bd4-8712-fe318ba90974</CTXCloudResourceID>
    <Vendor>Citrix</Vendor>
    <Product>Cloud Connector</Product>
    <PackageName>cwcconnector</PackageName>
    <InstallerType>exe</InstallerType>
    </CloudConnector>
</Setup>
```

Datei und Ordner Bezeichnung darf nicht abweichen!

2.6.4 Prozessdiagramm

Zur Vorstellung eines Kunden und ebenfalls dem besseren Verständnis des Nutzers wurde zudem noch ein Prozessdiagramm angefertigt. In dem folgenden Prozessdiagramm kann der Ablauf des gesamten Arbeitsprozesses abgearbeitet werden. Unterteilt wird der Ablauf hier in drei Abschnitte System, Installation und Software. Es folgen lediglich auf das System und die Installation Aktionen und Ereignisse, da die Software als finaler Abschnitt zu betrachten ist.

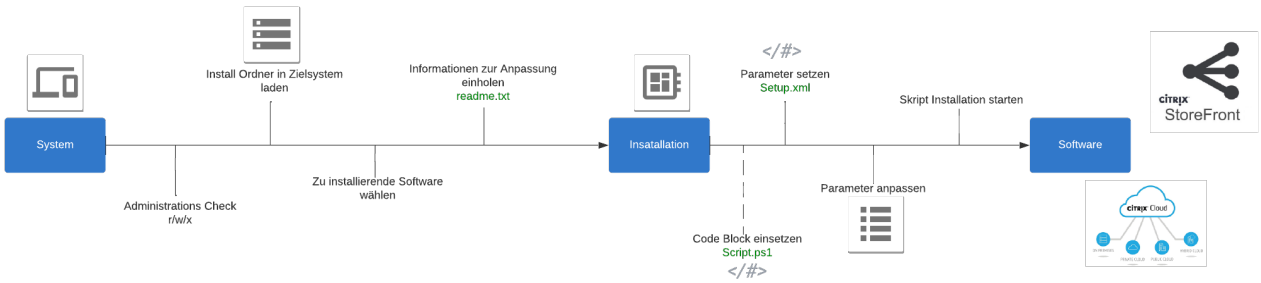


Figure 3: Prozessdiagramm

Beginnt man nun bei dem System, so ist der erste Schritt die Überprüfung der Administrationsrechte. Der Nutzer des Skripts sollte schauen, dass er Lese-, Schreib- und Ausführrechte auf dem System hat. Diese administrativen Rechte sind notwendig, damit der Nutzer jegliche, für die Installation notwendigen, Aktionen auf dem System durchführen kann.

Nach diesem Check kann der heruntergeladene Installationsordner in das Zielverzeichnis geladen werden. Im nächsten Schritt wird die zu Installierende Software gewählt und sich zum weiteren Vorgehen über die readme.txt informiert.

Dann kommt es zu dem eigentlichen Teil der Installation. Dort wird zuerst der Codeblock, der für die Installation notwendig ist, ausgewählt und in das Script.ps1 eingesetzt. Nun können die Parameter, die für die Installation wichtig sind in der Setup.xml gesetzt und angepasst werden. Final wird dann das Skript ausgeführt und die Installation automatisch durchgeführt. Als finales Produkt erzeugt ist dann die gewünschte Software installiert.

2.6.5 Kommentare

Die klassische Herangehensweise beim Programmieren ist es Kommentare passend zu implementierten Code zu verfassen. Der Vorteil darin liegt augenscheinlich darin, dass der Code schneller verständlich ist und man sich auch noch nach einer längeren Zeit einfach in den Code wieder einarbeiten kann. Dabei werden Zeilen oder Funktionen an Code einzeln kommentiert.

Von Powershell aus gibt es die sogenannte kommentarbasierte Hilfe. Kommentiert man seine Funktionen nach diesem Schema, so ermöglicht dies eine praktikable Erweiterung. Man kann sich die Möglichkeiten der Funktion, sowie die Bedienung, in Bezug auf die Parameterübergabe, ganz einfach mit dem Get-Help Command anzeigen lassen [11].

```
Get-Help inputXML -full
```

Diese Erweiterung galt es testweise zu implementieren und auszuprobieren:

```
function inputXML
{
  <#
      .SYNOPSIS
      Setup.xml gets read
      .DESCRIPTION
      Setup.xml gets read, has to be in the current folder from which
      the script is executed
      .INPUTS
      No Inputs needed file Path static
      .OUTPUTS
      All elements which are read from the selected node are outputted.
      .NOTES
      The selected data path has to be adjusted by the prefered data.
  #>

...
```

2.6.6 GUI

Um dem Nutzer den Gebrauch der Software noch einfacher zu machen gab es zu diesem Projekt zuletzt noch die Aufgabe sich an einem grafischen Benutzer Interface zu versuchen. Hierzu wurden die Kenntnisse aus der Ersten Praxisphase und dem Programmieren Modul zwei angewendet. Dies Bot sich an, da in beiden Projekten bereits mit der Erstellung einer GUI gearbeitet wurde.

Angefangen wurde mit der Einbindung spezifischer .NET Bibliotheken zur Erzeugung von grafischen Elementen. Der Grundaufbau ist der folgenden Grafik zu entnehmen:

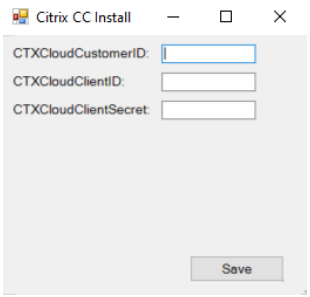


Figure 4: GUI Aufbau

Es wurde eine Main Form erstellt, welche die Hauptseite des Programms

darstellt. Auf der Form wurden Label zur Kennzeichnung der einzelnen Parameter eingetragen. Hinter die Label wurden die Textboxen gesetzt, in welchen der Nutzer die Werte für die einzelnen Parameter eingeben kann. Die Textboxen wurden mit einem Button *Save* verbunden, welcher die Eingabe überprüft und in einem Array *$data* zwischenspeichert.

Nachdem dieses Kernkonzept ausgearbeitet und funktionsfähig war, gab es noch die folgende Funktionen zu erledigen. Je nach Art der Konfiguration sollen die Parameter Label dynamisch angepasst werden. Zudem soll ebenfalls die Menge der Angezeigten Label und Textboxen dynamisch sein. Zuletzt muss noch die Übergabe der gesammelten Werte in die Setup.xml Datei stattfinden.

Als Erstes wurde sich an das dynamische erzeugen der einzelnen Label und Textboxen gesetzt, da dies viel redundanten Code mit sich brachte, demnach aus Effizienzgründen gekürzt werden konnte. Die Idee bestand darin, dass die Parameternamen in einem Array gespeichert werden. Dieses Array soll dann mit Hilfe einer for-Schleife durchlaufen werden und die entsprechenden Label und Textboxen angezeigt werden. Die Schleife sah wie folgt aus:

```
    $data = @("CTXCloudCustomerID", "CTXCloudClientId", "CTXCloudClientSecret",
     "CTXCloudResourceID", "Vendor", "Product", "PackageName", "InstallerType")

    for($i=0; $i -le $data.Length; $i++){
        $current = $data[$i]
        $value = New-Object System.Windows.Forms.Label
        $value.Text = $current
        $y_Label = 10
        $value.Location = New-Object System.Drawing.Point(10,$y_Label)
        $value.AutoSize = $true
        $y_Label+30 | out-null
        $main_form.Controls.Add($value)
        $value = 0 | out-null
}
```

Da diese GUI nur eine Erweiterung zu dem bereits fertigen Skript war, wurde nach dieser Schleifen Fertigstellung nicht weiter an der GUI gearbeitet. Die erwähnten noch zu implementierenden Funktionen blieben also aus. Das Skript ist unter *Project/GUI* zu finden.

Der Grund für das Abrupte abbrechen dieser Aufgabe, lag darin, da akut Hilfe bei bei einem anderen Projekt benötigt wurde. Dieses wird im nächsten Kapitel erläutert.

2.6.7 XML Parsing

2.7 Automatisierung Active Directory

Nach dem Sammeln an Erfahrung im Umgang mit Power Shell und der gelungenen Implementierung des Cloud Connector Skripts konnte zu einem neuen Projekt und Anwendungsgebiet gewechselt werden - dem Active Directory.

2.7.1 Active Directory

Zunächst musste ein Verständnis für das Active Directory (AD) erlangt werden. Dieses ist ein Verzeichnis, in welchem Benutzer auf einer zentralen Datenbank gespeichert sind. Der Vorteil einer zentralen Speicherung ermöglicht die einfachere Verwaltung und Handhabung der Accounts. Will beispielsweise ein Nutzer die Möglichkeit haben auf jedem Rechner ein Benutzerkonto zu haben, so muss nun nicht mehr auf jedem Rechner ein Benutzerkonto erstellt werden, da es zentral Abgerufen wird. Vorteilhaft ist ebenfalls der beispielhafte Fall eines Passwortwechsels. Das Passwort wird lediglich im AD geändert und nicht auf jedem einzelnen System.

Jedem Konto sind durch klare Festlegung spezifische Rechte zugeteilt. Möchte man jenes Konzept größer skalieren, so stößt man schnell auf das Erstellen von Gruppen. Diese Gruppen werden auch Domänen genannt. Eine Domäne gruppiert verschiedene Accountarten, wie Admin-, Berater- oder einfacher Nutzeraccounts. Durch die Richtlinien für unterschiedliche Gruppen lässt sich eine Anpassung deutlich schneller durchführen [12].

Zur besseren Verwaltung wird mit sogenannten GPO (Group Policy Objects) gearbeitet. Diese sind eine Art virtueller Sammlung von Richtlinien Einstellungen. Diese Richtlinien werden in einer Datei gespeichert, die einen eindeutigen Namen zugewiesen bekommt. Jene Datei wird in der Regel in einem Ordner des Computers gespeichert und redundant dazu im AD [13].

Einleitend gab es dazu kleine Aufgaben zu lösen, wie das Anlegen verschiedener Organisationseinheiten. Da es sich um einzeilige Kommandos handelte musste lediglich kurze Recherche betrieben werden, um diese zu finden. Das Ziel dieser Aufgabe lag darin, den erhaltenen Zugang zum AD zu testen.

Aufgaben [14]:

- Script zum Anlegen von Active Directory Organisationseinheiten (Multi)

```
New-ADOrganizationalUnit "test"
```

- Script zum Anlegen von Gruppen (Global Groups, Local Groups, Universal Groups)

```
New-ADGroup -Name PS-Test -GroupScope DomainLocal
```

2.7.2 GPO

Nach den allgemeinen Tests auf dem AD konnte sich mit den Aufgaben zu den GPOs beschäftigt werden. Für diese galt es folgende Dinge zu lösen:

- Backup GPO
- Rollback GPO from JSON File
- Transfer all GPO's from OU's to other OU's
- Transfer single GPO to OU's
- Report GPO to HTML
- Compare List of linked GPO's from OU to another OU
- Remove unlinked GPO's

Um all diese Funktionen umsetzen zu können musste sich zuerst eine kleine Übersicht gemacht werden. Hierfür wurde also erneut auf das Konzept des Flussdiagramm zurückgegriffen. Dieses konnte stark vereinfacht dargestellt werden, da es lediglich der eigenen Übersicht diente.

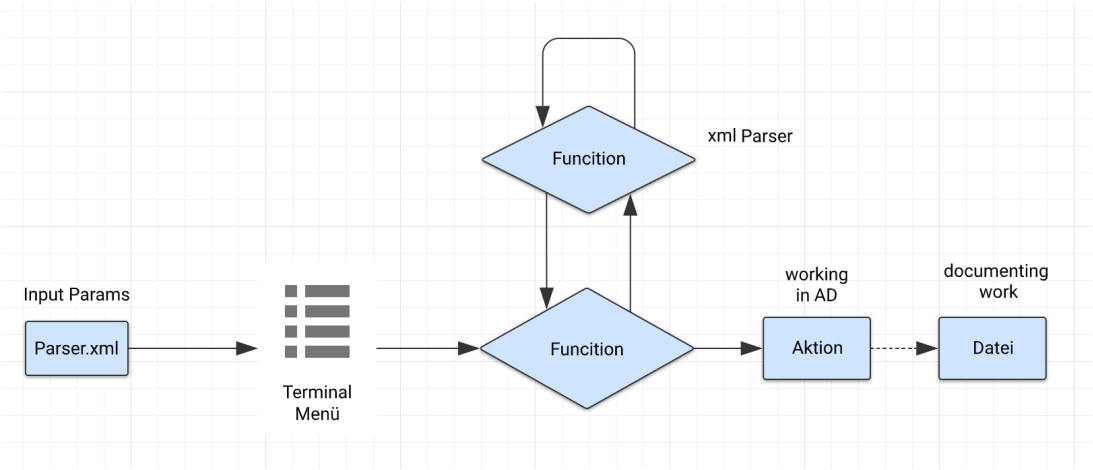


Figure 5: GPO Aufbau

Die Funktion zur Festlegung von Eingabeparametern wurde zunächst so gehandhabt wie bei dem vorherigen Cloud Connector Skript. Es wurde also mit einer Parser.xml gearbeitet, welche die OU Pfade als Input Parameter enthielt. Dabei fand eine Aufschlüsselung in Source, Target und Reporting statt. Die Idee hinter der separaten Abspeicherung in der XML war, dass bei der Verwendung beispielsweise mehrerer Targets eine bessere Übersicht und einfachere Eingabe gewährleistet werden kann. Die extra Herausforderung an das Skript wäre dabei, dass eine Art Erkennung implementiert werden muss, wie viel Pfade pro Variable eingetragen wurden. Alternativ könnte hierzu die Eingabe erst beim Aufrufen des Menüpunkts im Terminal statt finden, dies ist sinnvoll bei einzelnen Eingaben. Da primär aber das Ziel des Skripts darin lag ein ordentliches Reporting zu erstellen, wurden demnach zunächst für Testzwecke die Parameter Werte in dem Skript definiert.

Als nächsten Schritt müsste dann ein kleines Menü implementiert werden, welches im Terminal angezeigt und über dieses bedient werden kann. Das Menü

sollte die Punkte aus der einleitend genannten Liste enthalten. Der angesprochene Menüpunkt kann dann ausgelesen werden und über verschiedene Funktionsaufrufe zu dem gewünschten Ergebnis führen. Dabei ist es Elementar verschiedene Sktiptteile mit Aufgabenbereichen in einzelne Funktionen zu packen. Dies wird zum einen gemacht, da viele Funktionen mehrfach genutzt werden und so Redundanz im Code vermieden wird. Außerdem erleichtert die Verschachtelung eine spätere Wartung und Erweiterung des Skripts. Wurden die notwendigen Funktionen ausgeführt, so soll das Ergebnis, je nach Anfrage im AD selbst umgesetzt oder in einer extra Datei gespeichert werden.

Implementierung:

Das Skript und alle damit zusammenhängenden Dateien kann im Verzeichnis *Project/GPO* gefunden werden. Wie bereits gesagt wurden die XML Parser Teile von den alten Skripten übernommen. Das Schema der XML sah dabei wie folgt aus:

```
<Path>
    <Source>OU=Servers,OU=PROD,OU=Aero,DC=do1,DC=c-zsi,DC=io</Source>
    <Rollback></Rollback>
    <Target>
        <Target1>OU=DG-Win2k19-Test,OU=CTX,OU=Servers,OU=PROD,OU=Aero,
                 DC=do1,DC=c-zsi,DC=ik
        </Target1>
        <Target2>OU=Zwei-Test,OU=CTX,OU=Servers</Target2>
        <Target3>OU=Drei-Test,OU=CTX,OU=Servers</Target3>
    </Target>
</Path>
```

Das Konzept für das Auslesen mehrerer Targets, war dabei, dass mit einer foreach Schleife gearbeitet werden sollte, die die einzelnen Target Elemente durch iteriert. Dabei sollten diese im Idealfall einzeln an andere Funktionen durchgegeben werden, was wiederum durch Funktionsaufrufe in der Schleife selbst gelöst werden könnte. Dies funktionierte jedoch noch nicht einwandfrei, da die einzelnen XML Knoten Source, Target und Rollback nicht als solche erkannt wurden. Da dieser Teil der Implementierung jedoch gegen Ende der Praxisphase Programmiert wurde, gab es keine Fertigstellung dieser Funktion.

Um ein Grundgerüst für die weitere Implementierung zu schaffen, wurde die Menüfunktion geschrieben. Diese wertet einen Konsolen Input, basiert auf einer Switch Case Funktion, aus. Die Inputs werden im Main Bereich des Skripts abgefragt. Dort wird mit einer do until Schleife solange die Menüabfrage durchlaufen bis ein Input getätigt, oder mit dem Shortcut 'q' beendet wird. Zur Vermeidung von fehlerhaften Eingaben, wurde das Array *$validInput* mit den möglichen Inputs gefüllt. Dieses wird dann mit dem input über eine if und else Anweisung verglichen.

Nach diesen formalen Teilen wurde sich der funktionalen Umsetzung des Skripts gewidmet. Generell war die Vorgehensweise für das Erstellen der verschiedenen Funktionen so, dass zuerst das Skript an sich in einer eigenen Skript Datei erzeugt wurde, um anfängliche Fehler zu vermeiden. Aus Gründen der Übersicht und besseren Verständlichkeit wird in den folgenden Abschnitten jede Menüfunktion nur kurz umrissen.

Backup GPO / Report GPO to HTML

Die Aufgabe dieser zwei Menüfunktionen ist es existierende GPOs einer Organisationseinheit als Backup abzuspeichern oder einen Report zu erstellen. Sie bekommt die Parameter $searcher*, die den Bezeichner für die GPO hat, *$OutputPath*, die den Pfad für die Ausgegebene Datei beinhaltet und *$type*, der Aussagt welche Art von Funktion ausgeführt werden soll, also Backup oder Reports, übergeben. Ist dies geschehen, werden alle GPOs in ein Array gespeichert, welches dann wiederum mit einer foreach Schleife durchlaufen wird. Der einzige Unterschied zwischen Backup und Report ist, dass bei einem Backup der volle Bezeichner der GPO abgelegt wird. Als kleinen Zusatz wurde die Funktion einer Progressbar integriert, die im Terminal angezeigt wird, während die GPOs durchlaufen werden.

Rollback GPO from JSON File

Das Ziel dieser Menüfunktion soll es sein einem Rollback durch eine JSON

Datei zu ermöglichen. Um diesen Rollback zu ermöglichen wird auf eine zuvor erzeugte JSON, mit den Informationen zu den einzelnen GPOs, zur Verfügung gestellt. Diese wird dann in den *$Target* Pfad kopiert. Dabei wird eine neue Rollback Datei mit Zeitstempel erstellt, die dann wieder verwendet werden kann.

Transfer all GPO's from OU's to other OU's

Hier sollen alle GPOs einer Organisationseinheit in eine andere Organisationseinheit übertragen werden. Umgesetzt wurde dies mit einer if else Anweisung, bei der überprüft wird ob die $Target*Variableleerist.IstdiesderFall, werdenalleGPOsi$ OU in die $Target * OUkopiert.Istdie*$*Target* Variable jedoch nicht leer, werden alle GPOs in der $Source * OUindie*$*Target* OU kopiert und anschließend gelöscht.

Transfer single GPO to OU's

Der Transfer einer einzelnen OU wurde ausgelassen, da dieser Fall nicht häufig vorkommt. Gelöst werden könnte dies mit Funktionsteilen von *Transfer all GPO's from OU's to other OU's*.

Compare List of linked GPO's from OU to another OU

Bei dieser Task sollen zwei OUs mit ihren Richtlinien verglichen werden. Der Fokus des Vergleichs liegt dabei auf der Existenz der GPO in der OU. Die Ausgabe findet dann, in Form einer Tabelle, im Terminal statt. Hierzu gab es die Überlegung die Spalten der GPOs, welche nicht in beiden OUs vorhanden sind, zur besseren Übersicht in einer extra Farbe darzustellen. Leider hat dies nicht wie gewünscht funktioniert, demnach wurde darauf zurück gegriffen, dass bei der Tabelle die OUs nach False und True gruppiert wurden. Um überhaupt eine solche Tabelle zu erstellen, wurde in der Funktion *CompareLinkedGPO* eine Klasse erstellt, die die Datenstruktur der Tabelle beschreibt. Die Klasse kann in gewissen Ansätzen einem Mehrdimensionalen Array nachempfunden werden.

Eine Klassendeklaration ist ein Blueprint, der zum Erstellen von Instanzen

von Objekten zur Laufzeit verwendet wird. Jede Instanz von einer Klasse kann unterschiedliche Werte in ihren Eigenschaften aufweisen, und individuell angesprochen werden, was einen großen Vorteil darstellt. [15]

Remove unlinked GPO's

Zur Optimierung des ADs sollte ebenfalls eine Funktion integriert werden, die durch Parameter Übergabe eines OU Pfades jenen auf ungenutzte GPOs durchsucht und diese entfernt. Die erste Herangehensweise hierfür sollte sein, dass zunächst nur eine spezifische GPO gelöscht werden sollte. Leider kam es aus zeitlichen Gründen nicht mehr zur Bearbeitung dieses Falls.

3 Implementierung von automatischen Standard Service Requests

Um besser nachvollziehen zu können, wie Aufgaben für Entwickler entstehen und aus welchen Prozessen diese hervorgehen, wurde sich in der darauf folgenden Praxisphase mit der Implementierung von automatischen Standard Service Requests beschäftigt. Das persönlich angestrebte Ziel dabei ist es ein genaueres Verständnis der übergeordneten Abläufe zu erlangen und dieses Wissen für eventuell spätere Projekte beziehungsweise Abteilungen im Unternehmen anzuwenden.

3.1 Aufgabenstellung

Der Kunde besitzt keine Standard Service Requests. Die Aufträge kommen als Incidents und können demnach nicht automatisiert abgearbeitet werden. Das Reporting ist demnach falsch und die ITIL Konformität nicht gegeben. In Audit ist dies als zu verbessernd identifiziert.

Das Ziel der Arbeit ist es sowohl die vollständige Automatisierung als auch Trennung von Incidents und Standard Service Requests durchzuführen.

Im Verlauf der Praxisphase kam die Organisation und das Managen der verschiedenen SSRs als weitere Aufgabe hinzu. Die SSRs sollten ursprünglich nur mitgehört werden, um Schlüsse für das Vorgehen im eigenen Projekt zu ermöglichen. Da aber die leitende Mitarbeiterin in den Urlaub ging und diese Projekte sonst die Zeit über stillgestanden hätten, bot es sich für mich an als Urlaubsvertretung einzuspringen. Somit galt es also noch Wissen und Erfahrung im Bereich Management zu erlangen.

3.2 Einordnung der Aufgabenstellung in übergeordnete Prozesse

Es soll die Analyse stattfinden, um dies an die jeweiligen Betriebseinheiten zur Umsetzung weiterzugeben.

Die Analyse setzt sich zusammen aus der Absprache mit den einzelnen Betriebseinheiten, zu aktuellem Vorgehen. Im weiteren Verlauf soll dann das erlangte Wissen an Spezialisten weitergetragen und entwickelt werden. Im Nachgang soll dann mit dem Kunden ein Ende zu Ende Test gemacht werden, um den Erfolg und die vollständige Abdeckung der Arbeit zu prüfen. Der Kunde aktiviert das Formular für den erhaltenen Prozess.

3.3 Praktische Lösung

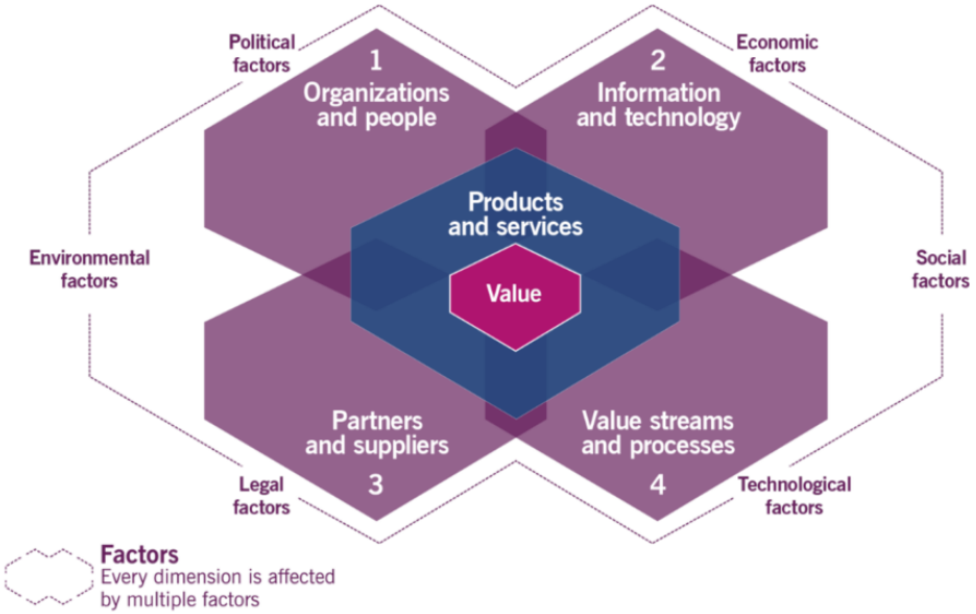
Die erste Aufgabe war das ganzheitliche Nachvollziehen von Prozessen und Strukturen. Gemacht wurde dies um die Analyse korrekt und zielgeführt durchzuführen. Dafür wurden einige verwendete Frameworks, die bei einem SSR (Standard Service Request) kurz erläutert. Im Folgenden werde ich diese kompakt erläutern und aufzeigen inwiefern jene im Zusammenhang mit dem angebotenen Produkt stehen.

3.3.1 ITIL

ITIL (Information Technology Infrastructure Library) ist ein standardisierter Leitfaden zum IT-Service-Management. Seine Best Practices sind eine Theorie zur besseren Umsetzung von Services. Das bedeutet, dass dieses Framework nicht zwingend zur Umsetzung von Services verwendet werden muss, sondern in der Praxis viel eher Fragmente daraus eingebunden und als vorteilhaft genutzt werden können. [16]

Generell besteht (spezifisch) ITIL 4 aus den zwei Kernelementen Service Value System (SVS) und dem Modell der vier Dimensionen. [17]

Die vier Dimensionen des Service Management dienen der effektiven und effizienten Förderung der Wertschöpfung für Kunden und Stakeholder, die durch Produkte und Dienstleistungen des Unternehmens widergespiegelt werden. Folgende Dimensionen werden hierbei thematisiert: Menschen, Produkte, Prozesse und Partner. [18]



[17]

Mit **Menschen** sind die Angestellten des Unternehmens gemeint, die für das Erbringen von Dienstleistungen, mit Hilfe ihrer Kompetenzen und Erfahrungen, verantwortlich sind.

Unter **Produkten** versteht man die *Werkzeuge*, die zum Erzeugen der Dienstleistung benötigt werden.

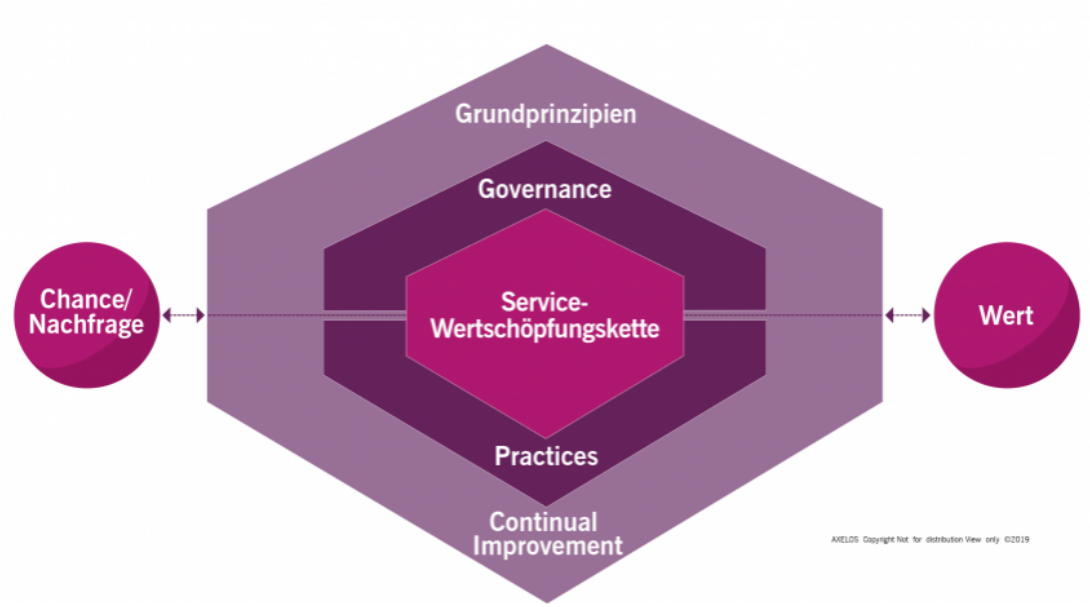
Prozesse unterstützen und verwalten Dienstleistungen, damit die Dienstleistung den Kundenerwartungen konform ist.

Partner sind praktisch als externe zu betrachten, die zur Erfüllung der Dienstleistungen beitragen können.

In häufigen Fällen ist es also sinnvoll gewisse Teile der Dienstleistung, respektiv zum Erzeugen einer Dienstleistung auf *Partner* zurückzugreifen. So kann beispielsweise ein *Produkt* von Externen bezogen werden, da es einen anderen Anbieter auf dem Markt gibt, der sich in diesem Bereich spezialisiert hat. Dieses Produkt wird zuvor über *Prozesse* mit dem Kunden in einem Service Level vereinbart und dann von den Mitarbeitern (*Menschen*) erarbeitet. [18]

Durch dieses Modell sollen also die Dimensionen einer Dienstleistung vermittelt werden. Das nächste Modell beschäftigt sich nun mit der Umsetzung und Einhaltung jener genannten Faktoren. [19]

Das Service Wertesystem beschreibt wie alle Faktoren und Aktivitäten einer Organisation zusammenwirken, um die gewünschte Wertschöpfung zu erzielen. Es ist ein Betriebsmodell zur Erzeugung, Bereitstellung und kontinuierlichen Verbesserung von Services. Zudem soll es Flexibilität fördern um Silodenken zu vermeiden. Nachfrage und Wert stehen dabei immer eng miteinander in Verbindung. Grafisch kann das SVS mit den folgenden fünf Schlüsselaktivitäten dargestellt werden:



[17]

Zwei dieser Schlüsselaktivitäten werden im weiteren Verlauf genauer betrachtet, da diese maßgeblich von Relevanz für die Arbeit waren: Der Themenblock Practices in Zusammenhang mit Continual Improvement. Darunter soll ein spezifisches Scope an Practice behandelt werden.

Auf die restlichen Komponenten wird nicht tiefer eingegangen, da diese für das Lösen der Aufgabe nicht relevant waren.

3.3.2 Service Request Practice

Das Scope an Practices, welches für die Praxisphase von Relevanz waren sind im Bereich des Service Management. Diese macht den größten Teil der gesamten Practices aus. Beschäftigt wird sich bei diesen Practices mit der Handhabung, Erstellung und Umsetzung von Services. Dabei wird zudem aktiv geschaut, wie neuer Mehrwert und entsprechende Wertschöpfung erreicht werden kann.

Generell wird eher ein vereinbarter Service ausgeführt und umgesetzt. Die spezifische Practice, um die es geht ist bei dieser Angelegenheit die Standard Service Request (SSR), auch Service Request genannt. Die Themenfelder, die von einer SSR abgedeckt werden können, sind Anfragen zur Servicebereitstel-

lung, Informationsanfragen und das Einholen von Feedback.

In den meisten Fällen handelt es sich hierbei um Changes. Dabei reicht dieser Service von einem minimalen Standard-Change, bei dem beispielsweise eine Passwort-Änderung eines Benutzerpassworts durchgeführt wird, bis hin zu einer Migration von einem ganzen System. [20]

Für die einzelnen SSRs, die in der Praxisphasen Abteilung aktuell bearbeitet wurden, gab es auf dem internen Sharepoint eine zentrale Excel Tabelle.

3.4 Produkte

In diesem Kapitel werden die einzelnen Produkte und deren Zusammenhänge im Prozess der Automatisierung generell aufgeführt. Die Ursprüngliche Idee diese Produkte mit der spezifischen Druckerautomatisierungsaufgabe anschaulich darzustellen wurde verworfen, da es sich bei dem Vorgehen mit lediglich einzelnen kleinen Abweichungen, für jede SSR relativ identisch gestaltet. Außerdem wurde dadurch, dass wie einleitend in der Arbeit bereits genannt eine Urlaubsvertretung stattfand, ein tieferer Einblick in die einzelnen SSRs ermöglicht. Dieser Einblick sorgte für extra Aufgaben, die wiederum mit den Produkten zu tun hatten. Somit wird nach der Produkterläuterung auf die Projekte und deren Produkt Zusammenhänge im einzelnen genauer eingegangen.

3.4.1 Top Desk

3.4.2 Service Now

Der Ansatz von Atos ist es Service Now dem Kunden als Service, in Form von ihrem Atos Technology Framework (ATF), anzubieten. Den ITIL Practice den es dabei übernimmt ist der des Service Request Managements.

Service Now ist eine Cloud basierte Plattform für Support Services. Es ermöglicht dem Nutzer eine einfache und schnelle Integration von Service Man-

agement im Unternehmen. Zudem erschafft es die Basis, Service Management in einem Unternehmen soweit zu automatisieren, dass es dadurch die Produktivität der Arbeitsabläufe potentiell steigert. [21]

3.4.3 Beat Box

Die von Service Now vorher erhaltenen Anfragen werden in Form von Workflow Routinen durch die Beat Box verarbeitet. Beatbox steht als Abkürzung für Back End Automation Tool in Box. Es wird also, wie sich aus dem Namen schon schließen lässt, mit der Beatbox ermöglicht die Automatisierung, die von Service Now gestartet wird, vollständig im Backend abzudecken. In der Toolbox enthalten sind Software wie der Microsoft Orchestrator, MS System Center Service Manager, MS System Center Configuration Manager und das Master Data Repository. Diese können nach Implementierung mit den folgenden Systemen agieren: Active Directory, Exchange, Office356, Skype for Business und User Application Management. Besteht also das Interesse des Kunden eines dieser Systeme aus dem eigenen Netzwerk zu automatisieren, kann dies durch die Beat Box vollständig ermöglicht werden. [**AtosTechnologyFramework?**]

3.4.4 Protokollierung und Dokumentation

Da es galt so viele Prozesse wie möglich zu automatisieren, wurde von der Abteilung eine Liste von Incidents gesammelt. Durch diese Incidents konnten dann SSRs für den Kunden erschlossen. Um einen umfassenden Überblick auf die einzelnen SSRs und deren aktuellen Status zu haben, wurde somit eine Excel Tabelle erstellt.

Zur Aufbauerklärung der Excel Tabelle wird diese mit dem Beispiel der Druckerautomatisierung durchgesprochen. Die erste Spalte beschäftigt sich mit der Kategorie, in unserem Fall *Drucker*. Danach kommt spezifischer die Sub-Kategorie,

also das Druckereinbinden, abmelden und umziehen. Gefolgt von Thema/Prozess und Ansprechpartner. In der Spalte Statusupdate, werden neue Termine eingetragen. Alte Termine sowie Erkenntnisse werden dabei nicht verändert und bleiben in der Zelle stehen, um den Verlauf besser einsehen zu können. Zuletzt gibt es noch den Status der SSR von der Seite des Dienstleisters und der des Kunden. Die Form die der Status dabei annehmen kann sind offen, in Analyse, in Umsetzung, bereit für End to End Test und abgeschlossen.

A	B	C	D	E	O	P	Q
Kategorie	Sub-Kategorie	Karport	Reihefolge Entwicklung	Thema / Prozess	Anprechpartner Atos	Statusupdate	Status Atos
Drucker	Druckereinbindung			Druckereinbindung (WLAN- bzw. A4-Drucker)	Schaack, Karl-Heinz	11.08: 3. Analysetermin mit BBX-Team und ServiceNow (Input Parameter 08.08: 2. Analysetermin mit BBX-Team 22.07.: Karl-Heinz passt nach seinem Urlaub ein Skript an, das dann die BeatBox ausführen kann.	in Analyse

Figure 6: SSR Vorschläge

Die Aufgabe war es die Excel Tabelle regelmäßig zu aktualisieren und speziell für das eigene Thema voranzubringen, sodass der Status am Ende im Idealfall als abgeschlossen gesetzt werden kann.

3.4.5 Reporting

Das genannte Excel Dokument war zudem eine Grundlage für die Erstellung eines Reports. Dieser Report musste einmal wöchentlich während des Urlaubs übernommen werden. Er besteht aus einer Übersicht über die SSRs, die in der Praxisphasen Abteilung aktuell aktiv bearbeitet wurden. Aufgebaut ist das Report Dokument mit einem Status Overview. Dieser enthält drei Ampeln mit den Sektoren Zeit, Ressourcen und Qualität. Im Idealfall stehen diese alle auf Grün. Zudem sind die drei aktuellsten SSRs, aufgelistet nach deren Priorität, inklusive ihres akuten Prozesses und Kommentars. Des Weiteren ist eine Übersicht über die Arbeitsergebnisse der letzten Woche aufzufinden. Diese beinhalten, die Menge an SSRs, welche in die Analyse gewechselt sind, die in die Entwicklung, sowohl seitens des Dienstleisters, als auch des Kunden, gewechselt sind, die im

End to End Test und erledigt wurden. Daneben zu finden, sind die Geplanten Arbeitspakete, mit den Inhalten die voraussichtlich in der darauffolgenden Woche angegangen werden.

Ziel eines solchen Reports ist es also dem Kunden und Unternehmen einen Eindeutigen Status über die aktuellen Aufgaben und SSRs zu vermitteln. Diese Vorgehensweise ist zudem ITIL konform. Ein weiterer Vorteil eines solchen Reports, ist die Incident Vorsoge und Bewältigung, beides Teile der Incident Management Practice. So können beispielsweise durch die Ampeln bevor sie auf Rot gesetzt werden, in der Stufe Gelb, noch behandelt und korrigiert werden. Gravierende Incidents werden vermieden und die Dienstleistung kann Qualitätskonform geliefert werden.

3.5 SSR Umsetzung

Wie zuvor in der Kapitel Einleitung *Produkte* angekündigt, wird sich nun mit den einzelnen SSRs, in spezifischen Unterkapiteln, auseinandergesetzt. Dabei wird, von der einleitenden Aufgabenstellung bis zur Umsetzung, der gesamte Prozess, aus einer Management Sicht, dargelegt. Der Stand der einzelnen Projekte wird soweit die Praxisphase bis zu diesem Zeitpunkt gegangen ist dokumentiert.

3.5.1 Druckerautomatisierung

Der Kunde erstellt jedes mal wenn er einen neuen Drucker bei sich in der Filiale einrichtet, umzieht oder entfernt, einen Service Request. Die dafür zuständige interne Abteilung hat hierfür ein eigenes Web Interface erstellt. In diesem lassen sich die von dem Ticket erhaltenen Daten eintragen. Die Daten werden in einer internen und Kunden externen Datenbank gespeichert. Somit ist der Request abgearbeitet. Nun gilt es den Request soweit zu vereinfachen, dass der Kunde

final beim erstellen die Parameter selbst eingeben kann und die Übergabe / der Eintrag automatisiert abläuft. Aufgeteilt wurde der Incident somit in drei SSRs: Drucker anmelden, umziehen und abmelden.

Drucker anmelden

Hierzu wurde sich mit dem Entwickler der Abteilung zusammen gesetzt und das Skript mit den relevanten Parametern in einer Usecaseanalyse dokumentiert. Diese Usecaseanalyse ist für den späteren Transfer zwischen Service Now (SNOW) und der Beat Box (BBX) notwendig gewesen, damit eine einheitliche Verwendung der Serviceparameter stattfinden kann. Als Erstes wurde sich dem SSR Anmeldung gewidmet. Zum definieren der Parameter gibt es ein fest vorgeschriebenen Aufbau. Der nach der Namens Benennung zur Art der Eingabe übergeht. Diese besteht aus Typ, also ob der Parameter später über einen Dropdown Menü oder ein Textfeld eingegeben werden soll und vordefinierten Parameter Werten. Abschließend wird noch die Bezugsquelle, in die der Parameter eingetragen werden soll, angegeben.

Für den ersten Parameter *$filiale* wird ein Dropdown Menü verwendet. Dieses besteht aus einem Array, welches die Filialenwerte enthält. Die Werte bestehen aus vier chars von 0000 bis 9999. Gewählt wird der Parameter auf einem Interface von Service Now. Dieser Aufbau zog sich für die anderen Parameter nahezu gleich durch.

Nach Vervollständigung der Parameter ging es über zum Dokumentieren des Vorgehens, im Falle einer Anfrage. Dafür gab es eine weitere Tabelle, die die Spalten Step, Action, System und Description hat. In der Spalte Action wird der Titel der Aktion angegeben, im ersten Fall die Verbindung mit der Drucker Datenbank, auf einem MySQL Server. Die Beschreibung hierzu beinhaltet die Aktion auf programmiertechnischer Ebene. Es werden also bereits implementierte Skriptteile, die zuvor manuell ausgeführt wurden aufgenommen. Diese können dann später bei der automatisierten Lösung mit integriert werden, so

kann Entwicklungszeit gespart werden. Somit konnte die Implementierung an das SNOW und BBX Team beauftragt werden.

Als nächste Aufgabe galt es für das SNOW Team eine Test Request zu erstellen. Diese bildet einen fiktiven Nutzer welcher eine Anfrage zum Anlegen eines Druckers hat und diese über das SNOW Interface erstellen möchte. Es handelt sich also um einen klassischen Test Case, zum Testen des implementierten Systems. Das Test Dokument sah wie folgt aus:

Neuer Drucker wird in Datenbank angelegt: *Service Parameter* der XML kursiv dargestellt

Eine RITM oder Ticket Nummer wird erstellt: *A4040*

Der Drucker *Optra T630* mit der ListenID *0001* soll in Frankfurt *0003* in der Innenstadt *0004* im *1. Stock, rechts* hinzugefügt werden. Ein IPP Protokoll ist vorhanden *Yes* . Die spezifische GeräteID lautet *GDD30491*

```
Beispiel XML:
    <ServiceParams>
        <Filiale>0003</Filiale>
        <Mandant>0004</Mandant>
        <PrinterName>OptraT630</PrinterName>
        <PrinterType>0001</PrinterType>
        <IPP>Yes</IPP>
        <Location>1. Stock,rechts</Location>
        <RITM>A4040</RITM>
    </ServiceParams>
```

Nach dem Auslesen der XML sollen folgende Variablen Werte ausgelesen sein: *Variablen Namen für mySQL/PowerShell*

```
$filiale = 0003
$mandant = 0004
$drucker_id = 0001
$ipp = 1
$location = '1. Stock,rechts'
$incident = 'A4040'
$rmt = 'GDD30491'
```

- Service Parameter vollständig, können die einzelnen Steps erfolgreich durchgeführt werden:

  - Create Record for new printer
  - Insert Ticket number

- Error/Exception Handeling
  - Validate the printer creation

Nun folgen die Skriptteile, welche mit den eingespeisten Variablen arbeiteten. Als Vorraussetzung gilt dafür, das der Benutzer, von dem die Scripte ausgeführt werden admin Rechte bestitzt und Zugang zu der Datenbank hat.

Create Record for new printer: Zum Einen wird dem Drucker eine spezifische Geräte URL zugewiesen.

```
if ($ipp==1){
    $url="http://$rmt.$domaene:631/ipp";
}
else {
    if ($druckertyp_id=='4')
        $url="lpd://$rmt.$domaene/com1";
    else
        $url="socket://$rmt.$domaene:9100/";
}
```

Zum Andern werden diese Werte dann mit SQL statements in die Datenbank gespeichert.

```
INSERT INTO Druckertool.drucker (mandant_id,rmt,druckertyp_id,ipp,location,url,datum)
VALUES ((SELECT mandant_id FROM Filialen.mandant M
JOIN Filialen.filiale F ON F.filiale_id=M.filiale_id WHERE F.filiale='$filiale' AND M.mandar
```

Nach diesem Eintrag muss ein weiterer incident Eintrag mit Ticket Nummer gesetzt werden.

```
get drucker_id from printer created in #2, e.g. lastinstertID

INSERT INTO Druckertool.incident (incident,drucker_id,neu,user,datum)
VALUES ('$incident','$drucker_id',1,'5574','$unixtime')
```

Zuletzt wird noch eine Routine aussgeführt, die abfragt, ob der Drucker mit der erstellten ID angelegt wurde. Falls dies nicht der Fall ist soll ein Fehler ausgegeben werden.

Zu den gegebnen Skripten wurde folgende grafische Darstellung des Interfaces erstellt, anhand der sich für die eigene Umsetzung orientiert werden konnte.

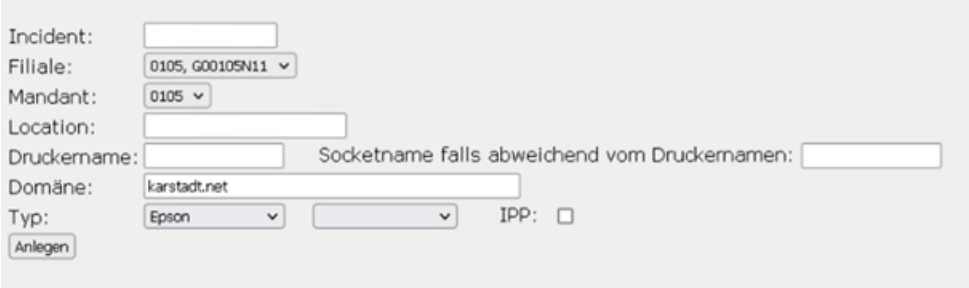


Figure 7: Drucker Interface

Da in dem Interface Felder mit Drop Down Menüs sind muss bei der Umsetzung drauf geachtet werden, dass diese während der Eingabe des Nutzers mit Daten aus der Datenbank befüllt werden. Die Datenbank muss also schon vor gewissen Eingaben ausgelesen werden. Dabei muss außerdem beachtet werden, dass bestimmte Fälle nicht möglich sind. So kann zum Beipiel nicht bevor die Filiale gewählt wird bereits der Mandant auszuwählt sein, da dieser von der Filiale abhängt.

Der SSR war damit fertig Analysiert und konnte über in die Implementierung von SNOW und der BBX gehen. Somit war das Projekt aus der Organisatorischen Sicht erfolgreich abgeschlossen.

Drucker umziehen/löschen

Nach einer Debatte, ob die Änderung, beziehungsweise der Umzug, als SSR angeboten werden soll, kam man zu dem Entschluss, dass sich der Mehraufwand eines extra SSRs, mit nur geringfügigem Mehrwert für den Kunden, nicht lohne. Somit wurde sich lediglich mit der Löschung eines angelegten Druckers in der Datenbank beschäftigt und der Fakt des Umziehens, als Löschen und Neuanlegen betrachtet.

Die Usecaseanalyse für diesen Request, konnte reduziert werden, da hiefür

nur die eindeutige DruckerID benötigt wir um einen Drucker aus der Datenbank zu entfernen. Demnach musste kein Code zur Verfügu8ng gestellt werden und dieser SSR war ebenfalls abgearbeitet.

3.5.2 Lizenzwechsel

Bei diesem SSR handelt es sich um Lizenzwechsel Anfragen von Microsoft365 Lizenzen. Als Erste Aufgabe galt es mit der zuständigen Abteilung und dem Kunden Rücksprache zu halten wie der Service Request bisher abgelaufen ist. Dafür wurde ein Word Dokument geschrieben, welches dann wiederum in ein Visio, zur besseren Übersicht, gewandelt wurde. Das Visio sieht wie folgt aus:

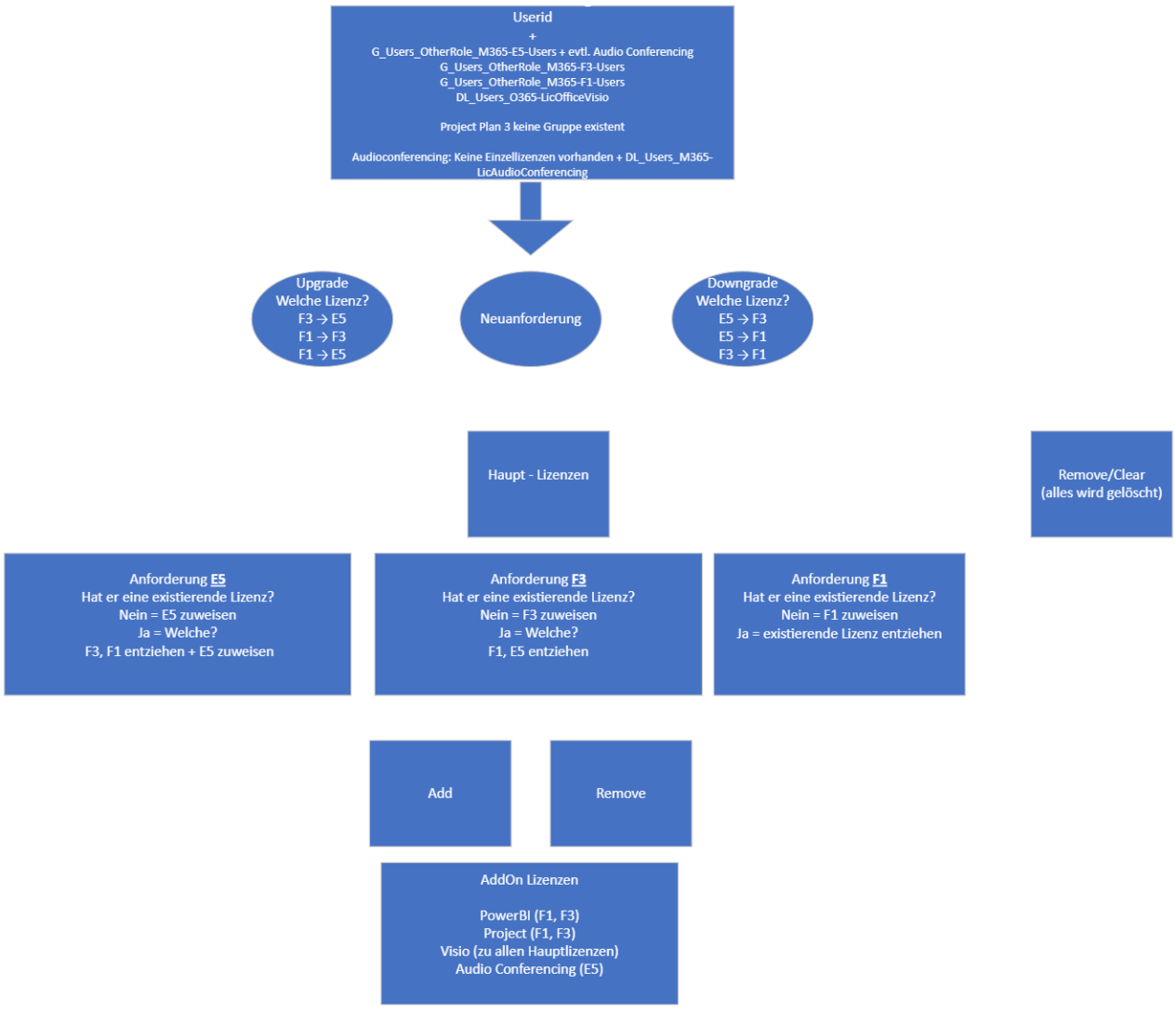


Figure 8: Lizenzen Visio

Ein neuer User mit spezifischen Lizenzanforderungen soll erstellt werden. Dafür wird die UserID mit seinen Rollen und bisherigen Lizenzen benötigt. Sind diese Informationen vorhanden, kann nun geschaut werden welche Art von Lizenz angefragt wird. Dabei wird zwischen einem Lizenz Upgrade und einem Lizenz Downgrade entschieden. Hierfür gibt es jeweils drei Fälle, da mit drei Hauptlizenzen gearbeitet wird. Handelt es sich um einen neuen User, hat er keine Lizenz und bekommt den Status Neuanforderung. Ebenfalls gibt es den umgekehrten Fall, dass der User entfernt wird und ihm entsprechend die Lizenzen entzogen werden müssen. Dafür wird nach der Entscheidung welche Lizenzanfrage ansteht ein Ebene weiter im Visio gegangen. Es wird zwischen Hauptlizenz oder Remove/Clear entschieden. Unter den Hauptlizenzen Fallen kategorisch drei verschiedene F1, F3 und E5, die mit steigender Nummer immer mehr Inhalte besitzen.

Die F1 Lizenz ist angedacht für Mitarbeiter im Service und Produktionsbereich, auch sogenannte Frontline-Worker. In diesem Beispiel wären es Kassen- und Beratungsmitarbeiter, die den Anspruch haben nicht unbedingt an einem fixen Arbeitsplatz zu arbeiten. Es handelt sich also um Lizenzen für Software die auf mobilen Geräten betrieben wird. Enthalten sind Apps von Microsoft365, Teams zur Kommunikation und Apps für den Bereich Arbeits- und Prozessmanagement.

Als nächste Lizenz wäre F3 zu erläutern. Diese ist für große Unternehmen angedacht, welche ihren Mitarbeitern Software zur Produktivität, dem Arbeitsmanagement, Filesharing, der Analyse und ebenfalls Kommunikation bieten wollen. Zu den vorherigen Apps kommen Email und Kalender durch Outlook und Exchange, OneDrive und das Azure Active Directory dazu. Außerdem werden grundlegende Security- und Compliance-Features geboten, ohne auf E5 hoch zu stufen.

Zuletzt gibt es noch den Premium Plan, in Form der E5 Lizenz. Diese Lizenz

hat ebenfalls alles, was F3 hat, ist aber im Bereich der Security- und Compliance-Features noch umfassender. Sie wird also benötigt, wenn ein Unternehmen ein großes Interesse an einer erweiterte Analyse-Möglichkeit für die genannten Features haben will.

Damit sind die drei Hauptlizenzen abgearbeitet im Schaubild zu erkennen sind nun die Bedingungen für die einzelnen Lizenzen. Dazu wird bei jeder Lizenzzuweisung geschaut, ob bereits eine Lizenz existiert. Ist dies der Fall, so wird die alte Lizenz entzogen und durch die neue ersetzt. Als Erweiterung werden weitere Einzellizenzen zur Auswahl angeboten. Diese dienen dazu, das der Kunde den User nicht komplett auf die nächste Hauptlizenz stufen muss, sondern in kleinen AddOns die Lizenz aufwerten und nur benötigte Inhalte dazu buchen kann.

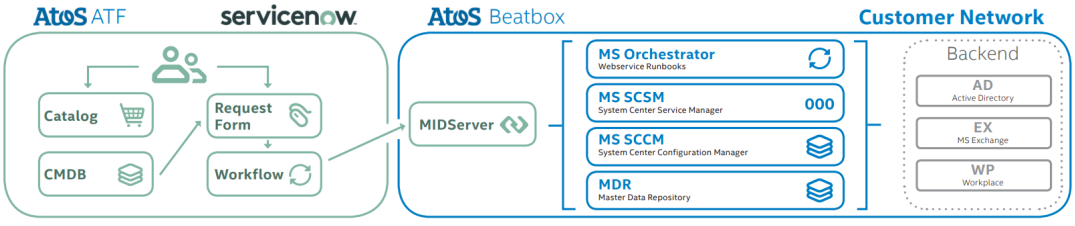
Nach der Veranschaulichung des Lizenzwechsel Ablaufs, kam es dazu die gesammelte Information an das BeatBox Team zu übermitteln. Diese fassten den SSR in eine Usecaseanalyse zusammen.

Um eine vollwertige Implementierung zu ermöglichen mussten die verschiedenen möglichen Namen der einzelnen Lizenzgruppen und Untergruppen zur Verfügung gestellt werden.

3.5.3 Sharepoint Usecases

3.6 SRR umsetzungs Prozess

Die Implementierungsroutine zur Erstellung einer automatisierten SSR sieht seitens Atos insgesamt wie folgt aus:



[AtosTechnology

3.7 Verknüpfung zu Vorlesungsinhalten

Softwareengeneering Userstories

3.8 Kritische, inhaltliche Reflexion von Theorie und Praxis

---

4 Ausblick von Automatisierung

5 Quellen

[1] Referenzarchitektur: Virtual Apps and Desktops Service von Citrix Service Provider. Im Internet: https://docs.citrix.com/de-de/tech-zone/design/reference-architectures/csp-cvads.html; Stand: 25.07.2022

[2] Citrix Cloud Connector | Citrix Cloud. Im Internet: https://docs.citrix.com/de-de/citrix-cloud/citrix-cloud-resource-locations/citrix-cloud-connector.html; Stand: 25.07.2022

[3] Citrix StoreFront. Im Internet: https://www.computerweekly.com/de/definition/Citrix-StoreFront; Stand: 26.07.2022

[4] sdwheeler. Was ist PowerShell? - PowerShell. Im Internet: https://docs.microsoft.com/de-de/powershell/scripting/overview; Stand: 21.07.2022

[5] Iwaya A. Does PowerShell Work on Other Operating Systems Besides Windows? Im Internet: https://www.howtogeek.com/306261/does-powershell-work-on-other-operating-systems-besides-windows/; Stand: 21.07.2022

[6] sdwheeler. Einführung in die Windows PowerShell ISE - PowerShell. Im Internet: https://docs.microsoft.com/de-de/powershell/scripting/windows-powershell/ise/introducing-the-windows-powershell-ise; Stand: 21.07.2022

[7] . Im Internet: http://www.powertheshell.com/; Stand: 21.07.2022

[8] Extensible Markup Language (XML) 1.0. 1998. Im Internet: https://web.archive.org/web/20060615212726/http://www.w3.org/TR/1998/REC-xml-19980210; Stand: 22.07.2022

[9] online heise. Besser zentral: Professionelles Logging. Im Internet: https://www.heise.de/ratgeber/Besser-zentral-Professionelles-Logging-2532864.html; Stand: 25.07.2022

[10] sdwheeler. Start-Transcript (Microsoft.PowerShell.Host) - PowerShell. Im Internet: https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.host/start-transcript; Stand: 22.07.2022

[11] joeyaiello. Beispiele für die kommentarbasierte Hilfe - PowerShell. Im Internet: https://docs.microsoft.com/de-de/powershell/scripting/developer/help/examples-of-comment-based-help; Stand: 29.07.2022

[12] Czernik A. Active Directory und Domäne – einfach erklärt. 2016. Im Internet: https://www.dr-datenschutz.de/active-directory-und-domaene-einfach-erklaert/; Stand: 27.07.2022

[13] REDMOND\\markl. Group Policy Objects. Im Internet: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/policy/group-policy-objects; Stand: 28.07.2022

[14] Verwalten von Organisationseinheiten mit PowerShell. Im Internet: https://blog.netwrix.de/2020/01/24/verwalten-von-organisationseinheiten-und-verschieben-ihrer-objekte-mit-powershell/; Stand: 29.07.2022

[15] sdwheeler. Informationen zu Klassen - PowerShell. Im Internet: https://docs.microsoft.com/de-de/powershell/module/microsoft.powershell.core/about/about_classes; Stand: 25.08.2022

[16] Was Ist ITIL? - Glossar - TOPdesk. Im Internet: https://www.topdesk.com/de/glossar/was-ist-itil/; Stand: 03.08.2022

[17] Beschreibung Zu Den ITIL 4 Best Practice | KESS-Buchsein.De. Im Internet: https://www.kess-buchsein.de/schulungen/itil-4/was-ist-itil-4; Stand: 03.08.2022

[18] Andenmatten M. ITIL4 – Der ganzheitliche Ansatz mit vier Dimensionen. 2019. Im Internet: https://blog.itil.org/2019/04/itil4-der-ganzheitliche-ansatz-mit-vier-dimensionen/; Stand: 08.08.2022

[19] Kempter S. ITIL 4 | IT Process Wiki. Im Internet: https://wiki.de.it-processmaps.com/index.php/ITIL_4; Stand: 03.08.2022

[20] Request Fulfilment | IT Process Wiki. Im Internet: https://wiki.de.it-processmaps.com/index.php/Request_Fulfilment; Stand: 09.08.2022

[21] Links R. About ServiceNow - Who We Are and What We Do - ServiceNow. Im Internet: https://www.servicenow.com/company.html; Stand: 10.08.2022