
Implementierung einer Datenschnittstelle für die MY-AOKApp in NiFi bei der AOK Sachsen-Anhalt

Abschlussdokumentation einschließlich der NiFi Implementierung und Handhabung

Dieses Dokument gibt einen Überblick über implementierte Gesamtlösung sowie eine Beschreibung der einzelnen Komponenten

Auftraggeber:	AOK Sachsen-Anhalt	Auftragsnummer:	
Auftragnehmer / Autor:	FREIHAFEN IT GmbH	Ausgabe oder Version:	0.1
Autorisiert durch:		Status:	Draft 0.1

I

Versionsnachweis

Version	Datum	Autor	Inhaltliche Änderung
0.1	25.05.2022	Oliver Perschke	First Draft

Inhaltsverzeichnis

Inhalt

1.	Ziel des Dokuments	4
2.	Einleitung / Ausgangssituation / Beschreibung der Lösung	4
2.1	Das Konzept.....	4
	ErrorFolder:	4
	AugabeVerzeichnis:	5
	BackupFolder:	5
	Weiter Anforderungen:	6
3.	Notwendige Installation (Setup Runtime Environment)	6
3.1	Test-Umgebung	6
3.2	Produktionsumgebung:	7
3.3	Alle Umgebungen	7
4.0	Technische Dokumentationen Nifi	8
4.1	Nifi	8
4.2	Nifi Registry	8
4.3	Nifi Toolkit	8
4.4	Installation	8
4.5	Konfiguration	9

4.51 Auf der Command Line	9
4.5.2 Die Konfiguration im NiFi Gui:	10
4.6 Nifi Referenzinformationen aus dem Internet	11
5.0 Technische Dokumentationen MeineAOKApp Nifi Schnittstelle	11
5.1 Allgemeine Informationen	11
5.2 Prozeduraler Ablauf:	11
5.3 Die Datenbank	12
MAOKA_Uebersetzung	13
6.0 Anhang	15
6.1 Dokumentation der Anforderungen	15
6.2 Das Programm	15
Die Klassen:	16
Das Programm:	16
Die KonfigDatei App.Config:	17
6.3 Klassenreferenz:	20
6.4 Die Datei App.Config	20
6.5 Das Programm Repository:	21
6.5 Der Anpassungsprozess für Änderungen der Datenbanktabellen:	22
Anpassen neuer Dokumententype	22
Technische ToDo:	22
Referenzen:	22
Datenbanklogin Test:	22
Datenbanklogin Prod:	22
Erstellung eines gecrypteten Passwords:	22

1. Ziel des Dokuments

Dieses Dokument fasst alle Ergebnisse und Teilergebnisse der Analyse- und Implementierungsphase bei der AOK Sachsen-Anhalt zusammen und erläutert die Ergebnisse und beschreibt die durchgeführten Maßnahmen.

Außerdem werden in diesem Dokument alle analysierten Prozesse noch einmal beschrieben, bzw. die bereits vorhandenen Beschreibungen zusammengefasst, um diese in einem Gesamtdokument zu bündeln. So dient dieses Dokument nicht nur für die weitere Optimierung der Fachlogikverarbeitung im RZ Magdeburg, sondern kann in Auszügen auch als Leitfaden für neue Mitarbeiter verwendet werden.

Im Rahmen der Einführung eines neuen Input-Managementsystems und der Einführung neuer Logikmodule kann das Dokument zudem als Dokumentation des Ist-Standes angesehen werden und als Grundlage für die Verarbeitung von MyAOKApp Daten dienen.

2. Einleitung / Ausgangssituation / Beschreibung der Lösung

Die AOK Sachsen-Anhalt (bzw. deren Dienstleister) implementiert zurzeit ein neues IPM System und erstellt dazu Datenverarbeitungs- und Transferlösungen.

Dieses Dokument beschreibt die für diesen Zweck zu implementierende MyAOKApp Schnittstelle als Interface zwischen Dokumenteneingang und dem CrossCap System zur Validierung der Eingänge.

Ausgangslage:

Über die „MeineAOKApp“ können Kunden der AOK beliebige Dokumente einreichen. Diese werden durch einen Dienstleister in ein PDF Dokument umgewandelt und mit einem begleitenden XML Dokument der Schnittstelle zur Verfügung gestellt.

Dabei dient eine .RDY Datei als Triggerdatei die die Anlieferung abschließt. Danach wird durch NiFi im Eingangsordner die Verarbeitung durch Einlesen der XML-Datei gestartet. Zur Verarbeitung werden verschiedene Informationen aus der XML Datei ausgelesen, in einer Reportdatenbank dokumentiert und die PDF und die XML Datei mit angepasstem Namen in einer Stapelstruktur mit anderen Dokumenten abgelegt.

Die Bearbeitung der eingereichten Dokumente erfolgt im Regelfall komplett ohne menschliche Interaktion (Dunkelverarbeitung). Nur im Fehlerfall ist ein manueller Eingriff vorgesehen.

2.1 Das Konzept

Die Entwicklung und Implementierung folgt der in der AOK üblichen Methodik des Workflowmanagementtools NiFi. Es wird eine NiFi Umgebung mit den folgenden Ausgabebereichen eingerichtet:

ErrorFolder:

Hier erscheint eine Datei, wenn es zu einem Programmabbruch gekommen ist und eine manuelle Interaktion notwendig ist => Automatische Überprüfung ist empfohlen!

Des Weiteren muss hier im Fehlerfall immer eine Einzelfallprüfung erfolgen, da hier immer eine Datei erscheint, wenn das Programm mit einem Exit Code != 0 oder einem Programmabsturz (keine Admin-E-Mail) beendet wurde.

Ausgabeverzeichnis:

Dies ist der Basisausgabefolder aus dem UC4 die Ausgabe abholt.

BackupFolder:

In diesen Folder werden die mit .rdy gekennzeichneten Verzeichnisse gesichert. Diese kann (und sollte wegen des Platzbedarfs) nach Belieben aufgeräumt werden. Dies kann und sollte automatisiert erfolgen.

Dieser Ordner kann nach Belieben aufgeräumt werden.

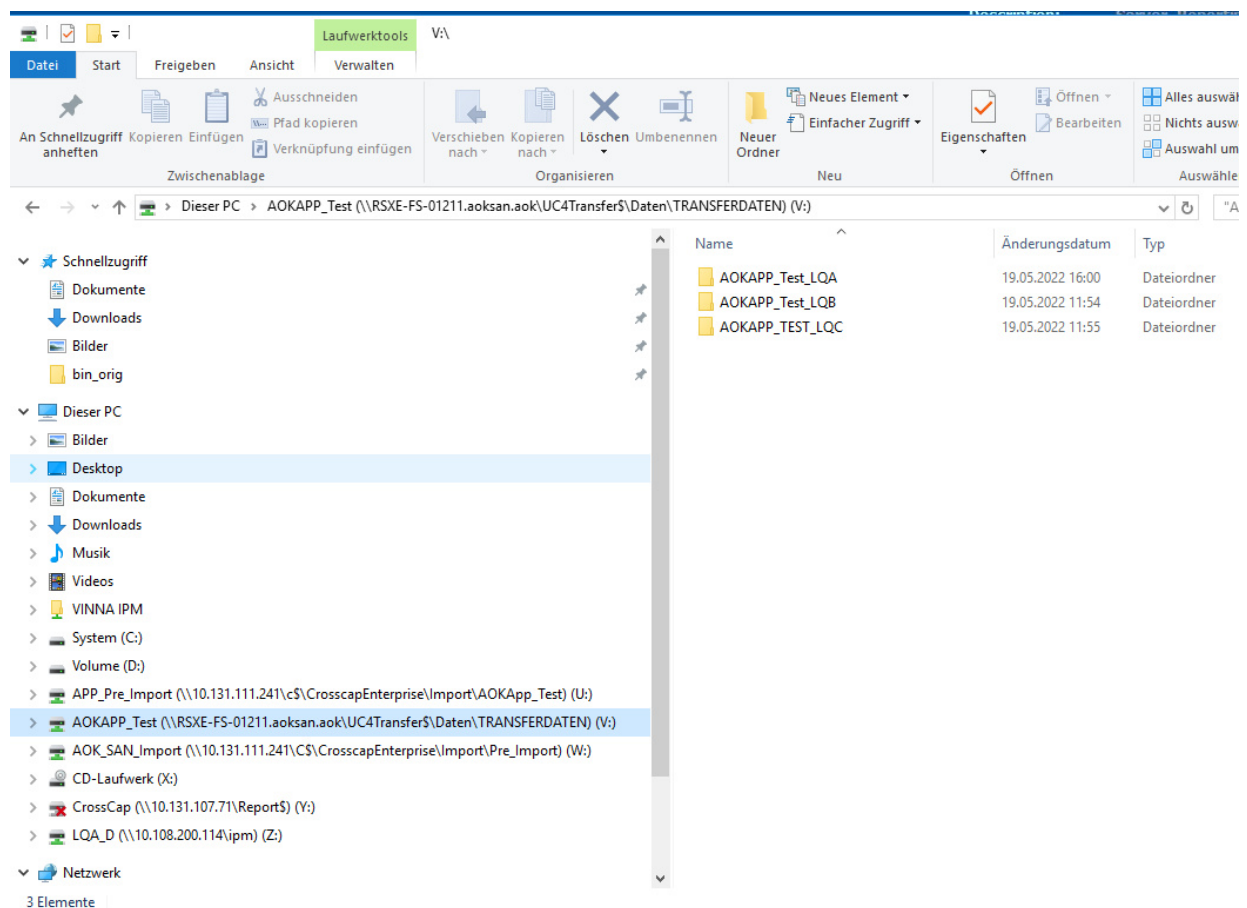
Diese werden in der zugehörigen Konfigurationsdatei wie folgt zugewiesen:

```
<add key="ErrorFolder" value="D:\AOKAPP\Base\Error"/>
<add key="BackupFolder" value="D:\AOKAPP\Base\Backup"/>
<add key="Ausgabeverzeichnis" value="U:\"/>
```

In der Test Umgebung:

Der Share ist als Laufwerk V:\ ist auf dem Server gemountet:

Dort liegen die Eingangsverzeichnisse der verschiedenen SAP Umgebungen



NiFi selbst verwaltet dabei nur den „error“ Folder.

Die anderen Verzeichnisse werden durch die Applikation genutzt. Innerhalb aller dieser genannten Verzeichnisse sollte ein Löschen nur mit hinreichend „gealterten“ Datei stattfinden. Manuell geöffnete Dateien können hier zu Problemen führen.

Nifi startet außerdem das Verarbeitungsprogramm Xtool4.exe durch Prozessaufruf. Pro Aufruf wird exakt eine E-Mail prozessiert.

Weiter Anforderungen:

- Eine Instanz einer MS-SQL-Server Datenbankinstanz pro Umgebung
- In der Testumgebung wird die Datenbank NiFi benutzt
 - o Tabelle 1: MAOKA_Sendungsnummern
 - o Tabelle 2: MAOKA_Uebersetzung
- In der Produktionsumgebung wird die Datenbank NiFi_Prod benutzt
 - o Tabellenamen sind identisch

3. Notwendige Installation (Setup Runtime Environment)

3.1 Test-Umgebung

Zunächst wurde zu Entwicklung und Testzwecken eine Testumgebung aufgebaut:
Dazu gehört

Basis:

- 1 Windows 2016 Server (10.131.107.71)
- DotNet Core 5.x Runtime Environment installiert
- Java 8 Runtime Environment installiert
- Das Eingangsverzeichnis der Test-Umgebung ist gemountet unter: V:\
- Installation der NiFi Software unter: C:\NiFiEnvironment\nifi-1.13.0
 - o nifi-1.13.0 auf Port 29443
 - o nifi-registry-0.8.0 auf Port 38443
 - o nifi-toolkit-1.13.0 (hier liegen die Zertifikate im Unterverzeichnis Target)
- Installation des Programmes unter:
 - o D:\AOKAPP\Bin
 - o Verzeichnisse sind angelegt unter D:\AOKAPP\base

SQL Server:	DS6D-DB-6B226.aoksan.aok
Datenbankinstanz:	NiFi
User:	SVC_NiFi

3.2 Produktionsumgebung:

TBD nach der Installation.

3.3 Alle Umgebungen

Für alle Umgebungen gilt:

Es gibt ein besonderes Verzeichniss das angelegt und überwacht werden muss:

1. Das Fehlerverzeichnis:

Dieses Verzeichnis wird im Fehlerausgangsmodul von Nifi genutzt, um bei einem technischen Fehler (Programmabbruch) die E-Mail-Datei abzulegen. Ein File in diesem Verzeichnis bedeutet einen technischen Fehler oder einen Programmabbruch. In jedem Falle ist ein manuelles Eingreifen und eine Einzelfallbetrachtung erforderlich.

Gegebenenfalls sind das Applikationsprotokoll und das Nifi Protokoll (C:\NiFiEnvironment\nifi-1.13.0\logs\) zur Fehleranalyse hinzuzuziehen.

Gegenwärtig in Test: D:\AOKAPP\Base>Error
(auf dem localhost)

Gegenwärtig in Produktion: TBD
(auf dem localhost)

4.0 Technische Dokumentationen Nifi

4.1 Nifi

Apache NiFi ist als Big Data System ein Tool für reibungslosen Dataflow. Dataflows sind Prozessketten, die Daten entgegennehmen, verarbeiten und weiterleiten. Ein einfaches Beispiel ist die automatische Einordnung von E-Mails. Weitere Beispiele wären Geschäftsabläufe zu Optimieren oder das Kundenverhalten auszuwerten, um Kosten zu senken, Risiken zu minimieren und die Gewinne zu vergrößern.

Apache NiFi ist ein Tool, das Daten nicht nur aus unterschiedlichen Quellen einsammeln kann, sondern im Standardumfang zahlreiche Transformationsmöglichkeiten bietet und viele unterschiedliche Ausgabekanäle. Bei diesem Projekt geht es sehr oft um die Einsammlung, Verarbeitung und Verteilung von XML-Daten.

Des Weiteren bietet Apache NiFi ein umfangreiches Angebot an Prozessoren für sehr viele Schnittstellen und Transformationen. NiFi bietet dabei ein Graphisches Frontend mit den zugehörigen Prozessoren. Es ist komplett Open Source und in Java geschrieben. Somit sind auch eine manuelle Erweiterung oder Anpassung jederzeit möglich.

Zum Betrieb der NiFi Anwendung ist eine „Java Runtime Umgebung“ notwendig. Gegenwärtig werden die Version 8 und Version 11 unterstützt. Will man allerdings auch das Registry einsetzen, so empfiehlt sich Java 8 JRE in der aktuellen Fassung (gegenwärtig 18.03.21 Version 101).

4.2 Nifi Registry

Das NiFi Registry ist ein separater JRE Prozess auf einem separaten Port, der eine Versionierung der Data Flows ermöglicht. Des Weiteren können über das Registry Strukturen verschiedener NiFi Server ausgetauscht werden. Dies geschieht über die Anbindung der jeweiligen Server mit dem Austausch Registry.

4.3 Nifi Toolkit

Das NiFi Toolkit beinhaltet alle notwendigen „Programme“ um die Konfiguration an die lokalen Sicherheitsbedürfnisse anzupassen.

Dabei stehen folgende Möglichkeiten zur Verfügung:

- lokale Zertifikate
- Erstellte Zertifikate
- LDAP Anbindung
- Kerberos Anbindung

Diese Umgebung verwendet lokal erstellte Zertifikate. (Details siehe unter Konfiguration:)

4.4 Installation

Die Installation der NiFi Umgebung setzt, wie bereits beschrieben, eine JRE der Version 8 voraus. Das Setzen der Variablen JAVA_HOME ist sehr sinnvoll. Alternativ kann die Datei java.exe auch in den PATH mit aufgenommen werden.

Ist Java erfolgreich eingerichtet, so können die 3 Installationsfiles für NiFi, Registry und Toolkit in einem oder verschiedenen Verzeichnissen abgelegt und ausgepackt werden (.zip Files).

Sind keine weiteren Sicherheitsoptionen gewünscht, so können NiFi und der Registry Server direkt gestartet werden. Dies geschieht mittels eines Skripts:

```
# cd Nififolder\bin  
# run-nifi.bat
```

bzw:

```
# cd Registry\bin  
# run-nifi-registry.bat
```

Weitere Konfigurationsmöglichkeiten sind unter 4.6 aufgeführt.

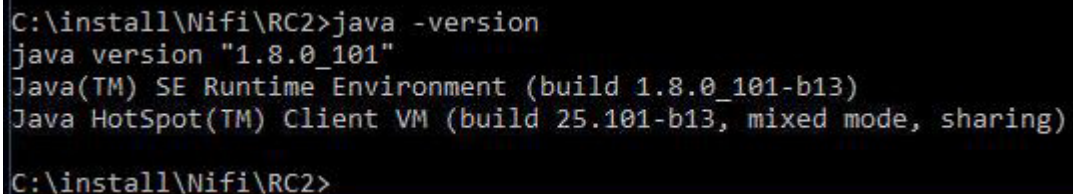
4.5 Konfiguration

4.51 Auf der Command Line

Nachdem die allgemeinen Informationen zu NiFi und der Registry erläutert wurden, kommen hier nun die detaillierten Informationen zur Installation in der AOK Sachsen-Anhalt.

1. In Allen Umgebungen:

Auf beiden Servern (Test:10.131.107.71 und Produktion: 10.131.107.72) ist Java 8 Build 101 installiert und im PATH aufgenommen:



```
C:\install\Nifi\RC2>java -version  
java version "1.8.0_101"  
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)  
Java HotSpot(TM) Client VM (build 25.101-b13, mixed mode, sharing)  
  
C:\install\Nifi\RC2>
```

Des Weiteren wurden auf beiden Servern die Arbeitsverzeichnisse als Laufwerk Z: gemountet. Auf beiden Servern wurden die Ports 29443 für NiFi und 38443 für Registry freigegeben und die 3 Installationsverzeichnisse in einem Folder installiert.

Die Installation und Konfiguration erfolgte auf der Testumgebung in c:\NifiFLTest. Nach erfolgreichem Test wurde der Server und die Registry als 1:1 Kopie auf die Produktion kopiert (unter c:\NifiFLProd)

2. Test-Umgebung:

Hier erfolgte die Installation wie folgt:

Installation:

- Anlegen des Verzeichnisses c:\NifiFLTest
- Kopieren und Auspacken der 3 Installationsfiles im Verzeichnis c:\NifiFLTest

Konfiguration:

- Sichern der Installation mittels lokaler Zertifikate:
- Dazu Wechsel in das Toolkitverzeichnis:
- #> cd toolkit

- Anlegen des Key- und Truststores und Admin Users mittels:
 - bin\tls-toolkit.bat standalone -n "localhost" -C "CN=SAN_SVC_REP, OU=AOKSAN" -o target
- Kopieren der Key- und Truststores (in beide Server „conf“ Verzeichnisse
Aus dem Unterverzeichnis localhost
 - copy keystore.jks C:\NifiFLTest\nifi-registry-0.8.0\conf
 - copy truststore.jks C:\NifiFLTest\nifi-registry-0.8.0\conf und
 - copy keystore.jks C:\NiFiEnvironment\nifi-1.13.0\conf
 - copy keystore.jks C:\NiFiEnvironment\nifi-1.13.0\conf
- Kopieren der generierten nifi.properties in das Nifi Conf Verzeichnis
 - Copy nifi.properties C:\NiFiEnvironment\nifi-1.13.0\conf
- Edit nifi-registry.properties und authorizers in Registry und Nifi
 - Anpassen sind der http (leer setzen) und der https (setzen 29443 und 39443) Port
 - Für die Registry: zusätzlich die Security Einstellungen aus der nifi.properties
- Ggfs. weiterer User:
 - bin\tls-toolkit.bat standalone -C "CN=sys_test, OU=NIFI" -o target
- Verschlüsseln der conf Dateien (nifi und registry)
 - Nifi:
 - encrypt-config.bat -b d:\NifiProd\nifi-1.13.0\conf\bootstrap.conf -n d:\NifiProd\nifi-1.13.0\conf\nifi.properties
 - Die Datei bootstrap.conf enthält dabei den Schlüssel und sollte zugriffsgeschützt abgelegt werden
 - Registry:
 - encrypt-config.bat --nifiRegistry -b d:\NifiProd\nifi-registry-0.8.0\conf\bootstrap.conf -r d:\NifiProd\nifi-registry-0.8.0\conf\nifi-registry.properties -v -t key aus_nifi_bootstrap
 - Beispiel für einen Schlüssel aus der Bootstrap.conf
 - 4DD5C4780E10589327FEC52E2DC7D824

3. Produktionsumgebung: TBD

4.5.2 Die Konfiguration im NiFi Gui:

Anzulegen ist eine Prozessgruppe mit dem Namen MyAOKApp Schnittstelle:

Darin sind anzulegen:

- Ein „GetFile“ Prozessor um jede Minute eine Datei mit dem Namen „check“ aus dem Verzeichnis base\check in das Input Verzeichnis zu kopieren um damit die Routine der Filesystemkontrolle zu starten. Dabei bleibt die Datei im „check“-Verzeichnis erhalten. Der Inputfolder wird in diesem Nifi Folder definiert.
Dies triggert die Überprüfung des Abschlusses der Ablagefolder mit der Endung „.rdy“ zur Weiterverarbeitung mit UC4 nach entweder 30 (konfigurierbar) E-Mail oder 30 min (konfigurierbar) Alter der Verzeichnisse.
- Ein GetFile Prozessor der mittels XML Datei oder Check File Verarbeitungsprozess anstößt.

- Ein ExecuteStreamProzessor der das eigentliche Programm startet und die Umgebung verwaltet
- Ein PutFile Prozessor zur Ablage eines ggfs. Auftretenden technischen Fehler (Errorhandling (Ablage Datei im Errorfolder)

4.6 Nifi Referenzinformationen aus dem Internet

Apache NiFi Walkthroughs Allgemeine Informationen:

<https://nifi.apache.org/docs/nifi-docs/html/walkthroughs.html>

Security Konfiguration:

<https://nifi.apache.org/docs/nifi-docs/html/walkthroughs.html#securing-nifi-with-tls-toolkit>

Der allgemeine Admin Guide

<https://nifi.apache.org/docs/nifi-docs/html/administration-guide.html>

User Identifikation:

https://nifi.apache.org/docs/nifi-docs/html/administration-guide.html#user_authentication

Alles andere zu Nifi:

<https://nifi.apache.org/docs.html>

5.0 Technische Dokumentationen MeineAOKApp Nifi Schnittstelle

5.1 Allgemeine Informationen

Alle Verarbeitungen sind in der Sprache C# implementiert und durch eine NiFi Funktion gestartet. In NiFi erfolgt keinerlei Verarbeitung (Ausnahme Austeuerung der Originaldatei und ggfs. Der Fehlerbehandlung).

5.2 Prozeduraler Ablauf:

1. Ein Kunde der AOK sendet ein Dokument über die MeineAOKApp an die AOK
2. Ein Dienstleister bereitet das Dokument als PDF auf und erstellt ein XML Begleitdokument. Beide Dokumente bekommen dabei ein gleichlautende UID als Namen. Parallel dazu wird eine .RDY gleichen Namens als Trigger Datei abgelegt.
3. Dieses Dokumentenpärchen wird in der Reihenfolge PDF dann XML im Eingangsverzeichnis angeliefert.
4. An dieser Stelle startet Nifi, dass die Eingangsfolder auf RDY Dateien überwacht, die Verarbeitung mittels der neuen Schnittstelle
5. A) Nifi checkt in Intervallen das Eingangsverzeichnis und startet die Verarbeitung der eingegangenen XML Dateien
B) Die „check“ Datei wird von NiFi in das „input“ Verzeichnis geschoben
6. Der NiFi Filewatcher sieht die Datei im „input“ Verzeichnis, startet das Verarbeitungsprogramm und „übergibt“ den Filenamen als Parameter.
7. Das Programm checkt nun das Eingangsfile auf seinen Namen

8. A) Ist der Name „check“, dann wird der Ablageordner auf Verzeichnisse kontrolliert, die entweder eine konfigurierbare Anzahl Sendungen enthalten oder älter als 30 Minuten sind. In diesem Falle wird das Verzeichnis gesichert und mit der Endung mit einer konfigurierbaren Endung versehen. Dies kennzeichnet den Abschluss der NiFi Verarbeitung und ist die Übergabe an UC4 zur Weiterverarbeitung in CrossCap. Somit sollten keine unverarbeiteten Stapel größer als 30 Mails oder keine Stapel älter als 30 Minuten existieren.

Folgende Verzeichnisnamen werden dabei zurzeit ignoriert (Dies ist konfigurierbar):

- leere Verzeichnisse
- Verzeichnisse mit der Endung .bsy
- Verzeichnisse mit der Endung .rdy
- Verzeichnisse mit der Endung .error
- Verzeichnisse mit der Endung .sendnumbererror
- Verzeichnisse mit der Endung .importererror

B) Ist das übergebene File eine valide XML Datei, dann folgt(6.):

9. Werden die formalen Voraussetzungen durch das Programm geprüft:
 (In diesem Bereich ist die Wahrscheinlichkeit von technischen Fehlern am höchsten)
- Korrekter Aufruf
 - Parameterzahl, etc.
 - Umgebungsparameter
 - Datenbankverbindungen
10. Die XML-Mail wird ausgelesen und die Werte in die Reportingdatenbank übertragen.
11. Bei einer Verarbeitung der Dokumente wird im Ausgabeverzeichnis eine XML Datei mit den Informationen zum PDF Dokument und das PDF Dokument mit neuen Namen abgelegt und in Stapelstrukturen organisiert
12. Im Falle eines technischen Fehlers gibt es 1 mögliche Ausgabe, die regelmäßig kontrolliert werden muss:

B) Das „error“ Verzeichnis. Im Falle eines Programmabbruches mit Fehlermeldung oder eines Programmabsturzes liegt in diesem Verzeichnis die XML-Datei, die den Abbruch verursacht hat.

In diesem Fall ist ein manueller Eingriff notwendig, da keine Verarbeitung stattgefunden hat.

5.3 Die Datenbank

Zur Applikation gehört zwei Datenbanken, die dazu dienen, bestimmte während der Laufzeit anpassbare Werte vorzuhalten oder Reportingdaten aufzunehmen. Diese Werte steuern die Verarbeitung eines Dokumentes z.B. durch Ermittlung eines Dokumententyps auf Basis eines Wertes in der XML Datei als Schlüssel für die Datenbank.

Diese Datenbank und ihr Inhalt wird durch die AOK vorgegeben und kann jederzeit aktualisiert werden. Ein Restart der Applikation ist dazu nicht notwendig. Die Umgebung muss aber zur Laufzeit erreichbar sein, sonst entsteht ein technischer Fehler, der manuell verarbeitet werden muss.

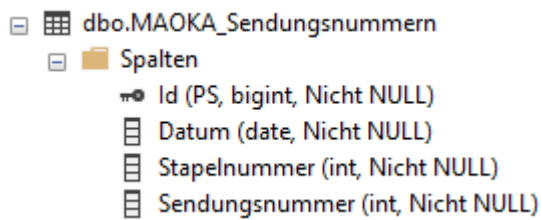
Benötigt wird hierzu eine Instanz eines Microsoft SQL Servers, die aus der jeweiligen NiFi Umgebung heraus zu erreichen sein muss.

Die Anwendung besteht aus 2 Tabellen:

- MAOKA_Sendungsnummern
- MAOKA_Uebersetzung

MAOKA_Sendungsnummern:

Mittels dieser Tabelle wird protokolliert und berechnet welche Stapel- und Sendungsnummern bereits vergeben und welche die nächsten sein werden. Diese Tabelle muss nicht vorbefüllt werden.



dbo.MAOKA_Sendungsnummern
Spalten
Id (PS, bigint, Nicht NULL)
Datum (date, Nicht NULL)
Stapelnummer (int, Nicht NULL)
Sendungsnummer (int, Nicht NULL)

Skript:

```
USE [NiFi]
GO

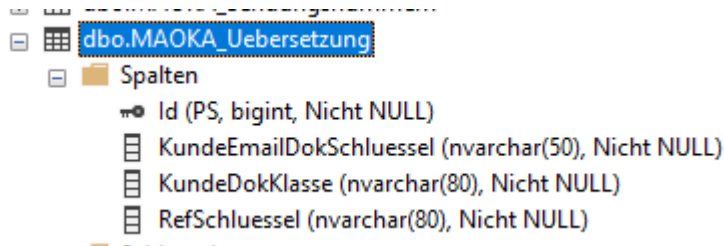
/***** Object: Table [dbo].[MAOKA_Sendungsnummern]    Script Date: 02.05.2022
12:24:08 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[MAOKA_Sendungsnummern](
    [Id] [bigint] IDENTITY(1,1) NOT NULL,
    [Datum] [date] NOT NULL,
    [Stapelnummer] [int] NOT NULL,
    [Sendungsnummer] [int] NOT NULL,
    PRIMARY KEY CLUSTERED
    (
        [Id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, AL-
    LOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

MAOKA_Uebersetzung

Diese Tabelle wird vorausgefüllt und liefert die YLAMI-Klasse und den Beschreibungstext für einen „RefSchlüssel“. Der RefSchlüssel kommt dabei aus der XML Datei



dbo.MAOKA_Uebersetzung	
Spalten	
Id (PK, bigint, Nicht NULL)	
KundeEmailDokSchluessel (nvarchar(50), Nicht NULL)	
KundeDokKlasse (nvarchar(80), Nicht NULL)	
RefSchluessel (nvarchar(80), Nicht NULL)	

- KundeEmailDokSchlüssel beinhaltet die Dokumentenart.
- KundeDokKlasse beinhaltet die Beschreibung der Dokumentenart

Skript:

```
USE [NiFi]
GO

/***** Object: Table [dbo].[MAOKA_Uebersetzung]    Script Date: 02.05.2022
12:23:59 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[MAOKA_Uebersetzung](
    [Id] [bigint] IDENTITY(1,1) NOT NULL,
    [KundeEmailDokSchluessel] [nvarchar](50) NOT NULL,
    [KundeDokKlasse] [nvarchar](80) NOT NULL,
    [RefSchluessel] [nvarchar](80) NOT NULL,
    PRIMARY KEY CLUSTERED
    (
        [Id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, AL-
    LOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Datensätze zum Produktionsstart:

```
USE [NiFi]
GO

INSERT INTO [dbo].[MAOKA_Uebersetzung]
    ([KundeEmailDokSchluessel]
    , [KundeDokKlasse]
    , [RefSchluessel])
VALUES
    ('ZISCMIO031'
    , 'APD 0031 KIKG CM_ANTRAG_KINDERKRANKENGELD'
    , 'KIKG')
GO

INSERT INTO [dbo].[MAOKA_Uebersetzung]
    ([KundeEmailDokSchluessel]
```

```
        , [KundeDokKlasse]
        , [RefSchluessel])
VALUES
    ( 'ZISCMIGAUB'
      , 'APD GAUB KG AU_BESCHEINIGUNG '
      , 'AU' )
GO

INSERT INTO [dbo].[MAOKA_Uebersetzung]
    ([KundeEmailDokSchluessel]
     , [KundeDokKlasse]
     , [RefSchluessel])
VALUES
    ( 'YLAMI072'
      , 'Keine Vorgabe'
      , 'DIGA' )
GO
```

6.0 Anhang

6.1 Dokumentation der Anforderungen

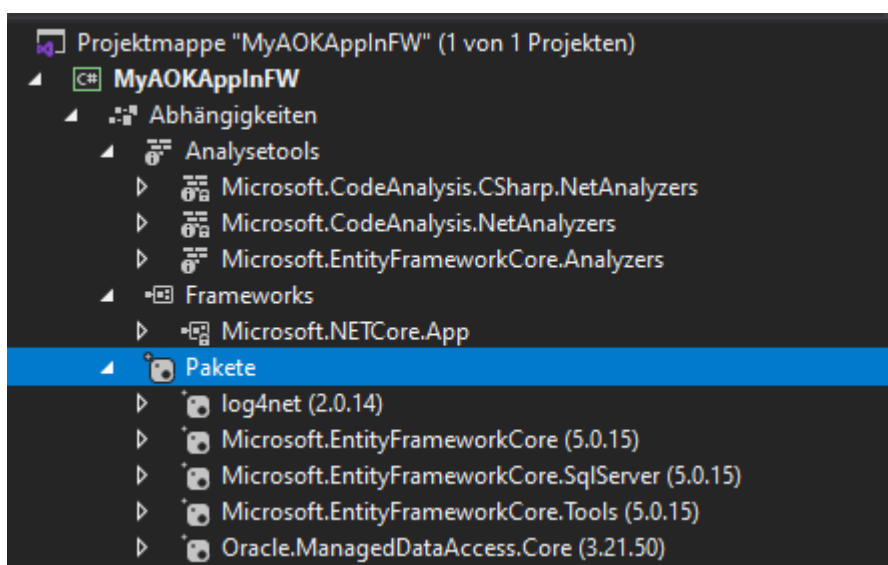


20220519_Technisc
hes Anforderungskc

6.2 Das Programm

Allgemein: Das Programm nutzt das .NET Core Framework 5.0 und das Entity Framework Core 5 zur Datenverarbeitung.

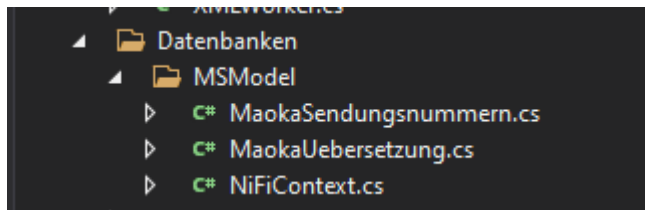
Für Loggingzwecke wird das Log4Net Framework angewandt.



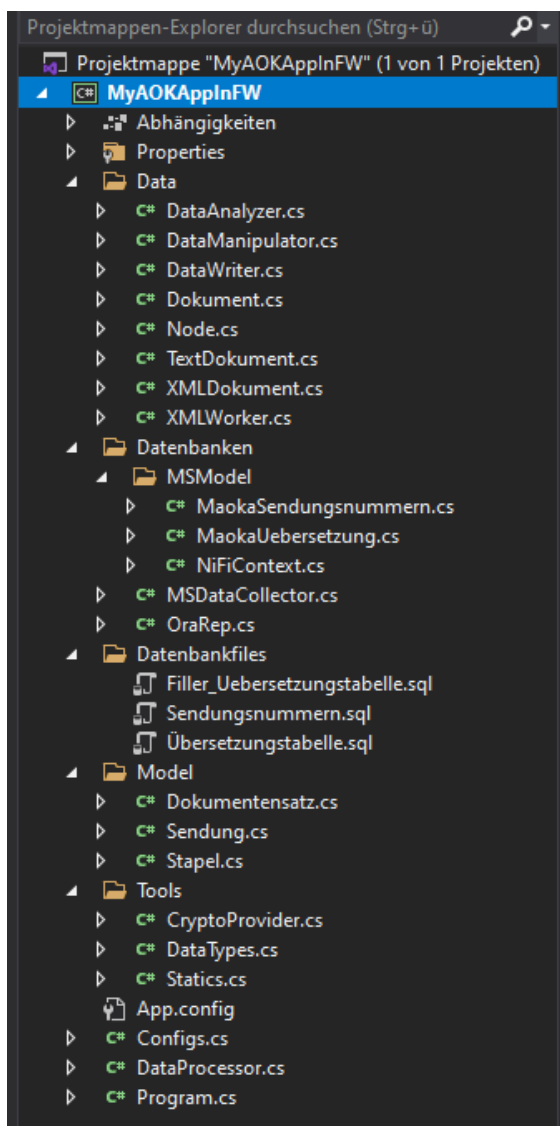
Damit ist die obige Liste die Referenz der benutzten Frameworks zur Erstellung des Programms.

Die Dateien im Verzeichnis Daten sind mittels EF 5 Framework aus der Datenbank „reverse engineered“ und werden für die Kommunikation zwischen Programm und DB benutzt.
Alle anderen Klassen dienen der Datenverarbeitung.

Die Klassen:



Das Programm:



- Die Klasse Program stellt den Einstiegspunkt des Programms dar und stellt die High-Level Verarbeitung. Sie startet die Konfigurationsanalyse in der Klasse Config und die Datenverarbeitung im DataProcessor.
- Allgemein benutzte Funktionen sind in der Klasse Statics abgebildet. Hier liegen nur statische Funktionen.
- Die Klassen unter Model und Datenbanken dienen der Interaktion mit den Datenbanken für Sendungsnummern und Reporterstellung.
- Die Klasse Datenmanipulation dient der Verarbeitung und Anpassung der XML Datei.
- Die Klasse Datenanalyse dient der Ausgabe der XML Datei.
- Die Klasse Datenanalyse dient dem Auslesen der XML Datei.
- Die restlichen Klassen sind ein Framework zur automatisierten Verarbeitung von XML Dateien im AOK Umfeld.
- Die Klasse CryptoProvider ist für das ver- und entschlüsseln der Datenbankpasswörter zuständig.
- Die Klasse DataTypes enthält die möglichen Fehlercodes und ermöglicht die Ausgabe aller definierten Fehlercodes.

Die KonfigDatei App.Config:

Diese Datei passt den Programmablauf jeweils an die Umgebungen an.

Passwörter müssen in dieser Datei verschlüsselt angegeben werden!

Editierbare Punkte:

1. Ablage der Protokolldatei:
Hier muss der Pfad der Umgebung angepasst werden.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <!--Definition des Logwriters-->
  <configSections>
    <section name="log4net" type="log4net.Config.Log4Net.ConfigurationSection-
Handler, log4net"/>
  </configSections>
  <!--Konfiguration des Log Writers-->
  <log4net>
    <appender name="RollingFileAppender" type="log4net.Appender.RollingFileAp-
pender">
      <file value="D:\AOKAPP\Base\Protokoll\protokoll.log" />
      <appendToFile value="true" />
      <rollingStyle value="Size" />
      <maxSizeRollBackups value="5" />
      <maximumFileSize value="100MB" />
      <staticLogFileName value="true" />
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date [%thread] %level %logger -
%message%newline" />
      </layout>
    </appender>
    <appender name="Console" type="log4net.Appender.ConsoleAppender">
      <param name="target" value="System.err"/>
      <threshold value="Error" />
    </appender>
  </log4net>
</configuration>
```

```

        <layout type="log4net.Layout.PatternLayout">
            <conversionPattern value="%date [%thread] %level %logger -
%message%newline" />
        </layout>
    </appender>
    <root>
        <level value="DEBUG"/>
        <appender-ref ref="RollingFileAppender" />
        <appender-ref ref="Console" />
    </root>
</log4net>

<appSettings>
    <!--Arbeitsverzeichnisse-->
    <add key="ErrorFolder" value="D:\AOKAPP\Base\Error"/>
    <add key="BackupFolder" value="D:\AOKAPP\Base\Backup"/>
    <add key="ClosingFolderNameExceptions" value=".rdy,.done,.bsy,.er-
ror,.sendnumbererror,.importererror,.importererr"/>
    <add key="StapelZeit" value="30"/>
    <add key="StapelSize" value="30"/>
    <!--Datenbankverbindung und User encoded-->
    <!--Scaffold-DbContext "Server=DS6D-DB-6B226.aoksan.aok;Database=NiFi;User
Id=SVC_NiFi;Password=Zg?ELj3dbV3%" Microsoft.EntityFrameworkCore.SqlServer -OutputDir
Datenbanken\MSModel-->
    <add key="MSSqlUser" value="SVC_NiFi"/>
    <add key="MSSqlPasswort" value="Ati/JJJG2PjtJ7PrtbztCnU1du-
xUvElqcYI4Weqyng0="/>
    <add key="MSConnectionString" value="Data Source=DS6D-DB-
6B226.aoksan.aok;Initial Catalog=NIFI"/>
    <!--Reporting-->
    <add key="ErrorFolderForOracle" value="D:\AOKAPP\Base\OraErr" />
    <add key="SQLFolderForOracle" value="D:\AOKAPP\Base\OraSQL" />
    <add key="SQLFileName" value="Reporting.SQL" />
    <!--Ora Report DB Connector-->
    <add key="OraReport_Connector" value="User Id=REPORTING_WRITE;Pass-
word=io+AKONKSPRnur0NF1EWSGH/k5fGE7scGy2exBxLDGU=;Data Source=10.108.209.77:1538/lipms-
tat.aokt" />
    <!--Ausgabeverzeichnisse-->
    <add key="AugabeVerzeichnis" value="U:\"/>
    <add key="M1Kennung" value="LQA"/>
    <add key="M1StapelKennung" value="1"/>
    <add key="M2Kennung" value="LQB"/>
    <add key="M2StapelKennung" value="2"/>
    <add key="M3Kennung" value="LQC"/>
    <add key="M3StapelKennung" value="3"/>
    <add key="ProdKennung" value="LQP"/>
    <add key="ProdStapelKennung" value="P"/>
    <!-- *** Endung zur Erkennung der Weiterverarbeitung eines Stapel -->
    <add key="Finalkennung_Folder" value=".rdy"/>
    <!-- *** Variablen für den Stapelnamen*** -->
    <add key="Kassennummer" value="97"/>
    <add key="Beleglesedienstleister_ID" value="A"/>
    <add key="Vinna_Eingangskanal" value="AP"/>
    <add key="E_Akte_Eingangskanal" value="E"/>
</appSettings>

```

`</configuration>`

Erklärung der Konfigurationselemente:

ErrorFolder:

Allgemeiner Fehlerordner für den Fall, das das Programm eine Fehlerdatei erzeugen soll.
Zurzeit nicht in Benutzung.

BackupFolder:

Hier wird jeder Ausgangstapel bei der Übergabe an Crosscap gesichert.

ClosingFolderNameExceptions

Alle Kennungen, die hier kommasepariert aufgelistet sind, erfolgt keine Stapelweiterverarbeitung. Dies dient sowohl der automatisierten wie manuellen Nacharbeit.

StapelZeit:

Dies ist die Zeitdauer in Minuten, die gewartet wird, bevor ein Stapel zur Weiterverarbeitung zur Verfügung gestellt wird.

StapelSize:

Dies ist die Anzahl der Elemente die in einem Stapel maximla gesammelt werden, bevor eine Weiterverarbeitung erfolgt.

ErrorFolderForOracle:

Wenn Oracle einer Fehlermeldung verursacht erscheint diese in diesem Ordner.

SQLFolderForOracle:

Wenn keine Reportingdaten geschrieben werden konnten, erfolgt eine Dateiausgabe der Daten in diesem Ordner.

SQLFileName

Die Oracle SQL Datei mit den Reportingdaten erhält diesen Namen. Es beinhaltet ein Skript , das alle Reporting Daten nachlaufbar macht.

AugabeVerzeichnis:

Dies ist das allgemeine Ausgabeverzeichnis für verarbeitete Stapel. Von hier werden die Daten mit UC4 angeholt, um sie für Crosscap zur Verfügung zu stellen.

M1Kennung

Der Namensbestandteil des Eingangsverzeichnisses, der die Umgebung M1 kennzeichnet

M1StapelKennung:

Der Namensbestandteil des Stapelnamens der die Umgebung M1 kennzeichnet

M2Kennung

Der Namensbestandteil des Eingangsverzeichnisses, der die Umgebung M2 kennzeichnet

M2StapelKennung

Der Namensbestandteil des Stapelnamens der die Umgebung M2 kennzeichnet

M3Kennung

Der Namensbestandteil des Eingangsverzeichnisses, der die Umgebung M3 kennzeichnet

M3StapelKennung

Der Namensbestandteil des Stapelnamens der die Umgebung M3 kennzeichnet

ProdKennung:

Der Namensbestandteil des Eingangsverzeichnisses, der die Umgebung Produktion kennzeichnet

ProdStapelKennung:

Der Namensbestandteil des Stapelnamens der die Umgebung Produktion kennzeichnet

Der Namensbestandteil des Stapelnamens der die Umgebung M1 kennzeichnet

Finalkennung_Folder

Die Namensextension des Stapels die er bei der Übergabe an Crosscap erhält

Beleglesedienstleister_ID

Fester Wert der den Beleglesedienstleister im Stapelnamen kennzeichnet

Vinna_Eingangskanal

Fester Wert der den Vinna_Eingangskanal im Stapelnamen kennzeichnet

E_Akte_Eingangskanal

Fester Wert der den E_Akte_Eingangskanal im Stapelnamen kennzeichnet

6.3 Klassenreferenz:

Siehe Repository.

6.4 Die Datei App.Config

Siehe Repository.

Als Schnellreferenz die App.Config zum Dokumentenerstellzeitpunkt.



App.config

6.5 Das Programm Repository:

https://vrs2d-so-1e097.aoksan.aok/svn/NIFI_MYAOKAPP

6.5 Der Anpassungsprozess für Änderungen der Datenbanktabellen:

Anpassen neuer Dokumententype

Neue Einträge oder Änderungen können jederzeit in der Tabelle durchgeführt werden und bedürfen keines Neustarts der Applikation.

Technische ToDo:

Für alle Tabellen gilt, dass bei Änderungen des Inhaltes zu prüfen ist, ob eine Programmanpassung notwendig ist. Eine generische Aussage kann hier nicht gemacht werden, da die Bedeutungen von beliebigen Änderungen nicht prognostiziert werden können.

Das Update kann manuell oder per Skript erfolgen. Eine Referenz der durchgeführten Änderungen sollte dokumentiert werden.

Technisch gesehen ist ein Datenbankupdate im laufenden Betrieb möglich, aber nicht ratsam.

Referenzen:

Datenbanklogin Test:

SQL-Server:	DS6D-DB-6B226.aoksan.aok
Datenbankinstanz:	NiFi
User:	SVC_NiFi
Password:	Zg?ELj3dbV3%

Datenbanklogin Prod:

SQL-Server:	DS6D-DB-6B226.aoksan.aok
Datenbankinstanz:	NiFi_Prod
User:	SVC_NiFi
Password:	Zg?ELj3dbV3%

Erstellung eines gecrypteten Passwords:

Das Programm enthält eine Funktion zum ver- und encrypten von Passwörtern.

Der Aufruf erfolgt wie folgt in der Kommandozeile:

```
># MyAOKAppInFW.exe c Passwort
```

Dass Programm zeigt der vercryptete Passwort an.

Zum Entschlüsseln kann man die Funktion d verwenden:

```
># MyAOKAppInFW.exe d EncryptedPassword
```

Dass Programm zeigt das originale Passwort an.