

# Hausarbeit

openMPS,

Erfassung und Übermittlung von Daten von Multifunktionsgeräten

Von Andreas Schreiner

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>3</b>
<b>2 Erläuterungen zu verwendeten Technologien</b>	<b>4</b>
2.1 Simple Network Management Protocol . . . . .	4
2.2 SocketIO . . . . .	6
<b>3 Pflichtenheft</b>	<b>7</b>
3.1 Zielbestimmung . . . . .	7
3.1.1 Kurzbeschreibung . . . . .	7
3.1.2 Muss-Kriterien . . . . .	7
3.1.3 Wunschkriterien . . . . .	8
3.2 Produktfunktionen . . . . .	8
<b>4 Datenbanken</b>	<b>9</b>
4.1 Übersicht der genutzten Datenbank . . . . .	9
4.2 Detailansicht der genannten Datenbanken . . . . .	10
4.2.1 Konfigurationsdatei	
Config.db . . . . .	10
4.2.2 OpenMPS Server Datenbank . . . . .	11
4.2.3 fnLog Server Datenbank . . . . .	12
4.2.4 Manastone Client Datenbank	
manastone.db . . . . .	13
4.2.5 Manastone Server Datenbank . . . . .	14
4.2.6 fnLog . . . . .	15
<b>5 Klassendiagramme</b>	<b>16</b>
5.1 openMPS . . . . .	16
5.2 de.fearvel.net . . . . .	17
<b>6 Sonstiges</b>	<b>18</b>
6.0.1 Aktivierung einer openMPS Installation	
jetziger Ablauf . . . . .	18
6.0.2 Übertragung der OID Tabelle an den openMPS Client . . . . .	19
6.0.3 Übertragung der OID Werte an den openMPS Server . . . . .	19
<b>7 Code Ausschnitte</b>	<b>20</b>
<b>8 Benutzerhandbuch</b>	<b>42</b>
8.1 Erststart und Aktivierung . . . . .	42
8.2 Automatische Suche . . . . .	43
8.2.1 Suchen von Geräten . . . . .	43
8.3 Manuelle Bearbeitung von Geräten . . . . .	44
8.3.1 Hinzufügen von Geräten . . . . .	44
8.3.2 Editieren von Geräten . . . . .	44
8.3.3 Löschen von Geräten . . . . .	44
8.4 Werte abfragen und senden . . . . .	45
8.5 Infoseiten . . . . .	46



# Kapitel 1

## Einleitung

In unserer sich stets modernisierenden Welt existieren Praktiken, die sich über Jahrzehnte bewährt haben, die jedoch in der heutigen Zeit als ineffektiv und zeitraubend betrachtet werden können.

Das openMPS-System widmet sich einem dieser Umstände, indem es die Erfassungszeiten der Daten eines Multifunktionsgerätes signifikant verringert und das Manipulations- und Fehlerpotenzial zudem weiträumig einschränkt. Zudem bietet das openMPS-System eine sichere Plattform, welche über Manastone DRM bereitgestellt wird. Aufgrund dieser abgesicherten Plattform ist es möglich, das openMPS-System schnell und einfach um Komponenten, welche eine eindeutige manipulationssichere Identifikation benötigen, wie z.B. Online Shop für Toner zu erweitern

Bei diesem Programm wird die Aktivierung des Clients, sowie die Erhebung und der Versand valider Daten im Vordergrund stehen. Die Benutzung dieses Programms muss für den Nutzer ohne Schulung auf dieses Produkt möglich sein.

Das Projekt wird aus drei Teilen bestehen:

- openMPS  
Das Hauptprogramm, welches von den nachfolgenden Bestandteilen abhängig ist.
- Manastone DRM  
Ein eigenes DRM System zur Verwaltung der Nutzbarkeit des openMPS Clients, sowie der Bereitstellung einer sicheren Plattform via Token.
- fnLog  
Ein Telemetrie-System zur Erfassung von Ereignissen lokal und via Server.

# Kapitel 2

## Erläuterungen zu verwendeten Technologien

### 2.1 Simple Network Management Protocol

Das **Simple Network Management Protocol (SNMP)** (deutsch Einfaches Netzwerkverwaltungsprotokoll) ist ein Netzwerkprotokoll, das von der IETF entwickelt wurde, um Netzwerklemente (z. B. Router, Server, Switches, Drucker, Computer usw.) von einer zentralen Station aus überwachen und steuern zu können. Das Protokoll regelt dabei die Kommunikation zwischen den überwachten Geräten und der Überwachungsstation. SNMP beschreibt den Aufbau der Datenpakete, die gesendet werden können und den Kommunikationsablauf. Es wurde dabei so ausgelegt, dass jedes netzwerkfähige Gerät mit in die Überwachung aufgenommen werden kann. Zu den Aufgaben des Netzwerkmanagements, die mit SNMP möglich sind, zählen:

- Überwachung von Netzwerkkomponenten.
- Fernsteuerung und Fernkonfiguration von Netzwerkkomponenten.
- Fehlererkennung und Fehlerbenachrichtigung.

Durch seine Einfachheit, Modularität und Vielseitigkeit hat sich SNMP zum Standard entwickelt, der sowohl von den meisten Managementprogrammen als auch von Endgeräten unterstützt wird.

#### Funktionsweise:

Zur Überwachung werden sogenannte Agenten eingesetzt. Dabei handelt es sich um Programme, die direkt auf den überwachten Geräten laufen, oder um Hardware, welche die gleichen Aufgaben erfüllt.[1] Diese Programme/Geräte sind in der Lage, den Zustand des Netzwerkergerätes zu erfassen und auch selbst Einstellungen vorzunehmen oder Aktionen auszulösen. Mit Hilfe von SNMP ist es möglich, dass die zentrale Managementstation mit den Agenten über ein Netzwerk kommunizieren kann. Dazu gibt es sieben verschiedene Datenpakete, die gesendet werden können:

#### GET-REQUEST

zum Anfordern eines Management-Datensatzes.

#### GETNEXT-REQUEST

um den nachfolgenden Datensatz abzurufen (um Tabellen zu durchlaufen).

#### SET-REQUEST

um einen oder mehrere Datensätze eines Netzelementes zu verändern. Manchmal verlangt ein Netzelement die gleichzeitige Änderung mehrerer Datensätze, um die Konsistenz zu überprüfen. Beispielsweise erfordert die Konfiguration einer IP-Adresse die gleichzeitige Angabe der Netzwerkmaske.

#### GET-RESPONSE

Antwort auf eines der vorherigen Pakete.

## **TRAP**

unaufgeforderte Nachricht von einem Agenten an den Manager, dass ein Ereignis eingetreten ist. Programme wie Wireshark, die zum Dekodieren von Protokollen wie SNMP benutzt werden, nennen dieses Datenpaket auch REPORT. Ein TRAP kann auch geschickt werden, wenn die in einem SET-REQUEST-Paket beschriebene(n) Datensatzänderung(en) nicht durchgeführt werden konnte(n), und nicht nur, um eine Fehlfunktion (z. B. einen Defekt eines Moduls eines Netzelements) zu melden.

## **INFORM-REQUEST**

aufgebaut wie ein Trap, nur dass dieser vom Empfänger quittiert wird. Die drei GET-Pakete (GET, GETNEXT, GETBULK) können vom Manager zu einem Agenten gesendet werden, um Daten über die jeweilige Station anzufordern. Dieser antwortet mit einem RESPONSE-Paket, das entweder die angeforderten Daten oder eine Fehlermeldung enthält.

Mit dem SET-REQUEST-Paket kann ein Manager Werte beim Agenten verändern. Damit ist es möglich, Einstellungen vorzunehmen oder Aktionen auszulösen. Der Agent bestätigt die Übernahme der Werte ebenfalls mit einem GET-RESPONSE-Paket.

Wenn der Agent bei der Überwachung des Systems einen Fehler erkennt, kann er diesen mit Hilfe eines TRAP-Paketes unaufgefordert an die Management-Station melden. Diese Pakete werden nicht vom Manager bestätigt. Der Agent kann daher nicht feststellen, ob das gesendete TRAP-Paket beim Manager angekommen ist.

Damit die Netzwerkbelaistung gering bleibt, wird zum Versenden der Nachrichten das verbindungslose Protokoll UDP verwendet. Der Agent und der Manager kommunizieren (Requests/Responses) auf dem Port 161, während der Port 162 zum Empfangen der TRAP-Meldungen vorgeschrieben ist.

## **Version 1**

Die erste Version von SNMP wurde 1988 in den folgenden RFCs definiert:

- RFC 1155 – Structure and Identification of Management Information for TCP/IP-based internets (Ersetzte die RFC 1065)
- RFC 1156 – Management Information Base for Network Management of TCP/IP-based internets (Ersetzte die RFC 1066)
- RFC 1157 – A Simple Network Management Protocol (Ersetzte die RFC 1067 und die RFC 1098)

Das Hauptproblem der ersten Version ist die unzureichende Sicherheit bei der Übertragung. Eines der Probleme ist die unverschlüsselte Übertragung des Passwortes, dadurch kann es leicht abgehört und durch unberechtigte Parteien verwendet werden.

Auszug aus Wikipedia, siehe Literaturverzeichnis ([Wik19a](#))

## 2.2 SocketIO

Socket.IO ist eine JavaScript-Bibliothek für Echtzeit-Webanwendungen. Es ermöglicht bidirektionale Echtzeit-Kommunikation zwischen Webclients und Servern. Es besteht aus zwei Teilen: einer clientseitigen Bibliothek, die im Browser des Benutzers läuft, und einer serverseitigen Bibliothek für Node.js. Beide Komponenten haben eine nahezu identische API. Wie Node.js auch, ist es ereignisgetrieben.

Socket.IO verwendet primär das WebSocket-Protokoll mit zyklischem Abfragen als Ersatzoption,[1] wobei es dafür dieselbe Schnittstelle bietet. Obwohl es auch als einfacher Wrapper für WebSocket verwendet werden kann, bietet es weitaus mehr Funktionen, einschließlich des Broadcastings an mehrere Sockets, Speichern von Daten, die mit dem jeweiligen Client verknüpft sind, und asynchrone E/A.

Es kann mit Hilfe des Paketmanagers npm installiert werden.

### Überblick

Socket.IO bietet Echtzeit-Analyse, binäre Datenströme, Sofortnachrichten und Zusammenarbeit an Dokumenten. Nennenswerte Nutzer sind unter anderen Microsoft Office, Yammer und Zendesk.

Socket.IO behandelt die Verbindung transparent. Es schaltet automatisch zum WebSocket-Protokoll um, falls dies möglich ist. Dadurch benötigt der Programmierer nur Kenntnisse von Socket.IO.

Socket.IO ist keine WebSocket-Bibliothek mit Ersatzoptionen für andere Echtzeitprotokolle. Es ist eine angepasste Implementierung eines Echtzeittransportprotokolls, aufbauend auf anderen Echtzeitprotokollen. Seine Protokollaushandlungsteile sorgen dafür, dass ein Client, der Standard-WebSocket unterstützt, nicht in der Lage ist, sich mit einem Socket.IO-Server zu verbinden. Und ein Socket.IO-implementierender Client kann nicht mit einem nicht-Socket.IO-basierten WebSocket oder Long Polling Comet-Server kommunizieren. Deshalb erfordert Socket.IO die Verwendung der Socket.IO-Bibliotheken sowohl auf dem Client als auch auf dem Server.

Ab Version 2.0 nutzt Socket.IO als zu Grunde liegende WebSocket-Bibliothek *μWebSockets*.

Auszug aus Wikipedia, siehe Literaturverzeichnis ([Wik19b](#))

# Kapitel 3

## Pflichtenheft

### 3.1 Zielbestimmung

#### 3.1.1 Kurzbeschreibung

Es ist eine Software von Nöten, die die Arbeitsabläufe bezüglich der Erfassung von Daten von Multifunktionsdruck-Systemen vereinfacht.

Durch die Verwendung des Simple Network Management Protocol, kurz SNMP, ist es möglich die benötigten Daten auszulesen, diese werden daraufhin an den openMPS Server übermittelt.

Diese Software muss über ein Kopierschutz-System abgesichert werden, um die Manipulationsmöglichkeiten einzuschränken. Zur Nutzung dieses Programms wird ein gültiger Produktschlüssel benötigt, welcher zur Inbetriebnahme der Software, sowie der Übermittlung der Daten zwingend notwendig ist.

Sollten Fehler während der Nutzung des Programms auftreten, werden diese protokolliert und an den fnLog Server übermittelt.

#### 3.1.2 Muss-Kriterien

- openMPS
  - openMPS Server
    - \* Sicherer Empfang der zuvor erfassten Daten.
    - \* Übermittlung der aktuellen OID Tabelle, nach Prüfung des Tokens.
    - \* Verwendung einer MySQL Datenbank.
  - openMPS
    - \* Eine lokale Konfigurationsdatei, welche abhängig der verwendeten Hardware verschlüsselt wird.
    - \* Integration von fnLog, ein Telemetrie System zur Übermittlung von Fehlerprotokollen und Nutzungsprotokollen.
    - \* Integration von Manastone DRM, ein System zur Lizenzverwaltung.
    - \* Verwaltungsmöglichkeiten, die ein Nutzer ändern können muss.
      - Gerät suchen.
      - Gerät manuell anlegen.
      - Gerät entfernen.
    - \* Erheben der Daten eines Multifunktionsgerätes über SNMP.
    - \* Versand über einen sicheren openMPS Server.

- fnLog
  - fnLog Server
    - \* Empfangen und Verarbeiten von eingehenden Logs.
    - \* Verwendung einer MySQL Datenbank.
  - fnLog Integration in openMPS
    - \* Lokales Aufzeichnen auftretender Fehler, sowie des Nutzungsverhaltens.
    - \* Optionen zum Übertragen der Logs mit folgenden Abstufungen.
      - Lokale Aufzeichnung, keine Übermittlung an den fnLog Server.
      - Automatisches Senden von Fehlern und Warnungen.
      - Automatisches Senden aller Logs.
- Manastone (DRM)
  - Manastone Server
    - \* Aktivierung eines Produktschlüssels.
    - \* Erstellung eines zeitlich begrenzten Tokens.
    - \* Verwendung einer MySQL Datenbank.
  - Manastone Integration in openMPS
    - \* Eingabemöglichkeit eines Produktschlüssels.
    - \* Anfordern und Empfangen eines Aktivierungsschlüssels.
    - \* Anfordern und Empfangen eines Tokens.
    - \* Überprüfung eines Tokens.
    - \* Freischaltung des Produktes.
    - \* Bereitstellen des Tokens an das Hauptprogramm, zwecks der Bildung einer sicheren Plattform.

### 3.1.3 Wunschkriterien

- Automatisches Update der OID Tabelle beim Start des Programms.

## 3.2 Produktfunktionen

Im Folgenden werden die Funktionen dargestellt, die bedienbar sein müssen.

- Aktivierung des Programms
- Geräteverwaltung
  - Gerät hinzufügen
    - \* Automatische Suche
    - \* Manuelles Hinzufügen
  - Gerät löschen
  - Gerät deaktivieren
- Abfrage und Versand
  - Gerät abfragen.
  - Gerätedaten versenden.
- Programminformationen
  - Anzeigen der Versionen der einzelnen Komponenten.

# Kapitel 4

## Datenbanken

### 4.1 Übersicht der genutzten Datenbank

Es werden zwei SQLite Datenbanken im openMPS Client verwendet.

- Config.db  
Speicherung der Nutzerdaten(z.B. die Geräteliste welche bei der Datenerhebung durchlaufen wird), der OID Tabelle, Versionsinformationen, Flags und der Logs über eine Integration von fnLog.
- manastone.db  
Datenbank zur lokalen Verwaltung der für das Manastone DRM System notwendigen Daten.

Diese SQLite Datenbaken sind mit einem hardwareabhängigen Schlüssel gesichert. Bei Erzeugung dieser Datenbanken werden diese mit der CPU1 ID verschlüsselt.

Die Dateiendung wurde trotz Verschlüsslung auf \*.db festgelegt.

Serverseitig werden folgende MySQL Datenbanken, welche zurzeit über einen Node.js Server verwaltet werden, verwendet.

- OpenMPS Server  
Speicherung der aktuellen OID Tabelle, der empfangenen Werten, der Serverlogs und der Speicherung von Key Value Pairs.
- fnLog Server  
Speicherung der erhobenen Logs, der AccessKeys, welche in naher Zukunft verwendet werden, und der Serverlogs.
- Manastone Server  
Komplexe serverseitige Lizenzverwaltung.

## 4.2 Detailansicht der genannten Datenbanken

### 4.2.1 Konfigurationsdatei Config.db

Es existieren folgende Tabellen:

- Oid

In dieser Tabelle werden die in Multifunktionsgeräten vorhandenen OID's für die späteren Abfragen gespeichert. Der openMPS Client bezieht nach der erfolgreichen Aktivierung des Produktschlüssels diese Werte über den openMPS Server.

- Devices

Diese Tabelle wird zur Speicherung der erkannten oder manuell definierten Geräten verwendet.

- Dictionary

Tabelle zur Speicherung von Key Value Pairs.

Diese Tabelle wird z.B. zum Speichern von Versionsdaten verwendet.

- Log

Integration von FnLog, siehe 4.2.6.

- Flags

Tabelle zur Speicherung von Flags in Form von string bool pairs.

Wird in der Abgabeversion noch nicht verwendet!

[1..1]		
Log		
• id	INTEGER	NN (PK)
LogType	int	NN
Title	varchar(200)	NN
Description	TEXT	NN
DateTimeOfIncident	Timestamp	NN
Devices		
Active	BOOL	NN
Ip	varchar(39)	NN
Model	varchar(250)	
SerialNumber	varchar(250)	
AssetNumber	varchar(250)	NN
• id	INTEGER	NN (PK)
HostName	varchar(250)	
Flags		
DiKey	varchar(200)	(AKÜ)
DVal	BOOL	
Dictionary		
DiKey	varchar(200)	(AKÜ)
DVal	TEXT	
Oid		
• id	INTEGER	NN (PK)
VendorName	varchar(100)	NN
OldPrivateId	varchar(100)	NN
ProfileName	varchar(100)	NN
DeviceName	varchar(100)	NN
DeviceType	varchar(100)	NN
Manufacturer	varchar(500)	NN
Model	varchar(500)	NN
SerialNumber	varchar(500)	NN
MacAddress	varchar(500)	NN
IpAddress	varchar(500)	NN
HostName	varchar(500)	NN
DescriptionLocation	varchar(500)	NN
AssetNumber	varchar(500)	NN
FirmwareVersion	varchar(500)	NN
PowerSleep1	varchar(500)	NN
PowerSleep2	varchar(500)	NN
TotalPages	varchar(500)	NN
TotalPagesMono	varchar(500)	NN
TotalPagesColor	varchar(500)	NN
TotalPagesDuplex	varchar(500)	NN
PrinterPages	varchar(500)	NN
PrinterPagesMono	varchar(500)	NN
PrinterPagesColor	varchar(500)	NN
PrinterPagesFullColor	varchar(500)	NN
PrinterPagesTwoColor	varchar(500)	NN
CopyPagesMono	varchar(500)	NN
CopyPagesColor	varchar(500)	NN
CopyPagesFullColor	varchar(500)	NN
CopyPagesTwoColor	varchar(500)	NN
CopyPagesSingleColor	varchar(500)	NN
FaxesSentFaxesReceived	varchar(500)	NN
ScansTotalScansTotalMono	varchar(500)	NN
ScansTotalColor	varchar(500)	NN
ScansCopyMono	varchar(500)	NN
ScansCopyColor	varchar(500)	NN
ScansEmail	varchar(500)	NN
ScansEmailMono	varchar(500)	NN
ScansNet	varchar(500)	NN
ScansNetMono	varchar(500)	NN
ScansNetColor	varchar(500)	NN
LargePagesMono	varchar(500)	NN
LargePagesFullColor	varchar(500)	NN
CoverageAverageBlack	varchar(500)	NN
CoverageAverageCyan	varchar(500)	NN
CoverageAverageMagenta	varchar(500)	NN
CoverageAverageYellow	varchar(500)	NN
BlackLevelMax	varchar(50)	
CyanLevelMax	varchar(50)	
MagentaLevelMax	varchar(50)	
YellowLevelMax	varchar(50)	
BlackLevel	varchar(50)	
CyanLevel	varchar(50)	
MagentaLevel	varchar(50)	
YellowLevel	varchar(50)	

## 4.2.2 OpenMPS Server Datenbank

Es existieren folgende Tabellen:

- Oid

In dieser Tabelle werden die in Multifunktionsgeräten vorhandenen OID's für die späteren Abfragen im Client gespeichert. Diese Tabelle wird zur Initialisierung oder Updates der Client OID's benötigt.

- OidData

Tabelle zur Speicherung der vom Client abgefragten und gesendeten Werte.

- Directory

Tabelle zur Speicherung von Key Value Pairs.

Diese Tabelle wird z.B. zum Speichern von Versionsdaten verwendet.

- ServerLog

Einfache Tabelle zur Speicherung von ServerLogs.

[1..1] Oid			OidData		
• Id	Bigint	NN (PK)	• id	Bigint	NN (PK)
VendorName	Text	NN	CustomerReference	Varchar(300)	NN
OldPrivateId	Text	NN	VendorName	Varchar(300)	NN
ProfileName	Text	NN	Model	Varchar(300)	NN
DeviceName	Text	NN	SerialNumber	Varchar(300)	NN
DeviceType	Text	NN	MacAddress	Varchar(300)	NN
Manufacturer	Text	NN	IpAddress	Varchar(300)	NN
Model	Text	NN	HostName	Varchar(300)	NN
SerialNumber	Text	NN	DescriptionLocation	Varchar(300)	NN
MacAddress	Text	NN	AssetNumber	Varchar(300)	NN
IpAddress	Text	NN	FirmwareVersion	Varchar(300)	NN
HostName	Text	NN	PowerSleep1	Varchar(300)	NN
DescriptionLocation	Text	NN	PowerSleep2	Varchar(300)	NN
AssetNumber	Text	NN	ProfileName	Varchar(300)	NN
FirmwareVersion	Text	NN	DeviceName	Varchar(300)	NN
PowerSleep1	Text	NN	DeviceType	Varchar(300)	NN
PowerSleep2	Text	NN	Manufacturer	Varchar(300)	NN
TotalPages	Text	NN	TotalPages	Bigint	NN
TotalPagesMono	Text	NN	TotalPagesMono	Bigint	NN
TotalPagesColor	Text	NN	TotalPagesColor	Bigint	NN
TotalPagesDuplex	Text	NN	TotalPagesDuplex	Bigint	NN
PrinterPages	Text	NN	PrinterPages	Bigint	NN
PrinterPagesMono	Text	NN	PrinterPagesMono	Bigint	NN
PrinterPagesColor	Text	NN	PrinterPagesColor	Bigint	NN
PrinterPagesFullColor	Text	NN	PrinterPagesFullColor	Bigint	NN
PrinterPagesTwoColor	Text	NN	PrinterPagesTwoColor	Bigint	NN
CopyPagesMono	Text	NN	CopyPagesMono	Bigint	NN
CopyPagesColor	Text	NN	CopyPagesColor	Bigint	NN
CopyPagesFullColor	Text	NN	CopyPagesFullColor	Bigint	NN
CopyPagesTwoColor	Text	NN	CopyPagesTwoColor	Bigint	NN
CopyPagesSingleColor	Text	NN	CopyPagesSingleColor	Bigint	NN
FaxesSentFaxesReceived	Text	NN	FaxesSentFaxesReceived	Bigint	NN
ScansTotalScansTotalMono	Text	NN	ScansTotalScansTotalMono	Bigint	NN
ScansTotalColor	Text	NN	ScansTotalColor	Bigint	NN
ScansCopyMono	Text	NN	ScansCopyMono	Bigint	NN
ScansCopyColor	Text	NN	ScansCopyColor	Bigint	NN
ScansEmail	Text	NN	ScansEmail	Bigint	NN
ScansEmailMono	Text	NN	ScansEmailMono	Bigint	NN
ScansNet	Text	NN	ScansNet	Bigint	NN
ScansNetMono	Text	NN	ScansNetMono	Bigint	NN
ScansNetColor	Text	NN	ScansNetColor	Bigint	NN
LargePagesMono	Text	NN	LargePagesMono	Bigint	NN
LargePagesFullColor	Text	NN	LargePagesFullColor	Bigint	NN
CoverageAverageBlack	Text	NN	CoverageAverageBlack	Bigint	NN
CoverageAverageCyan	Text	NN	CoverageAverageCyan	Bigint	NN
CoverageAverageMagenta	Text	NN	CoverageAverageMagenta	Bigint	NN
CoverageAverageYellow	Text	NN	CoverageAverageYellow	Bigint	NN
BlackLevelMax	Text	NN	BlackLevelMax	Bigint	NN
CyanLevelMax	Text	NN	CyanLevelMax	Bigint	NN
MagentaLevelMax	Text	NN	MagentaLevelMax	Bigint	NN
YellowLevelMax	Text	NN	YellowLevelMax	Bigint	NN
BlackLevel	Text	NN	BlackLevel	Bigint	NN
CyanLevel	Text	NN	CyanLevel	Bigint	NN
MagentaLevel	Text	NN	MagentaLevel	Bigint	NN
YellowLevel	Text	NN	YellowLevel	Bigint	NN
			DataDate	Datetime	NN

ServerLog			Directory		
• Id	Int	NN (PK)	• DKey	Varchar(400)	NN (PK)
SocketId	Varchar(100)	NN	• DValue	Varchar(400)	NN (PK)
Message	Text	NN			
DateOfIncident	Datetime	NN			

### 4.2.3 fnLog Server Datenbank

Es existieren folgende Tabellen

- Log  
Tabelle zur Speicherung von Logs.
- AccessKeys  
Tabelle für AccessKeys zum späteren Auslesen und Verarbeiten der Logs
- ServerLog  
Einfache Tabelle zur Speicherung von ServerLogs.

[1..1]																																																						
<table border="1"><thead><tr><th colspan="5">ServerLog</th></tr></thead><tbody><tr><td>Id</td><td>Int</td><td>NN (PK)</td><td></td><td></td></tr><tr><td>SocketId</td><td>Varchar(100)</td><td>NN</td><td></td><td></td></tr><tr><td>Message</td><td>Text</td><td>NN</td><td></td><td></td></tr><tr><td>DateOfIncident</td><td>Datetime</td><td>NN</td><td></td><td></td></tr></tbody></table>					ServerLog					Id	Int	NN (PK)			SocketId	Varchar(100)	NN			Message	Text	NN			DateOfIncident	Datetime	NN																											
ServerLog																																																						
Id	Int	NN (PK)																																																				
SocketId	Varchar(100)	NN																																																				
Message	Text	NN																																																				
DateOfIncident	Datetime	NN																																																				
<table border="1"><thead><tr><th colspan="5">Log</th></tr></thead><tbody><tr><td>Id</td><td>Int</td><td>NN (PK)</td><td></td><td></td></tr><tr><td>ProgramName</td><td>Varchar(300)</td><td>NN</td><td></td><td></td></tr><tr><td>ProgramVersion</td><td>Varchar(50)</td><td>NN</td><td></td><td></td></tr><tr><td>FnLogClientVersion</td><td>Varchar(50)</td><td>NN</td><td></td><td></td></tr><tr><td>Title</td><td>Varchar(500)</td><td>NN</td><td></td><td></td></tr><tr><td>Description</td><td>Text</td><td>NN</td><td></td><td></td></tr><tr><td>LogType</td><td>Int</td><td>NN</td><td></td><td></td></tr><tr><td>UUID</td><td>Varchar(100)</td><td>NN</td><td></td><td></td></tr><tr><td>DateOfIncident</td><td>Datetime</td><td>NN</td><td></td><td></td></tr></tbody></table>					Log					Id	Int	NN (PK)			ProgramName	Varchar(300)	NN			ProgramVersion	Varchar(50)	NN			FnLogClientVersion	Varchar(50)	NN			Title	Varchar(500)	NN			Description	Text	NN			LogType	Int	NN			UUID	Varchar(100)	NN			DateOfIncident	Datetime	NN		
Log																																																						
Id	Int	NN (PK)																																																				
ProgramName	Varchar(300)	NN																																																				
ProgramVersion	Varchar(50)	NN																																																				
FnLogClientVersion	Varchar(50)	NN																																																				
Title	Varchar(500)	NN																																																				
Description	Text	NN																																																				
LogType	Int	NN																																																				
UUID	Varchar(100)	NN																																																				
DateOfIncident	Datetime	NN																																																				
<table border="1"><thead><tr><th colspan="5">AccessToken</th></tr></thead><tbody><tr><td>Id</td><td>Int</td><td>NN (PK)</td><td></td><td></td></tr><tr><td>Token</td><td>Varchar(200)</td><td>NN</td><td></td><td></td></tr><tr><td>DateOfCreation</td><td>Datetime</td><td>NN</td><td></td><td></td></tr><tr><td>DateOfExpiry</td><td>Datetime</td><td>NN</td><td></td><td></td></tr><tr><td>Notes</td><td>Text</td><td></td><td></td><td></td></tr></tbody></table>					AccessToken					Id	Int	NN (PK)			Token	Varchar(200)	NN			DateOfCreation	Datetime	NN			DateOfExpiry	Datetime	NN			Notes	Text																							
AccessToken																																																						
Id	Int	NN (PK)																																																				
Token	Varchar(200)	NN																																																				
DateOfCreation	Datetime	NN																																																				
DateOfExpiry	Datetime	NN																																																				
Notes	Text																																																					

#### 4.2.4 Manastone Client Datenbank manastone.db

Es existieren folgende Tabellen:

- License  
Beinhaltet lizenz-abhängige Informationen, wie z.B. SerialNumber, ActivationKey, etc.
- Log  
Integration von FnLog, siehe 4.2.6.
- Directory  
Tabelle zur Speicherung von Key Value Pairs.  
Diese Tabelle wird z.B. zum Speichern von Versionsdaten verwendet.

[1..1]																											
<table border="1"><thead><tr><th colspan="4">Directory</th></tr></thead><tbody><tr><td>DKey</td><td>varchar(200)</td><td>(AK0)</td><td></td></tr><tr><td>DVal</td><td>TEXT</td><td></td><td></td></tr></tbody></table>				Directory				DKey	varchar(200)	(AK0)		DVal	TEXT														
Directory																											
DKey	varchar(200)	(AK0)																									
DVal	TEXT																										
<table border="1"><thead><tr><th colspan="4">Log</th></tr></thead><tbody><tr><td>id</td><td>INTEGER</td><td>NN (PK)</td><td></td></tr><tr><td>LogType</td><td>int</td><td>NN</td><td></td></tr><tr><td>Title</td><td>varchar(200)</td><td>NN</td><td></td></tr><tr><td>Description</td><td>TEXT</td><td>NN</td><td></td></tr><tr><td>DateTimeOfIncident</td><td>Timestamp</td><td>NN</td><td></td></tr></tbody></table>				Log				id	INTEGER	NN (PK)		LogType	int	NN		Title	varchar(200)	NN		Description	TEXT	NN		DateTimeOfIncident	Timestamp	NN	
Log																											
id	INTEGER	NN (PK)																									
LogType	int	NN																									
Title	varchar(200)	NN																									
Description	TEXT	NN																									
DateTimeOfIncident	Timestamp	NN																									
<table border="1"><thead><tr><th colspan="4">License</th></tr></thead><tbody><tr><td>id</td><td>INTEGER</td><td>NN (PK)</td><td></td></tr><tr><td>SerialNumber</td><td>varchar(36)</td><td>NN</td><td></td></tr><tr><td>ActivationKey</td><td>varchar(36)</td><td>NN</td><td></td></tr><tr><td>Token</td><td>varchar(36)</td><td>NN</td><td></td></tr><tr><td>CustomerReference</td><td>varchar(100)</td><td>NN</td><td></td></tr></tbody></table>				License				id	INTEGER	NN (PK)		SerialNumber	varchar(36)	NN		ActivationKey	varchar(36)	NN		Token	varchar(36)	NN		CustomerReference	varchar(100)	NN	
License																											
id	INTEGER	NN (PK)																									
SerialNumber	varchar(36)	NN																									
ActivationKey	varchar(36)	NN																									
Token	varchar(36)	NN																									
CustomerReference	varchar(100)	NN																									

## 4.2.5 Manastone Server Datenbank

Es existieren folgende Tabellen:

- Customer

Tabelle zur Speicherung kundenbezogener Informationen.

Beinhaltet eine CustomerReference, die vom Client bezogen werden kann.

- Product

Beinhaltet Informationen zu einem Projekt.

Bietet die Möglichkeit der Zuordnung von SerialNumber zu Produkt.

- Token

Wird zur Speicherung eines zeitlich begrenzten Tokens verwendet.

Dieses Token ist zur Authentifizierung eines Clients bei einem

programmabhängigen Server. z.B. openMPS Client zu openMPS Server.

Dieses Token schafft eine sichere Plattform, die einfach um Funktionen erweitert werden kann. z.B. ein Online Shop.

- Activation

Dient zur Speicherung einer Aktivierung.

Der generierte ActivationKey wird ausschließlich zur Authentifizierung eines Manastone Clients zum Manastone Server verwendet.

- SerialNumber

Wird zur Speicherung eines Produktschlüssels verwendet.

Der Produktschlüssels ist zwingend Produkt- und Kunden-bezogen.

- Directory

Tabelle zur Speicherung von Key Value Pairs.

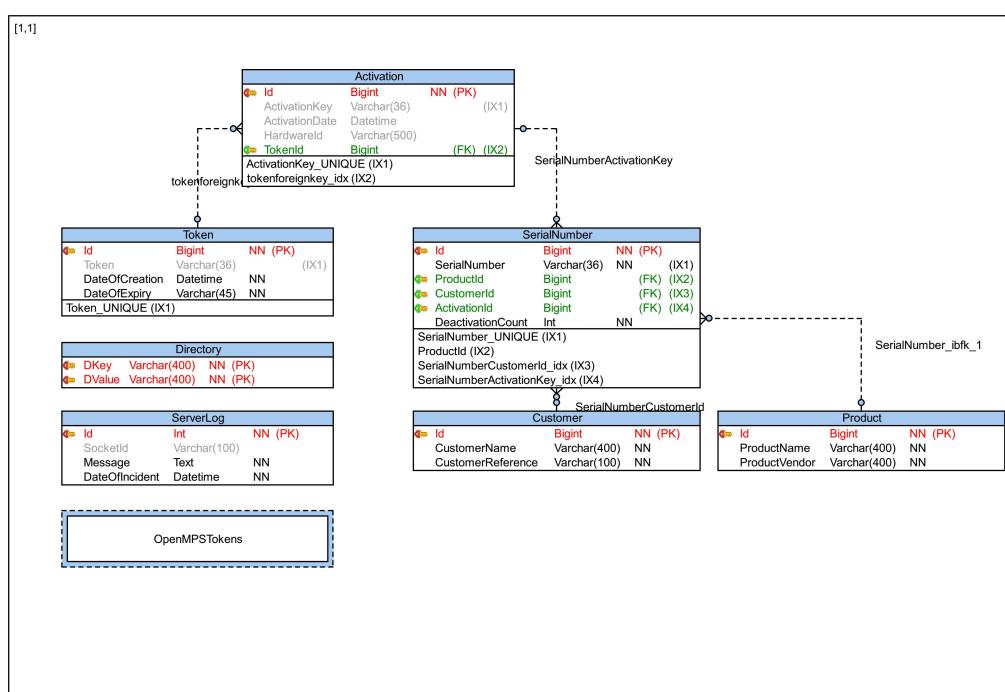
Diese Tabelle wird z.B. zum Speichern von Versionsdaten verwendet.

- ServerLog

Einfache Tabelle zur Speicherung von ServerLogs.

- OpenMPSTokens (VIEW)

Zugriffsmöglichkeit für den openMPS Server auf die zugehörige Token Tabelle.



#### 4.2.6 fnLog

Es existieren folgende Tabellen:

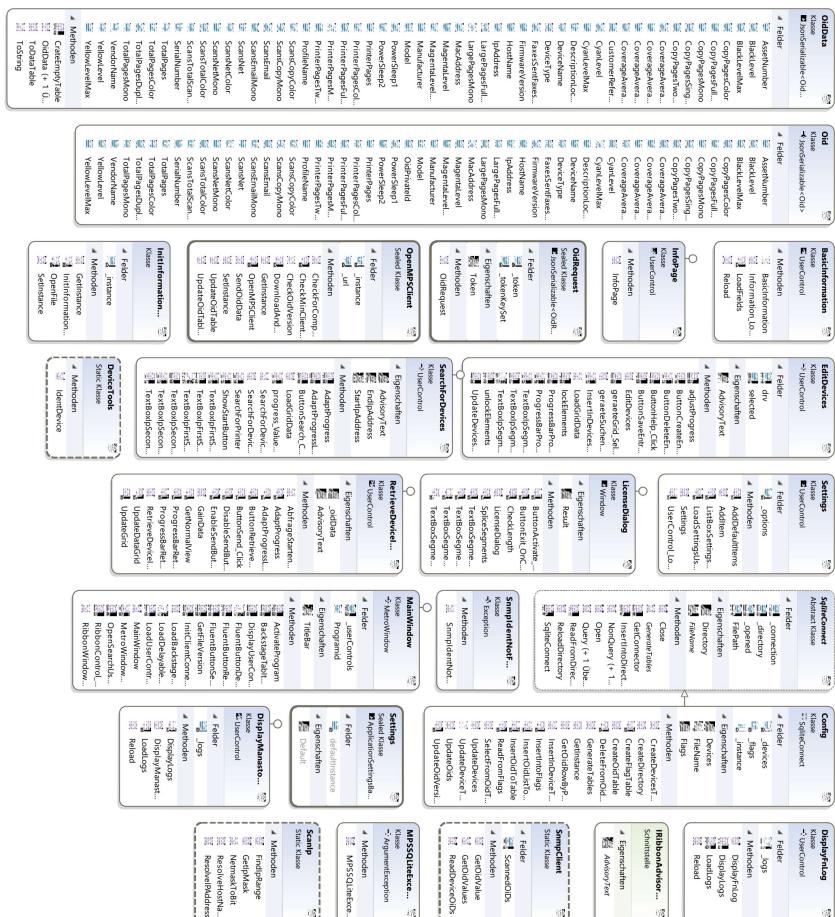
- Log  
Tabelle zur Speicherung von Logs.
- Directory  
Einfache Key Value Tabelle.

**Für die Nutzung im openMPS Client oder aber im Manastone client wird die fnLog Datenbank in die config.db, sowie die manastone.db integriert!**

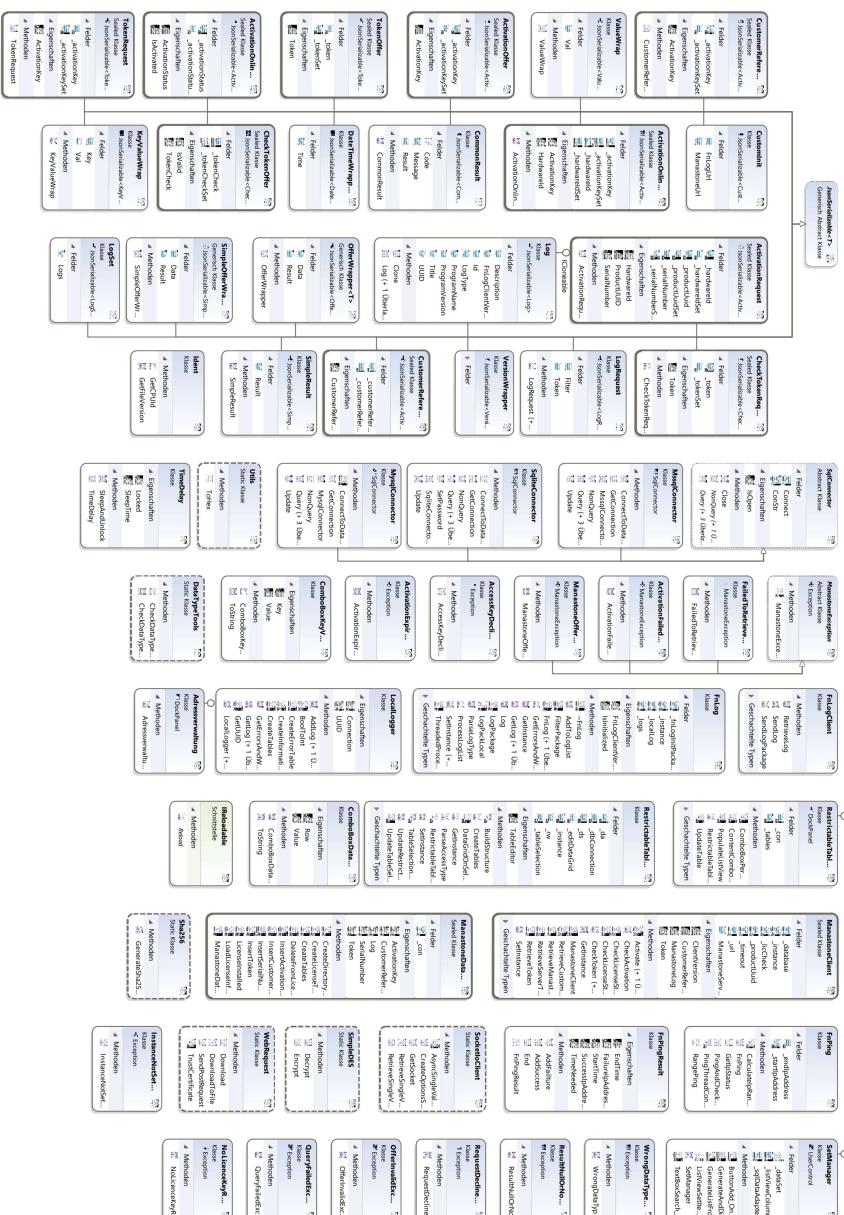
# Kapitel 5

# Klassendiagramme

## 5.1 openMPS



## 5.2 de.fearvel.net

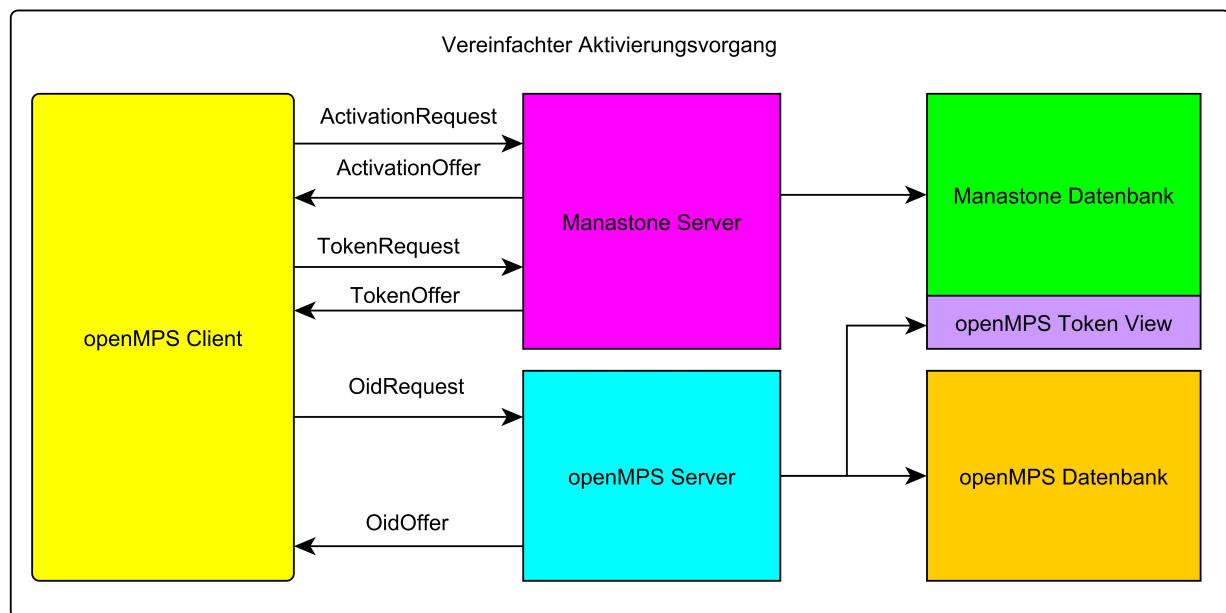


# Kapitel 6

## Sonstiges

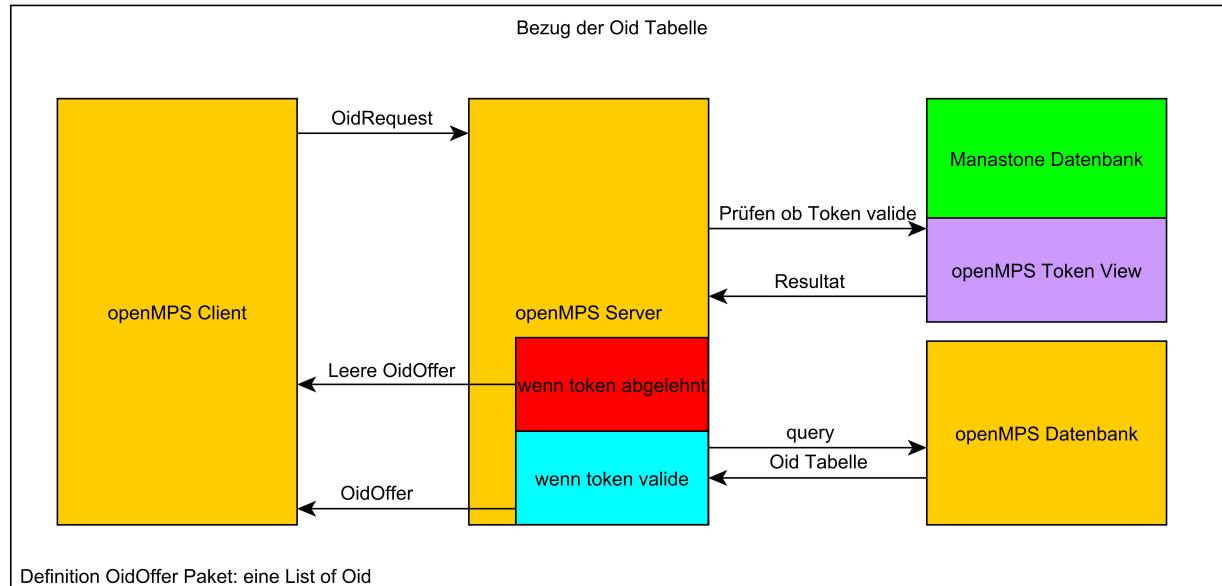
### 6.0.1 Aktivierung einer openMPS Installation jetziger Ablauf

Während des Erststarts des openMPS Tools wird der Nutzer zur Eingabe eines Produktschlüssels aufgefordert. Nach der Eingabe und Bestätigung wird ein ActivationRequest an den Manastone Server gesendet, dieser antwortet mit einer ActivationOffer, für den Fall, dass der Produktschlüssel ungültig oder bereits aktiviert ist, wird eine leere ActivationOffer versendet und vom Client via Exception behandelt. Mit dem ActivationKey, der in der ActivationOffer enthalten ist, fordert der Client ein Token vom Manastone Server an, welches eine begrenzte Gültigkeit besitzt. Dieses Token ist notwendig um sich bei nicht-Manastone-Servern zu authentifizieren.



## 6.0.2 Übertragung der OID Tabelle an den openMPS Client

Der openMPS Server hat Zugriff auf eine View die die aktuell gültigen Tokens für den openMPS Server beinhaltet. Der openMPS Server prüft das Token welches Bestandteil eines OidRequest Paketes ist Sollte das Token gültig sein wird der Inhalt der OID Tabelle als JSON serialisierte Liste an den Client versandt, im Falle eines fehlerhaften Tokens wird eine leere JSON serialisierte Liste übertragen und vom Client erkannt.



## 6.0.3 Übertragung der OID Werte an den openMPS Server

Der Client sendet ein OidData Paket an den Server. Auf eine Übertragung eines Tokens wird verzichtet, stattdessen wird eine über den Manastone Server empfangene CustomerReference übertragen, die den Kunden eindeutig identifiziert.

# Kapitel 7

## Code Ausschnitte

Die nachfolgenden Seiten zeigen Codeausschnitte, welche für das Projekt essenziell sind.

Datei: SocketIoClient.cs 10.05.2019, 12:28:50

---

```
1  using System.Threading.Tasks;
2  using de.fearvel.net.DataTypes;
3  using de.fearvel.net.DataTypes.Exceptions;
4  using Quobject.SocketIoClientDotNet.Client;

5  namespace de.fearvel.net.SocketIo
6  {
7      /// <summary>
8      /// Class for sending socketio requests
9      /// <copyright>Andreas Schreiner 2019</copyright>
10     /// </summary>
11     public static class SocketIoClient
12     {
13         /// <summary>
14         /// Sends a request to the SocketIo and receives a Task containing a JSON string
15         /// that can be
16         /// deserialized to Type T
17         /// </summary>
18         /// <typeparam name="T">T</typeparam>
19         /// <param name="serverUrl">serverUrl</param>
20         /// <param name="receiverEventName">receiverEventName</param>
21         /// <param name="senderEventName">senderEventName</param>
22         /// <param name="senderEventValue">senderEventValue</param>
23         /// <param name="acceptSelfSigned">acceptSelfSigned</param>
24         /// <param name="timeout">timeout in ms</param>
25         /// <returns></returns>
26         public static async Task<T> RetrieveSingleValueAsync<T>(string serverUrl, string
27             receiverEventName,
28             string senderEventName,
29             string senderEventValue, bool acceptSelfSigned = true, int timeout = 5000) =>
30             await AsyncSingleValueRetriever<T>(serverUrl, receiverEventName,
31                 senderEventName, senderEventValue,
32                 acceptSelfSigned, timeout);

33         /// <summary>
34         /// Sends a request to the SocketIo and receives a Task containing a JSON string
35         /// that can be
36         /// deserialized to Type T
37         /// </summary>
38         /// <typeparam name="T">T</typeparam>
39         /// <param name="serverUrl">serverUrl</param>
40         /// <param name="receiverEventName">receiverEventName</param>
41         /// <param name="senderEventName">senderEventName</param>
42         /// <param name="senderEventValue">senderEventValue</param>
43         /// <param name="acceptSelfSigned">acceptSelfSigned</param>
44         /// <param name="timeout">timeout in ms</param>
45         /// <returns></returns>
46         private static Task<T> AsyncSingleValueRetriever<T>(string serverUrl, string
47             receiverEventName,
48             string senderEventName,
49             string senderEventValue, bool acceptSelfSigned = true, int timeout = 5000) =>
50             Task.Run<T>(() => RetrieveSingleValue<T>(serverUrl, receiverEventName,
51                 senderEventName, senderEventValue,
52                 acceptSelfSigned, timeout));

53         /// <summary>
54         /// Sends a request to the SocketIo Server and receives a JSON string that can be
55         /// serialized to Type T
56         /// </summary>
57         /// <typeparam name="T"></typeparam>
58         /// <param name="serverUrl">serverUrl</param>
59         /// <param name="receiverEventName">receiverEventName</param>
60         /// <param name="senderEventName">senderEventName</param>
61         /// <param name="senderEventValue">senderEventValue</param>
62         /// <param name="acceptSelfSigned">acceptSelfSigned</param>
63         /// <param name="timeout">timeout in ms</param>
64         /// <returns></returns>
```

---

Seite: 1

Datei: SocketIoClient.cs 10.05.2019, 12:28:50

---

```
62     public static T RetrieveSingleValue<T>(string serverUrl, string receiverEventName,
62     string senderEventName,
63     string senderEventValue, bool acceptSelfSigned = true, int timeout = 5000)
64     {
65         var delay = new TimeDelay(timeout);
66         bool wait = true;
67         T result = default(T);
68         var socket = GetSocket(serverUrl, acceptSelfSigned);
69         socket.On(Socket.EVENT_CONNECT, () => { socket.Emit(senderEventName,
70             senderEventValue); });
71         socket.On(receiverEventName, (data) =>
72         {
73             result = data.GetType() == typeof(Newtonsoft.Json.Linq.JObject)
74                 ? DataTypes.AbstractDataTypes.JsonSerializable<T>.DeSerialize(
75                     ((Newtonsoft.Json.Linq.JObject) data).ToString().Replace("\r",
76                     "").Replace("\n", ""));
77                 : DataTypes.AbstractDataTypes.JsonSerializable<T>.DeSerialize((string)
78                     data);
79             socket.Disconnect();
80             socket.Close();
81         });
82     });
83     socket.On(Socket.EVENT_DISCONNECT, () => { wait = false; });
84     while (wait && delay.Locked)
85     {
86     }
87
88     if (wait || result.Equals(default(T)))
89     {
90         throw new ResultNullOrNotReceivedException();
91     }
92     return result;
93 }
94
95 /// <summary>
96 /// Creates a Socket
97 /// if accept acceptSelfSigned is true certificate validity will be ignored
98 /// </summary>
99 /// <param name="serverUrl">URL</param>
100 /// <param name="acceptSelfSigned">boolean</param>
101 /// <returns></returns>
102 public static Socket GetSocket(string serverUrl, bool acceptSelfSigned = true) =>
103     acceptSelfSigned ? IO.Socket(serverUrl, CreateOptionsSecure()) :
104     IO.Socket(serverUrl);
105 }
106 }
```

---

Seite: 2

Datei: DeviceTools.cs 10.05.2019, 11:26:40

---

```
1  using System.Data;
2  using System.Diagnostics;
3  using de.fearvel.net.FnLog;
4  using de.fearvel.openMPS.Database;
5  using de.fearvel.openMPS.DataTypes.Exceptions;

7  namespace de.fearvel.openMPS.Net
8  {
9      /// <summary>
10     /// Tools for MPS Identification
11     /// <copyright>Andreas Schreiner 2019</copyright>
12     /// </summary>
13     public static class DeviceTools
14     {
15         /// <summary>
16         /// Identifies the device.
17         /// </summary>
18         /// <param name="ip">The ip.</param>
19         /// <returns></returns>
20         public static string IdentDevice(string ip)
21         {
22             FnLog.GetInstance().AddToList(FnLog.LogType.RuntimeInfo, "DeviceTools",
23                 "IdentDevice");
24             var dt = Config.GetInstance().SelectFromOidTable();
25             var profile = SnmpClient.GetOidValue(ip, "1.3.6.1.2.1.1.2.0");
26             if (profile.Length == 0)
27             {
28                 FnLog.GetInstance()
29                     .AddToList(FnLog.LogType.Error, "DeviceTools", "IdentDevice - " +
30                         profile.Length == 0);
31                 throw new SnmpIdentNotFoundException();
32             }
33             for (var i = 0; i < dt.Rows.Count; i++)
34             {
35                 Debug.WriteLine("\n\n " + SnmpClient.GetOidValue(ip, "1.3.6.1.2.1.1.2.0"));
36                 Debug.WriteLine("\n\n " + dt.Rows[i].Field<string>("ProfileName"));
37                 if (profile.Contains(dt.Rows[i].Field<string>("ProfileName")))
38                     return dt.Rows[i].Field<string>("OidPrivateId");
39                 FnLog.GetInstance().AddToList(FnLog.LogType.RuntimeInfo, "DeviceTools",
40                     "IdentDevice - " + dt.Rows[i].Field<string>("OidPrivateId"));
41             }
42             FnLog.GetInstance().AddToList(FnLog.LogType.RuntimeInfo, "DeviceTools",
43                 "IdentDevice - GENERIC");
44             return "Generic";
45         }
46     }
```

---

Seite: 1

```
1  using System;
2  using System.Collections.Generic;
3  using System.Runtime.CompilerServices;
4  using System.Threading;
5  using System.Windows;
6  using de.fearvel.net.DataTypes;
7  using de.fearvel.net.DataTypes.Exceptions;
8  using de.fearvel.net.DataTypes.SocketIo;
9  using de.fearvel.net.FnLog;
10 using de.fearvel.net.Manastone;
11 using de.fearvel.net.SocketIo;
12 using de.fearvel.openMPS.Database;
13 using de.fearvel.openMPS.DataTypes;
14 using Newtonsoft.Json;
15 using Formatting = Newtonsoft.Json.Formatting;

17 namespace de.fearvel.openMPS.Net
18 {
19     /// <summary>
20     /// The openMPS Client
21     /// <copyright>Andreas Schreiner 2019</copyright>
22     /// </summary>
23     // ReSharper disable once InconsistentNaming
24     public sealed class OpenMPSClient
25     {
26         /// <summary>
27         /// The Server Url
28         /// </summary>
29         private readonly string _url;

32         /// <summary>
33         /// the Instance of this Singleton
34         /// </summary>
35         private static OpenMPSClient _instance;

37         /// <summary>
38         /// GetInstance for the Singleton
39         /// </summary>
40         /// <returns>instance</returns>
41         [MethodImpl(MethodImplOptions.Synchronized)]
42         public static OpenMPSClient GetInstance()
43         {
44             return _instance ?? throw new InstanceNotSetException();
45         }

47         /// <summary>
48         /// Sets the Instance of OpenMPSClient
49         /// Used to preset values like the Server URL
50         /// </summary>
51         /// <param name="serverUrl"></param>
52         /// <returns></returns>
53         [MethodImpl(MethodImplOptions.Synchronized)]
54         public static void SetInstance(string serverUrl)
55         {
56             _instance = new OpenMPSClient(serverUrl);
57         }

59         /// <summary>
60         /// Private Constructor
61         /// </summary>
62         /// <param name="serverUrl"></param>
63         private OpenMPSClient(string serverUrl)
64         {
65             _url = serverUrl;
66         }

68         /// <summary>
```

Datei: OpenMPSClient.cs 10.05.2019, 11:26:46

---

```
69     /// Checks Oif Version
70     /// Returns true if Version greater/equal than Server Version
71     /// Returns false if Version smaller than Server Version
72     /// </summary>
73     /// <returns></returns>
74     public bool CheckOidVersion(out string oidServerVersion)
75     {
76         FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
77             "OpenMPSClient", "CheckOidVersion");
78         if (Config.GetInstance().Directory.TryGetValue("OidVersion", out string
79             instVer))
80         {
81             FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
82                 "OpenMPSClient",
83                 " CheckOidVersion TryGetValue done");
84
85             var ver = SocketIoClient.RetrieveSingleValue<VersionWrapper>(_url,
86                 "OidVersionOffer", "OidVersionRequest", null, timeout: 30000);
87             FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
88                 "OpenMPSClient",
89                 "OidVersionOffer received");
90
91             if (System.Version.TryParse(instVer, out Version instVersion) &&
92                 System.Version.TryParse(ver.Version, out Version version))
93             {
94                 FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
95                     "OpenMPSClient",
96                     "before assigning out");
97
98                 oidServerVersion = ver.Version;
99                 FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
100                    "OpenMPSClient",
101                    "CheckOidVersion Complete");
102
103                 return (instVersion.CompareTo(version) >= 0);
104             }
105
106             FnLog.GetInstance().AddToList(FnLog.LogType.CriticalError, "OpenMPSClient",
107                 "Missing Directory OidVersion Value");
108             throw new QueryFailedException();
109         }
110
111         /// <summary>
112         /// Checks for the minimum openMPS server version
113         /// </summary>
114         /// <returns></returns>
115         private bool CheckMinClientVersion()
116         {
117             try
118             {
119                 FnLog.GetInstance()
120                     .AddToList(FnLog.LogType.MajorRuntimeInfo, "OpenMPSClient",
121                     "CheckMinClientVersion");
122                 var ver = SocketIoClient.RetrieveSingleValue<VersionWrapper>(_url,
123                     "MPSMinClientVersionOffer", "MPSMinClientVersionRequest", null, timeout:
124                     30000);
125                 var progVersion =
126                     System.Reflection.Assembly.GetExecutingAssembly().GetName().Version.ToString
127
128                 if (System.Version.TryParse(progVersion, out Version programVersion) &&
129                     System.Version.TryParse(ver.Version, out Version minversion))
130                 {
131                     FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
132                         "OpenMPSClient",
133                         "CheckMinClientVersion Complete");
134
135                     return (programVersion.CompareTo(minversion) >= 0);
136                 }
137             }
138         }
139     }
```

---

Seite: 2

Datei: OpenMPSClient.cs 10.05.2019, 11:26:46

---

```
127             FnLog.GetInstance().AddToList(FnLog.LogType.Error, "OpenMPSClient",
128                 "CheckMinClientVersion ResultNullOrNotReceivedException");
129             throw new ResultNullOrNotReceivedException();
130         }
131     } catch (Exception e)
132     {
133         FnLog.GetInstance().AddToList(FnLog.LogType.Error, "OpenMPSClient",
134             "CheckMinClientVersion" + e.Message);
135         MessageBox.Show("Could not reach the openMPS Server");
136         throw new ResultNullOrNotReceivedException();
137     }
138 }

140     /// <summary>
141     /// calls CheckMinClientVersion
142     /// can display messages
143     /// </summary>
144     public void CheckForCompatibleVersion()
145     {
146         try
147         {
148             if (!CheckMinClientVersion())
149             {
150                 FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
151                     "OpenMPSClient",
152                     "CheckForCompatibleVersion expired version found");
153                 FnLog.GetInstance().ProcessLogList();
154                 MessageBox.Show("openMPS Version veraltet!!\nBitte laden Sie die neuste
155                 Version herunter");
156                 Environment.Exit(1);
157             }
158         } catch (Exception e)
159         {
160             FnLog.GetInstance().AddToList(FnLog.LogType.Error, "OpenMPSClient",
161                 "CheckMinClientVersion" + e.Message);
162             MessageBox.Show("Could not reach the openMPS Server");
163             Environment.Exit(1);
164         }
165     }

167     /// <summary>
168     /// Downloads and updates the oid table
169     /// </summary>
170     /// <param name="oidVersion"></param>
171     private void DownloadAndUpdateOidTable(string oidVersion)
172     {
173         ManastoneClient.GetInstance().CheckToken();

175         FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
176             "OpenMPSClient",
177             "DownloadAndUpdateOidTable - " + ManastoneClient.GetInstance().Token);
178         var oid = SocketIoClient.RetrieveSingleValue<List<Oid>>(_url,
179             "OidOffer", "OidRequest", new
180             OidRequest(ManastoneClient.GetInstance().Token).Serialize(),
181             timeout: 30000);
182         FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
183             "OpenMPSClient",
184             "DownloadAndUpdateOidTable received" + oid.Count);
185         Config.GetInstance().UpdateOids(oidVersion, oid);
186         FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
187             "OpenMPSClient",
188             "DownloadAndUpdateOidTable Complete");
189     }

190     /// <summary>
191     /// updates the oid table
192 }
```

---

Seite: 3

```
189     /// </summary>
190     public void UpdateOidTable()
191     {
192         FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
193             "OpenMPSClient", "UpdateOidTable");
194         var thread = new Thread(UpdateOidTableThreaded);
195         thread.Start();
196     }
197
198     /// <summary>
199     /// Updates the oid table threaded
200     /// </summary>
201     public void UpdateOidTableThreaded()
202     {
203         try
204         {
205             FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
206                 "OpenMPSClient",
207                 "UpdateOidTableThreaded");
208             if (!CheckOidVersion(out string oidServerVersion))
209             {
210                 FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
211                     "OpenMPSClient",
212                     "UpdateOidTableThreaded newer version detected");
213                 DownloadAndUpdateOidTable(oidServerVersion);
214             }
215         }
216         catch (Exception e)
217         {
218             FnLog.GetInstance().AddToList(FnLog.LogType.Error, "OpenMPSClient",
219                 "UpdateOidTableThreaded " + e.Message);
220         }
221
222     /// <summary>
223     /// sends the oid data
224     /// </summary>
225     /// <param name="data"></param>
226     public void SendOidData(List<Oiddata> data)
227     {
228         FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
229             "OpenMPSClient", "SendOidData");
230         var dataStr = JsonConvert.SerializeObject(data, Formatting.Indented).Trim()
231             .Replace(System.Environment.NewLine, "");
232         var res = SocketIoClient.RetrieveSingleValue<SimpleResult>(_url,
233             "closer", "SendData", dataStr, timeout: 30000);
234         FnLog.GetInstance().AddToList(FnLog.LogType.MajorRuntimeInfo,
235             "OpenMPSClient", "SendOidData sent");
236     }
237 }
238 }
```

Datei: ManastoneClient.cs 10.05.2019, 12:33:51

---

```
1  using de.fearvel.net.DataTypes;
2  using de.fearvel.net.DataTypes.Exceptions;
3  using de.fearvel.net.DataTypes.Exceptions.Manastone;
4  using de.fearvel.net.DataTypes.Manastone;
5  using de.fearvel.net.SocketIo;
6  using System;
7  using System.Data;
8  using System.Runtime.CompilerServices;

10 namespace de.fearvel.net.Manastone
11 {
12     /// <summary>
13     /// Manastone DRM Client Class
14     /// used for the communication with the Manastone server
15     /// this class will create an instance on ManastoneDatabase
16     /// </summary>
17     public sealed class ManastoneClient
18     {
19         #region "Private"
20
21         /// <summary>
22         /// Timeout of the SocketIo RetrieveSingleValue function
23         /// </summary>
24         private readonly int _timeout = 30000;
25
26         /// <summary>
27         /// Url of the Manastone Server
28         /// </summary>
29         private readonly string _url;
30
31         /// <summary>
32         /// Instance of ManastoneDatabase, which will be used for the local storage of the
33         /// License information
34         /// </summary>
35         private readonly ManastoneDatabase _database;
36
37         /// <summary>
38         /// The uuid of the Program, which will be checked
39         /// Every program which witch uses Manastone DRM needs to have a UUID that is known
40         /// by the Manastone Server
41         /// </summary>
42         private readonly string _productUuid;
43
44         /// <summary>
45         /// Enum which determines if:
46         /// The Program will be checked for Activation Online, Offline or Periodically
47         /// online(Mixed)
48         /// This is for changing the Telemetry mode of the FnLog Telemetry system
49         /// </summary>
50         private readonly LicenseCheckType _licCheck;
51
52         /// <summary>
53         /// the Instance of this Singleton
54         /// There can only be one instance of this
55         /// </summary>
56         private static ManastoneClient _instance;
57
58         /// <summary>
59         /// Property for accessing the Manastone log by Programs that implement Manastone
60         /// DRM
61         /// </summary>
62         public DataTable ManastoneLog => _database.Log.GetLog();
63
64         /// <summary>
65         /// Private Constructor to ensure that there will be only one instance
66         /// this will initialize properties
```

---

Seite: 1

Datei: ManastoneClient.cs 10.05.2019, 12:33:51

---

```
65     /// creates the ManastoneDatabase instance
66     /// It will also retrieve the Manastone serverVersion
67     /// and try to Receive the CustomerReference
68     /// </summary>
69     /// <param name="serverUrl"></param>
70     private ManastoneClient(string serverUrl, string productUUID, LicenseCheckType
71     lCheck)
72     {
73         _url = serverUrl;
74         _database = new ManastoneDatabase();
75         _database.Log.AddToList(FnLog.FnLog.LogType.DrmLog, "MANASTONE STARTED", "");
76         _productUuid = productUUID;
77         _licCheck = lCheck;
78         ManastoneServerVersion = RetrieveManastoneServerVersion().ToString();
79         RetrieveCustomerReference(new
80             CustomerReferenceRequest(_database.ActivationKey));
81
82     /// <summary>
83     /// Returns the Version of the Manastone Server
84     /// if this fails it will throw a ManastoneOfferNotReceivedCorrectlyException
85     /// </summary>
86     /// <returns></returns>
87     private Version RetrieveManastoneServerVersion()
88     {
89         try
90         {
91             _database.Log.AddToList(FnLog.FnLog.LogType.MinorDrmLog,
92             "RetrieveManastoneServerVersion", "Start");
93             var serVer =
94                 Version.Parse(SocketIoClient.RetrieveSingleValue<VersionWrapper>(_url, "ManastoneVersion"
95                 "ManastoneVersionRequest", null, timeout: _timeout).Version);
96             _database.Log.AddToList(FnLog.FnLog.LogType.MinorDrmLog,
97                 "RetrieveManastoneServerVersion", "Complete");
98             _database.Log.ProcessLogList();
99             return serVer;
100         }
101         catch (Exception e)
102         {
103             _database.Log.AddToList(FnLog.FnLog.LogType.DrmError, "ERROR on
104             RetrieveManastoneServerVersion", e.Message);
105             _database.Log.ProcessLogList();
106             throw new ManastoneOfferNotReceivedCorrectlyException();
107         }
108
109     /// <summary>
110     /// Returns the DateTime of the Manastone Server
111     /// Currently unused, will be used later to locally check the Token expiry date
112     /// </summary>
113     /// <returns></returns>
114     private DateTime RetrieveServerTime()
115     {
116         try
117         {
118             _database.Log.AddToList(FnLog.FnLog.LogType.MinorDrmLog,
119             "RetrieveServerTime", "Start");
120             var serTime = SocketIoClient
121                 .RetrieveSingleValue<DateTimeWrapper>(_url, "ServerTime",
122                 "ServerTimeRequest", null)
123                 .Time;
124             _database.Log.AddToList(FnLog.FnLog.LogType.MinorDrmLog,
125                 "RetrieveServerTime", "Complete");
126             _database.Log.ProcessLogList();
127             return serTime;
128         }
129         catch (Exception e)
130         {
```

---

Seite: 2

Datei: ManastoneClient.cs 10.05.2019, 12:33:51

---

```
124         _database.Log.AddToList(FnLog.FnLog.LogType.DrmError, "ERROR on
124             RetrieveServerTime", e.Message);
125         _database.Log.ProcessLogList();
126         throw new ManastoneOfferNotReceivedCorrectlyException();
127     }
128 }

130     /// <summary>
131     /// Activates a License
132     /// the License must be valid and non activated
133     /// there can only be one activation of a SerialNumber
134     /// If the Activation is successful the SerialNumber and ActivationKey will be
134     inserted into the local database.
135     /// It will throw a ActivationFailedException if the ManastoneServer delivers no or
135     an empty Offer.
136     /// </summary>
137     /// <param name="req">an ActivationRequest containing the SerialNumber and the CPU1
137     ID</param>
138     private void Activate(ActivationRequest req)
139     {
140         try
141         {
142             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog, "Activate", "req
142                 " + req.SerialNumber +
143
144                 " " +
143
144             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog, "Activate",
145                 "Start");
146             var activationKey =
147                 SocketIoClient.RetrieveSingleValue<ActivationOffer>(_url,
147                     "ActivationOffer", "ActivationRequest",
148                     req.Serialize(), timeout: _timeout);
149             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog, "Activate",
149                 "ActivationOffer " + activationKey.ActivationKey.Length);
150             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog, "Activate",
150                 "ActivationOffer " + activationKey.Serialize());
151
152             if (activationKey.ActivationKey.Length == 0)
153             {
154                 throw new ActivationFailedException("Length == 0");
155             }
156             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog, "Activate",
156                 "Before SerialInsertion");
157             _database.InsertSerialNumber(req.SerialNumber);
158             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog, "Activate",
158                 "Before ActivationKeyInsertion");
159             _database.InsertActivationKey(req.SerialNumber,
159                 activationKey.ActivationKey);
160             RetrieveToken(new TokenRequest(activationKey.ActivationKey));
161             RetrieveCustomerReference(new
161                 CustomerReferenceRequest(activationKey.ActivationKey));
162             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog, "Activate",
162                 "Complete");
163             _database.Log.ProcessLogList();
164         }
165         catch (Exception e)
166         {
167             _database.Log.AddToList(FnLog.FnLog.LogType.DrmError, "ERROR on
167                 ACTIVATE", e.Message);
168             _database.Log.ProcessLogList();
169             throw new ActivationFailedException();
170         }
171     }

173     /// <summary>
174     /// Retrieves the Customer Reference from the Manastone Server and saves it to the
174     Local database
```

---

Seite: 3

Datei: ManastoneClient.cs 10.05.2019, 12:33:51

---

```
175     /// </summary>
176     /// <param name="req">an CustomerReferenceRequest containing the
177     ActivationKey</param>
178     private void RetrieveCustomerReference(CustomerReferenceRequest req)
179     {
180         try
181         {
182             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog,
183             "RetrieveCustomerReference", "Start");
184             var offer =
185                 SocketIoClient.RetrieveSingleValue<CustomerReferenceOffer>(_url,
186                 "CustomerReferenceOffer",
187                 "CustomerReferenceRequest", req.Serialize(), timeout: _timeout);
188             _database.InsertCustomerReference(req.ActivationKey,
189             offer.CustomerReference);
190             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog,
191             "RetrieveCustomerReference", "Complete");
192         }
193     }
194
195     /// <summary>
196     /// Retrieves the Token and saves it to the Local database
197     /// </summary>
198     /// <param name="req">an TokenRequest containing the ActivationKey</param>
199     private void RetrieveToken(TokenRequest req)
200     {
201         try
202         {
203             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog, "RetrieveToken",
204             "Start");
205             var token =
206                 SocketIoClient.RetrieveSingleValue<TokenOffer>(_url, "TokenOffer",
207                 "TokenRequest",
208                 req.Serialize(), timeout: _timeout);
209             if (token.Token.Length == 0)
210                 throw new FailedToRetrieveTokenException();
211             _database.InsertToken(req.ActivationKey, token.Token);
212             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog, "RetrieveToken",
213             "Complete");
214             _database.Log.ProcessLogList();
215         }
216     }
217
218     /// <summary>
219     /// Checks if an entry of an Activation is written to the local encrypted Manastone
220     database
221     /// </summary>
222     /// <returns>a bool that is true if the license is valid</returns>
223     private bool CheckLicenseStatusLocally()
224     {
225         var status = _database.LicenseInstalled();
226         _database.Log.AddToList(FnLog.FnLog.LogType.MinorDrmLog,
227             "CheckLicenseStatusLocally", status.ToString());
228         _database.Log.ProcessLogList();
229         return status;
230     }
```

---

Seite: 4

Datei: ManastoneClient.cs 10.05.2019, 12:33:51

---

```
231     }
232
233     /// <summary>
234     /// Checks Activation status Online.
235     /// this will contact the Manastone server and check if the activation is valid
236     /// </summary>
237     /// <param name="req">ActivationOnlineCheckRequest</param>
238     /// <returns>if valid returns true</returns>
239     private bool CheckLicenseStatusOnline(ActivationOnlineCheckRequest req)
240     {
241         try
242         {
243             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog,
244             "CheckLicenseStatusOnline", "Start");
245             var offer =
246                 SocketIoClient.RetrieveSingleValue<ActivationOnlineCheckOffer>(_url,
247                 "ActivationOnlineCheckOffer",
248                 "ActivationOnlineCheckRequest", req.Serialize(), timeout: _timeout);
249             if (offer.IsActive)
250             {
251                 _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog,
252                 "CheckLicenseStatusOnline", "ACTIVATION Confirmed");
253                 RetrieveCustomerReference(new
254                     CustomerReferenceRequest(req.ActivationKey));
255             }
256             _database.Log.AddToList(FnLog.FnLog.LogType.MajorDrmLog,
257                 "CheckLicenseStatusOnline", "Complete");
258             _database.Log.ProcessLogList();
259             return offer.IsActive;
260         }
261         catch (Exception e)
262         {
263             _database.Log.AddToList(FnLog.FnLog.LogType.DrmError, "ERROR on
264             CheckLicenseStatusOnline", e.Message);
265             _database.Log.ProcessLogList();
266             return false;
267         }
268     }
269
270     /// <summary>
271     /// Checks if the Token is valid
272     /// this will contact the Manastone server and check if the token is valid
273     /// a token check has to be Online only because the token is to authenticate a
274     /// program client to a program server or service
275     /// if the server doesn't answer it will throw a
276     /// ManastoneOfferNotReceivedCorrectlyException
277     /// </summary>
278     /// <param name="req">CheckTokenRequest</param>
279     /// <returns></returns>
280     public bool CheckToken(CheckTokenRequest req)
281     {
282         try
283         {
284             _database.Log.AddToList(FnLog.FnLog.LogType.DrmLog, "CheckToken",
285             "Start");
286             var offer =
287                 SocketIoClient.RetrieveSingleValue<CheckTokenOffer>(_url,
288                 "CheckTokenOffer",
289                 "CheckTokenRequest", req.Serialize(), timeout: _timeout);
290             _database.Log.AddToList(FnLog.FnLog.LogType.DrmLog, "CheckToken",
291                 "Complete");
292             _database.Log.ProcessLogList();
293             return offer.IsValid;
294         }
295         catch (Exception e)
296         {
297             _database.Log.AddToList(FnLog.FnLog.LogType.DrmError, "ERROR on
298             CheckToken", e.Message);
299             _database.Log.ProcessLogList();
```

---

Seite: 5

Datei: ManastoneClient.cs 10.05.2019, 12:33:51

---

```
287             throw new ManastoneOfferNotReceivedCorrectlyException();
288         }
289     }
290 
291     #endregion
292 
293     #region "Public"
294 
295     /// <summary>
296     /// The Version of this Manastone Client Class
297     /// Can be used later to determine if this client is outdated
298     /// </summary>
299     public static string ClientVersion => "1.000.0001.0000";
300 
301     /// <summary>
302     /// CustomerReference Property
303     /// Get only
304     /// Gets the CustomerReference for the activated SerialNumber
305     /// accesses a property of the ManastoneDatabase instance _database
306     /// </summary>
307     public string CustomerReference => _database.CustomerReference;
308 
309     /// <summary>
310     /// Token Property
311     /// Get only
312     /// Gets the Token for the activated SerialNumber
313     /// accesses a property of the ManastoneDatabase instance _database
314     /// </summary>
315     public string Token => _database.Token;
316 
317     /// <summary>
318     /// The Version of this Manastone Server
319     /// Can be used later to determine if this client is outdated
320     /// </summary>
321     public string ManastoneServerVersion;
322 
323     /// <summary>
324     /// GetInstance for the Singleton
325     /// if the instance is not set throws InstanceNotSetException
326     /// </summary>
327     /// <returns>instance</returns>
328     [MethodImpl(MethodImplOptions.Synchronized)]
329     public static ManastoneClient GetInstance()
330     {
331         return _instance ?? throw new InstanceNotSetException();
332     }
333 
334     /// <summary>
335     /// Sets the Instance of ManastoneClient
336     /// Used to preset values like the Server URL
337     /// </summary>
338     /// <param name="serverUrl">the URL of the Manastone Server</param>
339     /// <param name="productUuid">the id of the Product to be activated</param>
340     [MethodImpl(MethodImplOptions.Synchronized)]
341     public static void SetInstance(string serverUrl, string productUuid,
342         LicenseCheckType licCheck = LicenseCheckType.Online)
343     {
344         _instance = new ManastoneClient(serverUrl, productUuid, licCheck);
345     }
346 
347     /// <summary>
348     /// Activates a License
349     /// The License can only be Activated once at the same Time for each SerialNumber
350     /// </summary>
351     /// <returns>bool true if activation successful</returns>
352     public bool Activate(string serialNumber)
353     {
354         try
```

---

Seite: 6

Datei: ManastoneClient.cs 10.05.2019, 12:33:51

---

```
355         {
356             Activate(new ActivationRequest(serialNumber, _productUuid));
357             return true;
358         }
359     catch (Exception)
360     {
361         return false;
362     }
363 }

365     /// <summary>
366     /// Checks the ActivationKey
367     /// The LicenseCheckType determines if this will be
368     /// online only, offline only, or mixed
369     /// Mixed 1 in 7 chance to be checked online
370     /// </summary>
371     /// <returns></returns>
372     public bool CheckActivation()
373     {
374         try
375         {
376             switch (_licCheck)
377             {
378                 case LicenseCheckType.Online:
379                     return CheckLicenseStatusOnline(new
380                         ActivationOnlineCheckRequest(_database.ActivationKey));
381                 case LicenseCheckType.Offline:
382                     return CheckLicenseStatusLocally();
383                 case LicenseCheckType.Mixed:
384                     var rand = new Random();
385                     return rand.Next(100) % 7 == 0
386                         ? CheckLicenseStatusOnline(new
387                             ActivationOnlineCheckRequest(_database.ActivationKey))
388                         : CheckLicenseStatusLocally();
389                 default:
390                     throw new ArgumentOutOfRangeException();
391             }
392         catch (Exception)
393         {
394             return false;
395         }
396     }

397     /// <summary>
398     /// Checks if the Token is valid
399     /// Online Check
400     /// </summary>
401     /// <returns></returns>
402     public bool CheckToken()
403     {
404         try
405         {
406             if (CheckToken(new CheckTokenRequest(_database.Token)))
407             {
408                 RetrieveToken(new TokenRequest(_database.ActivationKey));
409             }
410
411             return true;
412         }
413         catch (Exception)
414         {
415             return false;
416         }
417     }

418     /// <summary>
419     /// Enum for the ActivationCheck type that will be used
420 
```

---

Seite: 7

Datei: ManastoneClient.cs 10.05.2019, 12:33:51

---

```
421     /// </summary>
422     public enum LicenseCheckType
423     {
424         Online,
425         Offline,
426         Mixed
427     }
428     #endregion
429 }
430 }
431 }
```

---

Seite: 8

Datei: ManastoneDatabase.cs 09.05.2019, 19:21:02

---

```
1  using System;
2  using System.Data;
3  using System.Data.SQLite;
4  using de.fearvel.net.Security;
5  using de.fearvel.net.SQL.Connector;

7  namespace de.fearvel.net.Manastone
8  {
9      /// <summary>
10     /// Manastone Database management class
11     /// used for the local storage of the License information
12     /// </summary>
13     internal sealed class ManastoneDatabase
14     {
15         private readonly SqliteConnector _con;

17         /// <summary>
18         /// the SerialNumber Key can be accessed by this property if an SerialNumber key
19         /// exists
20         /// </summary>
21         internal string SerialNumber { get; private set; }

22         /// <summary>
23         /// internal FnLog
24         /// </summary>
25         internal FnLog.FnLog Log {get; private set; }

27         /// <summary>
28         /// the ActivationKey can be accessed by this property if an ActivationKey exists
29         /// </summary>
30         internal string ActivationKey { get; private set; }

32         /// <summary>
33         /// Token Property
34         /// </summary>
35         internal string Token { get; private set; }

37         /// <summary>
38         /// the CustomerReference Key can be accessed by this property if an
39         /// CustomerReference exists
40         /// </summary>
41         internal string CustomerReference { get; private set; }

42         /// <summary>
43         /// public constructor
44         /// Creates or opens the *.db
45         /// May create missing tables
46         /// Uses the Cpu1 id for encryption
47         /// </summary>
48         internal ManastoneDatabase()
49         {
50             // _con = new SqliteConnector("Manastone.db"); //DEBUG
51             _con = new SqliteConnector("Manastone.db", Ident.GetCPUId());
52             Log = new FnLog.FnLog(new
53                 FnLog.FnLogInitPackage("https://app.fearvel.de:9020/",
54                     "MANASTONE", Version.Parse(ManastoneClient.ClientVersion),
55                     FnLog.FnLog.TelemetryType.LogLocalSendAll, "", ""));
56             _con);
57             CreateTables();
58             LoadLicenseInformation();
59         }

61         /// <summary>
62         /// Reads the License Information from the Database and fills the properties with it
63         /// </summary>
```

---

Seite: 1

Datei: ManastoneDatabase.cs 09.05.2019, 19:21:02

---

```
64      private void LoadLicenseInformation()
65      {
66
67          if (LicenseInstalled())
68          {
69              Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog,
70                  "LoadLicenseInformation", "License Found");
71              _con.Query("SELECT * FROM `License`;", out DataTable dt);
72              var row = dt.Rows[0];
73              SerialNumber = row.Field<string>("SerialNumber");
74              ActivationKey = row.Field<string>("ActivationKey");
75              Token = row.Field<string>("Token");
76              CustomerReference = row.Field<string>("CustomerReference");
77
78          }
79      }
80
81
82
83
84
85
86
87      /// <summary>
88      /// bool which reflects if a license is installed
89      /// </summary>
90      /// <returns>a bool true if license existent</returns>
91      internal bool LicenseInstalled()
92      {
93          _con.Query("SELECT EXISTS (SELECT * FROM `License`) as LicenseInstalled;", out
94          DataTable dt);
95          var licInst = dt.Rows[0].Field<long>("LicenseInstalled") == 1;
96          Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "LoadLicenseInformation",
97                      licInst.ToString());
98          return licInst;
99
100
101
102      private void DeleteFromLicense()
103      {
104          Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "DeleteFromLicense", "");
105          _con.NonQuery("DELETE FROM `License`;");
106
107
108      /// <summary>
109      /// Inserts a SerialNumber into the Database
110      /// also triggers LoadLicenseInformation();
111      /// </summary>
112      /// <param name="serialNumber"></param>
113      internal void InsertSerialNumber(string serialNumber)
114      {
115          Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "InsertSerialNumber",
116                      "Start");
117          if (LicenseInstalled())
118              DeleteFromLicense();
119          var command = new SQLiteCommand("INSERT INTO `License` (`SerialNumber`) VALUES
120              (@SerialNumber);");
121          command.Parameters.AddWithValue("@SerialNumber", serialNumber);
122          command.Prepare();
123          _con.NonQuery(command);
124          LoadLicenseInformation();
125          Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "InsertSerialNumber",
126                      "Complete");
127      }
```

---

Seite: 2

```
126     /// <summary>
127     /// Inserts a ActivationKey into the Database
128     /// also triggers LoadLicenseInformation();
129     /// </summary>
130     /// <param name="serialNumber">the serialNumber for identification of the row to be
131     /// used</param>
132     /// <param name="activationKey">the activationKey which will be updated</param>
133     internal void InsertActivationKey(string serialNumber, string activationKey)
134     {
135         Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "InsertActivationKey",
136         "Start");
137         var command =
138             new SQLiteCommand(
139                 "UPDATE `License` set `ActivationKey` = @ActivationKey where
140                 `SerialNumber` = @SerialNumber;");
141         command.Parameters.AddWithValue("@ActivationKey", activationKey);
142         command.Parameters.AddWithValue("@SerialNumber", serialNumber);
143         command.Prepare();
144         _con.ExecuteNonQuery(command);
145         LoadLicenseInformation();
146         Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "InsertActivationKey",
147         "Complete");
148     }
149
150     /// <summary>
151     /// Inserts a Token into the Database
152     /// also triggers LoadLicenseInformation();
153     /// </summary>
154     /// <param name="activationKey">the activationKey for identification of the row to
155     /// be used</param>
156     /// <param name="token">the token which will be updated</param>
157     internal void InsertToken(string activationKey, string token)
158     {
159         Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "InsertToken", "Start");
160         var command =
161             new SQLiteCommand("UPDATE `License` set `Token` = @Token where
162                 `ActivationKey` = @ActivationKey;");
163         command.Parameters.AddWithValue("@Token", token);
164         command.Parameters.AddWithValue("@ActivationKey", activationKey);
165         command.Prepare();
166         _con.ExecuteNonQuery(command);
167         LoadLicenseInformation();
168         Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "InsertToken", "Complete");
169     }
170
171     /// <summary>
172     /// Inserts a CustomerReference into the Database
173     /// also triggers LoadLicenseInformation();
174     /// </summary>
175     /// <param name="activationKey">the activationKey for identification of the row to
176     /// be used</param>
177     /// <param name="customerReference">the customerReference which will be
178     /// updated</param>
179     internal void InsertCustomerReference(string activationKey, string
180     customerReference)
181     {
182         Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "InsertCustomerReference",
183         "Start");
184         var command = new SQLiteCommand("UPDATE `License` set `CustomerReference` =
185             @CustomerReference" +
186                 " where `ActivationKey` = @ActivationKey;");
187         command.Parameters.AddWithValue("@CustomerReference", customerReference);
188         command.Parameters.AddWithValue("@ActivationKey", activationKey);
189         command.Prepare();
190         _con.ExecuteNonQuery(command);
191         LoadLicenseInformation();
192         Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "InsertCustomerReference",
```

Datei: ManastoneDatabase.cs 09.05.2019, 19:21:02

---

```
182         "Complete");
183     }

184     /// <summary>
185     /// function to create the Tables
186     /// creates the basic tables if they are not existent
187     /// </summary>
188     private void CreateTables()
189     {
190         Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "CreateTables", "Start");
191         CreateDirectoryTable();
192         CreateLicenseTable();
193         Log.AddToList(FnLog.FnLog.LogType.DrmDatabaseLog, "CreateTables",
194         "Complete");
195     }

196     /// <summary>
197     /// Creates the Directory Table if not exists
198     /// which is there for later use
199     /// </summary>
200     private void CreateDirectoryTable()
201     {
202         _con.ExecuteNonQuery("CREATE TABLE IF NOT EXISTS `Directory` (" +
203             " `DKey` varchar(200)," +
204             " `DVal` Text," +
205             " CONSTRAINT uq_Version_Identifier UNIQUE (`DKey`));");
206     }

207     /// <summary>
208     /// Creates the License Table if not exists
209     /// </summary>
210     private void CreateLicenseTable()
211     {
212         _con.ExecuteNonQuery("CREATE TABLE IF NOT EXISTS `License` (" +
213             " `Id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
214             " `SerialNumber` varchar(36) NOT NULL DEFAULT '',
215             " `ActivationKey` varchar(36) NOT NULL DEFAULT '',
216             " `Token` varchar(36) NOT NULL DEFAULT '',
217             " `CustomerReference` varchar(100) NOT NULL DEFAULT ''");
218     }
219 }
220 }
```

---

Seite: 4

```
1  using System;
2  using System.Linq;
3  using System.Management;
4  using System.Net;
5  using System.Net.Sockets;
6  using de.fearvel.net.FnLog;

8  namespace de.fearvel.openMPS.Net
9  {
10     /// <summary>
11     /// IP Scan Tools
12     /// <copyright>Andreas Schreiner 2019</copyright>
13     /// </summary>
14     internal static class ScanIp
15     {

17         /// <summary>
18         /// Finds the ip range.
19         /// </summary>
20         /// <param name="ipMask">The ip mask.</param>
21         /// <returns></returns>
22         public static IPAddress[] FindIpRange(string[] ipMask)
23         {
24             FnLog.GetInstance().AddToList(FnLog.LogType.MinorRuntimeInfo, "ScanIp",
24             "FindIpRange");

26             var ip = IPAddress.Parse(ipMask[0]);
27             var bits = NetmaskToBit(ipMask[1]);
28             var mask = ~(uint.MaxValue >> bits);
29             // Convert the IP address to bytes.
30             var ipBytes = ip.GetAddressBytes();
31             // BitConverter gives bytes in opposite order to GetAddressBytes().
32             var maskBytes = BitConverter.GetBytes(mask).Reverse().ToArray();
33             var startIpBytes = new byte[ipBytes.Length];
34             var endIpBytes = new byte[ipBytes.Length];
35             // Calculate the bytes of the start and end IP addresses.
36             for (var i = 0; i < ipBytes.Length; i++)
37             {
38                 startIpBytes[i] = (byte) (ipBytes[i] & maskBytes[i]);
39                 endIpBytes[i] = (byte) (ipBytes[i] | ~maskBytes[i]);
40             }
41             FnLog.GetInstance().AddToList(FnLog.LogType.MinorRuntimeInfo,
42             "ScanIp", "FindIpRange NETMASK FROM " +
43             new IPAddress(startIpBytes).ToString() +
44             " TO " + new IPAddress(endIpBytes).ToString());

46             // Convert the bytes to IP addresses.
47             return new[] {new IPAddress(startIpBytes), new IPAddress(endIpBytes)};
48         }

50         /// <summary>
51         /// Resolves a HostName to a IPAddress using the DNS Server
52         /// this will only return a IPV4 Address,
53         /// because SnmpSharpNet can only handle IPV4
54         /// </summary>
55         /// <param name="hostName"></param>
56         /// <returns></returns>
57         public static IPAddress ResolveHostName(string hostName)
58         {
59             return Dns.GetHostAddresses(hostName).FirstOrDefault(
60               ip => ip.AddressFamily == AddressFamily.InterNetwork);
61         }

63         /// <summary>
64         /// Resolves an IPAddress to a IPHostEntry
65         /// </summary>
66         /// <param name="ip"></param>
67         /// <returns></returns>
```

Datei: ScanIp.cs 10.05.2019, 11:26:06

---

```
68     // ReSharper disable once InconsistentNaming
69     public static IPHostEntry ResolveIPAddress(IPAddress ip)
70     {
71         return Dns.GetHostEntry(ip);
72     }
73
74     /// <summary>
75     /// Netmask to bit.
76     /// </summary>
77     /// <param name="mask">The mask.</param>
78     /// <returns></returns>
79     public static int NetmaskToBit(string mask)
80     {
81         FnLog.GetInstance().AddToList(FnLog.LogType.MinorRuntimeInfo, "ScanIp",
82         "NetmaskToBit");
83         var totalBits = 0;
84         foreach (var octet in mask.Split('.'))
85         {
86             var octetByte = byte.Parse(octet);
87             while (octetByte != 0)
88             {
89                 totalBits += octetByte & 1; // logical AND on the LSB
90                 octetByte >>= 1; // do a bitwise shift to the right to create a new LSB
91             }
92         }
93         FnLog.GetInstance().AddToList(FnLog.LogType.MinorRuntimeInfo, "ScanIp",
94         "NetmaskToBit - " + totalBits);
95         return totalBits;
96     }
97
98     /// <summary>
99     /// Gets the ip mask.
100    /// </summary>
101    /// <returns></returns>
102    public static string[] GetIpMask()
103    {
104        FnLog.GetInstance().AddToList(FnLog.LogType.MinorRuntimeInfo, "ScanIp",
105        "GetIpMask");
106        string[] address = null;
107        string[] subnetMask = null;
108        var networkInfo =
109            new ManagementObjectSearcher(
110                "SELECT * FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled =
111                'TRUE'");
112        var moc = networkInfo.Get();
113        foreach (var mo in moc)
114        {
115            address = (string[]) mo["IPAddress"];
116            subnetMask = (string[]) mo["IPSubnet"];
117        }
118        FnLog.GetInstance().AddToList(FnLog.LogType.MinorRuntimeInfo, "ScanIp",
119        "GetIpMask - " + address[0] + " - " + subnetMask[0]);
120
121        return new[] {address[0], subnetMask[0]};
122    }
123 }
```

---

Seite: 2

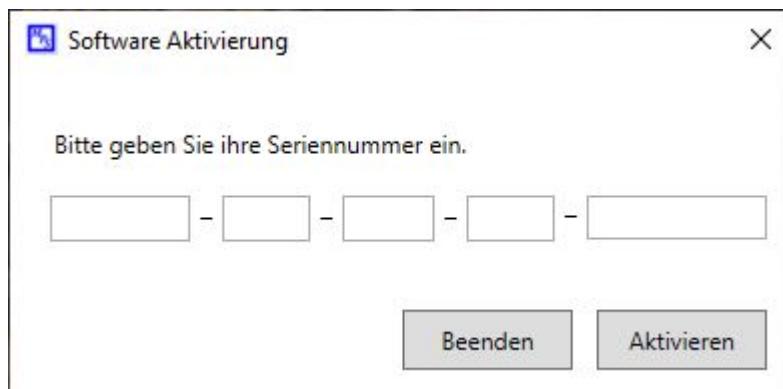
# Kapitel 8

## Benutzerhandbuch

Zur Nutzung des openMPS Clients ist ein Produktschlüssel sowie der Zugang zum Internet erforderlich! Der openMPS Client verwendet folgende Ports: 9020, 9040, 9060.

### 8.1 Erststart und Aktivierung

Der openMPS Client wird beim Erststart den Nutzer zur Eingabe eines Produktschlüssels auffordern. Der Produktschlüssels hat dieses Format 00000000-0000-0000-0000-000000000000. Nach Eingabe und Bestätigung("Aktivieren" Button) eines korrekten Produktschlüssels wird dem Nutzer die Standardansicht, die openMPS Suche, angezeigt.



## 8.2 Automatische Suche

In der openMPS Suche kann der Nutzer ein MPS Gerät über eine Bereichssuche auffinden, dazu gibt der Nutzer eine Start- und End-Adresse ein und betätigt den "Suchen" Button. Die benötigte Zeit der Suche ist abhängig von:

1. der Anzahl der Adressen.
2. der Anzahl der Geräte, welche keine MPS Geräte sind.
3. den Antwortzeiten eines MPS Gerätes.

Um effizient ein MPS Gerät zu suchen ist eine Suche von 2 - 20 Geräten empfohlen.

### 8.2.1 Suchen von Geräten

1. Füllen Sie die von und zu Felder aus.
2. Betätigen Sie den "Gerät suchen" Button.
3. warten Sie bis ihr Gerät gefunden wurde.

The screenshot shows the openMPS 2.7.0 software interface. The title bar says "openMPS 2.7.0". Below it, there are two tabs: "Datei" (selected) and "Geräte". A status message "Die automatische Suche nach Druckern kann einige Minuten in Anspruch nehmen" is displayed. Below the tabs, there are three buttons: "Geräte Suchen", "Geräte bearbeiten", and "Werte abfragen und senden". In the center, there is a search form with fields for "IPv4 Adressbereich": "192 . 168 . 44 . 0" and "bis": "192 . 168 . 44 . 255", followed by a "Geräte suchen" button. Below the form, a table displays search results:

Active	Ip	Model	SerialNumber	AssetNumber	HostName
<input checked="" type="checkbox"/>	192.168.1.100	CLX-4190 Series	Z8NLBJFH8000EHZ		SEC30CDA7F279AB.fritz.box

## 8.3 Manuelle Bearbeitung von Geräten

### 8.3.1 Hinzufügen von Geräten

Sollte sich der Nutzer zum manuellen Hinzufügen entscheiden müssen folgende Schritte durchgeführt werden:

1. "Gerät Hinzufügen" Button betätigen.
2. IP-Adresse oder Hostname eingeben.
3. "Speichern" Button betätigen.

### 8.3.2 Editieren von Geräten

Der Nutzer kann Geräte wie folgend bearbeiten.

1. Der Nutzer klickt auf das zu bearbeitende Gerät.
2. Die bisher ausgegrauten Felder Aktiv, IP sind nun bearbeitbar.
3. Nach erfolgter Bearbeitung muss der Nutzer den "Speichern" Button betätigen.

### 8.3.3 Löschen von Geräten

Zum Löschen eines Gerätes führt der Nutzer Schritt 1 von Editieren von Geräten aus und betätigt darauf folgend den "Löschen" Button.

Active	Ip	Model	SerialNumber	AssetNumber	HostName
<input checked="" type="checkbox"/>	192.168.1.100	CLX-4190 Series	Z8NLBJFH8000EHZ		SEC30CDA7F279AB.fritz.box

## 8.4 Werte abfragen und senden

Sobald der Nutzer seine MPS Geräte über die automatische Suche oder die manuelle Bearbeitung von Geräten hinzugefügt hat kann er den Button "Werte Abfragen" betätigen.

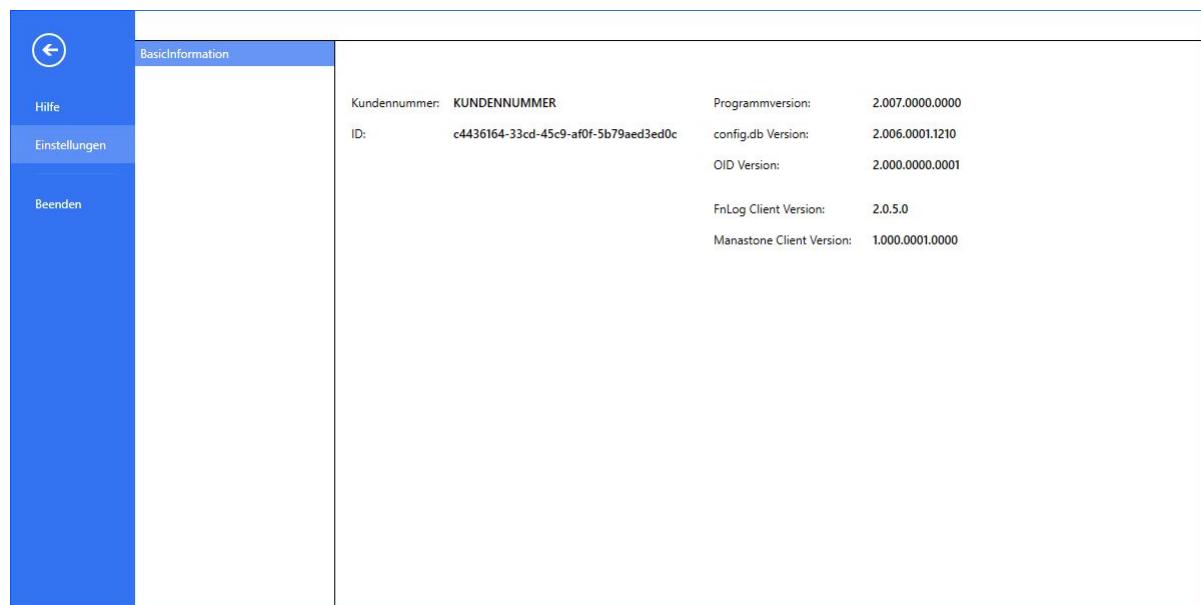
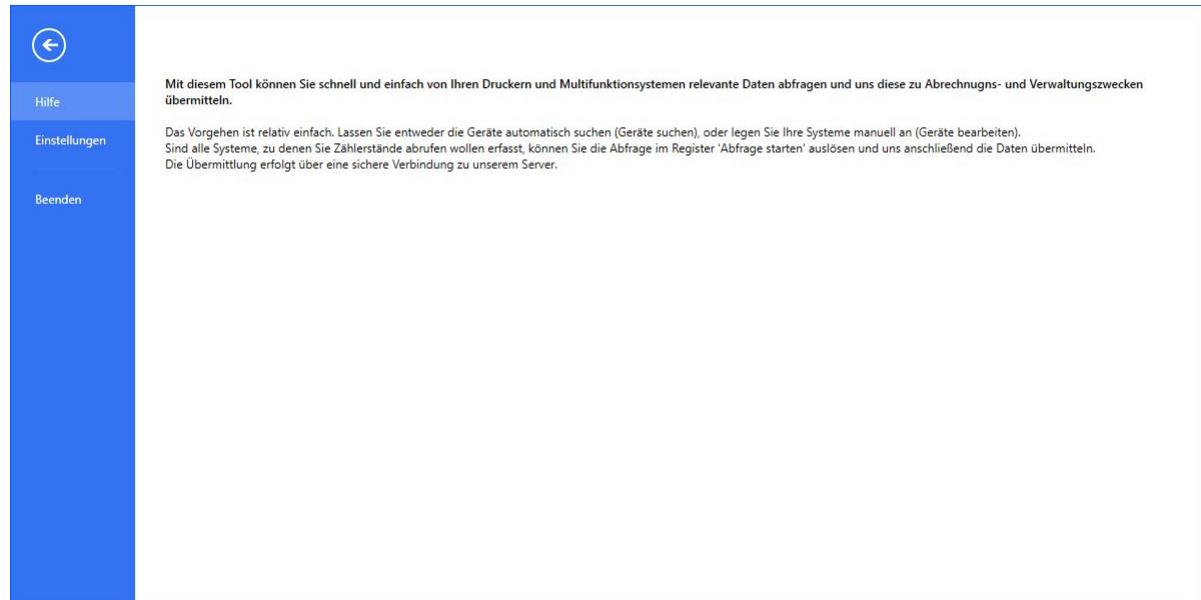
**Eine Abfrage ohne Angabe eines Gerätes wird ein leeres Resultat liefern!**

Sobald die Abfrage erfolgreich durchgeführt wurde, wird das Ergebnis in tabellarischer Form angezeigt und der Nutzer kann den "Senden" Button betätigen.  
Nach kurzer Zeit wird der Client den Nutzer über die erfolgreiche oder gescheiterte Übertragung informieren.

openMPS 2.7.0.0											
Datei		Geräte									
Geräte Suchen		Geräte bearbeiten		Werte abfragen und senden							
Werte abfragen								Senden			
CustomerReference		VendorName		Model		SerialNumber		MacAddress		IpAddress	
KUNDENNUMMER		CLX-4190 Series		Z8NLBJFH8000EHZ		30 CD A7 F2 79 AB		192.168.1.100		SEC30CDA7F279AB	
DescriptionLocation		AssetNumber		FirmwareVersion		PowerSleep1		PowerSleep2		V4.00.03.16 JULY-01-2016	

## 8.5 Infoseiten

Auf den nachfolgenden Seiten kann der Nutzer eine Kurzanleitung oder aber Programminformationen einsehen, diese können über den "Datei" Button im oberen linken Bereich des Programms angezeigt werden.



# Kapitel 9

## Literaturverzeichnis

- [Wik19a] Wikipedia. Simple network management protocol. [https://de.wikipedia.org/w/index.php?title=Simple\\_Network\\_Management\\_Protocol&oldid=187406207](https://de.wikipedia.org/w/index.php?title=Simple_Network_Management_Protocol&oldid=187406207), 2019.
- [Wik19b] Wikipedia. Socket.io. <https://de.wikipedia.org/w/index.php?title=Socket.IO&oldid=180121497>, 2019.