Abteilung Informatik
ITA – Institut für Internet-Technologien und Anwendungen

Entwicklung eines Besuchermonitoring-Systems

Autor: **Marcel Germann**
Kürzel: mg

Gewerk

Betreuer: Prof. Dr.-Ing. Andreas Rinkel
Co-Betreuung: Mathias Manz
Experte: Prof. Dr. Lothar Müller

Erstellt am: **25. September 2009**
Abgabedatum: **18. Dezember 2009**

Änderungsnachweis

Version	Änderungsgrund	Kurz-Z.	Datum
1.0	Erstellen des Dokumentes	mg	25.9.09

Inhalt

1 J2ME – Applikation

Im Kapitel 1 werden die Installation, die für die Erstellung einer Natelapplikation nötig sind, erklärt. Kapitel 1.6 befasst sich mit Problemen, die bei der Entwicklung aufgetaucht sind.

1.1 Installation Entwicklungsumgebung NetBeans

NetBeans ist eine Opensource Entwicklungsumgebung die komplett in Java geschrieben wurde und von Sun Microsystems gesponsert wird. NetBeans unterstützt alle Java Plattformen wie zum Beispiel Java SE, Java Web, Java Enterprise Java Beans und auch Java ME (Micro Edition). Die Entwicklung von Mobilen Applikationen unterstützt NetBeans mit nützlichen Erweiterungen wie den Flow Designer oder den GUI Builder mit dem Namen Visual Form Designer.

NetBeans kann direkt von der offiziellen Homepage www.netbeans.org heruntergeladen werden. Zu beachten ist dass ein Bundle gewählt wird, das Java ME integriert hat.

Die Installation kann mit der Standardkonfiguration durchgeführt werden.

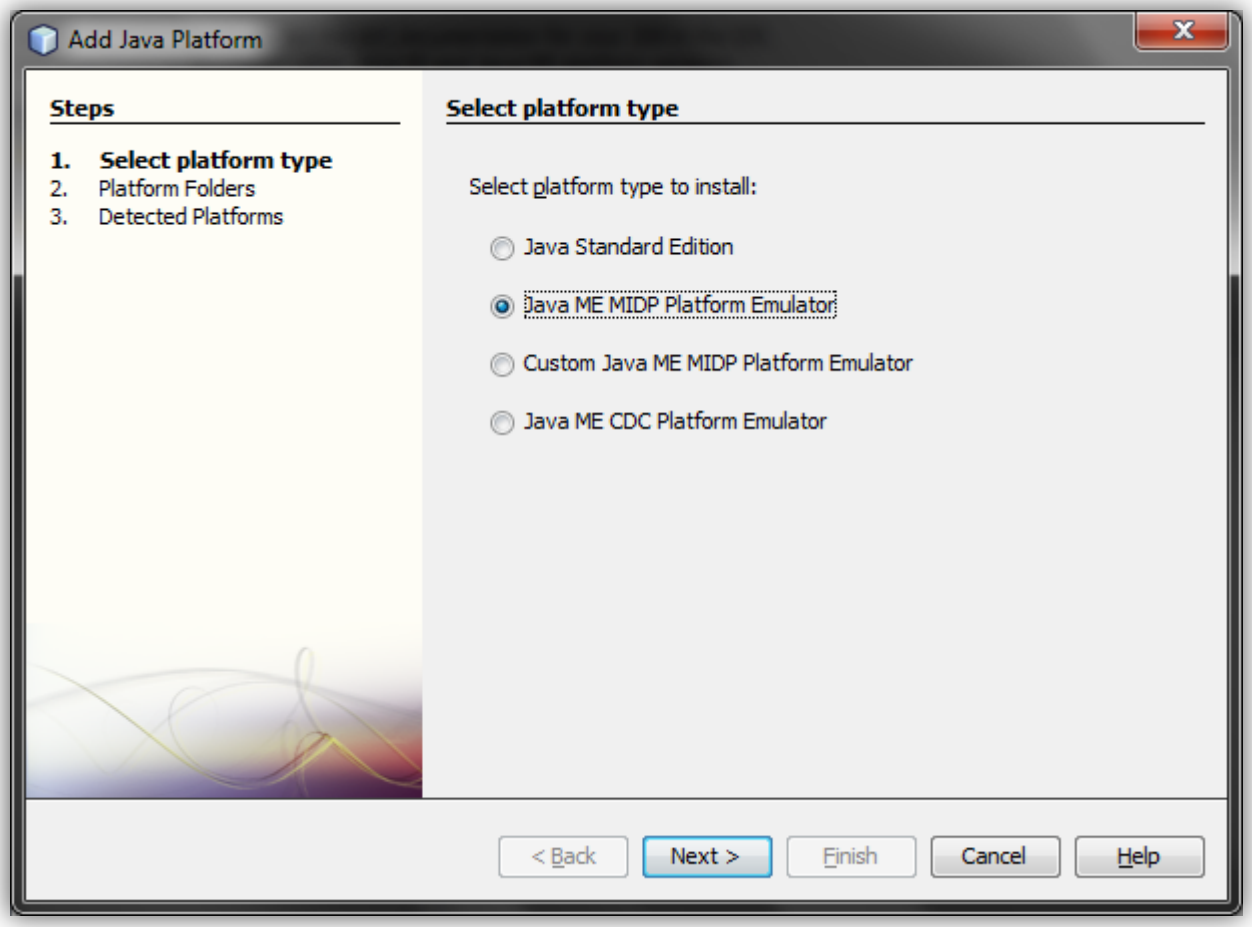
Die Entwicklung der Natelapplikation für Visiman wird mit der Version 6.7.1 von NetBeans durchgeführt.

1.2 Installation Nokia Series 40 SDK

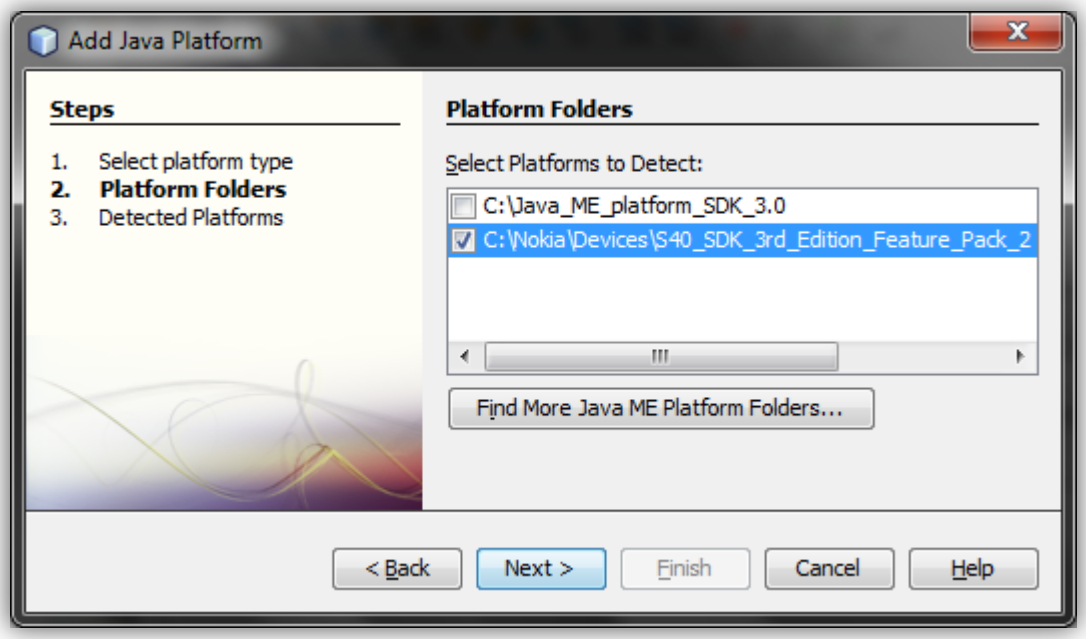
Damit die entwickelte Applikation in der Entwicklungsumgebung getestet werden kann und damit das Debugging funktioniert wird ein SDK benötigt. In der Semesterarbeit wird das Nokia Series 40 SDK benutzt. Dieses ist auf der Nokia Homepage nach einer kostenlosen Registrierung erhältlich. Die Installation kann an einem selbstdefinierten Ort durchgeführt werden. Damit keine Fehler entstehen, wird ein Pfad ohne Leerzeichen empfohlen.

Das SDK kann nun als Plattform in NetBeans integriert werden. Dies geschiet in NetBeans über das Menu „Tools → Java Platform → Add Plattform“.

Als Plattformtyp soll der „Java MIDP Platform Emulator“ ausgewählt werden.



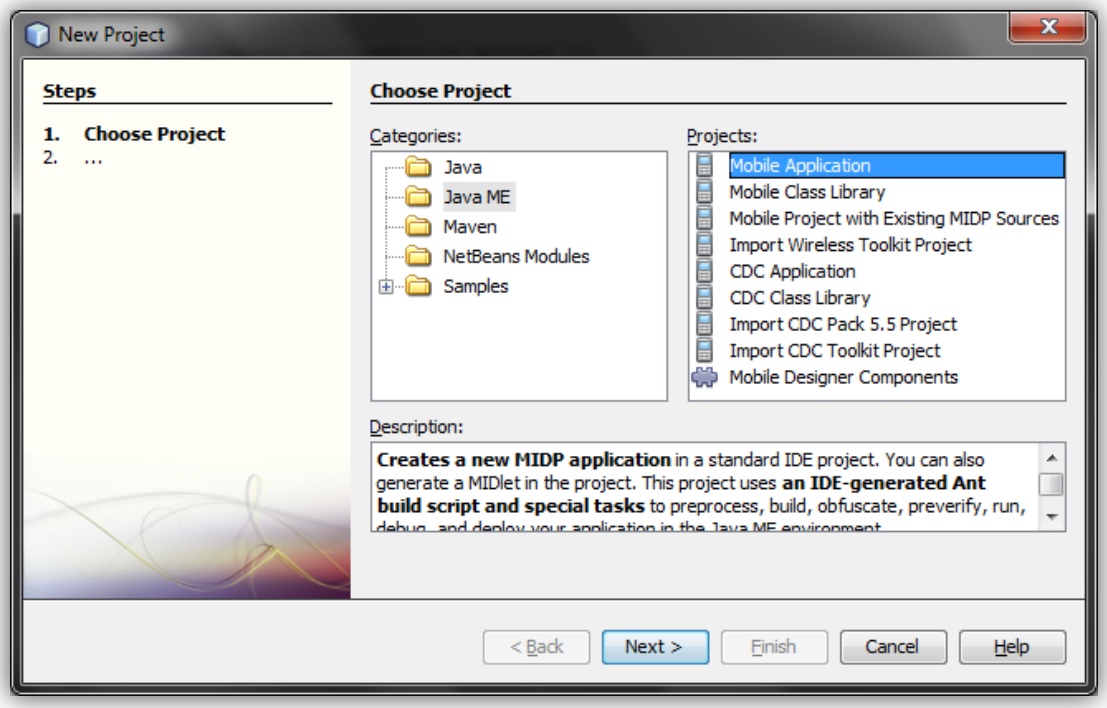
NetBeans untersucht nun den Computer nach verfügbaren Plattformen. Wird die vorher installierte Nokia Plattform nicht gefunden, kann der Pfad auch manuell über „Find More Java ME Platform Folders…“ eingegeben werden.



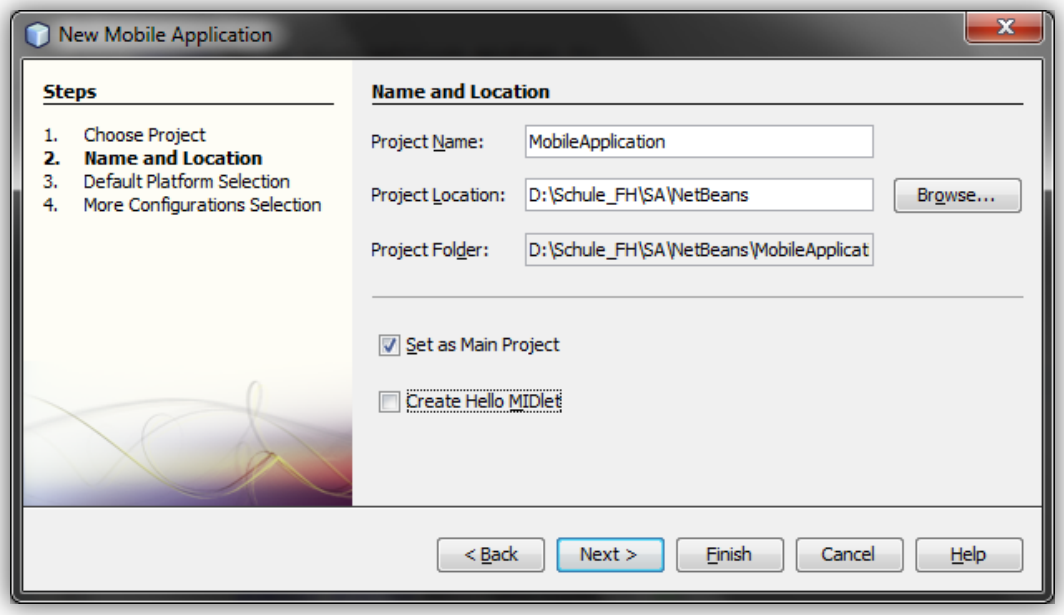
Nach der Auswahl der zu installierenden Plattform, wird diese nach Profilen durchsucht und kann nach der Durchsuchung mit einem Klick auf „Finish“ installiert werden.

1.3 Java ME Projekt erstellen

NetBeans ist nun fertig installiert und bereit für die Entwicklung von Mobile Applikationen. Um ein neues Java ME Projekt zu erstellen geht man auf „File → New Project“.
Als Kategorie wählt man „Java ME“ und als Projekt „Mobile Application“.



Im nächsten Schritt kann der Applikation einen Namen gegeben und der Projektpfad definiert werden.



Mit dem setzen des Häckchen bei „Set as Main Projekt“ wird das Projekt als Standardprojekt gesetzt. Das Standardprojekt wird zum Beispiel ausgeführt, wenn kein Projekt ausgewählt wird und der Run-Knopf gedrückt wird.

Beim zweiten Häckchen wird die Möglichkeit gegeben, ein Hello MIDlet zu erzeugen. Dieses erzeugt eine kleine Applikation, die „Hello World!“ auf dem Natelbildschirm ausgibt. Diese Startapplikation ist nützlich, um die Installation von NetBeans und des Plattform Emulators zu testen.

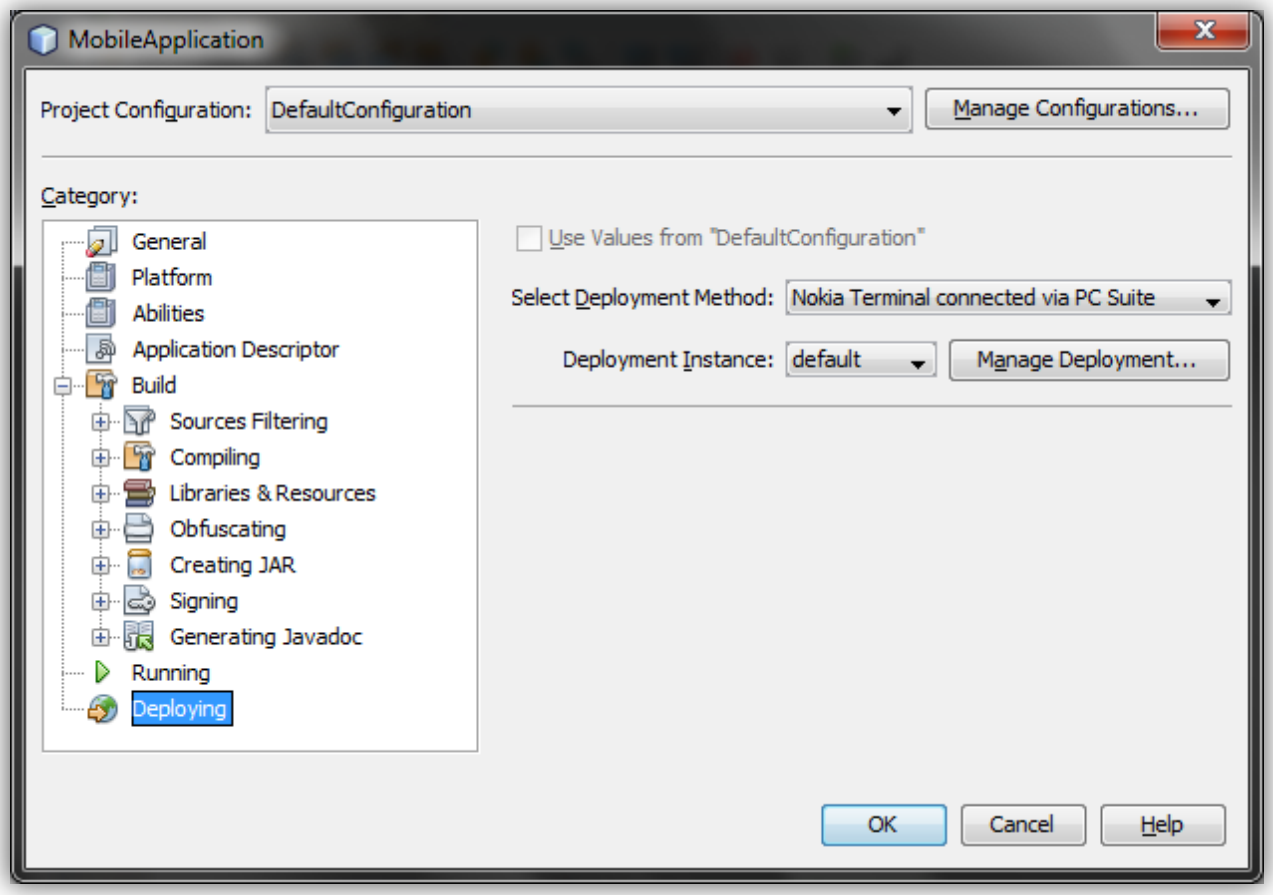
Im nächsten Schritt wird die Standard Plattform ausgewählt. Diese kann später auch in der Projektkonfiguration geändert werden. Als Standard wird hier die vorher installierte Series 40 Plattform gewählt.

Im 4. Installationsschritt können Projekt Konfigurations Vorlagen ausgewählt werden. Da in der Studienarbeit keine Vorlagen gebraucht werden, kann dieser Schritt ausgelassen werden. Mit einem Klick auf „Finish“ wird das Projekt erstellt und ist nun bereit für die Entwicklung.

1.4 Konfiguration des Projektes anpassen

Damit das Projekt einfach auf einem Natel installiert werden kann bietet NetBeans die Möglichkeit, die Applikation direkt auf das Natel zu installieren. Dies kann in den Einstellungen des Projektes definiert werden. Die Einstellungen können mit einem Rechtsklick auf das Projekt, über „Properties“ eingesehen und bearbeitet werden.

Die Methode zur direkten Installation nennt sich „Deploying“ und kann im gleichnamigen Reiter konfiguriert werden.



Während dieser Arbeit wird mit einem Nokia 3110 Classic getestet, daher ist im Bild die Methode „Nokia Terminal connected via PC Suite“ ausgewählt.

Um die Applikation nun auf das Natel zu installieren, muss die Nokia PC Suite installiert sein. Sobald die Nokia PC Suite das Natel erkannt hat, kann die Natelapplikation mit einem Rechtsklick auf das Projekt, über den Punkt „Deploy“ auf dem Natel installiert werden.

1.5 J4ME installieren

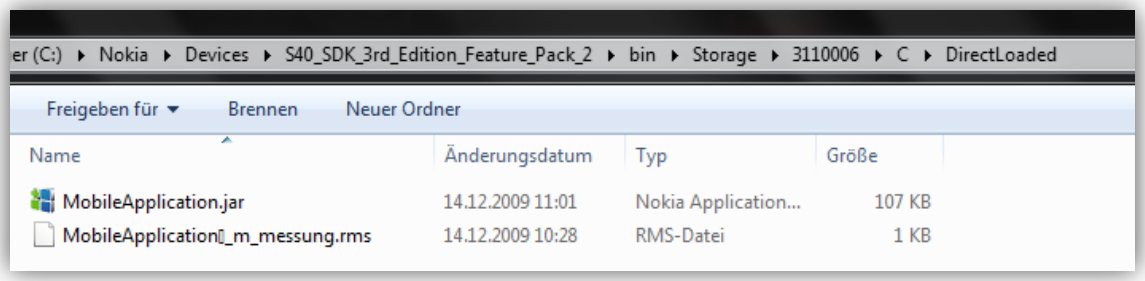
J2ME bietet keine Möglichkeit einen Stil, das heisst Farben, Schriftgrössen oder Schriftarten, einheitlich in die Applikation zu bringen. Auch die Elemente sehen bei jedem Nateltyp wieder anders aus. Es gibt nun viele Anbieter von Bibliotheken die diese Probleme verbessern und das Erstellen von Benutzeroberflächen erleichtern. In dieser Studienarbeit wird die Open Source Bibliothek J4ME verwendet, da diese klein und einfach anzuwenden ist. Zusätzlich zur Benutzeroberfläche bietet J4ME auch Unterstützung mit GPS Daten.

J4ME kann direkt von der Projektwebseite heruntergeladen werden und ist als Eclipse Projekt erhältlich, welches auch in NetBeans importiert werden kann.

1.6 Probleme und Lösungen

1.6.1 Null Pointer Exception auf Emulator

Beim Zugriff auf den RecordStore erscheint im Emulator eine Null Pointer Exception. Dies wird verursacht wenn eine frühere Instanz des Emulators nicht richtig beendet wurde.
Das Problem kann behoben werden indem der die .jar, sowie die .rms Datei der jeweiligen Emulator Instanz gelöscht wird. Diese Dateien befinden sich im Ordner „ *<EmulatorSDK>\bin\Storage\<Instanznummer>\C\DirectLoaded\“*



1.6.2 Zählung wird bei vielen Messwerte sehr langsam

Wenn die Zählung über 100 Messwerte besitzt, kommt die Applikation ins stocken. Je mehr Messwerte, desto länger dauert es, bis die Anzeige aktualisiert wird. Problem ist die lange Bearbeitungszeit, da die Messwerte immer direkt aus dem Record Store gelesen werden und dazu der ganze Record Store durchgearbeitet werden muss.
Als Lösung wurde ein Messwertobjekt „total“ erstellt. Beim einfügen neuer Messwerte, werden diese zwar immer noch direkt im Record Store gespeichert, doch zusätzlich wird auch das total-Objekt aktualisiert. Die Anzeige wird mit den Daten des total – Objektes aktualisiert. Die Zählung läuft nun auch mit vielen Messwerten schnell.

2 Visiman Datenbank

Im folgenden Kapitel wird der Aufbau der Visiman Datenbank erklärt.

2.1 Datenmodell

Nagios speichert nur die Statuswerte. Bei der Besucherzählung kommen viele weitere Informationen dazu, daher wird eine Datenbank verwendet. In der Visiman – Datenbank werden die Messwerte mit Datum und Attributen gespeichert.

Die Visiman Datenbank baut auf zwei Haupttabellen auf: „sensor_einsatz“ und „messung“.

Der Einsatz definiert eine Besucherzählung während einer bestimmten Zeit an einem bestimmten Ort. Dem Einsatz ist immer ein Sensor zugeordnet und hat einen Standort. Wenn der zugeordnete Sensor umplatziert wird, muss ein neuer Einsatz erstellt werden. Dies wird so definiert, damit bei der Auswertung keine Fehlinterpretationen entstehen.

Die zweite Tabelle ist die Messung. Sie bestimmt wieviele Personen, während einer bestimmten Zeit, den Sensor aktiviert haben. Der Messung können verschiedene Attribute zugeordnet werden.

Die Tabelle „person“ beinhaltet Informationen über die Zählperson und kann einem Einsatz zugeordnet werden.

Die Tabelle „ereignis“ wird für die Auswertung benötigt und beinhaltet wichtige Feiertage oder spezielle Ereignisse und Feste. Diese können entweder für alle Einsätze gültig sein oder einem Einsatz speziell zugeordnet werden.

2.1.1 Die Tabellen

sensor_einsatz

Der Einsatz eines Sensors hat eine Startzeit und bekommt beim beenden eine Endzeit. Der Einsatz kann nur an einem Standort stattfinden. Wird am Sensor eine Modifikation vorgenommen oder wechselt der Sensor den Standort muss ein neuer Einsatz erstellt werden.

sensor_definition

Die Definition ist die Hardware des Sensors, z.B. die Zählmatte oder die Person die eine Zählung durchführt.

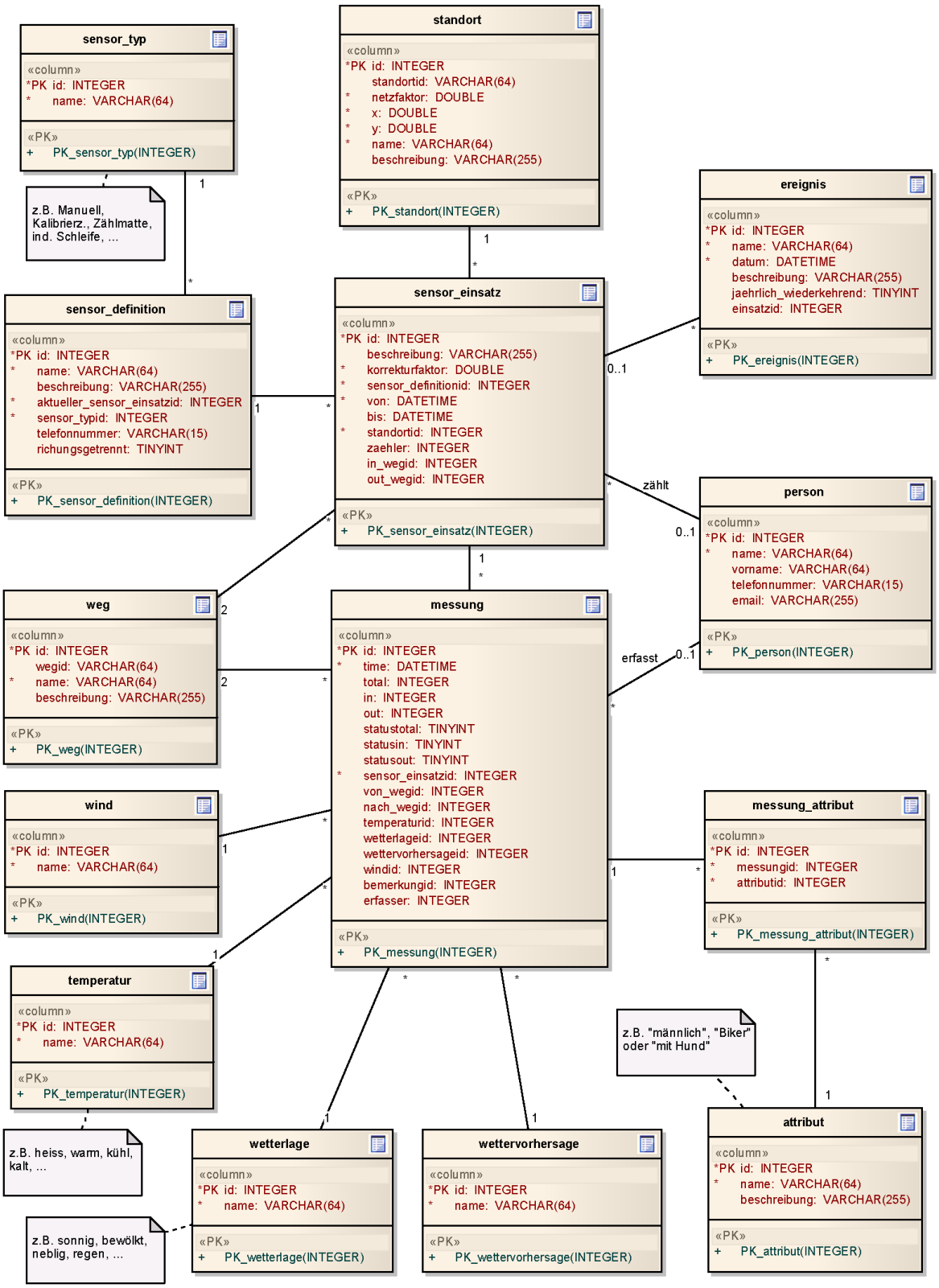
Zur Überprüfung damit keine falschen Messwerte definiert werden kann hier eine Telefonnummer definiert werden. Nur wenn Telefonnummer und Einsatz übereinstimmt werden die Messwerte in die Datenbank gespeichert (wird vom Check Plugin überprüft).

sensor_typ

Ein Sensor (*sensor_definition*) hat ein bestimmter Typ, z.B. „Zählmatte v1.2“ oder „Natel-Applikation“. Dieser ist notwendig um die per SMS empfangene Nachricht zu parsen und die richtigen Daten in die Datenbank einzutragen

standort

Jeder Einsatz findet an einem bestimmten Standort statt. Dieser kann mit *x* und *y*-Koordinaten positioniert werden.


sensor_typ
«column»
*PK id: INTEGER
* name: VARCHAR(64)
«PK»
+ PK_sensor_typ(INTEGER)
z.B. Manuell, Kalibrierz., Zählmatte, ind. Schleife, ...
standort
«column»
*PK id: INTEGER
standortid: VARCHAR(64)
* netzfaktor: DOUBLE
* x: DOUBLE
* y: DOUBLE
* name: VARCHAR(64)
beschreibung: VARCHAR(255)
«PK»
+ PK_standort(INTEGER)
ereignis
«column»
*PK id: INTEGER
* name: VARCHAR(64)
* datum: DATETIME
beschreibung: VARCHAR(255)
jaehrlich_wiederkehrend: TINYINT
einsatzid: INTEGER
«PK»
+ PK_ereignis(INTEGER)
sensor_definition
«column»
*PK id: INTEGER
* name: VARCHAR(64)
beschreibung: VARCHAR(255)
* aktueller_sensor_einsatzid: INTEGER
* sensor_typid: INTEGER
telefonnummer: VARCHAR(15)
richungsgetrennt: TINYINT
«PK»
+ PK_sensor_definition(INTEGER)
sensor_einsatz
«column»
*PK id: INTEGER
beschreibung: VARCHAR(255)
* korrekturfaktor: DOUBLE
* sensor_definitionid: INTEGER
* von: DATETIME
bis: DATETIME
* standortid: INTEGER
zaehler: INTEGER
in_wegid: INTEGER
out_wegid: INTEGER
«PK»
+ PK_sensor_einsatz(INTEGER)
zählt
person
«column»
*PK id: INTEGER
* name: VARCHAR(64)
vorname: VARCHAR(64)
telefonnummer: VARCHAR(15)
email: VARCHAR(255)
«PK»
+ PK_person(INTEGER)
weg
«column»
*PK id: INTEGER
wegid: VARCHAR(64)
* name: VARCHAR(64)
beschreibung: VARCHAR(255)
«PK»
+ PK_weg(INTEGER)
messung
«column»
*PK id: INTEGER
* time: DATETIME
total: INTEGER
in: INTEGER
out: INTEGER
statustotal: TINYINT
statusin: TINYINT
statusout: TINYINT
* sensor_einsatzid: INTEGER
von_wegid: INTEGER
nach_wegid: INTEGER
temperaturid: INTEGER
wetterlageid: INTEGER
wettervorhersageid: INTEGER
windid: INTEGER
bemerkungid: INTEGER
erfasser: INTEGER
«PK»
+ PK_messung(INTEGER)
erfasst
wind
«column»
*PK id: INTEGER
* name: VARCHAR(64)
«PK»
+ PK_wind(INTEGER)
messung_attribut
«column»
*PK id: INTEGER
* messungid: INTEGER
* attributid: INTEGER
«PK»
+ PK_messung_attribut(INTEGER)
temperatur
«column»
*PK id: INTEGER
* name: VARCHAR(64)
«PK»
+ PK_temperatur(INTEGER)
z.B. "männlich", "Biker" oder "mit Hund"
z.B. heiss, warm, kühl, kalt, ...
wetterlage
«column»
*PK id: INTEGER
* name: VARCHAR(64)
«PK»
+ PK_wetterlage(INTEGER)
wettervorhersage
«column»
*PK id: INTEGER
* name: VARCHAR(64)
«PK»
+ PK_wettervorhersage(INTEGER)
attribut
«column»
*PK id: INTEGER
* name: VARCHAR(64)
beschreibung: VARCHAR(255)
«PK»
+ PK_attribut(INTEGER)
z.B. sonnig, bewölkt, neblig, regen, ...

messung
Wenn ein Besucher bei einem Sensor vorbeiläuft wird eine neue *messung* erstellt. Beschrieben wird dabei die Anzahl Personen, die Richtung, feste Attribute wie z.B. Wind, Wetter, Temperatur und Wettervorhersage. Zusätzlich können für jede Zählung weitere Attribute definiert werden wie z.B. Fussgänger, Biker, mit Hund, …

weg
Die Tabelle *weg* ist vordefiniert und kann als Attribut hinzugefügt werden.

wind
Die Tabelle *wind* ist vordefiniert und kann bei einer Messung ausgewählt werden.

temperatur
Die Tabelle *temperatur* ist vordefiniert und kann bei einer Messung ausgewählt werden. Beschrieben wird hier nicht die Temperatur in °Grad, sondern die gefühlte Temperatur wie z.B. „kühl“, „warm“, „heiss“, …

wetterlage
Die Tabelle *wetterlage* ist vordefiniert und kann bei einer Messung ausgewählt werden.

wettervorhersage
Die Tabelle *wettervorhersage* ist vordefiniert und kann bei einer Messung ausgewählt werden.

attribut
In der Tabelle *attribut* befinden sich vordefinierte Attribute wie z.B. Fussgänger, Biker, mit Hund, …

messung_attribut
Die Tabelle *messung_attribug* dient als Zwischentabelle um die bei einer Messung weitere Attribute der Tabelle *attribut* hinzufügen zu können.

person
In der Tabelle *person* können Informationen über die Zählperson gespeichert werden. Diese Tabelle wird nur in Verbindung mit einer manuellen Zählung gebraucht. Das Feld „telefonnummer“ dient zur Abfrage, ob die gesendete Nachricht mit Messwerten gültig ist.

ereignis
Die Tabelle *ereignis* kann Feiertage oder spezielle Ereignisse festhalten. Diese Tabelle ist für die Zählung irrelevant und wird erst bei der Auswertung der Messwerte gebraucht.

3 SMS – Gateway Interface

Im folgenden Kapitel wird erklärt, warum ein PHP Interface erstellt wird und welche Funktionen dieses Interface implementieren muss.

3.1 Anforderungen an den SMS Gateway

Es gibt verschiedene Anbieter von SMS Gateways, daher möchte man den Natur- und Nationalparks offen lassen welchen Anbieter sie benutzen. Der Gateway muss also austauschbar sein.

- Unterstützung von HTTP oder HTTPS für den Datenaustausch
- 2-Weg Nachrichten (Empfang und Versand)
- Versand von Binären Nachrichten (Bestätigungsnachricht auf Port 16738)

3.2 Aufbau eines Interfaces

Für den SMS Gateway wird ein Interface in PHP erstellt. Das Interface ist eine formale Deklaration, welche Funktionen existieren und wie diese angesprochen werden müssen. Der Vorteil dieser Methode ist, dass ein Modul, welches dieses Interface implementiert, ohne Probleme gegen ein anderes Modul ausgetauscht werden kann. Dies kann sogar ohne Umprogrammierung direkt in der Konfiguration („config.php“) gemacht werden.

Ein SMS Gateway Modul muss folgende Funktionen implementieren:

3.2.1 getSender()

Liest die Absendernummer aus der Nachricht und gibt diese als String zurück.

3.2.2 getValues()

Liest den Text aus der Nachricht und gibt diesen als String zurück.

3.2.3 getTime()

Liest die Sendezeit aus der Nachricht und gibt diese als Zeitstempel (Integer) zurück.

3.2.4 getKeyword()

Liest das Keyword aus der Nachricht und gibt dieses als String zurück.

3.2.5 sendConfirmation($phonenumber, $text)

Sendet eine Nachricht mit dem übergebenen Text an die übergebene Telefonnummer

3.3 SMS Gateway iNetWorx

Für das Senden wird die vorgeschlagene Funktion „auth_https_post“ aus dem iNetWorx API Beispiel (siehe Anhang C) verwendet.
Für eine Überprüfung, ob das SMS gesendet wurde, wird der Rückgabewert (HTTP Status) und allfällige Fehlermeldungen in die Logdatei „send_log_<datum>“ gespeichert.

3.4 Senden einer Binären Nachricht

Die Bestätigungsnachricht soll direkt an die Natelapplikation geschickt werden. Damit dies möglich ist, muss der *User Data Header* (Kopfteil der Nachricht) um einen Ziel- und einen Absender-Port erweitert werden. Dies ist nur mit sogenannten binären Nachrichten möglich.
Ein möglicher UDH kann folgendermassen aussehen:

```
06 05 04 41C5 41C6
```

Der Header setzt sich aus folgenden Teilen zusammen:

06	Länge des ganzen Headers. Im Beispiel besteht der Header aus 6 Bytes.
05	Format der Nummern im Headers. 05 steht z.B. für Hexadezimal.
04	Anzahl Charakter für jeden Port. Im Beispiel sind dies 4 Charakter damit hohe Ports benutzt werden können.
41C5	Ziel-Port im vorher definierten Format. 41C5 steht für Dezimal 16837.
41C6	Absender-Port im vorher definierten Format. 41C6 steht für Dezimal 16838.

Jeder Gateway hat eine eigene Spezifikation wie der User Data Header definiert und übertragen werden muss. Beim SMS Gateway von iNetWorx wird der Header über die POST Variable „udh“ übertragen. Die einzelnen Bytes müssen mit einem %-Zeichen getrennt werden. In einer fertigen Abfrage kann das folgendermassen aussehen:

```
user=user&pass=passwd&sender=VISIMAN&rcpt=+41791234567&msgbody=
id=54,result=1&udh=%06%05%04%41%C5%41%C6
```

Beim Zusammenstellen der Variablen ist zu beachten, dass alle zusätzlichen Informationen URL – Encoded sind. Das Bedeutet dass keine Sonderzeichen vorkommen.
Speziell zu erwähnen ist die Variable „sender“. Hier kann entweder eine Absendernummer eingetragen werden oder, wie im Beispiel, ein Namen verwendet werden. Dieser Name wird beim Empfang in der normalen Inbox als Absender angezeigt. Leider bereiten Sonderzeichen und Leerzeichen im Namen, vielen Nateltypen Problemen. Daher ist es anzuraten als Absender eine Natelnummer oder ein Namen ohne Leerzeichen und ohne Sonderzeichen zu verwenden.

3.5 Probleme und Lösungen

Der in der Bachelorarbeit „ATOM“ verwendete Provider ASPSMS unterstützt ein einem erweiterten Packet zwar den Empfang von Nachrichten, es wird jedoch eine Internationale Nummer verwendet. Dies würde die Kosten für den Empfang stark erhöhen, daher wird ein anderer Provider gesucht, der diesen Dienst in der Schweiz anbietet.
Nach einer Evaluation (siehe Techn. Bericht, Kapitel 2.9) wurde iNetWorx als SMS Gateway für die Studienarbeit ausgewählt.

4 Server

Im folgenden Kapitel wird das Check Plugin für den Server erklärt. Im Kapitel 4.3 werden Problem die mit dem Server aufgetreten sind aufgezeigt.

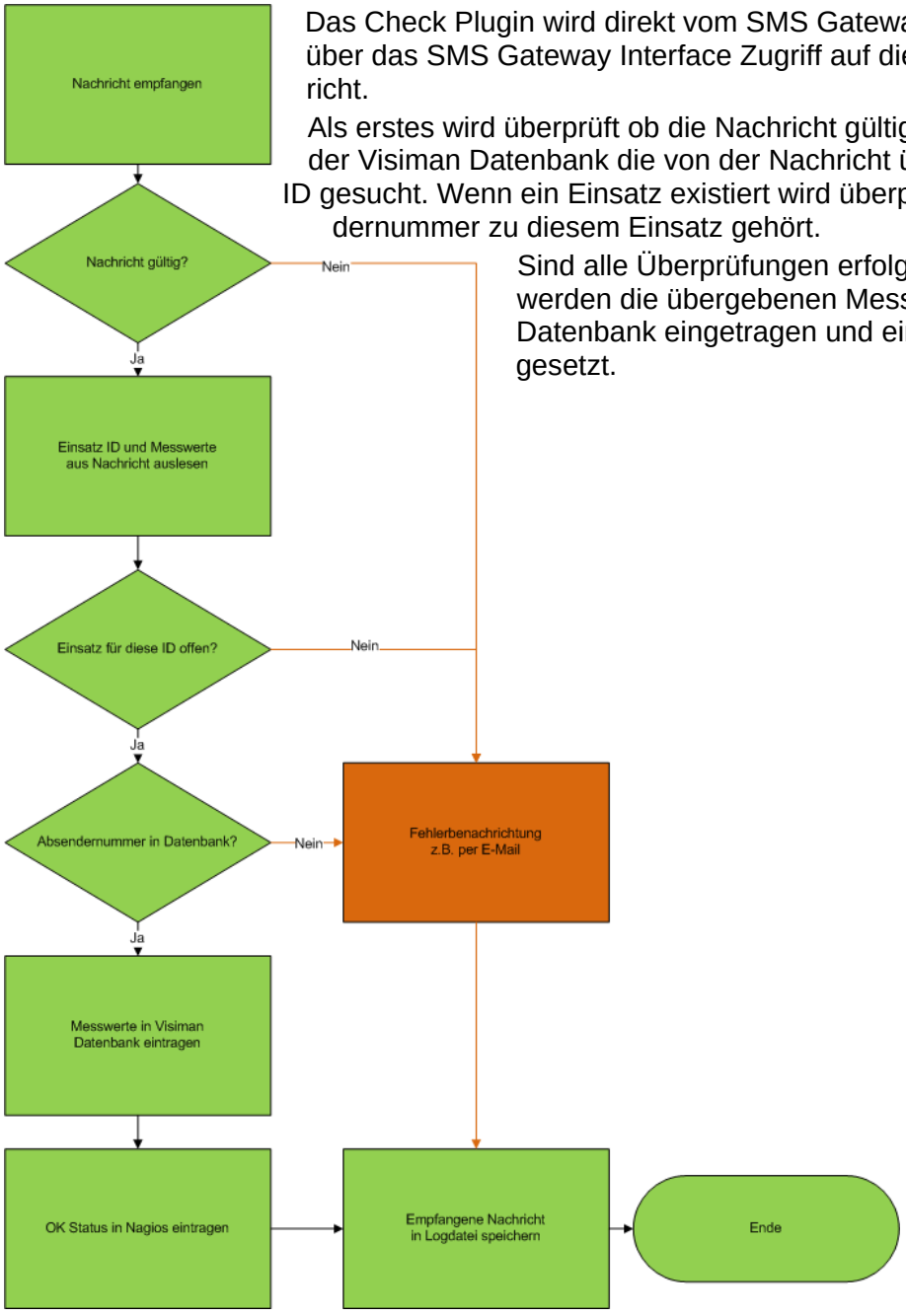
4.1 Übernahme der Installation von der Technologiestudie

Diese Studienarbeit wird als Folgearbeit der Bachelorarbeit „Technologiestudie VISIMAN“ durchgeführt. Die Serverinfrastruktur wird von der Bachelorarbeit direkt übernommen. Daher wird keine neue Installations- und Konfigurationsanleitung erstellt.

4.2 Check Plugin in PHP

Um die per SMS gesendeten Messwerte empfangen zu können, wird ein Check Plugin benötigt. Dieses kann in diversen Programmiersprachen geschrieben werden. Da wir die Nachricht per HTTP(S) erhalten, wird das PlugIn in der Skriptsprache PHP geschrieben. Dies ermöglicht einen einfachen Zugriff auf die Datenbank MySQL.

4.2.1 Ablauf des Check Plugin



Das Check Plugin wird direkt vom SMS Gateway aufgerufen und hat über das SMS Gateway Interface Zugriff auf die Variablen der Nachricht.

Als erstes wird überprüft ob die Nachricht gültig ist. Danach wird in der Visiman Datenbank die von der Nachricht übergebene Einsatz ID gesucht. Wenn ein Einsatz existiert wird überprüft, ob die Absendernummer zu diesem Einsatz gehört.

Sind alle Überprüfungen erfolgreich durchgelaufen, werden die übergebenen Messwerte in die Visiman Datenbank eingetragen und ein OK Status in Nagios gesetzt.

4.2.2 Eintragen der Werte in die Visiman Datenbank

Die übergebene Nachricht wird untersucht und jeder Messwert der gefunden wird, wird mit folgendem Code in die Datenbank eingefügt:

```
$sql = "INSERT INTO `messung` (`time`, `total`, `in`, `out`, `statusto-
tal`, `statusin`, `statusout`, `sensor_einsatzid`, `von_wegid`,
`nach_wegid`, `temperaturid` ,`wetterlageid`, `wettervorhersageid`
,`windid`) VALUES('". $time ."', '".$total."', '".$in."', '".$out."', 0,
0, 0, 1, 1, 3, 1, 2, 2, 1);";
$sql2 = "INSERT INTO `messung_attribut`(`messungid`,`attributid`) VALUES
(2,1);";

$ergebnis = mysql_query($sql);
if(!$ergebnis)
    $error .= "\n</br />Fehler: ".mysql_error();
$ergebnis2 = mysql_query($sql2);
```

4.2.3 Eintragen der Statuswerte in Nagios

Um die Statuswerte in Nagios einzutragen kann der Status kann in die Command-Line geschrieben werden, welche von Nagios periodisch während einem konfigurierbaren Zeitintervall abgearbeitet wird. Als Beispiel hier ein Ausschnitt aus dem Skript *„checkplugin.php“*:

```
...
$fh = fopen(_NAGIOS_COMMAND, 'w') or die("can't open command file");
$text = "[".mktime()."]
PROCESS_SERVICE_CHECK_RESULT;fussmatte1;total;0;OK - total ".$total."\n";
$text .= "[".mktime()."] PROCESS_SERVICE_CHECK_RESULT;fussmatte1;in;0;OK
- in ".$in."\n";
$text .= "[".mktime()."] PROCESS_SERVICE_CHECK_RESULT;fussmatte1;out;0;OK
- out ".$out."\n";
fwrite($fh, $text);
fclose($fh);
...
```

_NAGIOS_COMMAND ist eine Konstante, wird in der „config.php“ definiert und enthält den Pfad zur Nagios Command-Line.

4.3 Probleme und Lösungen

4.3.1 MySQL lässt sich nicht mehr starten

Beim Aufruf von phpMyAdmin erfolgt die Fehlermeldung: „Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock'“. Der Webserver funktionert jedoch problemlos. Normalerweise ist dieses Problem nach einem Neustart des MySQL Servers behoben. Der Server lässt sich mit folgendem Befehl neu starten:

```
/etc/init.d/mysql restart
```

Falls sich der Server nicht neu starten lässt ist ein Blick in das Logfile notwendig. Während der Studienarbeit standen folgende Zeilen im Logfile:

```
[ERROR] /usr/libexec/mysqld: Error writing file
'/var/run/mysqld/mysqld.pid' (Errcode: 28)
[ERROR] Can't start server: can't create PID file: No space left on de-
vice
```

Die Logdatei befindet sich normalerweise am folgenden Ort: *„/var/log/mysqld.log“*.

Das Problem war, dass auf dem System kein Speicherplatz mehr vorhanden war, da die Logdateien zu gross worden sind. Die oben genannte Fehlermeldung bestätigt diese Aussage und zur Überprüfung kann mit dem Befehl

```
df- h
```

die Speicherplatzbenutzung angezeigt werden.

Damit nicht lange nach der zu grossen Datei gesucht werden muss, können mit folgendem Befehl alle Dateien angezeigt werden, die grösser als 20MB sind.

```
find / -type f -size +20000k -exec ls -lh {} \; 2> /dev/null | awk '{
print $NF ": " $5 }'  | sort -nrk 2,2
```

5 VirtualBox

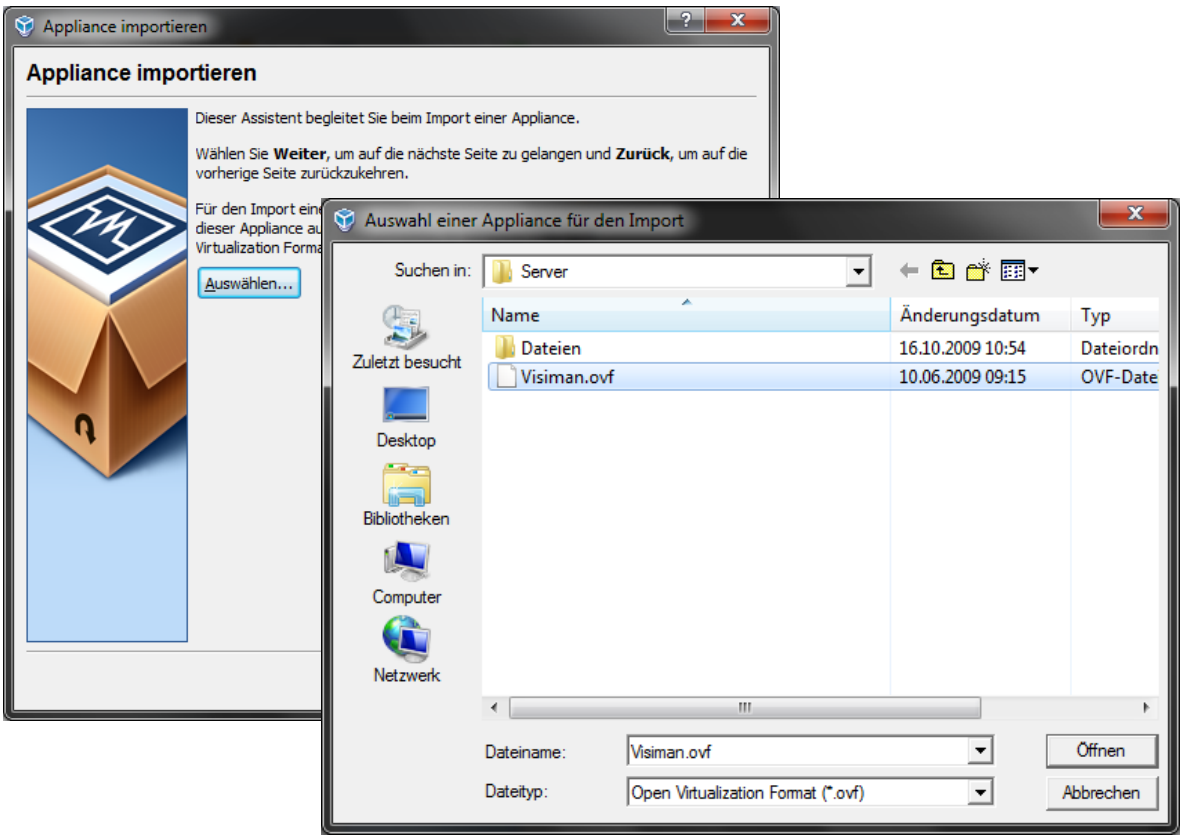
Im folgenden Kapitel wird die Installation von VirtualBox aufgezeigt und erklärt, was am Demosystem geändert wurde.

5.1 Installation

VirtualBox ist eine Virtualisierungssoftware für x86 Hardware. Mit VirtualBox ist es möglich auf einem Hostsystem ein weiteres Betriebssystem virtual laufen zu lassen. Dies wird in dieser Arbeit gebraucht, um den Server auf dem lokalen System auszutesten.

Die Software kann direkt von der offiziellen Homepage (http://www.virtualbox.org/) heruntergeladen und mit der Standardkonfiguration installiert werden.

Die Systeminstallation wurde von der Vorgängerarbeit übernommen. Die Installation kann über „Datei / Appliance importieren“ importiert werden



Nach dem Import kann das System gestartet werden.

5.2 Port Weiterleitung

Bei der Virtualisierung mit VirtualBox muss entschieden werden, wie das Gastsystem auf das Netzwerk zugreifen darf. Die Methode „Netzwerkbrücke“ bietet die Funktion dass das Gastsystem selbstständig per DHCP eine IP aus dem Netzwerk holt. Dies wird von vielen DHCP-Servern nicht unterstützt, da pro Netzwerkkarte nur eine IP-Adresse zugelassen wird.

Eine Alternative stellt NAT dar. NAT ermöglicht dass das Host-, sowie das Gastsystem über die gleiche IP-Adresse aufs Netzwerk zugreifen können. Bei dieser Einstellung kann vom Hostsystem nicht mehr direkt auf das Gastsystem zugegriffen werden. Lösung für dieses Problem ist die Port Weiterleitung. Diese kann für jeden Dienst eingerichtet werden. Am einfachsten geht Einrichtung über eine Batchdatei die einmal ausgeführt werden muss.

Für die Einrichtung muss die virtuelle Maschine heruntergefahren und gestoppt sein und der auf dem Hostsystem verwendete Port darf nicht von einem anderen Dienst schon in Benutzung sein.

Die Port Weiterleitung wird über die Commandozeile konfiguriert. Die in dieser Arbeit benutzte Konfiguration wird in der Batch-Datei „forward.bat“ bereitgestellt

```
"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/ssh/Protocol" TCP
"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/ssh/GuestPort" 22
"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/ssh/HostPort" 22

"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/http/Protocol" TCP
"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/http/GuestPort" 80
"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/http/HostPort" 80

"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/https/Protocol" TCP
"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/https/GuestPort" 443
"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/https/HostPort" 443

"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/imap/Protocol" TCP
"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/imap/GuestPort" 143
"%ProgramFiles%"\Sun\VirtualBox\VboxManage.exe setextradata "Visiman"
"VBoxInternal/Devices/pcnet/0/LUN#0/Config/imap/HostPort" 143
```

5.3 Was wurde geändert

Die Vorgängerarbeit hat Nagios und das Visiman System bereits auf der VirtualBox installiert. Diese Installation wurde nun aktualisiert. Von VirtualBox 3.0.8 wurde auf die Version 3.1.2 aktualisiert. Mit dem integrierten Update Manager wurden alle Packete auf die neuste Version aktualisiert.

Für das Visiman Projekt wurde das neue Datenbankstruktur integriert und das CheckPlugin installiert.

6 Anhang

A Verwendete Abkürzungen

DB	Datenbank
GSM	Global Standard for Mobile Communication
J2ME	Java 2 Micro Edition
J4ME	Open Source Bibliothek für J2ME Anwendungen
JVM	Java Virtual Machine
HTTP	Hypertext Transfer Protocol
HTTPS	Secure HTTP
KVM	Kernel-based Virtual Machine
SQL	Structured Query Language (Abfragesprache für Datenbanken)
SMS	Short Messaging Service
UDH	User Data Header
UI	User Interface (Benutzeroberfläche)

B Passworttabelle

VirtualBox

Name	Adresse	Benutzername	Passwort
phpMyAdmin	http://localhost/phpmyadmin/	root	nagi0s
Nagios	http://localhost/nagios3/	root	nagi0s
Benutzer	Linuxbenutzer	visiman	nagi0s
Administrator	Linuxadministrator	root	nagi0s

Virtueller Server HSR

Name	Adresse	Benutzername	Passwort
phpMyAdmin	http://sinv-56014.edu.hsr.ch/phpmyadmin/	root	nagi0s
Nagios	http://sinv-56014.edu.hsr.ch/nagios/	nagiosadmin	nagi0s

SMS Gateway iNetWorx

Name	Adresse	Benutzername	Passwort
Benutzer	Gateway Benutzer	hsrsms	gRe75Ws8
Auth	HTTP – Authentication Login	hsrsms	axess2smsapp

C iNetWorx API

Description iNetWorx SMS Gateway API

Revision 1.5 - 16-SEP-2009

General

The iNetWorx AG SMS gateway provides the possibility to send SMS through an HTTP(S)-interface. Therefore, the SMS message is submitted as a POST request to the gateway as URL-encoded string with all mandatory and optional parameters. In addition, the POST request must be prepended with an HTTP authorization header to gain access to the interface itself. The return value from the request either contains the status message from the gateway upon successful forwarding the messsage to the mobile operator's SMSC or an error message describing the problem.

Messages once accepted by the SMSC can not be withdrawn or deleted anymore whether by iNetWorx AG nor by the mobile operator. All SMS that are accepted by the SMSC for delivery will be charged whether or not the message can be successfully terminated to the mobile phone number provided in the request. It is the customers responsibility to submit a correct set of data to the gateway, in particular supplying valid and existing recipient numbers.

In addition to the broadcast solution via an SMSC there is also an interface available for incoming messages through a shared mobile number (+41-79-303-2000). This interface forwards incoming SMS identified by the first word in the message – also known as keyword – to the corresponding user via HTTP(S) GET/POST requests or email. As an additional feature, incoming SMS can automatically be replied to the sender with a predefined message.

NOTICE: You are encouraged to skip SSL certificate checking when connecting to the broadcast API. Although the server uses a certificate signed by a public CA (Geocerts/Equifax) it may cause problems on systems not having this CA added to their trusted root authorities.

SMS Gateway:

Host:	`sms.inetworx.ch`
Broadcast URL:	`/smsapp/sendsms.php` **→ Use this URL to send messages.**
Options URL:	`/smsapp/options.php` **→ Use this URL to query the gateway for options.**
Authorization:	client-specific, base64-encoded as `<user>:<password>`
	NOTICE: The URLs above can either be accessed via a secure HTTPS- or an unencrypted HTTP-connection. We strongly encourage you to connect to the gateway using SSL only.

Broadcast POST Variables (URL: `/smsapp/sendsms.php`):

`user=`	SMS account user name (independent from authorization above).
`pass=`	SMS account password
`sender=`	Sender-id, either as number or up to 11-characters string, URL-encoded If the value for the sender is empty or the same as the recipient number, it will be overwritten with the gateway's short-id (30315) because of technical reasons. **IMPORTANT:** iNetWorx AG is obliged by the mobile operators to prevent inappropriate use of this functionality. In particular, it is strictly prohibited to specify any business numbers (090x) as sender-id. iNetWorx AG reminas the right to take action upon fraudulent use of this feature. **IMPORTANT:** The following characters cannot be displayed and may cause the message not to be shown on the phone at all: `|^€{}[]~\` **NOTICE:** Some newer Phones – mostly Smartphones such as APPLE's iPhone – do have problems displaying messages with alpha numeric sender-id. These phones recieve the message properly but are not able to display it. Messages sent to SMS-capable fixed network phones are **rejected** by the operator if the sender-id contains alpha numeric characters. In these cases you must use numbers only as sender-id.
`rcpt=`	Recipient number, both international or national notation is supported (+41/0041/07x), URL-encoded. Multiple recipients can be included as a semicolon (`;`) delimited string. **IMPORTANT:** Spaces in the recipient numbers are treated as delimiter by the gateway! For bulk messages the recipient numbers must be submitted as one string, delimited with a semicolon. It is not advised to execute multiple recipient messages as single POST-Requests due to throughput restrictions on the SMSC side. Exceeding this rate may result in arbitrary dropped messages.

iNetWorx AG · Sägereistrasse 29 · CH-8152 Glattbrugg
Phone +41 44 810 05 55 · Fax +41 44 810 05 65 · info@inetworx.ch · www.inetworx.ch

iNetWorx AG
NETWORK SERVICE PROVIDER

`msgbody=` SMS message, URL-encoded.
Messages longer than 160 characters/bytes are automatically split into multiple SMS (max. 5 SMS) and reassembled as one single message on the recipients mobile. There is an additional overhead of about 8 characters/bytes per SMS (operator-dependant) for concatenation.
IMPORTANT: The message must be URL-encoded!
IMPORTANT: The following characters require two bytes: `|^€{}[]~\`

`mclass=` (optional) Flash SMS, SMS will be sent directly to mobile phone's display.
`mclass = 1`: Flash-SMS

`defer=` (optional) Deferred delivery.
`defer > 0`: postponed delivery in minutes (max. 7 days, minus tolerance.)

`dlrurl=` (optional) URL to forward delivery reports to.
Delivery reports from the mobile operator will be forwarded to the URL provided as an HTTP(S) GET request. When this option is present the message ID is returned by the server (see also below).
Every time the status of a message changes the SMS gateway is executing the `dlrurl` with the variables `msgid=<id>&status=<code>` appended to the URL initially passed as `dlrurl`. The status code is one of the following integers:
`1`: Message delivered, `2`: Message failed, `4`: Message buffered, `8`: Message acknowledged by SMSC
`16`: Message rejected by the SMSC
Usually, two dlr requests will be executed: The first one when the SMSC acknowledged the message and the second one upon delivery/failure.

Additional Broadcast POST Variable (Binary SMS):

`udh=` User Data Header part of message, URL-encoded.

Status Codes Returned by Server on Broadcast API:

`200: OK`	Message(s) successfully submitted to SMSC.
`200: OK (<rcpt>:<id>)`	Only when optional `dlrurl` is present, Message(s) successfully submitted to SMSC where `<rcpt>` is the recipient and `<id>` is an integer. With multiple recipients specified, the `<rcpt>:<id>` pairs are separated by a semi-colon (`;`).
`, Message Splits: <count>`	Appended to either one of the above for concatenated messages.
`401: Unauthorized`	Invalid user credentials (`user` or `pass`) submitted.
`402: Payment Required`	Quota is exhausted.
`404: Recipient Not Found`	Request did not have a valid `rcpt` value.
`500: Internal Server Error`	The interface did encounter an internal error. A closer description of the problem is appended in brackets to the error code.

Options POST Variables (URL: `/smsapp/options.php`):

`user=` SMS account user name (independent from authorization above).

`pass=` SMS account password

`option=` Selected option/information to be returned. Valid values are:

`quota:` Return the account's remaining amount of messages. Possible return values are:

`200: -1`	No quota set for account
`200: <number>`	Number of remaining messages.
`404: Invalid Option`	No action for option available.

iNetWorx AG · Sägereistrasse 29 · CH-8152 Glattbrugg
Phone +41 44 810 05 55 · Fax +41 44 810 05 65 · info@inetworx.ch · www.inetworx.ch

iNetWorx AG
NETWORK SERVICE PROVIDER

Variables Provided by Inbound SMS:

SMS received on the shared mobile number +41-79-303-2000 will be forwared to the destination configured in a well defined format. The destination can either be specified as an email address or an URL. In case of a web request it can be configured to execute the request as GET or POST via HTTP or HTTPS.

Note: In case of a web request configuration the incoming SMS will **not** be queued for further delivery attempts if the destination is unavailable.

`sender=`	Sender-ID SMS, in international notation (e.g. +4179123456789), may also be a string.
`keyword=`	Service identification keyword (first word in message).
`values=`	All values/text following the keyword.
`tstamp=`	Unix timestamp of request (local time zone: Zurich, CET)

Email Format (plain-text, iso-8859-1 coding):

```
Sender:    <sender-ID>
Keyword:   <string>
Values:    <string>
Timestamp: <Unix timestamp>
```

If you want to make use of the additional auto-reply feature of incoming messages, please specify the reply message as well as the sender ID of SMS replied upon account set up. These values cannot be changed interactively by the user, however, they will be altered by the iNetWorx staff at no charge.

iNetWorx AG · Sägereistrasse 29 · CH-8152 Glattbrugg
Phone +41 44 810 05 55 · Fax +41 44 810 05 65 · info@inetworx.ch · www.inetworx.ch

iNetWorx AG
NETWORK SERVICE PROVIDER

Code Example in PHP (Plain Text Message):

```php
function auth_https_post ($inarray) {
  // AUTH_HTTPS_POST: Request POST URL using basic authorization and SSL.
  // Input: inarray[0]: host name
  //        inarray[1]: service port
  //        inarray[2]: user:password
  //        inarray[3]: URL request
  //        inarray[4]: POST variables
  // Output: Message returned by server.

  // Build the header.
  $header = "POST ".$inarray[3]." HTTP/1.0\r\n";
  $header .= "Authorization: Basic ".base64_encode($inarray[2])."\r\n";
  $header .= "Host: ".$inarray[0]."\r\n";
  $header .= "Content-type: application/x-www-form-urlencoded\r\n";
  $header .= "Content-length: ".strlen($inarray[4])."\r\n\r\n";
  // Connect to the server.
  $connection = fsockopen("ssl://".$inarray[0], $inarray[1], &$errnum, &$errdesc, 10);
  if (! $connection) {
    $msg = $errdesc." (".$errnum.")";
  }
  else {
    socket_set_blocking($connection,false);
    fputs($connection,$header.$inarray[4]);
    while (! feof($connection)) {
      $newline = fgets($connection,128);
      switch ($newline) {
      // Skip http headers.
      case (strstr($newline, 'Content-')): break;
      case (strstr($newline, 'HTTP/1')): break;
      case (strstr($newline, 'Date:')): break;
      case (strstr($newline, 'Server:')): break;
      case (strstr($newline, 'X-Powered-By:')): break;
      case (strstr($newline, 'Connection:')): break;
      case "": break;
      case "\r\n": break;
      // Append output.
      default: $msg .= $newline;
      }
    }
    fclose($connection);
  }
  return $msg;
} // End of AUTH_HTTPS_POST


// Create function input array.
$inarray[0] = "sms.inetworx.ch";                // Gateway server.
$inarray[1] = "443";                            // Gateway server port (SSL).
$inarray[2] = "<myAUTHuser>:<myAUTHuserPass>";  // HTTP authorization user login.
$inarray[3] = "/smsapp/sendsms.php";            // Requested URL.
$inarray[4] = "";                               // SMS message details.

// Basic SMS message.
$inarray[4] = "user=<mySMSuser>&pass=<mySMSuserPass>";
$inarray[4] .= "&sender=".urlencode("0791234567")."&rcpt=".urlencode("0797654321");
$inarray[4] .= "&msgbody=".urlencode("Hello SMS World!");

// Full featured SMS message to multiple recipients and using flash and delayed delivery by 10 minutes.
$inarray[4] = "user=<mySMSuser>&pass=<mySMSuserPass>";
$inarray[4] .= "&sender=".urlencode("0791234567")."&rcpt=".urlencode("0797654321;0793456789");
$inarray[4] .= "&msgbody=".urlencode("Hello SMS World!")."&mclass=1&defer=10";


// Execute request and output return status.
echo auth_https_post($inarray);
```

More code examples in Perl, C++ (libcurl) or C# are available upon request.

iNetWorx AG · Sägereistrasse 29 · CH-8152 Glattbrugg
Phone +41 44 810 05 55 · Fax +41 44 810 05 65 · info@inetworx.ch · www.inetworx.ch

D Interface SMS Gateway

```php
<?php
/**
 * Interface fuer ein SMS Gateway im Projekt VISIMAN
 * Das Interface ist zuständig für die Kommunikation
 * zwischen dem Server (Check Plugin) und dem SMS Gateway
 */
interface SMSGateway{

    /**
     * Liest die Absenderadresse aus dem Request
     * des Gateways und gibt diese als String zurück
     *
     * @return String Absenderadresse der empfangenen Nachricht
     */
    public function getSender();

    /**
     * Liest den Text der Nachricht aus dem Request
     * des Gateways und gibt diesen als String zurück
     *
     * @return String Text der empfangenen Nachricht
     */
    public function getValues();

    /**
     * Liest die Zeit der Nachricht aus dem Request
     *
     * @return String Timestamp der Nachricht
     */
    public function getTime();

    /**
     * Liest das Keyword der Nachricht aus dem Request
     *
     * @return String Keyword der Nachricht
     */
    public function getKeyword();

    /**
     * Sendet eine Nachricht mit dem übergebenen Text
     * an die übergebene Telefonnummer
     *
     * @param String $phonenumber
     * @param String $text
     */
    public function sendConfirmation($phonenumber, $text);
}
?>
```

E Gateway Modul iNetWorx

```php
<?php

require_once 'SMSGateway.interface.php';

class inetworx implements SMSGateway{

    public function getSender(){
          return $_POST['sender'];
    }

    public function getValues(){
          return $_POST['values'];
    }

    public function getTime(){
          return $_POST['tstamp'];
    }

    public function getKeyword(){
          return $_POST['keyword'];
    }

    public function sendConfirmation($phonenumber, $text){
          // Create function input array.
          $inarray[0] = "sms.inetworx.ch"; // Gateway server.
          $inarray[1] = "443"; // Gateway server port (SSL).
          $inarray[2] = "hsrsms:axess2smsapp"; // HTTP auth user login.
          $inarray[3] = "/smsapp/sendsms.php"; // Requested URL.
          $inarray[4] = ""; // SMS message details.

          // Binary SMS message.
          $inarray[4] = "user=hsrsms&pass=gRe75Ws8";
          $inarray[4] .= "&sender=".urlencode("VISIMAN")."
          $inarray[4] .= &rcpt=".urlencode($phonenumber);
          $inarray[4] .= "&msgbody=".urlencode($text);
          // User Data Header erweitern mit Zielport 13837
          $inarray[4] .= "&udh=%06%05%04%41%C5%41%C6";

          $log = $this->auth_https_post($inarray);

          // Resultat in Sendlog speichern
          $msg = "\n[".date("Y-m-d H:i")."] ".$log." - ".$txt;
          $fh = fopen("log/send_log_".date("Ymd"),"a");
          fwrite($fh, $msg);
          fclose($fh);
    }

    private function auth_https_post ($inarray) {
```

```
          // AUTH_HTTPS_POST: Request POST URL using basic auth and SSL.
          // Input: inarray[0]: host name
          // inarray[1]: service port
          // inarray[2]: user:password
          // inarray[3]: URL request
          // inarray[4]: POST variables
          // Output: Message returned by server.
          // Build the header.
          $header = "POST ".$inarray[3]." HTTP/1.0\r\n";
          $header .= "Authorization: Basic
".base64_encode($inarray[2])."\r\n";
          $header .= "Host: ".$inarray[0]."\r\n";
          $header .= "Content-type: application/x-www-form-
urlencoded\r\n";
          $header .= "Content-length: ".strlen($inarray[4])."\r\n\r\n";

          $msg = "";
          // Connect to the server.
          $connection = fsockopen("ssl://".$inarray[0], $inarray[1],
&$errnum, &$errdesc, 10);
          if (! $connection) {
                $msg = $errdesc." (".$errnum.")";
          } else {
                socket_set_blocking($connection,false);
                fputs($connection,$header.$inarray[4]);
                while (! feof($connection)) {
                      $newline = fgets($connection,128);
                      switch ($newline) {
                            // Skip http headers.
                            case (strstr($newline, 'Content-')): break;
                            case (strstr($newline, 'HTTP/1')): break;
                            case (strstr($newline, 'Date:')): break;
                            case (strstr($newline, 'Server:')): break;
                            case (strstr($newline, 'X-Powered-By:')):
break;
                            case (strstr($newline, 'Connection:')): break;
                            case "": break;
                            case "\r\n": break;
                            // Append output.
                            default: $msg .= $newline;
                      }
                }
                fclose($connection);
          }
          return $msg;
    } // End of AUTH_HTTPS_POST


}
?>
```