monitord - Funkauswerter

Benutzerhandbuch, Stand 04/2011, Rev. 447 (SVN)

Inhalt

Inhalt

1 Einleitung

1.1 Der monitord, ein Funkauswerter

Im Bereich Amateurfunk, BOS-Funk (Behörden und Organisationen mit Sicherheitsaufgaben, z.B. Polizei und Feuerwehren, Rettungsdienste etc.) sowie im Betriebsfunk werden verschiedene Signalisierungsverfahren genutzt, um Geräte bzw. Personen zu rufen (z.B. zur Alarmierung von Einsätzkräften), Statusmeldungen abzugeben und Kurznachrichten zu übermitteln (zum Beispiel Einsatzorte). Der monitord ist in der Lage, drei der in Deutschland meist genutzten Signalisierungsverfahren zu dekodieren: Fünftonfolgen nach ZVEI-Standard, POCSAG-Telegramme und FMS-Meldungen.

Dekodierte Meldungen können per TCP-Socket "live" empfangen oder vom Auswerter in einer mySQL-Datenbank für die spätere Verwendung abgelegt werden.

1.1.1 Features

- Der monitord ist quelloffene Software (OpenSource) unter der GPL (v3) und damit kostenfrei
- Der monitord ist in Windows und Linux-Systemen auf 32 und 64 Bit-Architekturen lauffähig
- Die Überwachung von Sinalquellen ist sehr einfach über Line/Mic-Eingang der Soundkarte möglich
- Eine getrennte Überwachung der Kanäle rechts/links sowie die Verarbeitung mehrerer Soundkarten ist möglich
- Die Nutzung der Daten durch bekannte Programme wie FMS32Pro und FMS Crusader ist möglich (der monitord fungiert dann als Auswerte-Server)
- Durch offene Formate und Quellen sind Anpassungen an den eigenen Bedarf einfach realisierbar
- Die Übertragung der aufgefangenen Daten per TCP-Socket über Netzwerke (auch das Internet) ist möglich; damit kann z.B. eine zentrale Empfangsstation an funktechnisch gutem Standort Stationen mit schlechtem Empfang mit korrekten Daten versorgen

1.1.2 Gibt es eine History-Funktion?

Eine History-Funktion wie man sie von anderen Auswertern kennt, ist im Auswerter selber bisher NICHT implementiert. Entwickler, die dieses Feature (z.B. als Plugin) implementieren können und möchten, sind herzlich zur Mitarbeit eingeladen. Derzeit lässt sich eine History entsprechend nur über die Nutzung des mySQL-Plugins zur Speicherung und Datenbank-Abfrage durch ein externes Frontend erreichen.

1.2 Warnhinweis

Auch wenn es vielen Menschen reizvoll erscheint, ist das Abhören des Funkverkehrs aus dem nicht öffentlichen Bereich (BOS-Funk und nichtöffentlicher mobiler Landfunk nömL) nicht erlaubt. Entsprechend bitten wir darum, den monitord nur im Rahmen der gültigen Rechtslage (z.B. in Funkwerkstätten oder Feuer- und Rettungswachen) durch entsprechend befugtes Personal einzusetzen.

2 Download und Installation

2.1 Binary Packages

2.1.1 Windows/Win32

Für Windows stellen wir einen Installer zur Verfügung:

- monitord-01042011-r447.exe [SVN rev. 447, diverse Änderungen [1])]
- monitord-mingw32-2.0svn-05012011.exe [SVN rev. 435, neues POC1k2-Modul]

[1]) Changelog der aktuellen Version:

- LUA-Skripte werden nur noch gesucht und ausgeführt, wenn in der monitord.xml definiert; eine fehlende Angabe führte bisher zu Fehlermeldungen im Log (da nach Default-Files gesucht wurde) oder dazu, dass diese Default-Skripte ohne Angabe in der monitord.xml ausgeführt wurden
- FMS jetzt mit Zeilenumbrucherkennung für \10 und \13 (CR _und_ LF); wird nicht mehr als "#" ausgegeben, sondern als \n und landet auch so in der Datenbank
- Logging "versinnvollt" (INFO/DEBUG geändert)
- Der Client bekommt auf jede Eingabe auf dem monitord-Socket eine Rückmeldung - und wenn es eine Fehlermeldung ist (101:004, ILLEGAL FORMAT)
- Channel-Info wieder aktiv
- Aufzeichnung (Kommando 204) deaktiviert (Feedback: 001:005, NOT IMPLEMENTED)

Die Installation ist denkbar einfach: Führen Sie den Installationsassistenten aus und erhalten Sie einen als Programm ausführbaren Auswerter einschließlich Minimal-Konfiguration und Beispielskripten. Wie Sie diese anpassen, erfahren Sie in weiteren Bereichen.

2.1.1.1 Die Einrichtung als Dienst

Der monitord ist als Server- oder Hintergrundanwendung konzipiert. Entsprechend gibt es die Möglichkeit, ihn als Systemdienst zu installieren. Eine entsprechende Batchdatei liegt bei, manuell können Sie auch *monitord.exe --install* aufrufen. Nach einmaliger Installation als Dienst steht ihnen der monitord dann bei jedem Systemstart zur Verfügung.

2.1.2 Linux

Für Linux bieten wir zur Zeit keine fertigen Pakete an. Bitte laden Sie die Quellen aus dem SVN und compilieren Sie selbst.

2.2 Sourcecode

Wir empfehlen den manuellen Checkout aus dem SVN-Repository, um eine Übersicht über den bestehenden aktuellen Code zu erhalten. Dennoch haben wir hier den /trunk des SVN aktuell als Tarball zusammengefasst (im 3-Stunden-Takt).

- http://builds.monitord.de/monitor-trunk.tar.bz (

2.2.1 SVN-Zugriff

Auf das Repository kann lesend ohne die Angabe von Benutzername und Kennwort zugegriffen werden. Wir haben die übliche Struktur eingerichtet, bestehend aus /trunk (die aktuelle Entwicklung), /tags (besondere Revisionen) und /branches (Weiterentwicklungen parallel zum eigentlichen Kern des monitord). Das SVN-Repository ist erreichbar unter:

- http://svn.monitord.de/monitor

Benutzername und Passwort für den schreibenden Zugriff auf das Repository können bei entsprechendem Engagement bei den Entwicklern angefragt werden. Einfachster Weg dafür ist das Forum (bei gleichzeitigem Kennenlernen und Beschnuppern (siehe "Ressourcen").

3 Konfiguration

Die Konfiguration des monitord geschieht ausschließlich über eine Konfigurationsdatei namens "monitord.xml", die im Programmverzeichnis liegen sollte. Zur Vorverarbeitung von Daten können LUA-Skripte genutzt werden, dazu mehr in einem späteren Kapitel.

3.1 Allgemeiner Aufbau der monitord.xml

Die Konfigurationsdatei beginnt mit einem allgemeinen Block. Hier sind die XML-Strukturangaben, der Name des monitord-Servers, Ort und Name des Logfiles des Hauptprogramms, der Loglevel und die Lage zweier Filterskripte definiert. Ausführlich werden die Angaben in weiteren Kapiteln erläutert.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<monitordconfig version="1.0">
<name> Monitord </name>

<logfile> monitord.log </logfile> <!-- screen = Bildschirm -->
<loglevel> INFO </loglevel>

<SocketFilterScript> socketfilter.lua </SocketFilterScript>
<PluginFilterScript> pluginfilter.lua </PluginFilterScript>

(... auth, tcpsocket, soundcard, dataplugins ...)

</monitordconfig>
```

Die Einstellungen zum Thema Authentifizierung (auth), dem Serververhalten und den Auswertungen und Speicherorten in dieser Datei sind speziellerer Natur. Daher sind ihnen eigene Unterkapitel gewidmet, als Platzhalter für diese Blöcke seien die (...) oben zu erkennen.

3.2 Logfile und Loglevel

logfile und loglevel sind zwei Tags, die auch bei den Plugins verwendung finden. Sie regeln die Ausgabe von Informationen durch den Auswerter und die Plugins. Da der monitord als Dienst konzipiert ist, schreibt dieser seine Ausgaben in der Regel eher in ein Logfile als sie am Bildschirm auszugeben. Auch das ist durch Angabe von "screen" jedoch möglich. Das Loglevel kann enthalten DEBUG, ERROR und INFO, wobei DEBUG sehr viele Ausgaben generiert und diese Einstellung nicht für den Dauereinsatz empfohlen ist.

3.3 Einzelne Abschnitte der monitord.xml

3.3.1 <auth>-Sektion

Die Auth-Sektion definiert, welche Clients welchen Zugriff auf die Sockets haben. Die Zugriffssteuerung geschieht entweder nach IP-Adressen oder durch Angabe von Benutzername und Kennwort. Pro Benutzer muss eine "login"-Sektion eingetragen werden. Mittels "deny" können Hosts auch generell ausgeschlossen werden.

```
<auth>
        <login>
                <name>test</name>
                <password>test</password>
        </login>
        <login>
```

```
                <name>crusader</name>
                <password>pw</password>
        </login>

        <!-- Bisher nur IP Adressen. Keine Netze oder Bereiche ! -->
        <!-- Mehrfachnennungen sind aber moeglich, sofern sie Sinn machen -->
        <!-- Suchreihenfolge: allow, login, deny -->
        <ip action=allow>192.168.0.1</ip> <!-- Diese IPs muessen sich nicht einloggen -->
        <ip action=allow>127.0.0.1</ip> <!-- Diese IPs muessen sich nicht einloggen -->
        <ip action=login> any </ip> <!-- Diese IPs muessen sich einloggen -->
        <ip action=deny> any </ip> <!-- Diese IPs koennen sich nicht einloggen -->
</auth>
```

3.3.2 <tcpsocket>-Sektion

Hier wird definiert, auf welchem Netzwerk-Interface bzw. welcher IP-Adresse der monitord erreichbar sein soll ("bind"). Die Port-Angaben stellen den TCP-Port dar, der Parameter "mode" definiert das auszugebende Format. Wie bereits in der Feature-Liste erwähnt, kann der monitord als Funkauswerter an bestehende Programme wie den FMS Crusader oder FMS32/FMS32 PRO gehängt werden. Das monitord-Protokoll ist im Bereich "Ressourcen" zum Download verlinkt.

```
<tcpsocket>
        <bind> * </bind>
        <port mode="monitord"> 9333 </port>
        <port mode="fms32pro"> 9300 </port>
        <port mode="crusader"> 7778 </port>
</tcpsocket>
```

3.3.3 <soundcard>-Sektion

Für jede im PC vorhandene Soundkarte wird eine "soundcard"-Sektion angegeben. In ihr werden die unten stehenden Angaben gemacht und sie damit für den Betrieb der eigentlichen Auswerter vorbereitet; die Platzhalter "(... Auswerter-Module rechter/linker Kanal ...)" werden im Anschluss erläutert. Sowohl die Soundkarte als auch die Kanäle enthalten ein "Name"-Attribut.

```
<soundcard num=0>
  <device>/dev/dsp0</device> <!-- das /dev/dsp0 meint die erste Soundkarte  -->
  <status>1</status> <!-- 1=aktiv, 0=deaktivert -->
  <baud>22050</baud>
  <name> Erste Sondkarte </name>
  <channel part="left">
  <name>Kanal 0</name>
        (... Auswerter-Module linker Kanal ...)
  </channel>
  <channel part="right">
  <name>Kanal 1</name>
        (... Auswerter-Module rechter Kanal ...)
  </channel>
 </soundcard>
```

3.3.4 (Auswerter-)Module im CHANNEL-Block

Die Module verrichten die eigentliche Arbeit. Entsprechend wird ihnen hier jeweils eine einzelne Erklärung gewidmet. Jedes Modul kann, muss aber nicht, pro Kanal aktiviert werden.

3.3.4.1 ZVEI

```
<module type="zvei">
        <squelch> 25 </squelch>
        <debugmodus> 0 </debugmodus>
```

```
</module>
```

Durch diese Angabe im Platzhalter "(... Auswerter-Module rechter/linker Kanal ...)" der Soundcard-Sektion wird es für den jeweiligen Kanal aktiviert. Der Parameter 'Squelch' ist eine Art Rauschsperre (0-100), der 'Debugmodus' ist hilfreich bei Problemen - interessante numerische Angaben sind hier 3, 6 und 10, wobei 10 sehr viele Ausgaben auch aus den Tiefen des Auswerters bewirkt und keinesfalls für den Dauereinsatz zu empfehlen ist.

3.3.4.2 FMS

```
<module type="fms">
        <algorithmus> 1 </algorithmus>
        <syncbits> 12 </syncbits>
        <crc-check> 1 </crc-check>
        <maxerrors> 3 </maxerrors>
</module>
```

Diesem Modul können verschiedene Parameter mitgegeben werden, die die Auswertung beeinflussen. In der Regel sind "crc-check" und "maxerrors" entscheidend, um fehlertolerant auszuwerten. maxerrors beschreibt dabei die maximale Anzahl von Decodierfehlern innerhalb eines Telegramms, ab denen es verworfen werden soll, der crc-check (0/1) wird (de-)aktiviert, um eine Checksumme über das Telegramm zu berechnen. Das Feld Algorithmus kann 0 oder 1 enthalten; 0 ist ein alter Algorithmus, 1 ein neu implementierter mit im Test deutlich besseren Ergebnissen.

3.3.4.3 POCSAG

POCSAG ist in zwei Varianten implementiert: 512 und 1200 Baud. Hier exemplarisch die Konfiguration für die 1200-Baud-Variante:

```
<module type="poc1200">
        <algorithm> 1 </algorithm>
        <crc-check> 1 </crc-check>
        <ecc> 0 </ecc>
        <maxerrors> 3 </maxerrors>
</module>
```

Diesem Modul können verschiedene Parameter mitgegeben werden, die die Auswertung beeinflussen. Zur Beschreibung siehe die Angaben für den FMS-Auswerter.

4 Das mySQL-Plugin

4.1 Einleitung

Der monitord kann mit dem mySQL-Plugin (wie der Name schon sagt) zu einer mySQL-Datenbank connecten und empfangene Tonfolgen und Telegramme darin speichern. In den hier im Handbuch erklärten Beispielen wird davon ausgegangen, dass eine mySQL-Datenbank auf einem mySQL-Server auf demselben Host wie der monitord läuft (localhost), und dass ein Datenbank-Benutzer namens "monitord" mit dem Passwort "monitord" existiert.

4.2 Verfügbare Telegramm-Elemente

Die Auswerter geben gewisse Resultsets zurück. Diese sind sowohl im Konfigurations-Block zum mySQL-Plugin verwertbar (ihre Inhalte können in die Datenbank geschrieben werden), im Kapitel zu LUA-Filterskripten werden sie in Form eines Arrays auch noch einmal auftreten. Für alle bestehenden Auswerter ist hier angegeben, welche Felder zur Speicherung bzw. für Filter und ähnliches zur Verfügung stehen.

4.2.1 ZVEI-Fünftonfolgen

- timestamp (aktuelle Systemzeit des Auswerters)
- uhrzeit
- datum
- servernamehex (Name des Servers, HEX-codiert)
- channelnamehex (Name des Kanals, HEX-codiert)
- channelnum (Nummer des Kanals)
- typ (hier immer "zvei" - bei anderen "fms" oder "pocsag")
- zvei (die Ziffernfolge, Klartext ohne Wiederholtöne)
- weckton (Sirenensteuerung/Wecktöne)
- text (Klartext zum Weckton, z.B. "Unklare Auslösung", "Melderauslösung" etc.)

4.2.2 FMS-Telegramme

- timestamp (aktuelle Systemzeit des Auswerters)
- uhrzeit
- datum
- servernamehex (Name des Servers, HEX-codiert)
- channelnamehex (Name des Kanals, HEX-codiert)
- channelnum (Nummer des Kanals)
- typ ("fms")
- fmskennung (Fahrzeug- oder Gerätekennung)
- status (Status oder Typ-Identifier)
- baustufe
- richtung (0 = vom Fahrzeug, 1 = von der Leitstelle)
- tki
- bosdezimal (FMS-Organisationskenner)
- landdezimal (FMS-Landeskenner)
- statusdezimal (Status oder Typ-Identifier)
- bos (Organisationskenner)
- land (Landeskennung)
- ort (Landkreiskennung)
- kfz (Fahrzeugkennung)

- textuebertragung (FMS-Telegramminhalt, falls vorhanden)

4.2.3 POCSAG-Telegramme

- timestamp (aktuelle Systemzeit des Auswerters)
- uhrzeit
- datum
- servernamehex (Name des Servers, HEX-codiert)
- channelnamehex (Name des Kanals, HEX-codiert)
- channelnum (Nummer des Kanals)
- typ ("pocsag")
- subhex (Subadresse)
- sub (Subadresse)
- ric (Adresse)
- text (Telegramntext)

4.3 mySQL-Plugin-Konfiguration in der monitord.xml

Zunächst bilden wir hier den relevanten Bereich aus der monitord.xml ab, der das mySQL-Plugin konfiguriert:

```
<dataplugins>
 <!-- Daten Plugin -->
 <plugin name="mysql">
        <file> plugins/libmplugin_mysql-0.dll</file>
        <parameters>
                <logfile> mysql.log </logfile>
                <loglevel> DEBUG </loglevel>
                <hostname> localhost</hostname>
                <port> 3306 </port>
                <username> monitord </username>
                <password> monitord </password>
                <database> monitord </database>

                <!----
                Attribut name=XYZ": Zielfeld in der mySQL Tabelle
                Attribut source="mysql": Inhalt nicht aus dem Resultset nehmen,
                sondern "AS IS", also im Klartext als Parameter einfuegen (default="result")
                Wert des Tags: Feldname im Resultset oder Klartext(je nach Attribut source)
                Beispiele:
                <field name="Uhrzeit"           source="mysql"  > NOW() </field>
                        (laesst MySQL die aktuelle Uhr Zeit in das Feld "Uhrzeit" einfuegen)
                <field name="Typ"                       source="mysql"  > "T" </field>
                        (fuegt ein "T" in das Feld "Typ" ein)
                <field name="Meldung"> text </field>
                        (fuegt das Element "text" aus dem Resultset in das Feld "Meldung" ein)
                --->

                <mapping typ="fms">
                        <table> monitord_fms </table>
                        <field name="uhrzeit" source="mysql"> now() </field>
                        <field name="status"> statusdezimal </field>
                        <field name="kennung"> fmskennung </field>
                        <field name="richtung"> richtung </field>
                        <field name="text"> textuebertragung </field>
                        <field name="tki"> tki </field>
                        <field name="quelle"> channelnum </field>
                </mapping>

                <mapping typ="pocsag">
                        <table> monitord_pocsag </table>
                        <field name="uhrzeit" source="mysql"> now() </field>
                        <field name="kennung"> ric </field>
                        <field name="sub"> sub </field>
                        <field name="text"> text </field>
                        <field name="quelle"> channelnum </field>
                </mapping>

                <mapping typ="zvei">
                        <table> monitord_zvei </table>
                        <field name="uhrzeit" source="mysql"> now() </field>
                        <field name="typ"> weckton </field>
                        <field name="kennung"> zvei </field>
                        <field name="text"> text </field>
                        <field name="quelle"> channelnum </field>
                </mapping>
        </parameters>

 </plugin>
</dataplugins>
```

Das mySQL-Plugin kann einmal oder auch mehrmals (z.B. zur Verbindung zu mehreren Datenbanken) eingebunden werden. Wichtig ist dabei ein eindeutiger Identifier, der im "name"-Attribut des "plugin"-Tags notiert wird.

4.4 CREATE TABLE-Beispiele

Für die oben gezeigte Konfiguration des mySQL-Plugins müssen natürlich passende Tabellen bestehen. Entsprechende CREATE TABLE-Statements sind hier kurz zusammen gefasst. Die Datenfelder sind so definiert, dass es nicht zu Problemen kommen sollte - wir empfehlen, alle drei Tabellen anzulegen, auch wenn z.B. "vorerst" nur Fünftonfolgen abgelegt werden sollen: Fügt man später weitere Auswerter hinzu oder nutzt ein an diese Struktur angepasstes Web-Frontend, kommt es nicht zu Fehlern, die durch die Anlage und Speicherung einer leeren Tabelle vermeidbar wären.

```
DROP TABLE IF EXISTS `monitord_fms`;
CREATE TABLE  `monitord_fms` (
  `id` int(11) NOT NULL auto_increment,
  `uhrzeit` datetime NOT NULL,
  `status` smallint(2) unsigned default NULL,
  `kennung` varchar(9) collate latin1_german1_ci NOT NULL,
  `richtung` char(10) collate latin1_german1_ci NOT NULL,
  `text` varchar(255) collate latin1_german1_ci NOT NULL,
  `tki` char(1) collate latin1_german1_ci NOT NULL default '',
  `quelle` varchar(2) collate latin1_german1_ci NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=0 DEFAULT CHARSET=latin1 COLLATE=latin1_german1_ci;

DROP TABLE IF EXISTS `monitord_pocsag`;
CREATE TABLE  `monitord_pocsag` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `uhrzeit` datetime NOT NULL,
  `kennung` varchar(45) collate latin1_german1_ci NOT NULL,
  `sub` varchar(45) collate latin1_german1_ci NOT NULL,
  `text` varchar(500) collate latin1_german1_ci NOT NULL,
  `quelle` tinyint(2) unsigned NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=0 DEFAULT CHARSET=latin1 COLLATE=latin1_german1_ci;

DROP TABLE IF EXISTS `monitord_zvei`;
CREATE TABLE  `monitord_zvei` (
  `id` int(11) NOT NULL auto_increment,
  `uhrzeit` datetime NOT NULL,
  `kennung` varchar(5) collate latin1_german1_ci NOT NULL,
  `typ` char(1) collate latin1_german1_ci NOT NULL,
  `text` varchar(80) collate latin1_german1_ci NOT NULL,
  `quelle` varchar(2) collate latin1_german1_ci NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=0 DEFAULT CHARSET=latin1 COLLATE=latin1_german1_ci;
```

4.5 Automatisches Löschen

MySQL ist in der Version 5 in der Lage, auch so genannte Trigger zu verwenden. Die hier gezeigten Zeilen legen drei Trigger an, die bei Einfügeoperationen durch den monitord bei den bearbeiteten Tabellen dafür sorgen, dass alle Einträge, die älter als 14 Tage sind, gelöscht werden. Natürlich kann die Zeitspanne angepasst werden.

```
CREATE TRIGGER del_zvei AFTER INSERT ON monitord_zvei FOR EACH ROW
        DELETE FROM monitord_zvei WHERE uhrzeit < DATE_SUB(now(), INTERVAL 14 DAY);

CREATE TRIGGER del_fms AFTER INSERT ON monitord_fms FOR EACH ROW
        DELETE FROM monitord_fms WHERE uhrzeit < DATE_SUB(now(), INTERVAL 14 DAY);
```

```
CREATE TRIGGER del_pocsag AFTER INSERT ON monitord_pocsag FOR EACH ROW
        DELETE FROM monitord_pocsag WHERE uhrzeit < DATE_SUB(now(), INTERVAL 14 DAY);
```

Es sei erwähnt, dass hier der monitord keinerlei Arbeit leisten muss, das Löschen geschieht datenbankintern nach dem Einfügen eines neuen Datensatzes in die jeweilige Tabelle. Es werden auch nur in dieser alte Daten gelöscht. Falls der Bedarf anderer Löschoperationen besteht, hilft ein Blick in die mySQL-Dokumentation nicht nur zum Thema Trigger.

5 Erweiterungen

5.1 LUA-Skriptunterstützung

5.1.1 Einleitung

LUA ist eine Skriptsprache (siehe http://www.lua.org). Die Einbindung in den monitord ermöglicht die serverseitige (Vor-)Verarbeitung der empfangenen Daten, im simpelsten Fall eine Filterung. Vorerst wird nur diese an dieser Stelle erklärt - mit dem LUA-Konstrukt "os.execute()" lassen sich serverseitig auch Aktionen bzw. externe Programme ausführen.

Generell gilt: Das LUA-Skript muss 0 oder 1 zurück geben. 0 bedeutet, die Daten sollen an einen Client gesendet bzw. in einer Datenbank gespeichert oder allgemein das Plugin zur Speicherung oder Varbeitung der Daten angewiesen werden. mit "return 1" wird die Weiterverarbeitung oder Speicherung unterdrückt. Dies ist z.B. für Systemnachrichten beim FMS oder POCSAG interessant, auch kann so ein Probelarm zu festen Zeiten unterdrückt oder anders behandelt werden.

Die auszuführenden Skripte werden in der monitord.xml definiert (siehe "Konfiguration"):

```
<SocketFilterScript> socketfilter.lua </SocketFilterScript>
<PluginFilterScript> pluginfilter.lua </PluginFilterScript>
```

Für alle Sockets wird dabei das SocketFilterScript ausgeführt, für alle Plugins (z.B. mySQL-Dataplugin) das PluginFilterScript. Die angegebenen Skripte gelten jeweils für alle alle Clients an allen Sockets bzw. alle Plugins.

5.1.2 Filtern der Socket-Ausgaben

Direkt zum Einstieg in dieses Kapitel stellen wir ein Beispielskript für die Filterung der Ausgaben am Socket bereit:

```
-- SocketFilter.lua
--
-- Filter für den SocketServer - wird pro aktiven Client aufgerufen
--
-- Globales Array "arg" enthält die Daten vom Auswertermodul
--
-- Zusätzliche Werte im Array:
--
-- client_authenticated:        0/1  (1=angemeldet)
-- client_ip:                   IP-Adresse des Clients
-- client_loginname:            Anmeldename
-- client_type:                 fms32, crusader, monitord
--
-- Rückgabewert:
-- 0 = an Client senden,
-- 1= nicht an Client senden,
-- alle anderen = an Client senden
--

local toShowFMS = {"11111111", "22222222"} ;
local toShowPOCSAG = {"1111111", "2222222"} ;
local toShowZVEI = {"99000", "99001"} ;

function myfilterFMS32()
        for index,testwert in pairs(toShowFMS) do
                if (testwert==arg["ric"]) then
                        return 0
```

```
                end
        end
        return 1 ;
end

function myfilterMONITORD()
        -- ZVEI
        if(arg["typ"] == "zvei") then
                for index,testwert in pairs(toShowZVEI) do
                        if(testwert == arg["zvei"]) then
                                return 0
                        end
                end
        end

        -- FMS
        if(arg["typ"] == "fms") then
                for index,testwert in pairs(toShowFMS) do
                        if(testwert == arg["fmskennung"]) then
                                return 0
                        end
                end
        end

        -- POCSAG
        if(arg["typ"] == "pocsag") then
                for index,testwert in pairs(toShowPOCSAG) do
                        if(testwert == arg["ric"]) then
                                return 0
                        end
                end
        end

        -- default: anzeige unterdruecken
        return 1;
end

function filter()

        local dummyValue=1 ;

        -- DEBUG-Info: Alles ausgeben
        for index,testwert in pairs(arg) do
                print(index)
                print(testwert)
        end

        -- wird für jedes Telegramm (pocsag, fms, zvei) aufgerufen
        if (arg["client_type"]=="fms32") then
                return 0; -- delete this line to enable and uncomment the next one!
                -- return myfilterFMS32() ;
        end

        if (arg["client_type"]=="crusader") then
                -- nix
                dummyValue=2 ;
        end

        if (arg["client_type"]=="monitord") then
                return 0; -- delete this line to enable and uncomment the next one!
                -- return myfilterMONITORD() ;
        end

        -- default: alles anzeigen; ändern auf "1" um nichts anzuzeigen!
        return 0;
```

```
end
```

Wir hoffen, die Zeilen sind mehr oder weniger selbsterklärend - im Groben passiert folgendes:

- Definition der anzuzeigenden Meldungen
- Definition von Filterfunktionen (hier nur einer) für die einzelnen Sockets, die je nach Typ die entsprechende Filterliste durchgeht und returnt
- Definition der Funktion filter(), die vom monitord erwartet und initial aufgerufen wird (vgl. 'main()')

5.1.3 Filtern der Datenbank-INSERTs

Das Filtern der Datenbank-INSERTs ist prinzipiell analog zum oben genannten socketfilter. Einziger Unterschied ist, dass die zu implementierende Funktion "pluginFilter()" heißen muss. Die Konstrukte sind ansonsten 1:1 übernehmbar.

5.1.4 Ausführen von Programmen

Viele User des monitord und anderer Auswerter benötigen die Möglichkeit, automatisch SMS zu versenden oder auch andere Dinge zu tun. Mit LUA ist auch das möglich, hierfür gibt es den Befehl 'os.execute()'. Über diesen kann z.B. ein externes Programm zum SMS-Versand angestoßen, aber auch ein PHP-Skript ausgeführt oder sonst eine interessante Methode genutzt werden.

Ein Beispiel zur Syntax bei Ausführung eines PHP-Skripts:

```
if(arg["typ"] == "zvei" and string.sub(arg["zvei"], 0, 3) == "007" and arg["channelnum"] == "0"
        toexecute = "cmd /c c:\\xampp\\php\\php.exe ";
        toexecute = toexecute .. "c:\\programme\\monitord\\sendsms.php ";
        toexecute = toexecute .. arg["zvei"] ;
        os.execute(toexecute);
end
```

Hier wird PHP an der Kommandozeile aufgerufen, das Skript 'sendsms.php' im monitord-Verzeichnis ausgeführt, und dieses bekommt als Parameter die aktuell alarmierte Schleife angezeigt - der IF-Abfrage vorher gemäß aber nur, wenn diese mit '007' beginnt und vom linken Kanal der ersten Soundkarte (channelnum) ausgewertet wurde. Das entsprechende PHP-Skript kann dann den Zugriff auf eine SMS-Schnittstelle enthalten, aber z.B. auch per Jabber/XMPP oder E-Mail einen Alarm auslösen, da sind der Fantasie ja keine Grenzen gesetzt.

5.1.5 Beispiel zum SMS-Versand

Als Basis wurde hier das AlarmierungsTool von http://ffwug.jimdo.com/ eingesetzt. Die Entwickler des monitord stehen in keiner Verbindung zu diesem Tool, daher können wir auch keine Aussagen zu Sicherheit und Zuverlässigkeit desselben machen - Benutzung auf eigene Gefahr. Der Programmierer des Tools bietet eine gute Anleitung zur Konfiguration der Handynummern und SMS-Anbieter, so dass wir dahingehend nur darauf verweisen, die Anleitung befindet sich dort im Download-Bereich.

Skript zum Aufruf des konfigurierten AlarmierungsTools durch den monitord:

```
if(arg["typ"] == "zvei" and string.sub(arg["zvei"], 0, 3) == "001") then
        toexecute = "cmd /c c:\\Programme\AlarmierungsTool\\Alarm.exe ";
        toexecute = toexecute .. arg["uhrzeit"] ;
        toexecute = toexecute .. " " ;
        toexecute = toexecute .. arg["zvei"] ;
        os.execute(toexecute);
end
```

Mit diesem Aufruf werden dem AlarmierungsTool die Alarmzeit als erster Parameter und die alarmierte Schleife als zweiter Parameter übergeben. Diese können dann bei der automatisch generierten Alarmmeldung im AlarmierungsTool als &1 bzw. &2 in die SMS eingetragen werden. Für digitale Alarmierungen (POCSAG) etc. kann der gleiche Weg gewählt werden, lediglich die Parameter und das dazwischen eingefügte Leerzeichen (wichtig!) müssen dann entsprechend der verfügbaren Daten angepasst werden.

5.1.6 Allgemeine Infos zu Skripts

- Immer auf den Typ der bearbeiteten Ausgaben achten, sonst kommt es zu Fehlern im Programmablauf - der Telegrammtyp 'POCSAG' hat kein Feld 'zvei', in dem eine Alarmschleife erwartet wird, während ein ZVEI-Telegramm keinen Alarmtext enthält...
- Die in 'arg' enthaltenen Felder sind im Kapitel mySQL beschrieben bzw. aufgelistet.
- Die Programmaufrufe sind blockierend. Bevor das aufgerufene Programm beendet ist, wird also das LUA-Skript nicht weiter ausgeführt. Dies kann unter Windows durch Vorstellen des Wortes 'start', unter Linux/Unix-Systemen durch Nachstellen des '&'-Zeichens unterbunden werden.

6 Frontends

Der monitord beinhaltet an sich keine Frontends. Entsprechend sind hier alle Entwickler und Interessierten gefragt, sich mit der Programmierung von Frontends zu befassen und uns mitzuteilen. Wir sind gerne bereit, in unserem SVN Platz für (Erfolg versprechende) Frontends bereit zu stellen.

6.1 Bestehende Frontends (Socket monitord)

Wir haben noch keine Informationen zu "echten" Frontends gesammelt und in diese Seiten integriert und bitten das zu entschuldigen. Informationen zu Frontends kommen, sobald wir Screenshots und Co. vorliegen und wir ein wenig Zeit für die Beschreibung und Integration der Daten in diese Seiten haben.

Nicht umfassend sondern nur rudimentär getestet funktioniert der RadioOperator recht gut im Zusammenspiel. Der monitord ist außerdem Grundlage des mitgelieferten aber im Funktionsumfang beschränkten Auswerters/Decoders (decoder.exe) dieses Programms.

- http://www.radio-operator.de (RadioOperator)

6.2 Bestehende Frontends (Socket crusader/fms32)

Der monitord kann auch für den FMS Crusader bzw. FMS32/Pro als Auswerter genutzt werden.

Links:

- http://www.heirue-soft.de/fms.htm (FMS32/Pro)
- http://www.fmscrusader.de (Crusader)
- http://download.arne.de/ (AllFMS)

6.3 Bestehende Frontends (PHP/HTML)

Auch hier haben wir leider noch keine aktuelle Liste erstellt. Im SVN-Repository befinden sich jedoch auch Ansätze für mehr oder weniger umfangreiche Web-Frontends.

Ein simples und leicht erweiterbares Web-Frontend auf Basis von PHP zur Darstellung der Daten aus einer mySQL-Datenbank stellen wir auf mehrfache Anfrage hier zur Verfügung. Es läuft auf einem Webserver mit PHP-Unterstützung und wird über die mitgelieferten und bereits grundlegend konfigurierten .ini-Dateien eingerichetet.

- Web-Frontend.zip () - das Frontend im SVN: http://svn.monitord.de/Web-Frontend/trunk/

6.4 BOSiX

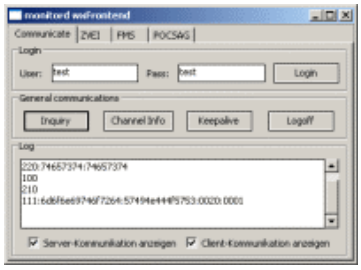
Eine modifizierte Knoppix-Variante, klein und als Boot-CD gehalten, mit dem monitord als Auswerter und einem minimalen aber übersichtlichen Frontend:
http://www.myandi.de/index.php/category/bosix/

6.5 Debug-Frontend

Gerade bei Tests ist es doch sehr angenehm, Klartext statt Hex-Ausgaben lesen zu können. Entsprechend stellen wir bei den Downloads ein minimalistisches Frontend bereit, das die Alarm-Ausgaben des monitord etwas formatiert und als Klartext darstellt. Auch sind bereits einige

Steuersequenzen implementiert, deren Kommunikationsdaten jedoch in Rohform dargestellt werden (in Hex).

- Debug-Frontend (nur das Frontend-Binary, win32), ca. 200kB [17.01.2008]
- Debug-Frontend (nur die benötigten Libraries/DLLs, win32), ca. 2.5MB [17.01.2008]
- Debug-Frontend (Sourcecode), ca. 6kB [17.01.2008]



7 Informationsquellen

7.1 Forum

http://www.funkmeldesystem.de/foren/forumdisplay.php?f=20

Speziell für den Einstieg und mit vielen Informationen zum Compilerlauf ist dieser Beitrag interessant: http://www.funkmeldesystem.de/foren/showthread.php?t=35783

7.2 Handbuch

Ein PDF-Handbuch mit den Informationen dieser Webseite ist herunterladbar unter http://www.monitord.de/download/monitord-handbuch.pdf

7.3 Bugtracker (Flyspray)

http://bts.monitord.de

7.4 Datenfluss-Überblick für Entwickler

monitord-datenfluss.pdf

7.5 Monitor-Wiki

http://monitor.08k.de

7.6 SVN-Zugang zu den Sourcen

http://svn.monitord.de/monitor

7.7 BOSiX

Eine modifizierte Knoppix-Variante, klein und als Boot-CD gehalten, mit dem monitord als Auswerter: http://www.myandi.de/index.php/category/bosix/

7.8 Protokolldefinition auf dem monitord-Socket

Vorrangig für Entwickler eigener Frontends interessant wird hier aufgeschlüsselt, welche Möglichkeiten zur Kommunikation mit dem monitord existieren und wie sie abzuwickeln sind. Ebenfalls erklärt ist, was die Ausgaben des monitord zu bedeuten haben. Für Anwender sind die "verfügbaren Telegramm-Elemente" im Kapitel "Das mySQL-Plugin" sicherlich ersteinmal hilfreicher.

- monitor-protokoll-03.pdf

7.9 Die Vorgänger und Väter des Gedanken

http://www.monitor.mgrohmann.de (ehemalige Heimat des monitor 1.8.1 - nicht mehr erreichbar)
http://www.baycom.org/~tom/ham/linux/multimon.html (mulitmon)

7.10 Andere Auswerte-Software (inklusive Frontend)

http://www.heirue-soft.de (FMS32/FMS32Pro)
http://www.fmscrusader.de (FMS Crusader)