

TiXML-Tutorial

Für Firmware bis 1.72.14.0

© 2004 Tixi.Com GmbH, Berlin

Redaktionsschluss: August 2005. Unterstützte Firmware Version 1.72.14.0

Dieses Handbuch ist durch Copyright geschützt. Jede weitere Veräusserung ist nur mit der Zustimmung des Herausgebers gestattet. Dies gilt auch für Kopien, Mikrofilme, Übersetzungen sowie die Speicherung und Verarbeitung in elektronischen Systemen.

In diesem Handbuch verwendete Firmen- und Markennamen sind eigenständige Markenzeichen der betreffenden Firmen, auch wenn sie nicht explizit als solche gekennzeichnet sind.

Inhalt

1. EINFÜHRUNG.....	3
1.1. SYNTAXERKLÄRUNG XML - TiXML	3
1.1.1. Was ist XML ?.....	3
1.1.2. Unterschied zu TiXML ?	4
1.2. ÜBERBLICK ZUSAMMENHÄNGE.....	5
1.3. VORGEHENSWEISE BEI DER PROJEKTERSTELLUNG	7
1.4. INFO ZU DEN BEGLEITENDEN PROJEKTBEISPIELEN	8
1.5. PROJEKTAUSTAUSCH ZWISCHEN TILA UND TICO	8
2. DIE TiXML-DATENBANKEN	9
3. DIE BASISKONFIGURATION	11
3.1. LOCATION (TiXML REFERENCE KAPITEL 3.3).....	11
3.2. USERDATA (TiXML REFERENCE KAPITEL 3.2)	11
3.3. ADDRESSBOOK (TiXML REFERENCE KAPITEL 3.4).....	12
3.4. ISP (TiXML-REFERENCE KAPITEL 3.5).....	12
3.5. LOGFILES (TiXML REFERENCE KAPITEL 4).....	13
3.6. ZUGRIFFSSCHUTZ (TiXML-REFERENCE KAPITEL 3.11).....	14
4. KONFIGURATION VON ALARMFUNKTIONEN	14
4.1. EINFACHE ALARMFUNKTION.....	14
4.1.1. Event Handler (TiXML Reference Kapitel 3.8).....	14
4.1.2. MessageJobTemplate (TiXML Reference Kapitel 3.7)	15
4.1.3. MessageText (TiXML Reference Kapitel 3.6).....	15
4.1.4. Testen des Alarms (TiXML Reference Kapitel 2.4.9.1).....	15
4.2. ERWEITERTE ALARMFUNKTIONEN	16
4.2.1. Erweitern der Meldetexte.....	16
4.2.1.1. Signaturen (TiXML Reference Kapitel 3.6)	16
4.2.1.2. Text mit Variablen (TiXML Reference Kapitel 12)	17
4.2.2. Multiple Sendmail (TiXML Reference Kapitel 3.8).....	17
4.2.3. Port löst Alarm aus	17
4.2.3.1. Der Service-Button (TiXML Reference Kapitel 6.7)	17
4.2.3.2. I/O-Ports (TiXML Reference Kapitel 6.3).....	18
4.2.3.3. ProcessVars (TiXML Reference Kapitel 6.2).....	19
4.2.4. Scheduler (TiXML Reference Kapitel 7).....	20
4.2.4.1. Schichtplan	20
4.2.4.2. Scheduler Conditions (Ferienprogramm)	22
4.2.5. erweiterte EventHandler Funktionen (TiXML Reference Kap. 3.8.1).....	22
4.2.5.1. Alarmprioritäten	22
4.2.5.2. Alarm setzt Ausgang - Set	23
4.2.5.3. Alarmkaskade „OnError“	23
4.2.5.4. Bestätigung über Port – „OnOK“	23
4.2.5.5. Quittieren von Nachrichten	24
4.2.5.6. Übergabe von Parametern	25
4.2.5.7. Loggen von Daten (TiXML Reference Handbuch Kapitel 4).....	26
4.2.5.8. Internetzeitsynchronisierung / Sommer-Winter-Zeitungstellung (TiXML-Reference Kap. 3.12)	29
4.3. FERNWIRKEN DURCH EINGEHENDE NACHRICHTEN (H. KAP. 8).....	31
4.3.1. Zugriffsschutz	31
4.3.2. Einfaches Fernwirken:.....	31
4.3.3. Fernwirken mit Parameterübergabe.....	32
4.3.4. Überwachung der Fernwirkfunktionen	32
4.3.4.1. Statische Rückantwort	32
4.3.4.2. Dynamische Rückantwort.....	33
4.3.4.3. Rückantwort mit Wertüberprüfung.....	34
4.3.5. Verhalten im Fehlerfall.....	34
4.3.5.1. ungültiges Passwort	35
4.3.5.2. ungültiges Event	35
4.3.5.3. Informationen bei OK-Fall loggen	36
4.3.6. E-Mails abholen.....	36
4.4. EVENT DURCH ANRUF (CALLERID) H. KAP. 8.2	36
5. ANBINDUNG AN EINE SPS (TiXML PLC HANDBUCH).....	37
6. WEITERE FUNKTIONEN.....	37

1. Einführung

1.1. Syntaxerklärung XML - TiXML

1.1.1. Was ist XML ?

Im Folgenden soll ein lediglich kurzer Einblick in XML-Strukturen zur Darstellung von Daten gegeben werden. (Weitergehende Informationen zu Stylesheets u.a. findet man bei www.XML.org oder als kurzer Überblick bei Hubert Partl¹.)

Start- und End-Tags für Blöcke von Daten

„Die meisten Befehle (Tags) in XML-Anwendungen - wie auch in HTML - treten paarweise als Start- und End-Tags auf und geben an, welche Bedeutung der dazwischen liegende (eventuell durch weitere Tags unterteilte) Text hat.“¹

Bei den Tag-Namen ist die Groß- und Kleinschreibung zu beachten. Parameter zu den Tags werden als Attribute bezeichnet. Die Werte der Attribute werden in Anführungszeichen (") oder Apostroph (' = ASCII-Zeichen: Single Quote) eingeschlossen (z.B. Autor = "Meyer").

```
<Buch> ... Text ... </Buch>
```

oder

```
<Buch Autor="Meyer" ... > ... Text ... </Buch>
```

Ein kleines Adress-Datenbank-Beispiel soll das XML-Prinzip aufzeigen:

```
<person id="p4681">
  <vorname>Hubert</vorname>
  <zuname>Partl</zuname>
  <titel>Dr.</titel>
  <strasse>Manngasse 118</strasse>
  <plz>A-1220</plz>
  <ort>Wien</ort>
  <geburtstag>
    <tag>28</tag>
    <monat>März</monat>
    <jahr>1949</jahr>
  </geburtstag>
</person>
```

Da Tags geschachtelt werden können, lassen sich auch komplizierte Datenstrukturen übersichtlich darstellen.

Einzelne Elemente ohne End-Tag

Einen Spezialfall stellt ein Element ohne End-Tag dar. Es hat folgende Form:

```
<Buch />
```

oder

```
<Buch Autor="Meyer" ... />
```

Hier braucht der Tag-Name am Ende nicht wiederholt zu werden, was aus Platzgründen für kleine CPUs mit wenig Speicher ein wesentlicher Vorteil ist.

¹ Hans Hubert Partl, Wien, siehe auch <http://www.boku.ac.at/html/inf/xmlkurz.html>

1.1.2. Unterschied zu TiXML ?

Tixi.Com hat für die Konfiguration der Alarm-Modems einen XML-Trick eingeführt und damit zwei Fliegen mit einer Klappe geschlagen: das Datenvolumen verringerte sich um 30 – 40 % und die Lesbarkeit der XML-Datei verbesserte sich erheblich, denn es stehen keine doppelten Informationen mehr in einer einzelnen Zeile.

Normale XML-Darstellung:

```
<vorname>Hubert</vorname>
```

TiXML-Darstellung:

```
<vorname _= "Hubert" />
```

Das in XML zugelassene Zeichen „_“ wird hier als Dummy-Attribut für den Tag benutzt, dem als Variable ein Wert zugewiesen werden soll.

Das Beispiel von oben sieht also in TiXML so aus:

```
<person id="p4681" >
  <vorname _="Hubert" />
  <zuname _="Partl" />
  <Titel _="Dr. " />
  <strasse _="Manngasse 118" />
  <plz _="A-1220" />
  <ort _="Wien" />
  <geburtstag>
    <tag _="28" />
    <monat _="März" />
    <jahr _="1949" />
  </geburtstag>
</person>
```

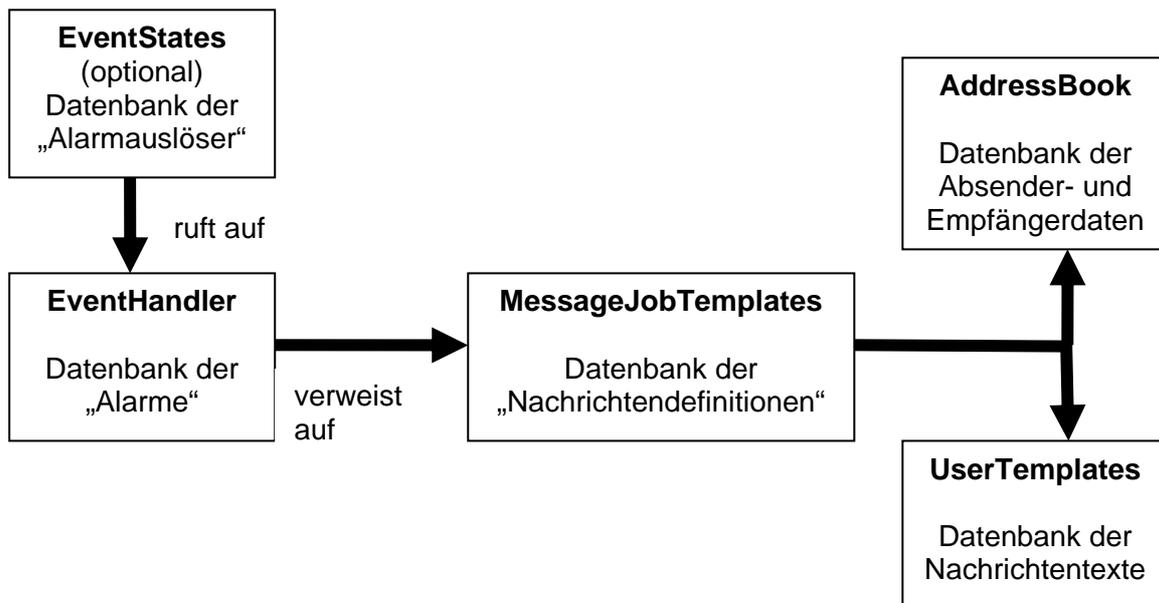
TiXML-Frames, d.h. TiXML-Datensätze oder Befehle, müssen beim Übertragen an das Alarm Modem in eckige Klammern gefasst werden. Dadurch erkennt das Modem, das es sich um Daten für den TiXML-Processor handelt, z.B.:

```
[ <DoOn _="Feuer" /> ]
```

1.2. Überblick Zusammenhänge

Um den Einstieg in die TiXML-Parametrierung zu erleichtern, werden in diesem Kapitel anhand einer Störmeldung die Zusammenhänge der einzelnen TiXML-Datenbanken erklärt.

Ein im Tixi Alarm Modem hinterlegter einfacher Alarm besteht im Wesentlichen aus Einträgen in fünf Datenbanken, die miteinander verkettet sind.



Der Zusammenhang der Daten lässt sich am einfachsten mit einem durch den Tixi Alarm Editor (TILA) erstellten Alarm darstellen.

In TILA wird dafür der Alarmwizard ausgeführt, der den Empfänger und den Nachrichtentext mit dem Alarm verknüpft.

In diesem Beispiel werden die Zusammenhänge eines Alarms „Feuer“, das als Faxmeldung an den Empfänger „Feuerwehr“ verschickt werden soll, erklärt.

An erster Stelle der Prozessabarbeitung steht der „**EventHandler**“:

```
[<SetConfig _="EVENTS" ver="y">
  <EventHandler>
    ...
    <Feuer>
      <SendMail _="MessageJobTemplates/Feuer" value="0"/>
    </Feuer>
    ...
  </EventHandler>
</SetConfig>]
```

Dieser verweist mit der Anweisung „SendMail“ auf die Datenbank mit den **MessageJobTemplates**:

```
[<SetConfig _="TEMPLATE" ver="y">
  <MessageJobTemplates>
    ...
    <Feuer _="TextFax">
      <Recipient _="/D/AddressBook/Feuerwehr"/>
      <Sender _="/D/AddressBook/MySelf"/>
      <Body _="/D/UserTemplates/Feuer"/>
      <Subject _="&#xae;/D/UserTemplates/S-Feuer/Subject;"/>
    </Feuer>
    ...
  </MessageJobTemplates>
</SetConfig>]
```

Das MessageJobTemplate verweist wiederum auf das **Adressbuch** (AddressBook) mit dem Absender „MySelf“ und Empfänger „Feuerwehr“, sowie auf den zu verschickenden **Nachrichtentext** (UserTemplate) „Feuer“:

```
[<SetConfig _="TEMPLATE" ver="y">
  <AddressBook>
    ...
    <MySelf hidden="1">
      <Fax _="+49-30-24037497"/>
      <SMS_No _="03024037497"/>
    </MySelf>
    <Feuerwehr>
      <Fax _="+49-30-10"/>
      <SMS_No _="017624434569"/>
      <SMS_Provider _="AnnyWay"/>
    </Feuerwehr>
    ...
  </AddressBook>
</SetConfig>]
```

```
[<SetConfig _="TEMPLATE" ver="y">
  <UserTemplates>
    <Feuer>
      <L _="Die Scheune brennt !"/>
    </Feuer>
    <S-Feuer>
      <Subject _="Feueralarm !"/>
    </S-Feuer>
  </UserTemplates>
</SetConfig>]
```

Um den Alarm auszulösen, müsste das Alarm Modem über die serielle Schnittstelle den Befehl [<DoOn _="Feuer"/>] erhalten. Der Parameter „Feuer“ ist der Name des auszulösenden EventHandlers.

Man kann den Alarm allerdings auch aufgrund der Änderung einer Variable auslösen lassen, z.B. über einen digitalen Eingang „P0“. Der Alarmwizard in TILA führt dies optional im letzten Schritt aus. Dazu wird eine weitere Datenbank „**EventStates**“ herangezogen:

```
[<SetConfig _="PROCCFG" ver="y">
  <EventStates>
    <Feuer_STATE>
      <Enabled _="TRUE"/>
      <ProcessVar _="/Process/C0/I/P0" flank="high"/>
      <Event _="Feuer"/>
    </Feuer_STATE>
  </EventStates>
</SetConfig>]
```

Der EventState ruft über den Eintrag <Event _="Feuer"/> den am Anfang konfigurierten EventHandler „Feuer“ auf. Das EventState steht also an erster Stelle der Prozessabarbeitung.

Ähnliche Zusammenhänge gibt es auch für alle weiteren Aktionen, die das Alarm Modem verarbeiten kann. Im EventHandler werden dazu anstelle des „SendMail“-Befehls, der für das Versenden von Nachrichten steht, die Befehle für Datenlogging („Log“, „BinLog“), Schalten („Set“), Alarmkaskaden („OnError“, „OnOK“) usw. eingefügt.

Die Verwendung der einzelnen Befehle wird in diesem TiXML-Tutorial erläutert. Im Handbuch finden Sie Verweise auf Kapitel im TiXML-Reference Manual. Dort können Sie detailliert alle Parameter und Einstellungsmöglichkeiten für die einzelnen TiXML-Funktionen nachschlagen.

1.3. Vorgehensweise bei der Projekterstellung

Bevor Sie sich tiefer mit dem Tutorial beschäftigen, sollten Sie zunächst einmal klären, ob die Version des Handbuchs sowie der TICO-TiXML-Konsole zu der auf dem Alarm Modem verwendeten Software (Firmware) passt, da es historisch bedingt zwei unterschiedliche Dokumentationen gibt.

Die Alarm Modem Firmware ab Version 2.0.0.0 ist auf die Software TILA2 und die „TICO für FW 2.0“ (siehe Start-Menü-Name) ausgerichtet, und wird in diesem Handbuch nicht behandelt. Verwenden Sie dafür das Handbuch mit Code TUT2-DE

Die Alarm Modem Firmware bis Version 1.72.14 ist auf die Software TILA1 und eine ältere TICO (ohne Zusatz „für FW 2.0“) ausgerichtet und ist Thema dieses Handbuchs.

Sollten Sie eine Zwischenversion (1.8x) verwenden, so raten wir zu einem Update auf FW 2.0.

Vor dem eigentlichen Erstellen von TiXML-Projekten sollte zunächst ein Basisprojekt mit Hilfe von TILA erstellt werden. Dieses Basisprojekt sollte bereits alle Modem- und Geräteeinstellungen, die Absender- und Empfängerdaten, Nachrichtentexte, einfache Alarmer sowie ggf. SPS-Variablen enthalten.

Dieses Basisprojekt kann dann mit der TiXML-Konsole importiert und weiterverarbeitet werden. Dadurch erspart man sich den größten Teil der TiXML-Parametrierung, und braucht das Projekt nur noch um wenige Zeilen erweitern.

Der Vollständigkeit halber wird im Kapitel 2 der projektrelevante TiXML-Inhalt des Basisprojektes erläutert. Da diese Werte aber komfortabler über TILA zu parametrieren sind, kann dieses Kapitel auch übersprungen werden.

1.4. Info zu den begleitenden Projektbeispielen

Dem TiXML-Tutorial liegen TICO-Projektbeispiele zur praktischen Veranschaulichung der TiXML-Funktionen bei.

Das in TILA erstellte „Basis“-Projekt dient dabei als allgemeine Vorlage.

Das Projekt „`Referenceproject.cnf`“ beinhaltet eine Sammlung aller derzeit verfügbaren TiXML-Funktionen. Es kann somit als Nachschlagewerk und Quelle für Funktionsbausteine dienen, die sie mittels Kopieren&Einfügen in Ihr aktuelles Projekt übernehmen können.

Zusätzlich gibt es zu jeder Funktion ein eigenständiges Beispielprojekt, welches Sie nach Ihren Bedürfnissen anpassen können.

1.5. Projektaustausch zwischen TILA und TICO

TILA und TICO sind nur in eine Richtung miteinander kompatibel. Ein mit TICO erweitertes Projekt sollte nachfolgend nicht mehr mit TILA verändert werden. TILA bietet zwar im Menü „Datei“ die Möglichkeit an, TICO Projekte zu importieren, verwirft aber beim Öffnen alle komplexen Ergänzungen wie z.B. Alarmkaskaden.

Um ein mit TILA erstelltes Projekt in TICO darzustellen, muss der Umweg über das Alarm Modem gewählt werden. Dazu überträgt man das Projekt mittels TILA in das Modem (Menüpunkt: `Sende Konfiguration an Gerät`), und läßt es anschließend in TICO wieder ein (Menüpunkt: `Projekt – Open from TiMo`).

TILA fügt an einigen Stellen zusätzliche XML-Parameter zur Organisation der Daten ein, z.B. `value="0`, die z.T. in den TiXML-Beispielprojekten zu sehen sind. Da diese im Alarm Modem selbst keine Funktion haben, werden sie im Tutorial weggelassen.

2. Die TiXML-Datenbanken

Die Software TICO stellt bei einem geöffneten Projekt jede TiXML-Datenbank auf einem eigenen Register dar. Derzeit sind dies:

- 1 - Location
- 2 - UserData
- 3 - AddressBook
- 4 - ISP
- 5 - UserLogin
- 6 - SMS Login
- 7 - Lofile Records
- 8 - Logfiles
- 9 - EventLoggin
- 10 - EventHandlerer
- 11 - MessageJobTemplates
- 12 - MessageText
- 13 - ProcessVars
- 14 - EventStates
- 15 - Schedule Conditions
- 16 - Schedule
- 17 - External
- 18 - SMS-Provider
- 19 - Incoming SMS-Provider
- 20 - Incoming CallerID Alarms

1 – Location

Hier stehen die Standortrelevanten Daten wie z.B. die Land, Telefonnummer und Amtsholung

2 – UserData

Hier sind die Einstellungen zur Modemkommunikation hinterlegt, z.B. Wahlwiederholungen, ISDN MSN, Faxkennung und Rufannahme.

3 – AddressBook

Die Kontaktadressen des Absenders und der Empfänger, z.B. E-Mail Adresse, Fax- oder SMS-Nummern.

4 – ISP

Die Daten des Internetzugangs, z.B. Rufnummer, Anmeldung und E-Mail-Konto sowie der Internetzeitsynchronisierung.

5 – UserLogin

Hier kann der Zugriffschutz festgelegt werden, durch welchen das Gerät lokal und bei Ferneinwahl geschützt ist.

6 – SMS Login

Hier ist das Passwort für das Fernwirken via SMS, E-Mail und Express-E-Mail gespeichert.

7 – Logfile Records

Die Struktur der Datensätze beim Binärlogging wird hier angegeben.

8 – Logfiles

Die Namen und Größen der verwendeten Logfiles für System- und Datenlogging.

9 – EventLogging

Hier wird die Art und der Umfang des Systemprotokolls definiert, z.B. Loggen von Fehlern bei der Alarmbearbeitung.

10 – EventHandler

Die Namen der Aktionen bzw. Alarme und die damit verknüpften Befehle, z.B. Nachricht senden oder Daten loggen.

11 – MessageJobTemplates

Hier werden Nachrichtentexte und Adreßbuchkontakte zu Alarmmeldungen verknüpft sowie deren Typ, z.B. SMS oder TextFax festgelegt.

12 – MessageText

Hier stehen die Nachrichtentexte und deren Betreffzeilen.

13 – ProcessVars

Hier können Variablen definiert und logische Operationen ausgeführt werden.

14 – EventStates

Die Bedingungen für eine Aktion werden hier beschrieben, z.B. bei welcher Variablenänderung ein bestimmter Alarm ausgelöst werden soll.

15 – Schedule Conditions

Hier können weitere Bedingungen für den Kalender angelegt werden, z.B. Ferienzeiten in denen eine Zeitschaltfunktion nicht ausgeführt werden soll.

16 – Schedule

Der Kalender des Alarm Modems wird hier parametrisiert, d.h. es können Zeitpunkte für Aktionen (z.B. Logging) angegeben werden (Zeitschaltuhr).

17 – External

Bei Verwendung einer SPS werden hier die Tabellen ihrer Variablen sowie die Einstellungen zur Kommunikation mit der SPS vorgenommen.

18 – SMS-Provider

Die Gateways für SMS-Versand werden hier mit den Einwahlrufnummern und Protokollen hinterlegt.

19 – IncomingSMSProviders

Die Rufnummern (Service Center) für den Empfang von Festnetz-SMS werden hier angegeben.

20 – Incoming CallerID Alarms

Hier können Rufnummern bestimmte Aktionen zugewiesen werden. Bei einem Anruf dieser Nummer kann z.B. eine Garage geöffnet werden.

3. Die Basiskonfiguration

Die Basiskonfiguration (`basis.cnf`) stellt die grundlegenden Daten wie z.B. Standortparameter, Absender- Empfängeradressen oder den Internetzugang zusammen. Diese Daten können alle über TILA parametrisiert werden.

Die nachfolgenden Parameter sind projektspezifisch und sollten Ihre persönlichen Gerätedaten enthalten.

3.1. Location (TiXML Reference Kapitel 3.3)

In der Location (Register 1-Location) muß die Rufnummer des Tixi Alarm Modem geändert werden. Es ist die Nummer des Anschlusses einzutragen, an dem das Modem angeschlossen ist (ohne Landes und Ortskennzahl):

```
<PhoneNumber _="12345678" />
```

Sollte die Ortskennzahl nicht die "30" (Berlin) sein, so ist diese hier ebenfalls zu ändern (immer ohne führende Null, welche in Dt. die nationale Vorwahl darstellt):

```
<AreaCode _="30" />
```

Wenn das Tixi Alarm Modem über eine Telefonanlage angeschlossen ist, kann zudem eine evtl. benötigte Amtsholung definiert werden. Bei einer "0" wäre dies z.B.:

```
<LocalDialPrefix _="0" />
```

```
<LongDialPrefix _="0" />
```

3.2. UserData (TiXML Reference Kapitel 3.2)

Im Register "2-UserData" muss der Name und die Kennung des Tixi Alarm Modem sowie ggf. die ISDN-Mehrgeräterufnummer eingetragen werden:

```
<BoxName _="Tixi Alarm Modem" />
```

wird z.B. bei Faxen in der Kopfzeile als verwendet.

```
<BoxNumber _="+49-30-1234567" />
```

wird in der Kopfzeile von Faxen als Kennung verwendet.

```
<IsdnDataChannelID _="*" />
```

Legt die MSN vom Tixi Alarm Modem am ISDN-Anschluß fest. Diese ist wichtig, wenn zum Tixi Alarm Modem auch eine Remote-Verbindung hergestellt werden soll. Ein "*" bedeutet "Rufe für jede Nummer annehmen".

Der „Ring Counter“ regelt die Anzahl der Klingelzeichen bis zur Rufannahme. Bei „0“ ist die Rufannahme abgeschaltet. Ein anderer Wert entspricht der Anzahl der abzuwartenden Klingelzeichen, z.B. Annahme nach einem Klingelzeichen:

```
<RingCounter _="1" />
```

Bei Verwendung eines Tixi Alarm Modem mit GSM-Erweiterung kann das GSM-Modul über die folgenden Einträge aktiviert werden:

Schnittstelle, an der das GSM-Modul vorhanden ist (Standard: C0):

```
<GSMModem _="C0" />
```

PIN der SIM-Karte, falls nötig:

```
<Pin1 _="1234" />
```

Achtung: Diese Einstellungen werden erst nach einem Neustart des Modems wirksam!

3.3. AddressBook (TiXML Reference Kapitel 3.4)

Im Adressbuch (Register 3-AddressBook) werden die eigenen Absenderdaten <MySelf> sowie die Empfänger <Receiver> definiert.

Für jeden verwendeten Nachrichtentyp (außer CityRuf) ist zwingend eine entsprechende Absenderadresse bzw. -nummer im Kontakt „MySelf“ nötig.

Die E-Mail Adresse, z.B.:

```
<Email _="TAM@freenet.de" />
```

Die SMS-Nummer , z.B.:

```
<SMS_No _="01721234567" />
```

und der dazugehörige SMS-Provider (nur beim Empfänger), z.B.:

```
<SMS_Provider _="Telekom" />
```

Mögliche Werte: D1 , D1_ISDN , D2 , D2_ISDN , Eplus , Telekom , AnnyWay

Die Faxnummer, z.B.:

```
<Fax _="+49-30-1234567" />
```

Die Express-Mail Adresse, z.B.:

```
<Express-Email _="INFO+49-30-1234567" />
```

Die CityRuf Nummer (nur beim Empfänger),z.B.:

```
<CityRuf _="3949000" />
```

Und der dazugehörige Pager-Provider:

```
<Pager_Provider _="CityRuf" />
```

Es können beliebig viele Empfänger mit beliebigen Namen hinzugefügt werden. Jeder Empfänger kann mehrere Adresstypen enthalten, z.B.:

```
<Receiver>
  <INet _="info@tixi.com" />
  <Fax _="+49-30-40608400" />
  <Express-Email _="INFO+49-30-40608555" />
  <SMS_No _="03040608300" />
  <SMS_Provider _="Telekom" />
  <CityRuf _="3949000" />
  <Pager_Provider _="CityRuf" />
</Receiver>
```

Der in den Beispielen vorhandene Kontakt „SMSAnswer“ wird später für automatische Rückantworten beim SMS-Fernwirken verwendet.

3.4. ISP (TiXML-Reference Kapitel 3.5)

Der Providerabschnitt (Register 4-ISP) ist für einen Internet-by-Call Testaccount von Tixi.Com vordefiniert und sollte nach Möglichkeit durch einen eigenen Internetzugang ersetzt werden.

Wichtig sind:

Die Benutzerdaten für die Einwahl:

```
<PPPUUserName _="Freenet" />
```

```
<PPPPassword _="Freenet" />
```

Der Server für ausgehende (Abschnitt SMTP) bzw. eingehende (Abschnitt POP3) Mails:

```
<mailserver_ip _="mx.freenet.de"/>
<mailserver_name _="mx.freenet.de"/>
```

Die POP3-Benutzerdaten des Mailservers:

```
<Username _="TIM-Test"/>
<Password _="TIM-Test"/>
```

Bei Bedarf kann das Verfahren POP-vor-SMTP im Abschnitt „SMTP“ aktiviert werden:

```
<Flags _="POPBeforeSMTP"/>
```

Die Einwahlnummer:

```
<RemotePhoneNumber _="+49-1019-01929"/>
```

Das Löschen der abgeholten Mails auf dem Mailserver kann deaktiviert werden:

```
<Flags _="DontDelete"/>
```

3.5. Logfiles (TiXML Reference Kapitel 4)

Das Tixi Alarm Modem bietet von sich aus 5 Logfiles an die der Überwachung von Prozessen dienen. Diese müssen jedoch erst im Dateisystem angelegt, sowie die Protokollierung aktiviert werden.

Angelegt werden die Logfiles im Register 8-Logfiles (size=Größe in Byte):

```
<LogFiles>
  <JobReport size="10240"/>
  <Event size="10240"/>
  <Login size="10240"/>
  <IncomingMessage size="10240"/>
  <FailedIncomingCall size="10240"/>
</LogFiles>
```

Im Register 9-EventLogging wird auch konfiguriert, welche Ergebnisse der Prozesse protokolliert werden (mode: erste Stelle: e=errors, o=OKs, a=all, zweite Stelle optional: v=verbous):

```
<EventLogging>
  <JobReport mode="av" file="Job"/>
  <Event mode="av" file="Event"/>
  <Login mode="av" file="Login"/>
  <IncomingMessage mode="av" file="Incoming"/>
  <FailedIncomingCall mode="av" file="failed"/>
</EventLogging>
```

Das Logfile "JobReport" enthält Informationen über die abgearbeiteten Prozesse.

Das Logfile "Event" listet alle ausgeführten Events auf.

Das Logfile "Login" protokolliert Anmeldungen (lokal und remote) am Gerät.

Das Logfile "IncomingMessage" enthält Informationen eingegangene Nachrichten (Fernwirken).

Das Logfile "FailedIncomingCall" enthält Informationen über Fehler beim Fernwirken.

3.6. Zugriffsschutz (TiXML-Reference Kapitel 3.11)

Die Konfiguration sowie die Fernkonfiguration lassen sich durch Benutzerdaten schützen. Diese müssen bei der Einwahl oder vor jeder Konfiguration übermittelt werden. Die Benutzerdaten können im Register 5-UserLogin eingefügt werden:

Der Schutz erfolgt durch einen Benutzernamen und ein Passwort, z.B.:

```
<Login>
  <Techniker _="Sonne" />
</Login>
```

Der Login Befehl sieht dann wie folgt aus:

```
[<Login _="PAP" user="Techniker" password="Sonne" ver="y"/>]
```

4. Konfiguration von Alarmfunktionen

Das Tixi Alarm Modem kann auf verschiedene Art zum Verschicken eines Alarms aufgefordert werden.

- per TiXML-Befehl an der RS232
- per SPS-Variable an der RS232 /422 /485
- per digitalen Eingang
- per eingehender Nachricht

Die folgenden Kapitel bauen diese unterschiedlichen Alarmmöglichkeiten Schritt für Schritt auf.

4.1. Einfache Alarmfunktion

Zunächst soll eine einzelne Nachricht verschickt werden. Da diese Funktion auch in TILA vorhanden ist, kann dies dort sicher einfacher parametrisiert werden. Aus Gründen der Vollständigkeit wird hier trotzdem die Konfiguration in TiXML erklärt. Im Beispielprojekt `basis-messages.cnf` ist für jeden Nachrichtentyp ein eigener Alarm parametrisiert. Im Folgenden wird exemplarisch das Verschicken eines TextFax erläutert.

Wie im Kapitel 1.2 erwähnt, arbeitet das Alarm Modem bei einem Alarm Datenbanken in einer bestimmten Reihenfolge ab:

EventHandler – MessageJobTemplates – AddressBook – MessageText

4.1.1. Event Handler (TiXML Reference Kapitel 3.8)

Als erstes wird ein Event-Handler erzeugt (Register 10-EventHandler):

```
<Fax-Alarm>
  <SendMail _="MessageJobTemplates/Fax-Alarm" />
</Fax-Alarm>
```

Der Name des Event-Handlers (hier : "Fax-Alarm") ist frei wählbar und wird zum Auslösen des Alarms per DoOn-Befehl verwendet.

Der EventHandler enthält Befehle, die beim Eintreten dieses Events abgearbeitet werden. Bei diesem einfachen Alarm ist dies zunächst nur das Verschicken einer Nachricht per „SendMail“.

Der Befehl `SendMail` verweist auf ein `MessageJobTemplate` "Fax-Alarm".

4.1.2. MessageJobTemplate (TiXML Reference Kapitel 3.7)

Der MessageJobTemplate (Register 11-MessageJobTemplates) enthält Angaben zum Nachrichtentyp, Absender/Empfänger sowie dem Nachrichteninhalte:

```
<Fax-Alarm _="TextFax">
  <Recipient _="/D/AddressBook/Receiver"/>
  <Sender _="/D/AddressBook/MySelf"/>
  <Body _="/D/UserTemplates/Fax-Alarm"/>
  <Subject _="&#xae;/D/UserTemplates/S-Fax-Alarm/Subject;"/>
</Fax-Alarm>
```

Recipient und Sender sind mit einem Eintrag im zuvor erstellten Adressbuch verknüpft. Der Betreff (Subject) und der Nachrichtentext (Body) verweisen auf die UserTemplates Datenbank.

Zu beachten ist, daß bei SMS und CityRuf der Body weggelassen wird.

4.1.3. MessageText (TiXML Reference Kapitel 3.6)

Die MessageTexte werden in den UserTemplates (Register 12-MessageText) abgelegt. Ein einfacher Text für den Fax-Alarm könnte z.B. so aussehen:

```
<Fax-Alarm>
  <L _="This is the message body of the fax alarm."/>
  <L _="Enter as many lines as you need."/>
</Fax-Alarm>
<S-Fax-Alarm>
  <Subject _="This is the subject of the fax alarm."/>
</S-Fax-Alarm>
```

Zur besseren Lesbarkeit wurden Body und Subject eines zusammengehörenden Alarms gleichlautend benannt, und das Subject durch ein vorangestelltes S- gekennzeichnet.

Bei SMS und CityRuf wird nur der Betreff verwendet. Der Inhalt darf 160 Zeichen für SMS bzw. 80 Zeichen für CityRuf nicht überschreiten.

Im Body wird jede neue Zeile durch <L _="Text"/> dargestellt.

4.1.4. Testen des Alarms (TiXML Reference Kapitel 2.4.9.1)

Der einfache Alarm ist nun vollständig parametrisiert. Nachdem das Projekt ins Alarm Modem geladen wurde, kann der Alarm über einen DoOn-Befehl an der seriellen Schnittstelle ausgelöst werden, z.B. [`<DoOn _="FaxAlarm"/>`].

4.2. Erweiterte Alarmfunktionen

Eine einfache Alarmfunktion kann durch viele Funktionen erweitert werden. Der Alarm kann z.B. über einen Port oder eine SPS-Variable ausgelöst werden, einen Port setzen, Alarmkaskadierungen enthalten oder sogar auf Betätigungen warten.

Die Kette der vom Tixi Alarm Modem abgearbeiteten Datenbanken wird dadurch, wie in Kapitel 1.2 beschrieben, erweitert:

EventStates – ProcessVars – EventHandler – MessageJobTemplates – AddressBook - MessageText

Dieses Kapitel wird den vorhandenen einfachen Alarm (Fax-Alarm) schrittweise erweitern.

4.2.1. Erweitern der Meldetexte

4.2.1.1. Signaturen (TiXML Reference Kapitel 3.6)

Ein großer Vorteil der Verwendung von XML ist die Möglichkeit über Referenzen Inhalte miteinander zu verknüpfen.

Als Beispiel (*basis-fax-signature.cnf*) dient hier ein Text mit Standortdaten, der als Signatur an jede Nachricht angehängt werden soll. Dies spart Platz im Filesystem und Änderungen in der Signatur werden dann in allen Nachrichten wirksam.

Die Signatur:

```
<Signature>
  <L _="Tixi.Com GmbH"/>
  <L _="Karmeliterweg 114"/>
  <L _="13465 Berlin"/>
</Signature>
```

In den Text wird es per „Include“ eingefügt:

```
<Fax-Alarm>
  <L _="This is the message body of the fax alarm."/>
  <L _="Enter as many lines as you need."/>
  <L _=""/>
  <Include _="/D/UserTemplates/Signature"/>
</Fax-Alarm>
<S-Fax-Alarm>
  <Subject _="This is the subject of the fax alarm."/>
</S-Fax-Alarm>
```

Beim Empfänger erscheint dann dieser Text:

```
This is the subject of the fax alarm.

This is the message body of the fax alarm.
Enter as many lines as you need.

Tixi.Com GmbH
Karmeliterweg 114
13465 Berlin
```

4.2.1.2. Text mit Variablen (TiXML Reference Kapitel 12)

Meldetexte können aber auch durch Variablen ergänzt werden, z.B. Systemvariablen wie Datum/Uhrzeit, Seriennummern oder Zustände von I/O-Ports oder SPS-Variablen (basis-fax-textvariables.cnf)

Hinweis: Das Beispielprojekt erfordert ein Alarm Modem mit I/O-Erweiterung. Ohne diese Erweiterung muss die Zeile mit der Referenz auf den Port 1 entfernt werden.

```
<Fax-Alarm>
  <L _="This is the message body of the fax alarm."/>
  <L _="Enter as many lines as you need."/>
  <L _=" "/>
  <L _="Systemdata: "/>
  <L _="Serial number: &#xae;/SerialNo;"/>
  <L _="Modemtype: &#xae;/Hardware/Modules/Modem0;"/>
  <L _="Date: &#xae;/TIMES/RFC822Date;"/>
  <L _=" "/>
  <L _="Value Port 1: &#xae;/Process/C0/I/P0;"/>
</Fax-Alarm>
<S-Fax-Alarm>
  <Subject _="This is the subject of the fax alarm."/>
</S-Fax-Alarm>
```

4.2.2. Multiple Sendmail (TiXML Reference Kapitel 3.8)

Bisher war dem EventHandler ein einziger SendMail Befehl zugewiesen. Es ist jedoch auch möglich mehrere SendMail Befehle in einem EventHandler zu verwenden, und so bei einem Alarm mehrere Empfänger und Nachrichtentypen zu bedienen (basis-multiplesendmail.cnf):

```
<Alarm>
  <SendMail _="MessageJobTemplates/Fax-Alarm" value="0"/>
  <SendMail _="MessageJobTemplates/Email-Alarm" value="0"/>
</Alarm>
```

Jeder SendMail Befehl verweist hier auf ein eigenes MessageJobTemplate für Fax und E-Mail, diese wiederum jedoch auf einen gemeinsamen Nachrichtentext.

4.2.3. Port löst Alarm aus

4.2.3.1. Der Service-Button (TiXML Reference Kapitel 6.7)

Man kann den Service-Button an der Rückseite des Geräts zum Auslösen eines Alarms verwenden (basis-fax-ServiceButton.cnf). Denkbar wäre auch die Verknüpfung mit einer Alarm-Quittierung (Kapitel 4.2.5.4).

Der Service-Button wird über einen System-Event-Handler eingebunden (Register 10-EventHandler):

```
<System>
  <OnButton>
    <SendMail _="MessageJobTemplates/Fax-Alarm"/>
  </OnButton>
</System>
```

4.2.3.2. I/O-Ports (TiXML Reference Kapitel 6.3)

Um einen Alarm mit einem digitalen Eingang zu verknüpfen wird auf die EventStates Datenbank (Register 14-EventStates) zurückgegriffen (basis-messages-ports.cnf):

Die EventStates verknüpfen einen Event-Handler mit einer Process-Variable. Hier soll z.B. der Eingang P0 auf einer Erweiterungskarte den Fax-Alarm auslösen

```
<Fax-Alarm_STATE>
  <Enabled _="TRUE" />
  <ProcessVar _="/Process/C0/I/P0" flank="low" />
  <Event _="Fax-Alarm" />
</Fax-Alarm_STATE>
```

Mit dem Parameter **“Enabled”** kann ein EventState (de-)aktiviert werden. Steht er auf TRUE, so nimmt das Tixi Alarm Modem den EventState in die Prozessabarbeitung auf, bei FALSE wird er ignoriert.

Der Parameter **“Event”** legt fest welcher Alarm beim Eintreten der Bedingung „ProcessVar“ ausgelöst wird.

Der Parameter **“ProcessVar”** kann nun auf zwei verschiedenen Weisen konfiguriert werden:

1. Direkt adressiert (schneller aber unflexibel)

Bei der direkten Adressierung werden die Portadresse sowie der Wert der Bedingung im EventState eingegeben.

```
<ProcessVar _="/Process/C0/I/P0" flank="low" />
```

In diesem Fall ist der Alarm mit dem Eingang P0 der Erweiterungskarte verknüpft. Der Parameter „low“ legt fest, dass die Bedingung bei der Änderung des Port von 1 auf 0 eintritt. Alternativ gibt es die Parameter „high“ (Wechsel auf 1) und „both“ (jeder Wechsel)

2. Über ProcessVars Datenbank (langsamer, aber flexibler)

In diesem Fall weist der Parameter ProcessVar auf einen Eintrag in der ProcessVars Datenbank (siehe nächstes Kapitel):

```
<ProcessVar _="/Process/PV/Fax-Alarm" />
```

4.2.3.3. ProcessVars (TiXML Reference Kapitel 6.2)

Die ProcessVars Datenbank (Register 13-ProcessVars) legt die Bedingungen für EventStates fest (basis-messages-ProcessVars.cnf).

Logische Operationen

Bei diesem Beispiel für den Fax-Alarm tritt die Bedingung immer dann ein, wenn der Port von 0 auf 1 wechselt.

```
<Fax-Alarm>
  <Value>
    <LD _="C0/I/P0" />
  </Value>
</Fax-Alarm>
```

Anstelle des LD-Befehls können noch viele andere logische Operationen verwendet werden, z.B. LDN, ST oder Operationen zum Verknüpfen mehrerer Ports:

Bei diesem Beispiel für den E-Mail-Alarm tritt die Bedingung immer dann ein, wenn beide Ports von 0 auf 1 wechseln.

```
<Email-Alarm>
  <Value>
    <LD _="C0/I/P1" />
    <AND _="C0/I/P2" />
  </Value>
</Email-Alarm>
```

Vergleichsoperationen

Neben den logischen Operationen sind auch noch Vergleichsoperationen möglich.

Hier für den Express-E-Mail-Alarm ein Beispiel mit Ports, jedoch machen

Vergleichsoperationen eigentlich nur bei Variablen mit größerem Wertebereich, z.B. SPS Variablen, Sinn:

```
<Express-Alarm>
  <Value>
    <GT v1="C0/I/P3" v2="C0/I/P4" />
  </Value>
</Express-Alarm>
```

Die Bedingung tritt ein, wenn v1 größer v2 ist. Bei v2 kann anstelle einer Variable auch direkt ein Wert eingegeben werden.

Zeitvergleich

Die letzte Bedingung für ProcessVariablen ist die Zeit, d.h. ein Alarm wird immer zu

bestimmten Zeiten ausgelöst. Dabei können Uhrzeit und/oder Datum ein Kriterium sein.

In diesem Beispiel erfolgt die SMS-Alarmierung täglich beim Wechsel in die angegebene Zeitspanne, also um 14:20.

```
<SMS-Alarm>
  <Value>
    <TIME _="14:20:00-16:30:00" />
  </Value>
</SMS-Alarm>
```

Die Zeit-Bedingung kann auch mit logischen Bedingungen kombiniert werden, so lassen sich z.B. mehrere Zeiträume angeben:

```
<CityRuf-Alarm>
  <Value>
    <TIME _="14:20:00-16:30:00" />
    <TIME _="18:00:00-20:00:00" />
    <OR _=" " />
  </Value>
</CityRuf-Alarm>
```

Für komplexere Zeitabhängige Alarmierungen ist die Scheduler-Datenbank zu verwenden (siehe nächstes Kapitel).

4.2.4. Scheduler (TiXML Reference Kapitel 7)

Der Scheduler (Register `16-Schedule`) wird für komplexe Zeitschaltfunktionen verwendet, z.B. Schichtpläne. Die Zeitkriterien reichen von Minuten, über Wochentage bis zu Monaten und Jahren. Kombinationen und Ausschlusszeiten (Ferienprogramm, Register `15-Schedule Conditions`) sind auch möglich.
(`basis-fax-scheduler.cnf`)

Dieser Scheduler verschickt z.B. Werktags um 18:00 den Fax-Alarm (denkbar als Tagesbericht):

```
<Workday-Fax _="Fax-Alarm">
  <Weekday _="Mo-Fr" />
  <Time _="18:00" />
</Workday-Fax>
```

Der durch den Kalendereintrag auszulösende EventHandler (hier: Fax-Alarm), wird in der ersten Zeile des Eintrags als Parameter angegeben.

4.2.4.1. Schichtplan

Das Beispiel `basis-fax-Shift.cnf` demonstriert die Verwendung des Schedulers, um Alarmempfänger zeitabhängig zu wählen. Ab 09:00 UHR soll die Tagesschicht (Receiver) den Fax-Alarm erhalten, ab 21:00 UHR erhält die Nachtschicht (Shift) die Alarme.

Zunächst wird der Empfänger der Nachtschicht im Adressbuch angelegt:

```
<Shift>
  <Email _="tixi-support@tixi.com" />
  <SMS_No _="03040608500" />
  <SMS_Provider _="AnnyWay" />
  <CityRuf _="3949000" />
  <Pager_Provider _="CityRuf" />
  <Fax _="+49-30-40608401" />
  <Express-Email _="SUPPORT+49-30-40608444" />
</Shift>
```

Desweiteren wird eine Prozess-Variable benötigt, in welche der Scheduler den derzeit aktiven Empfänger schreiben kann (Register 13-ProcessVars):

```
<Shiftworker/>
```

Das Ändern dieser Prozess-Variable, und somit des Empfängers, wird über zwei Event-Handler ausgeführt. Diese schreiben den Empfängernamen des Adressbuchkontaktes in die Variable:

```
<Shiftchange_Day>
  <Set _="/Process/PV/Shiftworker" value="Receiver"/>
</Shiftchange_Day>
```

```
<Shiftchange_Night>
  <Set _="/Process/PV/Shiftworker" value="Shift"/>
</Shiftchange_Night>
```

Diese Event-Handler werden nun zum Schichtwechsel über den Scheduler aufgerufen:

```
<Shift_Day _="Shiftchange_Day">
  <Time _="09:00"/>
</Shift_Day>

<Shift_Night _="Shiftchange_Night">
  <Time _="21:00"/>
</Shift_Night>
```

Damit der Alarm den Wert der Process-Variable als Empfänger verwendet, wird im dazugehörigen MessageJobTemplate der Pfad auf den Adreßbuchkontakt über die Prozess-Variable manipuliert:

```
<Fax-Alarm _="TextFax">
  <Recipient _="/D/AddressBook/&#xae;/Process/PV/Shiftworker,Receiver;"/>
  <Sender _="/D/AddressBook/MySelf"/>
  <Body _="/D/UserTemplates/Fax-Alarm"/>
  <Subject _="&#xae;/D/UserTemplates/S-Fax-Alarm/Subject;"/>
</Fax-Alarm>
```

Nach dem Auflösen der Referenz ®/Process/PV/Shiftworker; findet das Alarm Modem je nach Tageszeit den Recipienten:

```
<Recipient _="/D/AddressBook/Receiver"/> oder
<Recipient _="/D/AddressBook/Shift"/>
```

Die im MessageJobTemplate durch ein Komma abgetrennte Angabe "Receiver" dient als Start- bzw. Alternativwert. Sie wird immer dann verwendet, wenn die Prozess-Variable noch keinen Wert enthält, z.B. nach dem Neustart des Modems.

Die Variable kann nach dem Neustart aber auch manuell über einen Set-Befehl initialisiert werden:

```
[<Set _="/Process/PV/Shiftworker" value="Shift" ver="y"/>]
```

4.2.4.2. Scheduler Conditions (Ferienprogramm)

In vielen Fällen benötigt eine Kalenderfunktion, insbesondere im Fall des Schichtplans, Ausnahmeregelungen. Diese werden im Register `15-Schedule Conditions` definiert.

Als Beispiel dient der zuvor konfigurierte Schichtplan, in welchen die festen deutschen Feiertage eingefügt werden. An den Feiertagen soll ganztägig der Empfänger der Nachtschicht „Shift“ verwendet werden (`basis-fax-shift-condition.cnf`):

Um den gewünschten Effekt zu erreichen, wird an den Feiertagen der Wechsel auf die Tages-Schicht weggelassen. Im Scheduler führt man dort eine Bedingung ein:

```
<Shift_Day _="Shiftchange_Day">
  <Time _="09:00"/>
  <Condition _="Condition/Holidays"/>
</Shift_Day>
```

Diese Bedingung verweist auf die Condition-Datenbank:

```
<Holidays>
  <Data not="1.1.,1.5.,3.10.,25.12.,26.12."/>
</Holidays>
```

Der Parameter „not“ legt fest, dass der Kalendereintrag an diesen Tagen nicht gültig ist.

Alternativ hätte diese Bedingung auch direkt im Schedulereintrag eingegeben werden können:

```
<Shift_Day _="Shiftchange_Day">
  <Time _="09:00"/>
  <Data not="1.1.,1.5.,3.10.,25.12.,26.12."/>
</Shift_Day>
```

Der Vorteil der Verwendung der Conditions-Datenbank ist aber, dass die Bedingung so für mehrere Kalenderfunktionen verwendet werden kann, und somit nur einmal eingegeben werden braucht, und zentral geändert werden kann.

4.2.5. erweiterte EventHandler Funktionen (TiXML Reference Kap. 3.8.1)

Im Tutorial wurde der EventHandler bisher nur zum Verschicken von Nachrichten per SendMail verwendet. Es sind allerdings noch viel komplexere Operationen möglich:

4.2.5.1. Alarmprioritäten

Jedem SendMail Befehl kann eine Priorität zugewiesen werden. Anhand der Priorität werden die zu versendenden Nachrichten in der Warteschlange sortiert, und zwar wird die Nachricht mit der höchsten Priorität (1=kleinste, 255=größte) zuerst verschickt.

Wenn also z.B. ein Feuersalarm mit Priorität 10 ausgelöst wird, und zeitgleich noch einige Nachrichten mit niedrigerer Priorität in der Warteschlange stehen, so wird die Nachricht sofort nach dem frei werden der Leitung verschickt. (`basis-messages-priorities.cnf`)

Die Priorität wird im Event-Handler wie folgt festgelegt:

```
<Fax-Alarm>
  <SendMail _="MessageJobTemplates/Fax-Alarm" value="0">
    <Priority _="5" />
  </SendMail>
</Fax-Alarm>
```

4.2.5.2. Alarm setzt Ausgang - Set

Per Set-Befehl können Ausgänge manuell gesetzt werden (TiXML Reference Kap. 6.5.3). Dieser Befehl kann jedoch auch in einen EventHandler eingebaut werden.

(basis-fax-setport.cnf)

In diesem Beispiel wird neben dem Versenden der Nachricht auch noch der Ausgang P0 auf 1 gesetzt (denkbar als Alarmkontakt):

```
<Fax-Alarm>
  <SendMail _="MessageJobTemplates/Fax-Alarm" />
  <Set _="/Process/C0/Q/P0" value="1" />
</Fax-Alarm>
```

4.2.5.3. Alarmkaskade „OnError“

Das Tixi Alarm Modem arbeitet grundsätzlich mit Wahlwiederholungen. Diese helfen jedoch in dem Fall nicht weiter, in dem ein Empfänger einfach nicht erreichbar ist.

Für diesen Fall sind Alarmkaskaden gedacht. Wenn eine Nachricht nicht abgesetzt werden konnte wird sie einfach an einen anderen Empfänger geschickt (basis-kaskade.cnf):

```
<Fax-Alarm>
  <SendMail _="MessageJobTemplates/Fax-Alarm">
    <OnError _="Email-Alarm" />
    <MaxRepeat _="1" />
    <Interval _="180s" />
  </SendMail>
</Fax-Alarm>
```

In diesem Beispiel wird eine Nachricht zweimal versucht zuzustellen (MaxRepeat=1 bedeutet eine Wahlwiederholung mit Interval=180s: 180s Abstand). Wenn beide Versuche fehlschlagen wird das Event „Email-Alarm“ ausgelöst.

Es wäre nun denkbar das Event „Email-Alarm“ auch wieder mit einer Kaskade auszustatten. Es könnte sogar zurück auf das Event „Fax-Alarm“ verweisen und so eine Schleife erstellen, die erst nach dem ordnungsgemäßen Absetzen der Nachricht abbricht.

4.2.5.4. Bestätigung über Port – „OnOK“

Neben dem Befehl „OnError“ gibt es auch noch den Befehl „OnOK“ der z.B. zum Schalten von Ausgängen in Verbindung mit einer erfolgreich abgesetzten Störmeldung verwendet werden kann (basis-kaskade.cnf).

In diesem Beispiel wird im Alarmfall ein Ausgang gesetzt und nach dem erfolgreichen Absetzen der Nachricht der Port über das Event „Setport“ zurückgesetzt.

```
<Express-Alarm>
  <Set _="/Process/C0/Q/P0" value="1" />
  <SendMail _="MessageJobTemplates/Express-Alarm">
    <OnOK _="Setport" />
    <MaxRepeat _="1" />
    <Interval _="180s" />
  </SendMail>
</Express-Alarm>
<Setport>
  <Set _="/Process/C0/Q/P0" value="0" />
</Setport>
```

Es können auch „OnOK“ und „OnError“ Befehle gemeinsam in einem SendMail-Befehl verwendet werden.

4.2.5.5. Quittieren von Nachrichten

Eine Alternative zur Alarmkaskade ist das Verschicken von Nachrichten mit Quittierungsanforderung. Dadurch ist gewährleistet, daß die Alarmnachricht den Empfänger tatsächlich erreicht hat.

Bestätigungen sind per SMS, Express-E-Mails, Internet E-Mail oder Variablen möglich. (basis-confirmation.cnf)

Der EventHandler sieht wie folgt aus:

```
<SMS-Alarm>
  <SendMail _="MessageJobTemplates/SMS-Alarm">
    <ConfirmID _="99" />
    <Timeout _="180s" />
    <OnTimeout _="SMS-Alarm" />
  </SendMail>
</SMS-Alarm>
```

Der dazugehörige Nachrichtentext:

```
<SMS_Alarm>
  <L _="" />
</SMS_Alarm>
<S-SMS_Alarm>
  <Subject _="This is the text of the SMS alarm. Enter max. 160
  characters. Confirmation requested ! &#xae;~/_Fingerprint;
  (&#xae;~/_ConfirmID;)" />
</S-SMS_Alarm>
```

Zudem ist ein spezieller System-EventHandler nötig:

```
<System>
  <Confirmation>
    <Confirm _="&#xae;~/_ConfirmID;" />
  </Confirmation>
</System>
```

Die Quittierungsfunktion arbeitet mit einer BestätigungsID (ConfirmID, im Beispiel "99") und einem eindeutigen, automatisch generierten Fingerprint. Jedem Alarm muss im EventHandler eine eigene Confirm-ID zugeordnet werden.

Beides wird über eine Referenz in den Betreff der Nachricht integriert, und muss als Quittierung lediglich vom Empfänger zurückgeschickt werden.

Der Befehl „OnTimeout“ legt dabei fest, was nach Ablauf der Timeout-Zeit, in der die Quittierung erfolgen muss, geschieht. Im Beispiel wird der gleiche Event nochmal ausgelöst, sodaß es solange wiederholt wird, bis eine Quittierung eingetroffen ist.

Denkbar wäre auch das Auslösen eines anderen Events mit alternativem Empfänger nach Ablauf des Timeouts.

4.2.5.6. Übergabe von Parametern

Wenn ein Alarm per „DoOn“-Befehl ausgelöst wird können ihm auch Parameter übergeben werden, die z.B. in Meldetexte eingebaut werden können. (*basis-fax-parameter.cnf*)

Als Beispiel dient der Fax-Alarm. In den Nachrichtentext soll eine Zimmernummer dynamisch eingefügt werden.

```
<Fax-Alarm>
  <L _="This is the message body of the fax alarm."/>
  <L _="Enter as many lines as you need."/>
  <L _=""/>
  <L _="Alarm in room number: &#xae;~/Room;"/>
</Fax-Alarm>
<S-Fax-Alarm>
  <Subject _="This is the subject of the fax alarm."/>
</S-Fax-Alarm>
```

Hier wird im Text der beim DoOn übergebene Parameter "Room" über eine Referenz *®~/Room;* eingefügt.

Der **DoOn-Befehl** muss dazu wie folgt aussehen:

```
[<DoOn _="Fax-Alarm" ver="y">
  <Room _="23.3"/>
</DoOn>]
```

Das generierte Fax sieht dann so aus:

```
This is the subject of the fax alarm.

This is the message body of the fax alarm.
Enter as many lines as you need.

Alarm in room number: 23.3
```

Die Anzahl der Parameter ist beliebig. Es ist auch denkbar die übergebenen Parameter als Wert für Variablen zu verwenden.

So könnte z.B. ein EventHandler mit „**Set-Befehl**“ einen Port auf den übergebenen Wert setzen:

DoOn-Befehl:

```
[<DoOn _="Setport" ver="y">
    <Parameter _="1"/>
</DoOn>]
```

EventHandler:

```
<Setport>
    <Set _="/Process/C0/Q/P0" value="&#xae;~/Parameter;"/>
</Setport>
```

4.2.5.7. Loggen von Daten (TiXML Reference Handbuch Kapitel 4)

Neben den bereits bekannten Logfiles können auch eigene Logfiles erzeugt und mit Daten beschrieben werden. Diese können zum einen als XML-Logfiles, zum anderen als platzsparende Binärlogfiles angelegt werden.

In diesem Beispiel sollen z.B. die I/O-Portzustände regelmäßig protokolliert werden.

XML-Logging (basis-portlog.cnf):

Zunächst wird das **Logfile** "PortLog" mit der gewünschten Größe (Ringspeicher) hinzugefügt:

```
<LogFiles>
    <JobReport size="10240"/>
    <Event size="10240"/>
    <Login size="10240"/>
    <IncomingMessage size="10240"/>
    <FailedIncomingCall size="10240"/>

    <PortLog size="10240"/>
</LogFiles>
```

Dieser **Event-Handler** schreibt den Zustand vom I/O-Port P0 in das Logfile:

```
<Logging>
    <Log _="PortLog">
        <Port1 _="&#xae;/Process/C0/I/P0;"/>
    </Log>
</Logging>
```

Der beim Befehl "Log" angegebene Parameter "PortLog" ist der Name des verwendeten Logfiles, gefolgt von den zu loggenden Werten (Im Beispiel eine Referenz auf den Eingang P0). Die Anzahl der zu loggenden Daten ist beliebig.

Binär-Logging (basis-portbinlog.cnf):

Für binäres Datenloggen wird als erstes ein **binäres Logfile** der gewünschten Größe (Ringspeicher) erzeugt:

```
<LogFiles>
  <JobReport size="10240"/>
  <Event size="10240"/>
  <Login size="10240"/>
  <IncomingMessage size="10240"/>
  <FailedIncomingCall size="10240"/>

  <PortBinLog size="10240" contenttype="binary" record="Struct"/>

</LogFiles>
```

Das Logfile verweist auf ein Record (Register 7-LogfileRecords), in welchem die Datenstruktur beschrieben ist:

```
<Struct>
  <Value1 _="int" size="1" value="#xae;/Process/C0/I/P0;"/>
</Struct>
```

Hier wird der Typ der Daten (integer "int", alternative: "string"), die Größe des Datensatzes (1=1 Byte) sowie der zu loggende Wert (via Referenz auf Eingang P0) angegeben.

Im **Event-Handler** muss lediglich der Befehl "BinLog" mit dem zu verwendenden Logfile aufgerufen werden:

```
<Logging>
  <BinLog _="PortBinLog"/>
</Logging>
```

Zeitgesteuert Daten loggen:

Damit die Logging-Funktion einen Sinn macht, sollte diese über den Scheduler (Kapitel 4.2.4) zyklisch ausgeführt werden, z.B. alle 5 Minuten:

```
<LogTime _="Logging">
  <Minute _="0,5,10,15,20,25,30,35,40,45,50,55"/>
</LogTime>
```

Da alle Logfiles als Ringpuffer verwendet werden, kann das Filesystem nicht volllaufen.

Senden von Daten:

Logfiles können auch per „IncludeLog“ Befehl in Nachrichtentexte eingefügt werden, und so z.B. regelmäßig per Scheduler an die Zentrale übermittelt werden (basis-portbinlog-email.cnf).

Nachrichtentext:

```
<Email >
  <IncludeLog _="PortBinLog" range="previous 24 hours"/>
</Email >
<S-Email >
  <Subject _="E-Mail with logged data."/>
</S-Email >
```

Scheduler:

```
<SendLogTime _="Email">
  <Time _="00:01"/>
</SendLogTime>
```

Beim IncludeLog Befehl wird der Name des Logfile sowie der Bereich "range" der zu sendenden Daten festgelegt. Im Beispiel werden die Daten der letzten 24 Stunden verschickt. Die Möglichen Bereiche sind im TiXML Reference Handbuch im Kapitel 2.4.9.2 beschrieben.

Senden von Daten als Attachment:

In vielen Fällen kann der Empfänger mit den geloggtten Daten in XML-Format nicht viel anfangen. Zur Weiterverarbeitung der Daten bietet sich das CSV-Format an, welches z.B. in Excel gelesen werden kann. Via „IncludeLogTXT“ Befehl können die Daten vor dem Einfügen formatiert werden, z.B. die IDs, Datum und Uhrzeit entfernt oder die Zeilen durch Start- und End- und Trennzeichen ergänzt werden. Zudem kann das Logfile in ein Attachment mit beliebigen Namen verpackt werden. (basis-portbinlog-email-attachment.cnf).

Der Nachrichtentext muss entsprechend dem Standard für E-Mail-Attachments formatiert werden. Dazu werden an bestimmten Stellen „Boundary“- sowie Leerzeilen eingefügt, anhand derer das empfangende E-Mail-Programm den Nachrichtentext vom Anhang trennen kann.

```
<Email >
  <L _="--&#xae;~/Boundary;"/>
  <L _="Content-type: text/plain; charset=ISO-8859-1"/>
  <L _="Content-description: Mail message body"/>
  <L _=""/>
  <L _="This is an email with attachment"/>
  <L _=""/>
  <L _="--&#xae;~/Boundary;"/>
  <L _="Content-type: text/plain; charset=ISO-8859-1"/>
  <L _="Content-disposition: attachment;
filename=&quot;data.csv&quot;"/>
  <L _=""/>
  <IncludeLogTXT _="PortBinLog" flags="NoId" tagstart="" tagend=""
colsep=";" range="#5-"/>
  <L _="--&#xae;~/Boundary;--"/>
<Email >
<S-Email >
  <Subject _="E-Mail with logged data attachment."/>
</S-Email >
```

Bei dem Parameter `filename="data.csv"` kann anstelle von `data.csv` jeder beliebige Dateiname angegeben werden. Denkbar wäre auch, den Dateinamen aus dem aktuellen Datum zu generieren:

```
filename=&quot;&#xae;/TIMES/DATE;.csv&quot;
```

Die umfangreichen Formatierungsmöglichkeiten sind den TiXML-Reference Handbuch Kapitel 4.8 zu entnehmen.

Zusätzlich muss das `MessageJobTemplate` um den `Boundary`-Parameter erweitert werden:

```
<Email _="SMTP" >
  <Recipient _="/D/AddressBook/Receiver"/>
  <Sender _="/D/AddressBook/MySelf"/>
  <Body _="/D/UserTemplates/Email"/>
  <Subject _="&#xae;/D/UserTemplates/S-Email/Subject;/>
  <Boundary _="Message-Boundary-TAM"/>
</Email>
```

4.2.5.8. Internetzeitsynchronisierung / Sommer-Winter-Zeitungstellung (TiXML-Reference Kap. 3.12)

Die RTC des Tixi Alarm Modems kann über einen Zeitserver im Internet z.B. mit der Zeit der Atomuhr abgeglichen werden. Diese Funktion kann daher auch zur automatischen Sommer-Winter-Zeitungstellung verwendet werden.

Es gibt zwei verschiedene Zeitserverprotokolle:

TIME liefert die UTC Zeit in einem standardisierten String

DAYTIME liefert die lokale Zeit in einem nicht-standardisierten String

Beim TIME-Protokoll muss die Umrechnung der Zeitzone (z.B. +0100 für Deutschland) selber durchgeführt werden. Die Sommer-Winterzeit ist ebenfalls noch nicht enthalten.

Beim DAYTIME-Protokoll liefert der Server die lokal gültige Uhrzeit inkl. Zeitzonenumrechnung und Sommer-Winterzeit. Dem Alarm Modem muss das Format des Strings vorgegeben werden. Da die Zeit bereits vollständig umgerechnet vorliegt, müssen die `TimeZone` und der `TimeDiff`-Parameter auf +0000 stehen.

Der Zeit-String eines Internet-TimeServers kann über eine Abfrage des Port 13 erfolgen. Wenn man mit dem Internet verbunden ist, kann über die Eingabeaufforderung der String wie folgt abgefragt werden:

```
telnet Server-URL 13, z.B.
telnet ptbtime1.ptb.de 13
```

Als Antwort erhält man z.B.:

```
10 JUN 2004 10:24:16 METDST
```

Das Beispiel `basis-timeserver.cnf` zeigt die Verwendung des deutschen Atomuhr-DAYTIME-Servers zur Sommer-Winterzeitumstellung mit Hilfe des Schedulers (siehe Kapitel 4.2.4).

Timeserver-Einstellungen

Im Register 4-ISP steht die URL des Timeservers, das Protokoll, die Zeitdifferenz sowie ggf. das Zeit-Format:

```
<TimeServer>
  <ServerName _="ptbtime1.ptb.de"/>
  <Protocol _="DAYTIME"/>
  <TimeDiff _="+0000"/>
  <TimeFormat _="d E y h:n:s "/>
</TimeServer>
```

TimeZone

Im Register 2-UserData ist die TimeZone auf +0000 gesetzt:

```
<TimeZone _="+0000"/>
```

EventHandler

Der EventHandler SyncTime auf Register 10-EventHandler ruft den Befehl INetTime zur Internetzeitsynchronisierung auf:

```
<SyncTime>
  <INetTime/>
</SyncTime>
```

Scheduler

Die Zeitsynchronisierung muss nun noch zu sinnvollen Zeitpunkten automatisch vom Alarm Modem ausgeführt werden, z.B. am ersten jeden Monats:

```
<Internetttime _="SyncTime">
  <Month _="1-12"/>
</Internetttime>
```

Um die Sommer- und Winterzeitumstellung zu realisieren, muss das Modem sich am entsprechenden Tag kurz nach der Zeitumstellung mit dem Server synchronisieren. Europaweit erfolgt dies einheitlich am letzten Sonntag des März und Oktober. Das ergibt folgende Scheduler-Konfiguration:

```
<Summertime _="SyncTime">
  <Month _="3"/>
  <Day _="25-31"/>
  <Weekday _="Su"/>
  <Time _="02:00"/>
</Summertime>
im März
letzten 7 Tage
am Sonntag
um 2 UHR

<Wintertime _="SyncTime">
  <Month _="10"/>
  <Day _="25-31"/>
  <Weekday _="Su"/>
  <Time _="03:00"/>
</Wintertime>
im Oktober
letzten 7 Tage
am Sonntag
um 3 UHR
```

4.3. Fernwirken durch eingehende Nachrichten (H. Kap. 8)

Das Gerät ist auch in der Lage E-Mails, Express-E-Mails und SMS zu empfangen und auszuwerten. So können z.B. Events ausgelöst werden um I/O-Ports oder SPS-Variablen zu ändern.

Das Beispiel bezieht sich auf den Empfang von SMS (`basis-remoteswitch.cnf`):

4.3.1. Zugriffsschutz

Zunächst wird ein Zugriffsschutz auf dem Register `6-SMS_LogIn` konfiguriert, damit nur autorisierte Personen Fernwirken können.

Einfacher Zugriffsschutz:

Als Passwort wird z.B. das Wort „TIXI“ gewählt:

```
<SMS_Login>
  <Default _="TIXI" />
</SMS_Login>
```

Zugriffsschutz mit Nummernüberprüfung (SMS, Express-E-Mail):

Wird anstelle von „Default“ das Kürzel „OA_“ gefolgt von einer Nummer eingetragen, z.B.:

```
<SMS_Login>
  <OA_01721234567 _="TIXI" />
</SMS_Login>
```

so wird neben dem Passwort auch die übermittelte Rufnummer überprüft, d.h. das Fernwirken ist nur von der angegebenen Rufnummer aus möglich.

Zugriffsschutz mit Absenderüberprüfung (Internet E-Mail):

Bei E-Mail-Adressen wird der Alpha-Parameter (E-Mail Alias) ausgewertet:

Aus der Absenderkennung „Tixi Support“ ergibt sich folgender Eintrag (Leerzeichen durch Unterstriche ersetzen!):

```
<SMS_Login>
  <Tixi_Support _="TIXI" />
</SMS_Login>
```

4.3.2. Einfaches Fernwirken:

Mit dem angegebenen Passwort kann jedes im Gerät konfigurierte Event (EventHandler) ausgelöst werden. Dazu muss der Betreff (E-Mail, Express-E-Mail) bzw. der SMS-Text der Nachricht das Passwort und den Eventnamen in folgender Syntax enthalten:

```
Password Eventname
z.B.
```

```
TIXI SETPORT
```

Bei diesem Beispiel lautet das Passwort „TIXI“ und der EventHandler „SETPORT“.

Der dazu passende **EventHandler** könnte so aussehen:

```
<SETPORT>
  <Set _="/Process/C0/Q/P0" value="1"/>
</SETPORT>
```

Bei der eingehenden SMS wird also der Ausgang P0 auf 1 gesetzt.

4.3.3. Fernwirken mit Parameterübergabe

Um einem durch Fernwirken ausgelösten Event Parameter zu übergeben, muss der Betreff (E-Mail, Express-E-Mail) bzw. der SMS-Text der Nachricht das Passwort, den Eventnamen und die Parameter in folgender Syntax enthalten:

Passwort Eventname Parameter1 Parameter2 ... Parameter6
z.B.

```
TIXI SETVALUE 1
```

Bei diesem Beispiel lautet das Passwort „TIXI“ und der EventHandler „SETVALUE“. Dem Event wird der Parameter „1“ zur Weiterverarbeitung übergeben.

Der dazu passende **EventHandler** könnte so aussehen:

```
<SETVALUE>
  <Set _="/Process/C0/Q/P1" value="&#xae;~/P1;"/>
</SETVALUE>
```

Der Parameter wird über die Referenz ®~/P1; (bei mehreren Parametern bis P6 hochzählen) dem Set-Befehl als neuer Wert für den Ausgang P1 übergeben, der Port wird also auf 1 gesetzt.

Mit der SMS TIXI SETVALUE 0 könnte der Port wieder auf 0 werden.

4.3.4. Überwachung der Fernwirkfunktionen

In vielen Fällen ist es ratsam das Fernwirken durch Rückantworten zu erweitern.

Den Event erweitert man dafür um einen Sendmail Befehl, der nach dem Set-Befehl ausgeführt wird: (*basis-remoteswitch-answer.cnf*)

Das Beispiel ist für ein Tixi Alarm Modem GSM.

```
<SETPORT>
  <Set _="/Process/C0/Q/P0" value="1"/>
  <SendMail _="MessageJobTemplates/AnswerOnSETPORT"/>
</SETPORT>
```

4.3.4.1. Statische Rückantwort

Als **Bestätigung** für den Vorgang wird eine Nachricht an einen Empfänger aus dem Adreßbuch erstellt. Das dazu passende MessageJobTemplate sieht wie folgt aus:

```
<AnswerOnSETPORT _="GSMSMS">
  <Recipient _="/D/AddressBook/Receiver"/>
  <Sender _="/D/AddressBook/MySelf"/>
  <Subject _="&#xae;/D/UserTemplates/S-SMS_Answer/Subject;"/>
</AnswerOnSETPORT>
```

In den Nachrichtentext kann zur weiteren Sicherheit eine Referenz auf den ausgeführten Event-Namen eingebaut werden:

```
<SMS_Answer>
  <L _=" "/>
</SMS_Answer>
<S-SMS_Answer>
  <Subject _="The Event: &#xae;~/Event; was processed."/>
</S-SMS_Answer>
```

4.3.4.2. Dynamische Rückantwort

Um das Fernwirken dynamischer zu gestalten, kann die Rückmeldung auch an den Absender der Fernwirknachricht erfolgen (Beispiellevent: SETVALUE).

Das MessageJobTemplate verweist in diesem Fall nicht auf einen Empfänger im Adressbuch, sondern erhält die Empfängeradresse bzw. -nummer aus der Absenderkennung der eingegangenen Nachricht:

Das dazu passende MessageJobTemplate sieht wie folgt aus:

```
<DynamicAnswer _="GSMSMS">
  <Recipient _="&#xae;~/OA;"/>
  <Sender _="/D/AddressBook/MySelf"/>
  <Subject _="&#xae;/D/UserTemplates/S-SMS_Answer/Subject;"/>
</DynamicAnswer>
```

Es ist zu beachten, dass die Referenz auf den Empfänger je nach Nachrichtentyp variiert.

Bei GSM-SMS: ®~/OA;

Bei E-Mail oder Express-E-Mail: ®~/Alpha;

Wichtiger Hinweis für SMS-Empfang bei Alarm Modems am Festnetz (V.90/ISDN):

Der SMS-Empfang im Festnetz erfordert einen Telekom-Telefonanschluß mit Rufnummernanzeige (CLIP/CLIR).

Die dynamische SMS-Rückantwort über den Parameter „OA“ funktioniert nur bei GSM Alarm Modems, da diese über das ServiceCenter der SIM-Karte an alle Handy-Netze SMS schicken können.

Festnetz-Modems müssen je nach Empfänger-Netz ein anderes Gateway anwählen, und dieses kann anhand der Absender-Rufnummer nicht mehr eindeutig zugeordnet werden (Rufnummernmitnahme bei Providerwechsel).

Das Problem kann umgangen werden, wenn als Empfänger ein spezieller Adreßbucheintrag „SMSAnswer“ verwendet wird (basis-remoteswitch-answer-AnnyWay.cnf):

Das passende MessageJobTemplate...:

```
<DynamicAnswer _="SMS">
  <Recipient _="/D/AddressBook/SMSAnswer"/>
  <Sender _="/D/AddressBook/MySelf"/>
  <Subject _="&#xae;/D/UserTemplates/S-SMS_Answer/Subject;"/>
</DynamicAnswer>
```

...verweist auf diesen Empfänger im Adressbuch:

```
<SMSAnswer>
  <SMS_No _="#xae;~/OA;" />
  <SMS_Provider _="AnnyWay" />
</SMSAnswer>
```

Der Trick dabei ist, dass die Rückantwort über den Festnetz-SMS-Service von „AnnyWay“ verschickt wird. AnnyWay ist in der Lage SMS in alle Mobilfunknetze zu senden („Telekom“ wäre auch möglich, erfordert aber im Gegensatz zu AnnyWay eine Anmeldung). Als SMS-Nummer wird im Adressbuch wiederum auf die Absendernummer der eingegangenen SMS verwiesen.

4.3.4.3. Rückantwort mit Wertüberprüfung

Zur weiteren Sicherheit kann in die Rückantwort neben dem ausgeführten Event-Namen auch der aktuelle Wert der veränderten Variable übermittelt werden. Dadurch kann überprüft werden ob der Schaltbefehl auch tatsächlich wirksam war.

Um den aktuellen Wert in der Nachricht zu erhalten, muss im EventHandler zwischen dem Set-Befehl und dem SendMail eine kleine Verzögerung eingefügt werden, da durch die schnelle Abarbeitung im System sonst ggf. noch der Wert vor wirksam werden des Set-Befehls ermittelt wird.

Der EventHandler sieht dann wie folgt aus:

```
<SETVALUE2>
  <Set _="/Process/C0/Q/P2" value="#xae;~/P1;" />
  <Delay _="5s" />
  <SendMail _="MessageJobTemplates/AnswerValue" />
</SETVALUE2>
```

Der Nachrichtentext der Rückantwort mit Angabe des aktuellen Zustands von P2:

```
<SMS_AnswerValue>
  <L _=" " />
</SMS_AnswerValue>
<S-SMS_AnswerValue>
  <Subject _="The Event: &#xae;~/Event; was processed. Value P2:
&#xae;/Process/C0/Q/P2; " />
</S-SMS_AnswerValue>
```

4.3.5. Verhalten im Fehlerfall

Neben dem erfolgreichen Abarbeiten einer eingehenden Nachricht sind auch noch zwei Fehlerfälle denkbar:

- ungültiges Passwort
- ungültiges Event

Beide Fälle werden durch so genannte System-Events in der EventHandler Datenbank verarbeitet (basis-remoteswitch-errors.cnf):

Es ist zu beachten, dass diese Events in einer eigenen Untergruppe „System“ innerhalb der EventHandler-Gruppe stehen.

4.3.5.1. ungültiges Passwort

Falls eine empfangene SMS ein ungültiges Passwort enthält, wird automatisch der Event „SMSInvalidPassword“ bzw. „TixiInvalidPassword“ bei Express-E-Mails oder „POPInvalidPassword“ bei Internet E-Mails ausgeführt.

Diese Systemevents können beliebige Event-Handler Befehle enthalten, z.B. Set- oder SendMail-Befehle.

Ein sinnvolles Beispiel wäre das Loggen des Absender (Parameter „OA“) und des empfangenen Textes (Parameter „Text“) zusammen mit der Meldung „SMS with invalid password received“:

```
<SMSInvalidPassword>
  <Log _="FailedIncomingCall">
    <Annotation _="SMS with invalid password received"/>
    <OA _="&#xae;~/OA;"/>
    <Text _="&#xae;~/Text;"/>
  </Log>
</SMSInvalidPassword>
```

Bei E-Mails und Express-E-Mails sollte zusätzlich der Parameter „Alpha“ (Absenderadresse) geloggt werden.

4.3.5.2. ungültiges Event

Falls eine empfangene SMS einen ungültigen Eventnamen enthält, wird automatisch der Event „SMSInvalidEvent“ bzw. „TixiInvalidEvent“ bei Express-E-Mails oder „POPInvalidEvent“ bei Internet E-Mails ausgeführt.

Diese Systemevents können beliebige Event-Handler Befehle enthalten, z.B. Set- oder SendMail-Befehle.

Ein sinnvolles Beispiel wäre eine Rückmeldung an den Absender, sodaß er über den Fehler unterrichtet wird, sowie das Loggen des Absender (Parameter „OA“) und des empfangenen Textes (Parameter „Text“) zusammen mit der Meldung „SMS with invalid event received“:

```
<System>
  <SMSInvalidEvent>
    <Log _="FailedIncomingCall">
      <Annotation _="SMS with invalid event received"/>
      <Sender _="&#xae;~/OA;"/>
      <Text _="&#xae;~/Text;"/>
    </Log>
    <SendMail _="MessageJobTemplates/SMSAnswerOnError"/>
  </SMSInvalidEvent>
</System>
```

Bei E-Mails und Express-E-Mails sollte zusätzlich der Parameter „Alpha“ (Absenderadresse) geloggt werden.

Die Fehlermeldung wird am sinnvollsten, wie in Kapitel 4.3.4.2 beschrieben, an den Absender der Fernwirknachricht zurückgeschickt, und sollte einen Hinweis auf den unbekanntenen Event-Namen enthalten:

```
<SMS_AnswerOnError>
  <L _=" " />
</SMS_AnswerOnError>
<S-SMS_AnswerOnError>
  <Subject _="Error: The Event &#xae;~/Event; is unknown." />
</S-SMS_AnswerOnError>
```

4.3.5.3. Informationen bei OK-Fall loggen

Wenn der ausgeführte EventHandler mit dem Log-Befehl erweitert wird, können auch im OK-Fall nützliche Daten des Fernwirkprozesses geloggt werden, z.B. die Absenderrufnummer und der Fernwirktext:

```
<SETPORT>
  <Log _="IncomingMessage" >
    <OA _="&#xae;~/OA;" />
    <Text _="&#xae;~/Text;" />
  </Log>
  <Set _="/Process/C0/Q/P0" value="1" />
</SETPORT>
```

Bei E-Mails und Express-E-Mails sollte zusätzlich der Parameter „Alpha“ (Absenderadresse) geloggt werden.

4.3.6. E-Mails abholen

Damit das Alarm Modem via E-Mail Schaltbefehle erhalten kann, muss dieses zunächst nach E-Mails schauen. Dazu ist ein EventHandler mit dem „POP3Query“-Befehl zu konfigurieren, z.B.:

```
<CheckMail>
  <POP3Query/>
</CheckMail>
```

Dieser EventHandler sollte regelmäßig über den Scheduler aufgerufen werden, oder aber durch einen CallerID-Trigger (Kapitel 4.4) um es mit unserem NewMailSignal-Service zu verwenden. Siehe dazu auch: <http://www.NewMailSignal.com>

4.4. Event durch Anruf (CallerID) H. Kap. 8.2

Das Gerät ist auch in der Lage durch einen simplen Telefonanruf aufgrund der übermittelten Rufnummer einen Event auszuführen, um z.B. einen Ausgang zum Öffnen eines Garagentores zu schalten. (*basis-calltrigger.cnf*)

Die Rufnummer und das damit verknüpfte Event (EventHandler) werden auf dem Register 20-Incoming CallerID Alarms definiert:

```
<IncomingCallTrigger>
  <No1 _="0307654321" event="Setport" />
</IncomingCallTrigger>
```

“0307654321” wäre in diesem Fall die übermittelte Rufnummer. Der Parameter „event“ verweist auf den auszuführenden EventHandler, in welchem z.B. der Ausgang gesetzt wird.

Es können auch verschiedene Rufnummer verschiedenen Events zugewiesen werden. Der Parameter „NoX“ muss dazu fortlaufend hochgezählt werden.

5. Anbindung an eine SPS (TiXML PLC Handbuch)

Die bisherigen Beispiele haben als Alarmauslöser oder beim Fernwirken digitale Ein- und Ausgänge des Alarm Modems verwendet.

Anstelle der I/O-Ports können auch SPS-Variablen für alle beschriebenen Funktionen verwendet werden, z.B. für Datenlogging, Alarme und Fernwirken.

In den Beispielen müssen dazu die I/O-Port-Referenzen (z.B. „/Process/C0/I/P0“) auf die SPS-Variablen (z.B. „/Process/Aux1/D0/Merker1“) geändert werden.

Die komplexe Datenbank der SPS-Variablen (Register 17-External) sollte der Einfachheit halber mit TILA erstellt werden.

Das Beispiel `basis-sps.cnf` zeigt exemplarisch eine Alarmfunktion mit Textvariablen, Datenlogging sowie SMS-Fernwirken in Zusammenarbeit mit einer SPS.

6. Weitere Funktionen

Die in diesem Tutorial vorgestellten Funktionen decken den Großteil der vom Anwender geforderten Anwendungen ab.

Für weitere Funktionen und die Verfeinerung der gezeigten Beispiele empfehlen wir das Studium des TiXML Reference Handbuchs, z.B. bei

Logik- und Vergleichsoperationen
Webserver
Datenformatierungen
Scheduler-Conditions
Variablenformatierungen
uvm.

Quellen:

-
- 1 Hans Hubert Partl, Wien, siehe auch <http://www.boku.ac.at/htmlleinf/xmlkurz.html>
 - 2 TiXML Reference Manual, Tixi.Com, 2003