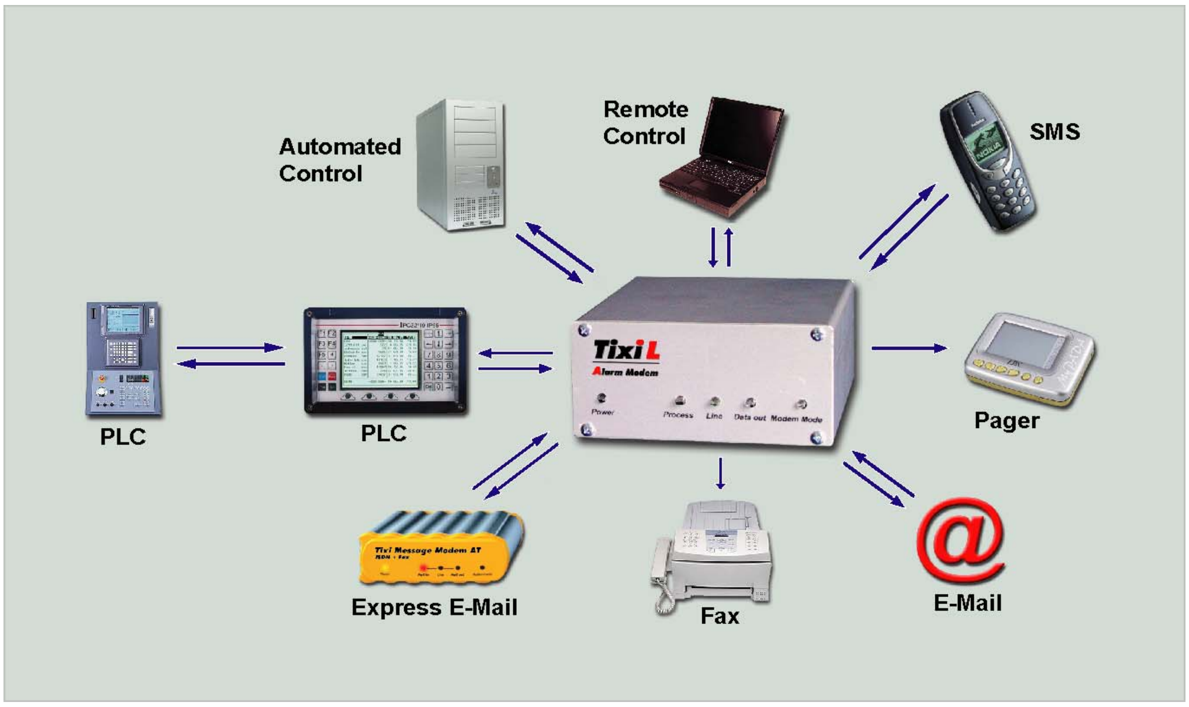
TUT-EN



TiXML-Tutorial

Until Firmware Version 1.72.14.0

Publication close:August 2005, Supported Firmware Version 1.72.14.0

Table Of Contents

1. Introduction

1.1. XML vs. TiXML Syntax

1.1.1. What is XML ?

This chapter will provide a short overview on XML structures containing and representing data only. More information on stylesheets etc. can be found at www.XML.org, or, as an overview, at Hubert Partl[1].

Start- and End-Tags **For Data Blocks**

Most commands (tags) in XML applications - as in HTML - will come in pairs of *Start*- and *End*-tags, which determine the meaning of the data contained. This data may be divided by sub-tags in the same way recursively.

Tag names are case sensitive. They may contain parameters (attributes) which have their values enclosed in quotation marks (ASCII character 34), as shown in the example:

```
      <Book>
        Text
      </Book>
```

or

```
      <Book Author="Smith">
        Text
      </Book>
```

A sample address record will show the principle of XML.

```
<person id="p4681">
  <prename>John</prename>
  <lastname>Smith</lastname>
  <title>Dr</title>
  <street>49 Sample Alley</street>
  <zip>EN56J</zip>
  <city>Sampleville</city>
  <birthday>
    <day>28</day>
    <month>March</month>
    <year>1949</year>
  </birthday>
</person>
```

As tags may be used recursively, even complicated structures can be arranged clearly.

Single Elements With No *End*-Tag

These are a special case and look like this:

```
      <Book/>
```

or

```
      <Book Author="Smith" ... />
```

This way, the tag name doesn't need to be repeated, which benefits small CPUs with less memory.

[1] Hans Hubert Partl, Wien, see http://www.boku.ac.at/htmleinf/xmlkurz.html

1.1.2. TiXML: The Difference

Tixi.Com improved XML for Tixi Alarm Modem configuration and got two birds with one stone: The amount of data is cut by 30...40% - while the XML files become much more readable, as there is no similar strings in one line anymore (see example above).

Common XML:
```
<prename>John</prename>
```

Improved TiXML:
```
<prename _="John"/>
```

The legal XML sign _ (underscore) serves as a dummy attribute for a tag that shall be assigned a value to.

The above example, when put in TiXML, looks like this:

```
<person id="p4681">
  <prename _"John"/>
  <lastname _"Smith"/>
  <title _"Dr"/>
  <street _"49 Sample Alley"/>
  <zip _"EN56J"/>
  <city _"Sampleville"/>
  <birthday>
    <day _"28"/>
    <month _"March"/>
    <year _"1949"/>
  </birthday>
</person>
```

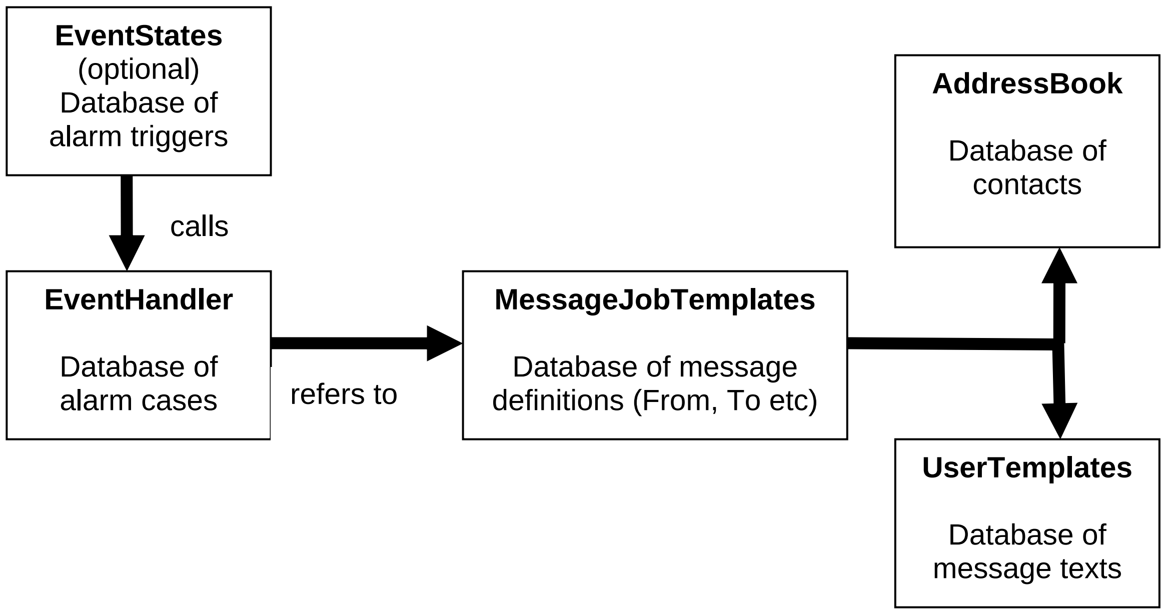
TiXML frames are to be put in square brackets when transmitted to the Tixi Alarm Modem. That way, it realizes the data to be meant for the TiXML processor, e.g.:

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

1.2. Connections Between XML Databases

In order to ease getting into TiXML configuration, this chapter describes the connections between TiXML databases, using a sample event to report.

An alarm configured inside the Tixi Alarm Modem, basically consists of five databases, which are connected as follows:



The connex is to be shown using an alarm edited by the Tixi Alarm Editor (TILA). When in TILA, the alarm wizard is executed, which assigns recipients, message text etc. to the alarm.

This sample alarm is sent via fax to the fire brigade, in case an event "fire" occurs.

At first, we've got an **EventHandler**:

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

The `SendMail` command refers to an entry inside a database **MessageJobTemplates**.

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

This MessageJobTemplate refers to the **AddressBook** with "Myself" and "FireBrigade" contacts, as well as to the **UserTemplate** (message text) "Fire" to send:

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

[<SetConfig _="TEMPLATE" ver="y">
<UserTemplates>
  <Fire>
    <L _="The Barn Is On Fire !!"/>
  </Fire>
  <S-Fire>
    <Subject _="FIRE ALERT !"/>
  </S-Fire>
</UserTemplates>
</SetConfig>]
```

In order to trigger the alarm, the Tixi Alarm Modem needed to receive the command `<DoOn _="Fire"/>]`, where "Fire" is the name of the EventHandler to trigger.

It is possible as well to trigger the alarm via change of a variable, e.g. the digital input port "P0". The TILA alarm wizard does that within the last step, using a database **EventStates**:

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

Via the `<Event _=”Fire”/>` entry, the "Fire" EventHandler (configured above) is called. Therefore, the **EventStates** are the first step of event proccesing.

There is similar connect between other actions to be performed by the Tixi AlarmModem. Therefore, the EventHandler may contain not only the `SendMail` command (see abve), but even such as `Log` or `BinLog` for data logging, `Set` for switching ports and variables or `OnOK` and `OnError` for alarm cascading.

Application of this commands is described inside this tutorial. It contains hints to the TiXML reference manual, which holds detailed description of all commands and parameters.

1.3. Creating Projects

Before you take a closer look at this manual, please check if this documentation and your TICO is made for the software (firmware) used on your Alarm Modem. Due to historical reasons there are two different versions of this document available.

Alarm Modem Firmware starting at version 2.0.0.0 is made for software TILA2 and „TICO for FW 2.0“ (see start menu name), and is not discussed in this manual. Use manual with code TUT2-EN instead.

Alarm Modem Firmware until version 1.72.14 is made for software TILA1 and an older TICO (without suffix “for FW 2.0) and used in this manual.

If you are using a 1.8X Firmware, we recommend to upgrade to FW 2.0.

Before actually starting, we'll create a basic project using TILA first. This will enclose all modem and device settings, contact data, message text, simple alarms and PLC variables.

This basic project is then to be imported into the TICO TiXML Console for accessing even more settings. This holds the advantage of saving a lot of time neccessary for typing basic configuration settings.

For the sake of completeness, chapter 2 describes the TiXML parts relevant for the project. As these are to be configured more easily using TILA, it may be skipped anyway.

1.4. Information Regarding The Sample Projects

This tutorial holds TICO sample projects in order to illustrate TiXML capabilities. The basic project created using TILA will serve as a template therefore.

The `Referenceproject.cnf` consists of the complete range of TiXML features currently available. So it may be used as a reference and glossary of all terms necessary, to be used for your own projects via copy&paste.

Each feature comes with a small sample project to be adapted to your own needs.

1.5. Moving Projects Between TILA And TICO

There is just one-way compatibility between TILA and TICO. A project improved using TICO shouldn't be changed by TILA afterwards. TILA got a capability to import TICO projects, but will skip all complex comfigurations as alarm cascading etc.

In order to get a TILA project into TICO, it is to be transferred via a Tixi Alarm Modem. Therefore, the project is uploaded to the device using TILA, and then downloaded by TICO.

TILA sometimes inserts additional XML parameters for organizing the data (e.g. `value="0"`) which then show up in TiXML sample projects. As these don't affect the Tixi Alarm Modem, this tutorial doesn't describe them.

2. TiXML Databases

TICO displays each database on an own registry sheet. Currently, these are:

```
 1 – Location
 2 – UserData
 3 – AddressBook
 4 – ISP
 5 – UserLogin
 6 – SMS Login
 7 – Lofile Records
 8 – Logfiles
 9 – EventLoggin
10 – EventHandler
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
Holds location data as country, phone number and outside line prefixes.

2 – UserData
Modem specific settings: Redial attempts, MSN, Fax ID and call acceptance.

3 – AddressBook
Contacts holding sender and recipient addresses are stored here, e.g. E-mail address or fax and SMS numbers.

4 – ISP
Internet access data - dialup, username, password etc. and internet time synchonization settings.

5 – UserLogin
Device access data for local and remote protection of the Tixi Alarm Modem.

6 – SMS Login
Extra Login data for remote control via SMS, E-mail and Express-E-Mail.

7 – Logfile Records
Data structures for binary logging are determined inside this database.

8 – Logfiles
Names and sizes of logfiles used for any data logging.

9 – EventLogging
Method and extent of system logging to be defined here., e.g. logging errors striking upon event processing or message dispatch.

10 – EventHandler
Event names and commands assigned upon them.

11 – MessageJobTemplates
This links message texts, and address contacts to alarm messages and defines their type, e.g. SMS or TextFax.

12 – MessageText
What we have here are message texts and subject lines.

13 – ProcessVars
In this database, variables and logical operations may be defined.

14 – EventStates
This holds conditions for specific actions, e.g. which variable change triggers which alarm.

15 – Schedule Conditions
For more complex schedules, whole sets of time definitions (e.g. hildays) can be stored here.

16 – Schedule
The Tixi Alarm Modems scheduler is to be configured here, ahich allow triggering actions upon specific times.

17 – External
For use with PLCs, this database stores variables and communication settings.

18 – SMS-Provider
These are gateways for sending SMS with dialup and protocol definitions.

19 – IncomingSMSProviders
For receiving PSTN SMS, the appropriate service center dialup numbers are saved here.

20 – Incoming CallerID Alarms
As incoming calls may trigger events upon their incoming caller ID, nexus of such numbers and events (e.g. opening a garage door upon mobile call) may be defined here.

3. Basic Configuration

This (`basis.cnf`) provides basic data as location parameters, sender, recipient or internet access. All data are to be set using TILA.

The following parameters are project specific and shall be adapted to your personal conditions.

3.1. Location (TiXML Reference Chapter 3.3)

The phone number of the Tixi Alarm Modem is to be entered here. Skip all country and local prefixes:

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

The area code is set to `30` by default. Change this to your own one:

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

In case the Tixi Alarm Modem is connected to a PABX, an outside line prefix may be defined as well. In case this prefix was `0`, the appropriate lines would look like this:

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

3.2. UserData (TiXML Reference Chapter 3.2)

Name, ID and ISDN MSN (if applicable) is to be stored here:

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

is used, e.g., for fax headlines.

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

will be displayed as sender ID in fax headlines.

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

Determines the MSN of an ISDN Tixi Alarm Modem. This is important if the device shall accept incomming calls for the purpose of remote configuration. The `*` character means "accept calls for any number".

Call acceptance behaviour is determined with the `RingCounter` value. Zero (`0`) means no call acceptance. Any other values defines a number of incoming call signs after which the call is accepted:

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

When using a GSM module, some extra entries are necessary. These will apply after restarting the modem only!
Extension card address that hold the GSM modem:

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

SIM PIN, if necessary:

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

3.3. AddressBook (TiXML Reference Chapter 3.4)

The sender (`MySelf`) and recipient (`Receiver`) contacts are defined in the AddressBook.
For any message type (except CityRuf), it is necessary to provide an address within the MySelf contact.

The E-Mail address, e.g.:
```
<Email _="TAM@freenet.de"/>
```

The SMS number, e.g.:
```
<SMS_No _="01721234567"/>
```

And the SMS provider (recipient only)
```
<SMS_Provider _="Telekom"/>
```
Possible Values: `D1,D1_ISDN,D2,D2_ISDN,Eplus,Telekom,AnnyWay`

Fax number, e.g:
```
<Fax _="+49-30-1234567"/>
```

The Express-E-Mail address, e.g.:
```
<Express-Email _="INFO+49-30-1234567"/>
```

CityRuf number (recipient only), e.g.:
```
<CityRuf _="3949000"/>
```

And the appropriate pager provider:
```
<Pager_Provider _="CityRuf"/>
```

Add as many contacts as you like. Every contact may hold addresses for one, some or all message types:
```
<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>
```

The SMSAnswer contact is to be used later for automatic replies on commands sent by SMS.

3.4. ISP (TiXML-Reference Chapter 3.5)

In the sample project, this database holds an call-by-call internet access by Tixi.Com, which shall be replaced by your own one:

Dialup user data:
```
<PPPUserName _="Freenet"/>
<PPPPassword _="Freenet"/>
```

Outgoing (section `SMTP`) resp incoming (section `POP3`) mail server:
```
<mailserver_ip _="mx.freenet.de"/>
<mailserver_name _="mx.freenet.de"/>
```

POP3 access data for above server:
```
<Username _="TIM-Test"/>
<Password _="TIM-Test"/>
```

If needed, the POP-before-SMTP method (section SMTP) may be used:
```
<Flags _="POPBeforeSMTP"/>
```

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

Deleting mails after retrieval may be disabled:
```
<Flags _="DontDelete"/>
```

3.5. Logfiles (TiXML Reference Chapter 4)

The Tixi Alarm Modem got five logfiles by default, which record system and user processes. Anyway, these must be defined within the file system, as well as the logging to be enabled.

Create logfiles in register `8–Logfiles` (size= in byte):

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

Register `9–EventLogging` configures which extent of data to log. (`mode`: first digit: `e=errors`, `o=OKs`, `a=all`, 2nd digit optional: `v=verbose`)

```
<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>
```

`JobReport` holds info on sent messages.
`Event` lists all events triggered.
`Login` keeps a record of logins and login attempts.
`IncomingMessage` saves info on incoming messages.
`FailedIncomingCall` saves info on problemswith remote control.

3.6. Access control (TiXML-Reference Chapter 3.11)

Local and remote configuration may be protected by user data, which have to be entered on dialup or any configuration attempt.

Username and password, e.g.:

```
<Login>
     <Staff _="Sunrise"/>
</Login>
```

The logion command required therefore would look like this:
```
[<Login _="PAP" user="Staff" password="Sunrise" ver="y"/>]
```

4. Configuring Alarms

There are different ways of getting the Tixi Alarm Modem to send alarm messages:

- by TiXML command via RS232
- by PLC variable via RS232 /422 /485
- by digital input
- by incoming message.

The following chapters configure these options step by step.

4.1. Simple Alarm

Let us begin with a single message. As TILA is capable of configuring this, it may be more simple to use it - but for the purpose of completeness, the TiXML configuration will be explained anyway. Sample project `basis-messages.cnf` got an alarm for any message type. See how a fax is sent, as follows.

As described in chapter 1.2, the Tixi Alarm Modem processes the databases in a specific order: EventHandler – MessageJobTemplates – AddressBook – MessageText

In our example, we'll stick to this order

4.1.1. Event Handler (TiXML Reference Chapter 3.8)

At first, we'll make an EventHandler (register `10-EventHandler`)

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

Its name (here: `"Fax-Alarm"`) is to be chosen freely and used for triggering the alarm.

The EventHandler holds commands to be processed on triggering the appropriate event. In case of a simple alarm, there is a single `SendMail` command only, which, in this case, refers to the `Fax-Alarm` MessageJobTemplate.

4.1.2. MessageJobTemplate (TiXML Reference Chapter 3.7)

Here (register `11-MessageJobTemplates`) we have message type, sender, recipient and text:

```
<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 and Sender are linked to the address book entries created above. Note that there is no `Body` for SMS and CityRuf messages.

4.1.3. MessageText (TiXML Reference Chapter 3.6)

Message Texts are to be found within the UserTemplates (Register `12-MessageText`) and could look like this:

```
<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>
```

For better readability, body and subject of messages are named similarly, where the subject is tagged by an `S-` prefix.

For SMS and CityRuf pager messages, the subject only is used. Its content must not extent 160 chars for SMS resp. 80 chars for pager messages.

Inside the body, any line is tagged by this notation:
`<L _="line of text here"/>`

4.1.4. Testing The Alarm (TiXML Reference Chapter 2.4.9.1)

A simple alarm is complete now. after uploading the project to the Tixi Alarm Modem, the alarm may be triggered using a DoOn command, e.g. `[<DoOn _="FaxAlarm"/>]`.

4.2. Improved Alarm Features

Alarm may be triggered by a PLC variable or digital input port of the Tixi Alarm Modem, use alarm cascading or even wait for confirmation.

The processing of databases, as described in chapter 1.2, is therefore extended:
EventStates – ProcessVars – EventHandler – MessageJobTemplates – AddressBook - MessageText

This chapter will improve the existing fax alarm step by step.

4.2.1. Enhancing Message Texts

4.2.1.1. Signatures (TiXML Reference Chapter 3.6)

A big plus of XML is the capability of linking contents by reference. For instance (`basis-fax-signature.cnf`) there is a text and location data to be appended to any message. This saves memory, and changes of the signature apply to all messages linked to it.

The signature:

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

Is to be inserted using the `Include` command:

```
<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>
```

At the recipient's site, this message shows up:

```
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 With Variables (TiXML Reference Chapter 12)

Message texts may be extended by variables as well, e.g. system data (timestamps, serial number etc.) or or values of I/O ports and PLC variables. (sample `basis-fax-textvariables.cnf`)

Note: The sample project requires a Tixi Alarm Modem with I/O extension card. Without such, the line with reference to port 1 must be removed.

```
<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 Chapter 3.8)

Up to now, we had one SendMail command per EventHandler. But we may even have more, so that several recipients can be notified, using several message types.
(Sample `basis-multiplesendmail.cnf`)

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

Every `SendMail` command refers to a specific MessageJobTemplate, one for fax and one for E-mail messages - which then both use the same message text.

4.2.3. Port Triggers Alarm

4.2.3.1. The Service-Button (TiXML Reference Chapter 6.7)

The service button at the device's rear may be used to "manually" trigger alarms. (`basis-fax-ServiceButton.cnf`), or even to confirm incoming alarm messages (see chapter 4.2.5.5).
This is configured via a special `System` EventHandler (Register `10-EventHandler`)

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

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

In order to link alarms to digital input ports, the EventStates database is used.
(`basis-messages-ports.cnf`)

This database links an EventHandler to a process variable. In this case, the existing Fax-Alarm EventHandler shall be assigned to the input port P0 at an extension card:

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

The `Enabled` parameter enables or disables EventStates.
If value is `TRUE`, the event will be processed upon the condition being fulfiled, while when FALSE it' ignored.

The Event parameter determines which event to trigger upon the condition being fulfiled.

The ProcessVar Parameter is to be configured in two possible ways:

1. **Directly (fast but not so flexible)**

Therefore, we'll use port address and value (which is our condition to fulfil) inside the EventState:

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

This alarm is assigned to an input port P0 at an extension card. The `low` parameter determines the condition being fulfiled if the value changes from 1 to 0. Alternatively, one may use `high` (change from 0 to 1) or `both` (any change).

2. **By ProcessVars Database (slow but flexible)**

In this case, the `ProcessVar` parameter refers to an entry inside the ProcessVars database (see next chapter).

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

4.2.3.3. ProcessVars (TiXML Reference Chapter 6.2)

The ProcessVars database (register `13-ProcessVars`) defines conditions for EventStates (`basis-messages-ProcessVars.cnf`).

Logical Operations
In this example, the condition for the Fax-Alarm is met if input port 0 changes from 0 to 1:

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

Instead of the LD command, many others may be used as well, e.g. LDN, ST or operations for linking more than one port. In this case, the E-mail alarm is triggered if input ports 1 and 2 change from 0 to 1:

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

Comparisons
Beside logical operations, even comparisons are possible. This example triggers an Express-E-Mail, but basically, comparisons do make more sens for variables other than boolean, e.g. PLC registers.

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

The condition is fulfiled if `v1` > `v2`. For `v2`, even an actual value may be entered.

Time Comparison
Another condition for process variables is time, i.e. an alarm is triggered upon a specific time being reached. Time and/or date may be used. This example daily sends an SMS when the given time is matched, which is 14:20.

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

One may combine this with logical operations, e.g. to provide more than one time span:

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

For complex schedules, use the scheduler database, which is described below.

4.2.4. Scheduler (TiXML Reference Chapter 7)

The Scheduler (Register `16-Schedule`) can hold complex time switching plans. Times may be combined and disabled as well (Holidays, Register `15-Schedule Conditions`, `basis-fax-scheduler.cnf`)

This Scheduler sends the fax alarm on weekdays at 18:00, which could e.. serve as a daily report of something:

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

The event to trigger (here: `Fax-Alarm`) is to be provided as an parameter inside the first line of the entry.

4.2.4.1. Shift Plan

The sample project `basis-fax-Shift.cnf` shows the usage of the Sheduler for notifying different recipients at different times. We assume the day shift to start at 09:00 when `Receiver` gets the message, and the night shift at 18:00 when `Shift` shall be notified. At first, we create the night shift recipient inside the address book:

```
 <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>
```

Furthermore, we do need a process variable to write the current recipient into (Register `13-ProcessVars`)

```
<Shiftworker/>
```

This is gonna be changed using two EventHandlers, which write the contact name of the current recipient into that variable:

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

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

These EventHandler are called by the appropriate Scheduler entries at shift change:

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

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

For the alarm using the value of the process variable as the recipient, the appropriate MessageJobTemplate holds a reference to this variable at the `Recipient` entry:

```
<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>
```

After resolving `&#xae;/Process/PV/Shiftworker;`, the Tixi Alarm Modem will know the proper recipient by what time it is:

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

The `Receiver` given in the MessageJobTemplate separated by comma, is an alternate value to be used if the process variable is empty (e.g. after restarting the modem between shift changes).

But the variable may be initialized after restart by a Set command:

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

4.2.4.2. Scheduler Conditions (Holidays)

Scheduler functions often do need exception handling, e.g. when holidays strike.
See `15-Schedule Conditions` therefore.

As an example, we'll use the shift plan configured above, in which the german public holidays are to be insertd. At these days, the night shif recipient Shift shall be alerted even during daytime. (`basis-fax-shift-condition.cnf`):

To achieve this, the shift change shall be disabled at the specific days. Therefore, this condition is inserted into the Scheduler:

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

It refers to the Conditions database:

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

The `not` parameter means that the entry is not valid at the given days.

We could put this even into the scheduler directly:

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

The advantage of using the `Conditions` databse is that the conditions defined there may be used for more than one scheduler entry and can be changed one-for-all.

4.2.5. Advanced EventHandler Features (TiXML Reference Chapter 3.8.1)

Up to now, we used the EventHandler for plainly sending messages only.
But it is capable of much more:

4.2.5.1. Priorities

Each `SendMail` command may be assigned a priority. This is used to sort outgoing messages if they occur more than one at a time, so the message with higher priority is sent befor the one with lower priority (1 is lowest, 255 highest).
So if an fire alarm with priority=10 is triggered at a time when some other messages with lower priority still are in the queue, the fire alarm is to be sent first nonetheless.
(See `basis-messages-priorities.cnf`)

The priority is defined inside the `SendMail` command as follows:

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

4.2.5.2. Alarm Sets Port - The `Set` Command

By Set command, output ports may be switched. (TiXML Reference Kap. 6.5.3) This command can be implemented into an EventHandler. (`basis-fax-setport.cnf`)
In our example, beside sending the message we'll even switch a port 0 from 0 to 1 (e.g. for activating a warning light)

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

4.2.5.3. Alarm Cascading: `OnError`

Basically, the Tixi Alarm Modem uses redial attempts. But these will be of little use in case that a recipient just isn't available.
Therefore, it provides alarm cascading. In case a message cannot be transmitted properly, it becomes sent to another recipient.
(`basis-kaskade.cnf`):

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

In this example, there are two attempts of sending the message (MaxRepeat=1 means one redial after 180sec). If both attempts fail, the "Email-Alarm" event is triggered.
This event could be fitted with some cascading as well. It could even refer back to "Fax-Alarm", so it is creating a loop that just breaks upon successful transmission.

4.2.5.4. Confirmation By Port: `OnOK`

Beside `OnError`, there is even `OnOK`, which may switch ports in connection to successful receipt of messages. (`basis-kaskade.cnf`)
In this example, in case of an alarm an output port is to be set and, after successfully sending the message, the `SetPort` event is used to reset the port.

```
<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>
```

The `OnOK` and `OnError` commands may be used together in an `SendMail` command.

4.2.5.5. Confirmation Of Messages

Even more security is granted by sending messages that require an active confirmation, which guarantee that a message actually was noticed by the recipient. Confirmations may be sent via SMS, Express-E-Mail, Internet E-mail or via variable changes.
(`basis-confirmation.cnf`)

The EventHandler looks like this:

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

The appropriate message text:

```
<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>
```

Additionally, a special `System` EventHandler is necessary:

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

The confirmation feature utilizes a ConfirmID (e.g. "99") and a unique fingerprint code which is generated automatically. Each alarm needs its own Confirm-ID. Both is used as reference and integrated into the message and just needs to be sent back by the recipient.

The OnTimeout command determines what to do after the given timeout expired without a confirmation being received. In this example, the same event is triggered again - over and over again until the confirmation arrives. It would be possible to even notify someone else after the timeout expiring.

4.2.5.6. Handing Over Of Parameters

If an alarm is to be triggered by `DoOn` command, it may be handed over parameters, which then can be inserted into message texts. (`basis-fax-parameter.cnf`)

For example, we'll send a fax in which a room number is inserted dynamically:

```
<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>
```

The parameter `Room` is inserted by the `&#xae;~/Room;` reference.

The appropriate `DoOn` command must look like this:

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

While the resulting fax message would then be as follows:

```
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
```

The number of parameters handed over is infinite. The parameters may be used even to set variables. Hence, an EventHandler with the `Set` command could set a port to the value handed over:

`DoOn` command:

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

EventHandler:

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

4.2.5.7. Logging Data (TiXML Reference Manual Chapter 4)

Beside the logfiles mentioned above, even custom ones may be defined. These could be XML or (space-saving) binary logfiles.

In this example, the port values shall be logged frequently:

XML Logging (`basis-portlog.cnf`):

At first, we need to define the logfile `PortLog` and its size:

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

  <PortLog size="10240"/>

</LogFiles>
```

This EventHandler writes the I/O port values into the logfile:

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

The parameter `PortLog` is the name of the logfile, followed by the values to log (here: a reference to port 0). The amount of logged data is infinite.

Binary Logging (`basis-portbinlog.cnf`):

For binary logging, we need to define a binary logfile with the desired ring buffer size:

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

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

</LogFiles>
```

The logfile refers to a record (register `7-LogfileRecords`) which defines the data structure::

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

This provides the type of data (integer “int“, alternative: “string“), its size (1=1 Byte) and the value to log (via reference to input P0).

Then, the `BinLog` command must be used inside the EventHandler, providing the logfile name:

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

Logging Data By Schedule:

For the Logging feature making sense, it shall be called via the Scheduler (Chapter 4.2.4) periodically, e.g. every five minutes:

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

As all logfiles are ring buffers, the logging system is overflow-proof.

Sending Logged Data:

Logfiles may be included into message texts using the `IncludeLog` command, thus e.g. being transmitted to a service center frequently.
(`basis-portbinlog-email.cnf`)

Message text:

```
<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>
```

With the IncludeLog command, the logfile name and range of entries to log are defined. In this example, entries of the past 24 hours are to be sent. The range values are described in detail within the TiXML Reference chapter 2.4.9.2.

Sending Data As MIME Attachment:

Often, the recipient will make little use of the XML format which the data comes in. Much more usable is the CSV format (e.g. to import into MS Excel™). The IncludeLog command is used to format data at will before sending, e.g. removing IDs, timestamps or adding start, end and separation characters. The logfile may be put into an attachment with random name.
(`basis-portbinlog-email-attachment.cnf`).

The message text is to be formatted in compliance to standards of E-mail attachments. Therefore, "Boundary" and blank lines are inserted at certain points, whil enable the receiving E-mail client to divide attachment from message text.

```
<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 >
```

At `filename=&quot;data.csv&quot;` the `data.csv` may be replaced by any filename, or even by reference e.g to the current date: `filename=&quot;&#xae;/TIMES/DATE;.csv&quot;`

Detailed info on TiXML formatting is to be found within TiXML-Reference chapter 4.8.

Additionally, we need to extend the MessageJobTemplate by the `Boundary` parameter:

```
<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. Internet time synchonization / Summer-Winter-Time (TiXML-Reference Chap. 3.12)

The RTC of the Tixi Alarm Modems may be synchonized with an internet time server, e.g. the german atom clock. This feature may also be used for automated Summer-Winter time changes.

Two different time server protocols are supported:
TIME offers the UTC time with a standard string
DAYTIME offers the local time with a non-standard string

With the TIME protocol the time zone calculation (e.g. +0100 for Germany) must be done by the device itself. The Summer-Winter time is also not included.

With the DAYTIME-Protokoll the server offers the local valid time including time zone and Summer-Winter time. The Alarm Modem has to know the format of the time string. The TimeZone and TimeDiff parameter has to be set to +0000 because the server already offers the calculated time.

The time string of an internet time server may be requested via TCP/IP port 13.
If connected to the internet, following command prompt command may be used:

`telnet Server-URL 13`, e.g.:
`telnet ptbtime1.ptb.de 13`

Server answers e.g..:
```
10 JUN 2004 10:24:16 METDST
```

The example `basis-timeserver.cnf` shows the usage of the german atom clock DAYTIME-server for automated Summer-Winter time changes using the scheduler (see chapter 4.2.4).

Timeserver settings
At register `4-ISP` the URL of the Timeserver, the protocol, the time difference and the time string (optional) is configured:

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

TimeZone
At Register `2-UserData` the TimeZone iss et to +0000:
```
<TimeZone _="+0000"/>
```

EventHandler
The EventHandler `SyncTime` on Register `10-EventHandler` calls the command `INetTime` for internet time synchronization.

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

Scheduler
The time synchronization must be called by the Alarm Modem scheduler at an usefull time, e.g. every month:

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

For a Summer- Wintertime change, the modem has to synchronize the time at a spezial time. In Europe its on the last Sunday in march and october. Following scheduler entries are necessary:

```
<Summertime _="SyncTime">
  <Month _="3"/>                     March
  <Day _="25-31"/>                   least 7 days
  <Weekday _="Su"/>                  on sunday
  <Time _="02:00"/>                  at 2 o’clock
</Summertime>

<Wintertime _="SyncTime">
  <Month _="10"/>                    October
  <Day _="25-31"/>                   last 7 days
  <Weekday _="Su"/>                  on sunday
  <Time _="03:00"/>                  at 3 o`clock
</Wintertime>
```

4.3. Remote Control By Incoming Messages (R. Chapter. 8)

The Tixi ALarm Modem may even receive E-mail, Express-E-Mail or SMS and process commands enclosed in these messages, e.g to set ports or PLC variables. The example utilizes SMS. (`basis-remoteswitch.cnf`):

4.3.1. Access Protection

At first we'll configure an access protection at register `6-SMS_LogIn`, in order to let authorized personnel only control the device:.

Simple Access Protection:
We'll use the word `sesame` as a password:

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

Access Protection With Sender Address Verification (SMS and Express-E-Mail)
If using the `OA_` parameter along with e.g. a number instead of `Default`, for instance:

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

Then, additionally to the password, even the sender number is checked, i.e. remote control is possible via the given number only.

Access Protection With Sender Verification (Internet E-Mail)
On Internet E-mail, the `Alpha` parameter (E-Mail Alias) is evaluated.
The sender ID `Tixi Support` results in this entry, where blanks are replaced by underscores:

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

4.3.2. Simple Remote Control:

With the given password, any event configured in the device can be triggered. Therefore, the incoming message needs to have the password and event name inside its subject line:

```
<Password> <Eventname>
```
z.B.

```
TIXI SETPORT
```

In this example, the password is `TIXI` and the event to trigger is `SETPORT`.
The appropriate EventHandler could look like this:

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

Upon receive of the SMS given above, the output port P0 is set to 1..

4.3.3. Remote Control With Handing Over Of Parameters

In order to remotely hand over parameters to an event, the subject of the incoming message must hold the password, event name and parameters:

```
<Password> <Eventname> <Parameter1> <Parameter2> ... <Parameter6>
```
z.B.

```
TIXI SETVALUE 1
```

In this example, the password is `TIXI` and the event to trigger is `SETVALUE` and the parameter `1` is handed over to the event.

The appropriate EventHandler could look like this:

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

The parameter is inserted via the `&#xae;~/P1;` reference (for more parameters, count up to `P6`) is handed to the `Set` command as new value for port P1, which is then set to 1.
With an SMS `TIXI SETVALUE 0`, the port could be set to 0 again.

4.3.4. Supervising Remote Control

Often it is useful to extend remote control by reply messages, which notify upon success or failure. The event therefore will be added an `SendMail` command to be executed after the `Set` command. (`basis-remoteswitch-answer.cnf`)
For our example, we'll use an Tixi Alarm Modem GSM.

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

4.3.4.1. Static Reply

As an confirmation of an action, a message to an addressbook contact shall be created. The MessageJopTemplate looks like this:

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

The message text may hold an reference to the name of the event executed:

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

4.3.4.2. Dynamc Reply

In order to make remote control more dynamic, the reply may be sent to the sender of the remote control command message. (Sample event: SETVALUE)

The MessageJobTemplate doesn't refer to a specific addressbook contact but contains a reference to the sender ID of the incoming message.

It looks like this:

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

Note that the reference type varies with message type
GSM-SMS: `&#xae;~/OA;`
E-Mail or Express-E-Mail: `&#xae;~/Alpha;`

Important notice on receive of SMS on PSTN lines (ISDN/V90)
This service is only available in Germany, Austria and Italy! In other countrys you'll have to use GSM devices.

4.3.4.3. Reply With Value Check

For even more security, the reply may hold the value of the variable just changed. This allows to check if the `Set` command was executed successfully.
To have the actual value inside the message, a small delay needs to be inserted into the EventHandler between `Set` and `SendMail` commands, as the fast processing otherwise would result in having the old value inserted.

The EventHandler:

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

The replymessage text containing the current value of 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. Behaviour In Case Of Error

Two error cases may be assumed:

- invalid password
- invalid event

Both cases are intercepted by so-called System-Events within the EventHandler Database.
(`basis-remoteswitch-errors.cnf`):
Note that these events are contained inside a `System` sub-section of the EventHandler database.

4.3.5.1. Invalid Password

In case an incoming message holds an invalid password, the `SMSInvalidPassword` (SMS) resp. `TixiInvalidPassword` (Express-E-Mails) oder „`POPInvalidPassword`“ Internet E-Mails is executed automatically.

These System events may hold any EventHandler command, e.g `Set` or `SendMail` commands.

A useful application would be logging the sender address and text, along with sending a message "SMS with invalid password received“:

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

For Internet and Express-E-Mails, the `Alpha` parameter (sender ID) shall be logged as well.

4.3.5.2. Invalid Event

In case an incoming message holds an invalid event, the `SMSInvalidEvent` (SMS) resp. `TixiInvalidEvent` (Express-E-Mails) oder `POPInvalidEvent` Internet E-Mails is executed automatically.

These System events may hold any EventHandler command, e.g `Set` or `SendMail` commands.

A useful application would be logging the sender address and text, along with sending a reply message "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>
```

For Internet and Express-E-Mails, the `Alpha` parameter (sender ID) shall be logged as well.

The error message shall be sent back to the sender of the errorneous message, as described in chapter 4.3.4.2, and shall give hint on the problem occured:

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

4.3.5.3. Logging Informationen In Case Of Success

When extending the EventHandler by a `Log` command, even in case of success some useful data may be logged, e.g. sender number and text of the remote control message.

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

For Internet and Express-E-Mails, the `Alpha` parameter (sender ID) shall be logged as well.

4.3.6. Query for Emails

To process incoming emails the alarm modem has to query a POP3 account first. Therefore an EventHandler with a “POP3Query” command is necessary, e.g.:

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

This EventHandler should be called by scheduler or by a CallerID-Trigger (Chapter 4.4) to use it with our NewMailSignal-Service (see: http://www.NewMailSignal.com)

4.4. Event By Incoming Call (CallerID) R. Chapter 8.2

Even a simple phone call may trigger an event by its sender number transmitted, e.g. to open a garage door. (`basis-calltrigger.cnf`)

Phone number and assigned event are to be defined at the register
`20-Incoming CallerID Alarms`

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

“`0307654321`” is the calling number, event refers to the EventHandler to execute, which could e.g. then set an output port.

Different numbers may be assigned different events to. Therefore, the `No`*X* parameter must be counted consecutively.

5. Connection To A PLC (TiXML PLC Manual)

The examples up to here utilized digital I/Os of the Tixi Alarm Modem to trigger alarms, perform switching actions or dynamically arrange message texts. Instead of these, even PLC variables may used for data loging, alarm triggers or remote control.
Hence, replace the I/O port references inside the examples by such referring to PLC variables (e.g. `/Process/Aux1/D0/Register1`).
The more complex PLC database(register `17-External`) shall be created using TILA, for the sake of plainness.

The `basis-sps.cnf` sample shows alarm features with text variables, data logging and remote control by SMS, along with a PLC.

6. More Features

The capabilities described in this tutorial will cover most of the applications.

For more and complex features and improvement of the examples shown, we recommend studying the TiXML reference manual, e.g. chapters on

Logic And Comparison Operations
Webserver
Data Formatting
Scheduler-Conditions
Variable Formatting
etc. pp.