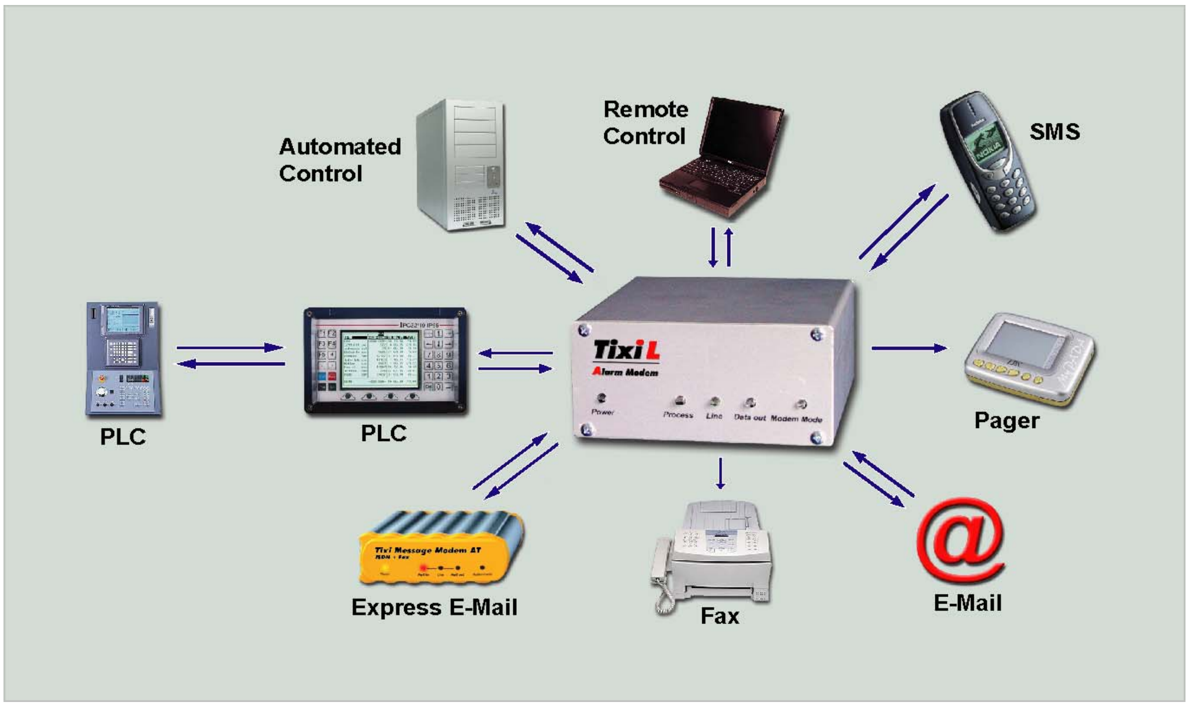
TUT2-EN



TiXML-Tutorial

For Firmware 2.2.12.0

Publication close: 29. May 2007, Supported Firmware Version 2.1.22.0

Table of Contents

1. INTRODUCTION 3
- 1.1. XML VS. TIXML SYNTAX 3
  - *1.1.1. What is XML ? 3*
  - *1.1.2. TiXML: The Difference 4*
- 1.2. CONNECTIONS BETWEEN XML DATABASES 5
- 1.3. CREATING PROJECTS 7
- 1.4. INFORMATION REGARDING THE SAMPLE PROJECTS 8
- 1.5. MOVING PROJECTS BETWEEN TILA2 AND TICO 8

2. TIXML DATABASES 9

3. BASIC CONFIGURATION 11
- 3.1. LOCATION (TIXML REFERENCE CHAPTER 3.3) 11
- 3.2. USERDATA (TIXML REFERENCE CHAPTER 3.2) 11
- 3.3. ADDRESSBOOK (TIXML REFERENCE CHAPTER 3.4) 12
- 3.4. ISP (TIXML-REFERENCE CHAPTER 3.5) 12
- 3.5. LOGFILES (TIXML REFERENCE CHAPTER 4) 13
- 3.6. ACCESS CONTROL (TIXML-REFERENCE CHAPTER 3.11) 14

4. CONFIGURING ALARMS 14
- 4.1. SIMPLE ALARM 14
  - *4.1.1. Event Handler (TiXML Reference Chapter 3.8) 15*
  - *4.1.2. MessageJobTemplate (TiXML Reference Chapter 3.7) 15*
  - *4.1.3. MessageText (TiXML Reference Chapter 3.6) 15*
  - *4.1.4. Testing The Alarm (TiXML Reference Chapter 2.4.9.1) 15*
- 4.2. IMPROVED ALARM FEATURES 16
  - *4.2.1. Enhancing Message Texts 16*
    - 4.2.1.1. Signatures (TiXML Reference Chapter 3.6) 16
    - 4.2.1.2. Text with Variables (TiXML Reference Chapter 13) 17
  - *4.2.2. Multiple Sendmail (TiXML Reference Chapter 3.8) 17*
  - *4.2.3. Port Triggers Alarm 17*
    - 4.2.3.1. The Service-Button (TiXML Reference Chapter 6.7) 17
    - 4.2.3.2. I/O-Ports (TiXML Reference Chapter 6.3) 18
    - 4.2.3.3. ProcessVars (TiXML Reference Chapter 6.2) 18
  - *4.2.4. Scheduler (TiXML Reference Chapter 7) 21*
    - 4.2.4.1. Shift Plan 21
    - 4.2.4.2. Scheduler Conditions (Holidays) 24
  - *4.2.5. Advanced EventHandler Features (TiXML Reference Chapter 3.8.1) 24*
    - 4.2.5.1. Priorities 24
    - 4.2.5.2. Alarm Sets Port - The `Set` Command 25
    - 4.2.5.3. Alarm Cascading: `OnError` 25
    - 4.2.5.4. Confirmation By Port: `OnOK` 25
    - 4.2.5.5. Acknowledge of Messages 26
    - 4.2.5.6. Handing over of Parameters 27
    - 4.2.5.7. Logic operations and calculations 28
    - 4.2.5.8. Logging Data (TiXML Reference Manual Chapter 4) 29
    - 4.2.5.9. IF-Condition (TiXML-Reference Manual Chapter 3.8.2) 30
    - 4.2.5.10. Internet time synchronization / Summer-Winter-Time (TiXML-Reference Chap. 3.12) 31
  - *4.2.6. Sequencer (TiXML-Reference Manual Chapter 8) 32*
- 4.3. REMOTE CONTROL BY INCOMING MESSAGES (R. CHAPTER. 9) 33
  - *4.3.1. Access Protection 33*
  - *4.3.2. Simple Remote Control: 34*
  - *4.3.3. Remote Control with Handing Over Of Parameters 34*
  - *4.3.4. Supervising Remote Control 34*
    - 4.3.4.1. Static Reply 35
    - 4.3.4.2. Dynamic Reply 35
    - 4.3.4.3. Reply With Value Check 36
  - *4.3.5. Behaviour in case of error 36*
    - 4.3.5.1. Invalid Password 36
    - 4.3.5.2. Invalid Event 37
    - 4.3.5.3. Logging information in case of success 37
  - *4.3.6. Query for Emails 37*
- 4.4. EVENT BY INCOMING CALL (CALLERID) R. CHAPTER 9.2 38
- 4.5. PROJECT UPLOAD VIA EMAIL (TIXML-REFERENCE CHAPTER 3.8.1) 38

5. CONNECTION TO A PLC (TIXML PLC MANUAL) 39

6. MORE FEATURES 39

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 style sheets 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>
```

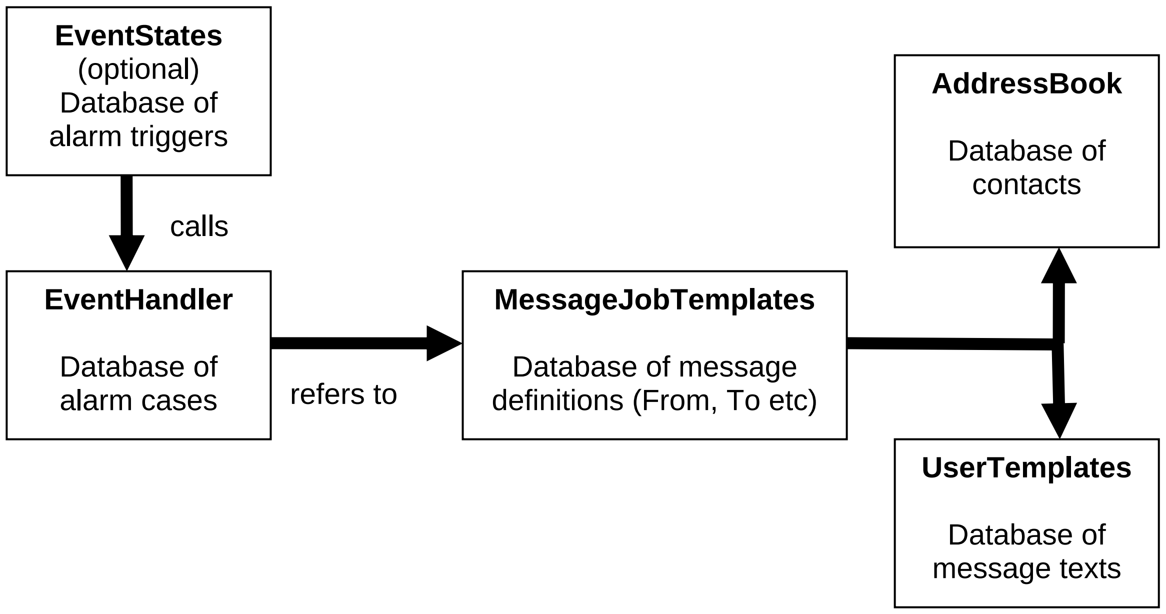
TiXML frames are to be put in square brackets when transmitted to the Tixi Alarm Modem. That way, it realizes the data to be meant for the TiXML processor, e.g.:

```
[<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 connection is to be shown using an alarm edited by the Tixi Alarm Editor (TILA2). When in TILA2, 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 _="+49-30-24037497"/>
    </MySelf>
    <FireBrigade>
      <Fax _="+49-30-10"/>
      <SMS_No _="+49-176-24434569"/>
      <SMS_Provider _="AnnyWay"/>
    </FireBrigade>
    ...
</AddressBook>
</SetConfig>]

[<SetConfig _="TEMPLATE" ver="y">
<UserTemplates>
  <Fire>
    <Subject _="FIRE ALERT !"/>
    <Body>
      <E _="The Barn Is On Fire !!"/>
    <Body>
  </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" using a database **EventStates**:

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

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

There is similar connect between other actions to be performed by the Tixi Alarm Modem. Therefore, the EventHandler may contain not only the `SendMail` command (see above), 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 these 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 until version 1.72.14 is made for software TILA1 and an older TICO (without suffix “for FW 2.0) and is not discussed in this manual. Use manual with code TUT-EN instead.

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 mainly used in this manual. Some of the described functions require at least FW 2.2.12.0. A complete compatibility list can be found in the TiXML-Reference Manual chapter “Firmware”.

If you are using an older firmware than 2.2.12.0, we recommend upgrading to the latest version.

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 necessary 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 examples are installed along with TICO, directory “Examples”. The “basic” project created using TILA2 is used as a template for all projects.

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

At first modify the “basic” project according to your needs (phone settings, address book) and distribute these settings to all other projects using the “Distribute Settings” entry within the TICO start menu program group.

All projects are written for HM Alarm Modems but may be also used with other Alarm Modems. If you are using a GSM modem (HG) some settings must be changed:

- AddressBook SMS-Provider should be “GSM”
- MessageJobTemplate type should be “GSMSMS” instead of “SMS”

1.5. Moving Projects between TILA2 and TICO

There is just one-way compatibility between TILA2 and TICO. A project improved using TICO shouldn't be changed by TILA2 afterwards. TILA2 got a capability to import TICO projects, but will skip all complex configurations that it can not interpret.

In order to get a TILA2 project into TICO, it is to be transferred via a Tixi Alarm Modem. Therefore, the project is uploaded to the device using TILA2, and then downloaded by TICO (menu: project -> open from TAM).

If you previously used TILA1 or the old TiXML-Tutorial (for firmware until 1.72.14.0) you will notice that the TiXML tags created by TILA2 will use static names (e.g. Contact_0, Switch_0 etc.) instead of user defined names (as still shown in chapter 1.2). Therefore projects are more difficult to read but this change was necessary for the flexibility of TILA2. The user gets his self defined names displayed by the “name” attribute. The firmware ignores this information.

TILA2 inserts additional XML parameters to the TiXML code and to the register “30-TILA2” to organize the data. As these don't affect the Tixi Alarm Modem, this tutorial doesn't describe them and they are written *italic*.

2. TiXML Databases

TICO displays each database on an own page which can be selected within the ProjectView list. It’s possible to add/remove pages via menu or context menu.

```
 1 – Location
 2 – UserData
 3 – AddressBook
 4 – ISP
5/6 – AccRights
7/8 – LogDefinition
 9 – EventLogging
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
21 - Sequencer
…
30 – TILA2
```

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 synchronization settings.

5/6 – AccRights
Device access data for local, remote and remote control protection of the Tixi Alarm Modem.

7/8 – LogDefinition
Names and sizes of logfiles used for any data logging as long as data structures for binary logging are determined inside this database.

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. holidays) can be stored here.

16 – Schedule
The Tixi Alarm Modems scheduler is to be configured here, which 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 centre 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.

21 – Sequencer
The Sequencer is used to set values out of a time table.

21-29 – advanced settings
These settings are used for special applications and are not part of this tutorial. They are described in the TiXML-Reference-Manual.

30 – TILA2
This register is used by TILA2 to organize its data. Since you will not modify TICO projects using TILA2, these data can be ignored.

3. Basic Configuration

This (`basis.cnf`) provides basic data as location parameters, sender and recipient or internet access. All data are to be set using TILA2.

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 value defines a number of incoming call signs after which the call is accepted:

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

SIM PIN, if necessary:

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

3.3. AddressBook (TiXML Reference Chapter 3.4)

The sender (`MySelf`) and recipient (`Receiver`) contacts are defined in `3-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 _="Tixi-Training1@gmx.net"/>
```

The SMS number, e.g.:
```
<SMS_No _="+49-172-1234567"/>
```

And the SMS provider (recipient only)
```
<SMS_Provider _="AnnyWay"/>
```
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:

```
<Contact_0>
     <INet _="info@tixi.com"/>
     <Fax _="+49-30-40608400"/>
     <Express-Email _="INFO+49-30-40608555"/>
     <SMS_No _="03040608300"/>
     <SMS_Provider _="AnnyWay"/>
     <CityRuf _="3949000"/>
     <Pager_Provider _="CityRuf"/>
</Contact_0>
```

The “OA” contact of some projects 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, the database “`4-ISP`” holds an call-by-call internet access which shall be replaced by your own one:

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

Outgoing (section `SMTP`) and incoming (section `POP3`) mail server:
```
<mailserver_name _="mail.gmx.net"/>
```

POP3 access data for above server:
```
<Username _="Tixi-Training1@gmx.net"/>
<Password _="xyz"/>
```

If needed, the POP-before-SMTP or ESMTP method (section SMTP) may be used. For ESMTP additional user and password tags have to be added (similar to POP3):
```
<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 `7/8–LogDefinition` (size= in byte):

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

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 problems with 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. Users may be added to register “`5/6-AccRights`”:

Username and password, e.g.:

```
<AccRights>
     <Groups>
          <Admin>
               <LocalLogin AccLevel="1"/>
               <RemoteLogin AccLevel="1"/>
          </Admin>
     </Groups>
     <User _="Plain">
          <Def_LocalLogin Plain="Sunrise" Group="Admin"/>
          <Def_RemoteLogin Plain="Sunset" Group="Admin"/>
     </User>
</AccRights>

<Login>
     <Technician _="Sun"/>
</Login>
```

The local login command would look like this:
```
[<Login _="PAP" password="Sunrise" ver="y"/>]
```

The local login command would look like this:
```
[<Login _="PAP" password="Sunset" 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.
- by scheduler

The following chapters configure these options step by step.

4.1. Simple Alarm

Let us begin with a single message. As TILA2 is capable of configuring this, it may be simpler 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`)

```
<Alarm_0 Name="Fax-Alarm">
      <SendMail _="MessageJobTemplates/Alarm_0"/>
</Alarm_0>
```

Its <tag> name (here: `"Alarm_0"`) is theoretically to be chosen freely and used for triggering the alarm. We recommend using the TILA2 syntax “Alarm_X”.
The tag name is used to trigger the alarm via `DoOn` command.

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 `Alarm _0` MessageJobTemplate.

4.1.2. MessageJobTemplate (TiXML Reference Chapter 3.7)

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

```
<Alarm_0 _="TextFax">
      <Recipient _="AddressBook/Contact_0"/>
      <Sender _="AddressBook/MySelf"/>
      <Subject _="&#xae;/D/UserTemplates/Message_3/Subject;"/>
      <Body _="/D/UserTemplates/Message_3/Body"/>
</Alarm_0>
```

Recipient and Sender are linked to the address book entries created above. Body and subject refers to the UserTemplates database
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:

```
<Message_3 Name="Fax-Alarm" Type="Mail" UseSignature="0">
      <Subject _="This is the subject line of the Fax."/>
      <Body>
            <E _="This is the message body of the Fax."/>
            <E _="Enter as many lines as you need."/>
      </Body>
</Message_3>
```

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

Inside the body, any line is tagged by this notation:
`<E _="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 _="Alarm_0"/>]`.

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 acknowledge.

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 advantage of XML is the capability of linking contents by reference. For instance (`basis-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. TILA2 already offers this feature.

The signature:

```
<LocationText hidden="1">
     <Email>
          <E _="Signature of the Alarm Modem"/>
     </Email>
     <SMS _="Modem Signature"/>
</LocationText>
```

Is to be inserted using the `Include` command:

```
<Message_0 Name="Fax-Alarm" Type="Mail" UseSignature="1">
     <Subject _="This is the subject line of the fax."/>
     <Body>
          <E _="This is the body of the fax."/>
          <E _="Enter as many lines as you need."/>
          <E _=""/>
          <E _="This is the signature :"/>
          <E _="----------------------------------------"/>
          <Include _="/D/UserTemplates/LocationText/Email"/>
     </Body>
</Message_0>
```

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

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

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

This is the signature :
----------------------------------------
Signature of the Alarm Modem
```

For SMS and CityRuf this signature is included via reference:

```
<Message_1 Name="SMS-Alarm" Type="SMS" UseSignature="1">
     <Subject _="This is the message of the SMS Alarm. This is the
     signature: / &#xae;/D/UserTemplates/LocationText/SMS;"/>
</Message_1>
```

4.2.1.2. Text with Variables (TiXML Reference Chapter 13)

Message texts may be extended by variables as well, e.g. system data (timestamps, serial number etc.) or values of I/O ports and PLC variables. TILA2 already offers some of these variables. (example `basis-fax-textvariables.cnf`)

Note: The sample project requires a Tixi Alarm Modem with I/O extension. Without such, the line with reference to port I00 must be removed.

```
<Message_0 Name="Fax-Alarm" Type="Mail" UseSignature="0">
     <Subject _="This is the subject line. Port I00:
          &#xae;/Process/MB/IO/I/P0;"/>
     <Body>
          <E _="This is the message body."/>
          <E _="You can add variables to the text:"/>
          <E _="Date&amp;Time: &#xae;/TIMES/RFC822Date;"/>
          <E _="Firmware: &#xae;/OEM/Firmware/Version;"/>
          <E _=""/>
     </Body>
</Message_0>
```

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_0 Name="Fax-Alarm">
     <SendMail _="MessageJobTemplates/Alarm_0"/>
     <SendMail _="MessageJobTemplates/Alarm_1"/>
</Alarm_0>
```

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 6.8)

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/Alarm_0"/>
  </OnButton>
</System>
```

Even TILA2 offers the service button as alarm trigger but uses an EventState with the system variable `/Process/MB/PollButton` instead. Both solutions are possible.

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 P1:

```
<Alarm_0 Var="Product_0_1">
     <Event _="Alarm_0"/>
     <ProcessVar _="/Process/MB/IO/I/P1" flank="low"/>
     <Enabled _="TRUE"/>
</Alarm_0>
```

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

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

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/MB/IO/I/P1" flank="low"/>
```

This alarm is assigned to input port P1. The `low` parameter determines the condition being fulfilled 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/Alarm_0_ProcVar"/>
```

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 is met if input port 0 changes from 1 (open) to 0 (closed):

```
<Alarm_0_ProcVar sys="1">
      <Value>
           <LDN _="MB/IO/I/P0"/>
     </Value>
</Alarm_0_ProcVar>
```

Instead of the LDN command, many others may be used as well, e.g. LD, 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 1 to 0:

```
<Alarm_1_ProcVar sys="1">
      <Value>
           <LDN _="MB/IO/I/P0"/>
           <ANDN _="MB/IO/I/P1"/>
     </Value>
</Alarm_1_ProcVar>
```

Comparisons

Beside logical operations, even comparisons are possible. TILA2 already uses this feature. This example triggers an Express-E-Mail. It is also possible to compare PLC variables to trigger an alarm.

```
<Alarm_2_ProcVar sys="1">
 <Value>
   <GE v1="/Process/MB/A/AI/P0" v2="5000"/>
   <LE v1="/Process/MB/A/AI/P0" v2="10000"/>
   <ANB _=""/>
 </Value>
</Alarm_2_ProcVar>
```

The condition is fulfilled if `v1` greater equal `v2` (GE) **and** `v1` less equal `v2` (value inside range 5000-10000). For `v2`, even a reference to another variable may be used.

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 04:20.

```
<Alarm_3_ProcVar sys="1">
 <Value>
  <TIME _="04:20:00-06:30:00"/>
 </Value>
</Alarm_3_ProcVar>
```

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

```
<Alarm_4_ProcVar sys="1">
 <Value>
  <TIME _="04:20:00-06:30:00"/>
  <TIME _="08:00:00-10:00:00"/>
  <OR _=""/>
 </Value>
</Alarm_4_ProcVar>
```

For complex schedules, use the scheduler database, which is described in the next chapter.

MSK / FIND_BIT

Some users of PLCs have the problem, that their PLC transmits alarm bits inside 16- or 32Bit variables. Therefore these must be filtered out of a word or dword value.
A similar problem is the maximum number of alarms (EventStates), which is currently limited to 100 by the Alarm Modem. If you want to trigger more than 100 alarms, they can be put together in a register inside the PLC, and the modems divides them again. With that theoretically 100 x 32 (DWort) = 3200 alarms are possible.

The simplest solution for that is the MSK instruction. With that a bit mask will be put over the word or dword. As soon as one or several bits inside the mask are set, the process variable is set to 1 and may trigger an alarm.

```
<Alarm_0_ProcVar sys=”1”>
 <Value>
  <MSK v1="/Process/PV/PV1" v2="5"/>
 </Value>
</ Alarm_0_ProcVar >
```

In this example a 16-Bit process variable PV1 is filtered by the mask value of 5.

The process variable consists of 16 single bits, being 0 or 1:
Bit 16…………………………..Bit 1
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The mask with value 5 in dual style:
Bit 16…………………………..Bit 1
   0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1

This means bit 1 (=1) and bit 3 (=4) are monitored (1+4=5).
As soon as bit 1 or bit 3 or both are set, the process variable is 1.

The disadvantage of this method is, that it is only determined that a bit is set but not which of them. Therefore the message texts can not be related to the bit.

For that the instruction FIND_BIT was introduced. It works similar to MSK, but the result of the process variable is not 1 but the position of the set bit.
`(basis-messages-ports-FIND_BIT.cnf)`

```
<Alarm_0_ProcVar>
 <Value>
  <LD _="/Process/PV/PV1"/>
  <FIND_BIT_ADDRESS _="1" range="1" mask="58759"/>
 </Value>
</Alarm_0_ProcVar>
```

The mask with value 58759 in dual style:
Bit 16…………………………..Bit 1
   1 1 1 0 0 1 0 1 1 0 0 0 0 1 1 1

The masked bits are numbered by the system starting with the lowest bit, in the example it would mean bit 1 to 9.
As soon as one or several of these bits are set, the process variable value becomes the position (1-9) of the first (specified by `_="1"`) detected bit.
If one loads the same 16bit variable into a second process variable, and puts the same mask over it but sets the bit position to _=“2“, the second set bit within the bit mask will be detected. With that, several set bits inside the 16bit variable can be detected at the same time.

Example:
If the value of the 16bit variable is 128, the bit with position number 4 is set. Therefore the process variable will be 4:

```
16 bit variable “128“:   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
Bit mask “58759“:        1 1 1 0 0 1 0 1 1 0 0 0 0 1 1 1
Bit position:            9 8 7 - - 6 – 5 4 - - - - 3 2 1
```

With this information alarm texts may be designed dynamically using intelligent references, so that every alarm sends a message text related to the bit.

Therefore inside the UserTemplates (Register 12 – MessageText) a set of text templates for all bit positions has to be created:

```
<TextTemplates>
 <BIT_0 _="No Bit detected !"/>
 <BIT_1 _="Error Bit 1 detected !"/>
 <BIT_2 _="Error Bit 2 detected !"/>
 <BIT_3 _="Error Bit 3 detected !"/>
 <BIT_4 _="Error Bit 4 detected !"/>
 <BIT_5 _="Error Bit 5 detected !"/>
 …
</TextTemplates>
```

Inside the alarm text these text templates are referenced depending on the process variable value, by replacing a part of the reference by a reference to the process variable.

```
<Message_0 Name="SMS-Alarm" Type="SMS" UseSignature="0">
 <Subject _="&#xae;/D/UserTemplates/TextTemplates/BIT_&#xae;/Process/PV/Alarm_0_ProcVar;;"/>
</Message_0>
```

Triggering of the alarm can be done by a greater zero comparison of the process variable (GT 0, see above),

```
<GT v1="/Process/PV/Alarm_0_ProcVar" v2="0"/>
```

or via a trigger variable set by the PLC together with the alarm register (see chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**)

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`). `(example basis-fax-scheduler.cnf)`

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

```
<Schedule_0 _="Alarm_0">
 <Weekday _="Mo-Fr"/>
 <Time _="18:00"/>
</Schedule_0>
```

The event to trigger (here: `Alarm_0`) 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 `Contact_0` gets the message and the night shift at 21:00 when `Contact_1` shall be notified. At first, we create the night shift recipient inside the address book:

```
<Contact_1 Name="Shiftworker">
  <Fax _="+49-30-40608401"/>
  …
```

```
</Contact_1>
```

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

```
<Shiftworker sys="1"/>
```

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

```
<Switch_0>
 <Set _="/Process/PV/Shiftworker" value="Contact_0"/>
</Switch_0>

<Switch_1>
 <Set _="/Process/PV/Shiftworker" value="Contact_1"/>
</Switch_1>
```

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

```
<Schedule_0 _="Switch_0">
 <Time _="09:00"/>
</Schedule_0>

<Schedule_1 _="Switch_1">
 <Time _="21:00"/>
</Schedule_1>
```

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:

```
<Alarm_0 _="TextFax">
 <Recipient _="/D/AddressBook/&#xae;/Process/PV/Shiftworker,Contact_0;"/>
 <Sender _="AddressBook/MySelf"/>
 <Subject _="&#xae;/D/UserTemplates/Message_3/Subject;"/>
 <Body _="/D/UserTemplates/Message_3/Body"/>
</Alarm_0>
```

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

`<Recipient _="/D/AddressBook/Contact_0"/>` or
`<Recipient _="/D/AddressBook/Contact_1"/>`

The `Contact_0` 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="Contact_0" ver="y"/>]
```

There is also an easier solution for the shift plan, but with that you would not know the ProcessVars now:

With this example no ProcessVar and even no modified MessageJobTemplate path.
Just let the EventHandler write directly into the MessageJobTemplate:

```
<Switch_0>
 <Set _="/TEMPLATE/MessageJobTemplates/Alarm_0/Recipient"
      value="AddressBook/Contact_0"/>
</Switch_0>

<Switch_1>
 <Set _="/TEMPLATE/MessageJobTemplates/Alarm_0/Recipient"
      value="AddressBook/Contact_1"/>
</Switch_1>
```

The set command may even be used to modify database entries.

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 inserted. At these days, the night shift 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:

```
<Schedule_0 _="Switch_0">
 <Time _="09:00"/>
 <Condition _="Condition/Holidays"/>
</Schedule_0>
```

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:

```
<Schedule_0 _="Switch_0">
 <Time _="09:00"/>
 <Data not="1.1.,1.5.,3.10.,25.12.,26.12."/>
</Schedule_0>
```

The advantage of using the `Conditions` database 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 before 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:

```
<Alarm_0 Name="Fax-Alarm">
<SendMail _="MessageJobTemplates/Alarm_0">
  <Priority _="5"/>
 </SendMail>
</Alarm_0>
```

4.2.5.2. Alarm Sets Port - The `Set` Command

By Set command, output ports may be switched. (TiXML Reference Chap. 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 P2 from 0 to 1 (e.g. for activating a warning light)

```
<Alarm_0 Name="Fax-Alarm">
 <SendMail _="MessageJobTemplates/Alarm_0"/>
 <Set _="/Process/MB/IO/Q/P2" value="1"/>
</Alarm_0>
```

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`):

```
<Alarm_0 Name="Fax-Alarm">
<SendMail _="MessageJobTemplates/Alarm_0">
  <OnError _="Alarm_1"/>
  <MaxRepeat _="1"/>
  <Interval _="180s"/>
 </SendMail>
</Alarm_0>
```

In this example, there are two attempts of sending the message (MaxRepeat=1 means one redial after 180sec). If both attempts fail, the "Alarm_1" event is triggered.
This event could be fitted with some cascading as well. It could even refer back to " Alarm_0", 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-cascade.cnf`)
In this example, in case of an alarm an output port is to be set and, after successfully sending the message, the `Switch_0` event is used to reset the port.

```
<Alarm_2 Name="Express-E-Mail-Alarm">
 <Set _="/Process/C0/Q/P2" value="1"/>
 <SendMail _="MessageJobTemplates/Alarm_2">
  <OnOK _="Switch_0"/>
  <MaxRepeat _="1"/>
  <Interval _="180s"/>
 </SendMail>
</Alarm_2>
```

```
<Switch_0 Name="Switch_0">
<Set _="/Process/MB/IO/Q/P2" value="0"/>
</Switch_0>
```

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

4.2.5.5. Acknowledge of Messages

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

The EventHandler looks like this:

```
<Alarm_0 Name="SMS-Alarm">
     <SendMail _="MessageJobTemplates/Alarm_0">
          <OnError _="Alarm_0"/>
          <OnTimeout _="Alarm_0"/>
          <ConfirmID _="0"/>
          <Timeout _="2m"/>
     </SendMail>
</Alarm_0>
```

The appropriate message text:

```
<Message_0 Name="SMS-Alarm" Type="SMS" UseSignature="0">
     <Subject _="SMS-Alarm with acknowledge request. Code:
     &#xae;~/_Fingerprint;"/>
</Message_0>
```

Additionally, a special `System` EventHandler is necessary:

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

The confirmation feature utilizes a ConfirmID (e.g. "0") and a unique fingerprint code which is generated automatically. Each alarm needs its own Confirm-ID. The fingerprint is 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:

```
<Message_3 Name="Fax-Alarm" Type="Mail" UseSignature="0">
      <Subject _="This is the subject line of the Fax."/>
      <Body>
            <E _="This is the message body of the Fax."/>
            <E _="Enter as many lines as you need."/>
            <E _=""/>
            <E _="Value parameter room: &#xae;~/Room;"/>
      </Body>
</Message_3>
```

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

The appropriate DoOn command must look like this:

```
[<DoOn _="Alarm_0" ver="y">
      <Room _="23.3"/>
</DoOn>]
```

While the resulting fax message would then be as follows:

```
This is the subject line of the Fax.

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

Value parameter room: 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 _="Switch_0" ver="y">
      <Parameter _="1"/>
</DoOn>]
```

EventHandler:

```
<Switch_0>
      <Set _="/Process/MB/IO/Q/P2" value="&#xae;~/Parameter;"/>
</Switch_0>
```

4.2.5.7. Logic operations and calculations

The logic operations, which you have already seen in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.** for calculating process variables, can be used directly inside an EventHandler. This makes it possible to easily copy values
`(basis-process.cnf)`:

Within an EventHandler (Register 10 – EventHandler) the instruction „Process“ has to be inserted. This can use the same logic and comparison instructions as for process variables:

```
<Alarm_0>
 <Process>
  <LD _="/Process/MB/IO/I/P0"/>
  <ST _="/Process/MB/IO/Q/P2"/>
 </Process>
</Alarm_0>
```

In the example above the value of input P0 will be copied to output P0 on every process of the event.

If one wants to do calculations at a given point (event controlled), the usage of math operations ADD,SUB,MUL and DIV inside EventHandler is possible `(basis-process-calculator.cnf):`

```
<Alarm_1>
 <Process>
  <LD _="/Process/PV/Counter"/>
  <ADD _="1"/>
  <ST _="/Process/PV/Counter"/>
 </Process>
</Alarm_1>
```

The example above creates a counter with increases by 1 on every process of the event. In combination with the scheduler, this may be used as a operation time (hours run) counter.

4.2.5.8. 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. We recommend to use binary Logfiles, therefore we do not describe XML logging in this tutorial.

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

For binary logging, we need to define a binary logfile with the desired ring buffer size.
The logfile refers to a record which defines the data structure (register `7/8-LogDefinition`):

```
<LogDefinition>
  <LogFiles>
     <JobReport size="10240"/>
       <Event size="10240"/>
       <Login size="10240"/>
       <IncomingMessage size="10240"/>
       <FailedIncomingCall size="10240"/>
          <Logfile_0 size="10240" contenttype="binary"
          record="Record_0"/>
  </LogFiles>
  <Records>
    <Record_0>
      <Value1 _="word" value="&#xae;/Process/MB/A/AI/P0;"/>
    </Record_0>
  </Records>
</LogDefinition>
```

Here is the type of data (byte, word, dword, int, string, etc.“) and the value to log (via reference to analogue input P0) defined.

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

```
<Logging_0>
     <BinLog _="Logfile_0"/>
</Logging_0>
```

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:

```
<Schedule_0 _="Logging_0">
     <Minute _="0,5,10,15,20,25,30,35,40,45,50,55"/>
</Schedule_0>
```

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

Sending Logged Data as attachments:

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 IncludeLogTXT 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`).

At first the attachment format (e.g. CSV) and the range of to be included data has to be defined within Register 12-Message Text:

```
<Attachments_0 hidden="1">
 <Attachment filename="Logfile_0.csv">
   <IncludeLogTXT _="Logfile_0" range="previous 24 hours"  type="CSV"/>
 </Attachment>
</Attachments_0>
```

At `filename=”Logfile_0.csv”` the `Logfile_0` may be replaced by any filename, or even by reference e.g to the current date: `filename=”&#xae;/TIMES/Date;.csv”`

Within the IncludeLogTXT command the “range“ of to be sent data has to be specified.
The example sends the data of the previous 24 hours. Other range parameters can be found in TiXML Reference Manual chapter 2.4.9.2.

The attachment block even may include several attachments (groups „Attachment“).

Detailed info on formatting (e.g. HTML, XML etc.) is to be found within TiXML-Reference chapter 4.9.

Additionally, we need to assign the Attachment block to the MessageJobTemplate:

```
<Alarm_1 _="SMTP">
 <Recipient _="AddressBook/Contact_0"/>
 <Sender _="AddressBook/MySelf"/>
 <Subject _="&#xae;/D/UserTemplates/Message_4/Subject;"/>
 <Body _="/D/UserTemplates/Message_4/Body"/>
 <Attachments _="/D/UserTemplates/Attachments_0"/>
</Alarm_1>
```

Periodical email with logged data can be sent via scheduler:

```
<Schedule_1 _="Alarm_1">
 <Time _="00:01"/>
</Scheule_1>
```

4.2.5.9. IF-Condition (TiXML-Reference Manual Chapter 3.8.2)

To process an EventHandler command depending on a condition, e.g. to log PLC data only if the PLC is connected (DeviceState=1), the usage of the IF condition is recommended (`basis-portbinlog-IF-condition.cnf`):

Around the EventHandler commands the Tag `"If"` is put, which refers (path) to a bit variable inside its start tag (Register 10 – EventHandler):

```
<Logging_0>
 <If _="/Process/MB/IO/I/P0">
  <BinLog _="Logfile_0"/>
 </If>
</Logging_0>
```

The enclosed commands are only processed if the bit variable is`"1"`. To process them on „0“, the bit variable has to be negated inside a process variable (see chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**).

Please note that the path of the IF condition is written without reference symbol (&#xae;).

4.2.5.10. Internet time synchronization / Summer-Winter-Time (TiXML-Reference Chap. 3.13)

The RTC of the Tixi Alarm Modems may be synchronized 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 is set 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 useful time, e.g. every month:

```
<Schedule_0 _="SyncTime">
  <Month _="1-12"/>
```

```
</Schedule_0>
```

For a Summer- Wintertime change, the modem has to synchronize the time at a special time. In Europe it is 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>
```

4.2.6. Sequencer (TiXML-Reference Manual Chapter 8)

The sequencer is used to set values related to a time, e.g. time dependent thresholds. The values and times are saved within a table `(basis-sequencer.cnf).`

The table can be sent in XML format or CSV format (with TiXML frame) via TiXML command `SetSequence`. CSV allows easy import of excel files.

Sequence in XML format:

```
[<SetSequence _="Sequence_0" priority="0" ver="v">
<T date="*.*.*" time="*:00" P1="1" P2="0" P3="1"/>
<T date="*.*.*" time="*:15" P1="0" P2="1" P3="0"/>
<T date="*.*.*" time="*:30" P1="1" P2="0" P3="1"/>
<T date="*.*.*" time="*:45" P1="0" P2="1" P3="0"/>
</SetSequence>]
```

Sequence in CSV format with TiXML frame:

```
[<SetSequence _="Sequence_0" mode="text">
<![CDATA[
!priority=1
!mask="d;t;1;2;3"
*.*.*;*:00;1;0;1
*.*.*;*:15;0;1;0
*.*.*;*:30;1;0;1
*.*.*;*:45;0;1;0
]]>
</SetSequence>]
```

Each sequence instruction has a date of format DD.MM.YYYY, a time of format HH:MM and a list of up to 6 values. Parts of date and time can be replaced by an asterisk (*), so that the date “*.*.*“ means “every day“ and the time “*:15“ means “quarter past each hour “.

The required sequences are defined within „Register 21 – Sequencer“:

```
<Sequencer>
 <Sequence_0 event="Switch_0" logfile="Sequence_0_Log"/>
</Sequencer>
```

An EventHandler (`event`) and a logfile (`logfile`) are assigned to the sequence.

Transmitted sequences will be copied into the logfile.

The EventHandler will be processed at the defined sequence times and the values will be hand over as P1 to P6.

```
<Switch_0>
 <Set _="/Process/MB/IO/Q/P0" value="&#xae;~/P1;"/>
 <Set _="/Process/MB/IO/Q/P1" value="&#xae;~/P2;"/>
 <Set _="/Process/MB/IO/Q/P2" value="&#xae;~/P3;"/>
</Switch_0>
```

4.3. Remote Control By Incoming Messages (R. Chapter. 9)

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. TILA2 supports remote switching via SMS and Express-E-Mail. The example utilizes SMS.
(`basis-remoteswitch.cnf`):

4.3.1. Access Protection

At first we'll configure an access protection at register `5/6-AccRights,` in order to let authorized personnel only control the device:.

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

```
<AccRights>
      <Groups>
            <MessageGroup>
                  <Message AccLevel="1"/>
            </MessageGroup>
      </Groups>
      <User _="Plain">
            <Def_Message Plain="TIXI" Group="MessageGroup"/>
      </User>
</AccRights>
```

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

```
      <User _="Plain">
            <OA_491721234567 Plain="TIXI" Group="MessageGroup"/>
      </User>
```

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

Within GSM networks the callerID is sent along with „+country code“ (e.g.. +491721234567). The „+“ has to be left out of the configuration.

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:

```
     <User _="Plain">
          <Tixi_Support Plain="TIXI" Group="MessageGroup"/>
     </User>
```

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>
```
e.g.

```
TIXI SETPORT
```

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

```
<SETPORT Name="SETPORT">
     <Set _="/Process/MB/IO/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>
```

e.g.

```
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 Name="SETVALUE">
     <Set _="/Process/MB/IO/Q/P2" value="&#xae;~/P1;"/>
</SETVALUE>
```

The parameter is inserted via the `&#xae;~/P1;` reference (for more parameters, count up to `P10`) 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 a `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 Name="SETPORT">
     <Set _="/Process/MB/IO/Q/P0" value="1"/>
     <SendMail _="MessageJobTemplates/Switch_0"/>
</SETPORT>
```

4.3.4.1. Static Reply

As an confirmation of an action, a message to an address book contact shall be created. The MessageJobTemplate looks like this:

```
<Switch_0 _="GSMSMS">
     <Recipient _="AddressBook/Contact_0"/>
     <Sender _="AddressBook/MySelf"/>
     <Subject _="&#xae;/D/UserTemplates/Message_0/Subject;"/>
</Switch_0>
```

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

```
<Message_0 Name="SMS Answer" Type="SMS" UseSignature="0">
     <Subject _="The Event &#xae;~/Event; was successfully processed."/>
</Message_0>
```

4.3.4.2. Dynamic 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 address book contact but contains a reference to the sender ID of the incoming message.

The corresponding MessageJobTemplate refers to the system address book entry “OA” which uses the received callerID via reference as the recipients address.

```
<Switch_1 _="GSMSMS">
     <Recipient _="AddressBook/OA"/>
     <Sender _="AddressBook/MySelf"/>
     <Subject _="&#xae;/D/UserTemplates/Message_0/Subject;"/>
</Switch_1>
```

AddressBook:

```
<OA hidden="1">
     <SMS_No _="&#xae;~/OA"/>
     <Email _="&#xae;~/OA"/>
     <Express-Email _="&#xae;~/Alpha"/>
     <SMS_Provider _="AnnyWay"/>
</OA>
```

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 some countries that offer landline SMS. In other countries you’ll have to use GSM devices.

If your country offers landline SMS, you’ll have to change the “OA”s SMS_Provider to your local SMS gateway.

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 checking 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 Name="SETVALUE2">
     <Set _="/Process/MB/IO/Q/P2" value="&#xae;~/P1;"/>
     <Delay _="5s"/>
     <SendMail _="MessageJobTemplates/Switch_2"/>
</SETVALUE2>
```

The reply message text containing the current value of P2:

```
<Message_1 Name="SMS Answer Value" Type="SMS" UseSignature="0">
     <Subject _="The Event &#xae;~/Event; was successfully processed.
     Value Q2: &#xae;/Process/MB/IO/Q/P2;"/>
</Message_1>
```

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) or `TixiInvalidEvent` (Express-E-Mails) or `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 erroneous message, as described in chapter 4.3.4.2, and shall give hint on the problem occurred:

```
<Message_0 Name="SMS-Alarm" Type="SMS" UseSignature="0">
 <Subject _="Error: The Event &#xae;~/Event; is unknown."/>
</Message_0>
```

4.3.5.3. Logging information 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/MB/IO/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 9.2

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="Switch_0"/>
</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.

Tixi provides a “CLIP-tool” to monitor the callerID format at your PSTN or ISDN connection.

4.5. Project Upload via email (TiXML-Reference Chapter 3.8.1)

If a lot of similar units are installed and parts of the configuration changes periodically (e.g. the address book or Sequencer tables), a project upload via emails makes more sense than dialling all sites. For that the Alarm Modem is able to receive projects via email `(basis-SetConfig.cnf)`.

At first a periodically email query is setup as described in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**.

The subject line of the collected email, which contains a remote control password (chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**), triggers an EventHandler with a SetConfig instruction (Register 10 – EventHandler):

```
<SETCONFIG>
 <SetConfig/>
</SETCONFIG>
```

The email sent to the Alarm Modem has to contain all SetConfig instructions for the databases being modified. The square brackets have to be replaced by <D> </D> Tag enclosed around all instructions:

```
<D>
<SetConfig _="TEMPLATE" ver="y">
<AddressBook>

<MySelf Name="MySelf" hidden="1">
 <SMS_No _="+49-172-6055321"/>
 <Fax _="+49-172-6055321"/>
 <Email _="Tixi-Training1@gmx.net"/>
 <Express-Email _="TAM+49-172-6055321"/>
</MySelf>

<OA hidden="1">
 <SMS_No _="&#xae;~/OA"/>
 <Email _="&#xae;~/OA"/>
 <Express-Email _="&#xae;~/Alpha"/>
 <SMS_Provider _="AnnyWay"/>
</OA>
```

```
<Contact_0 Name="Receiver">
 <Email _="demo@tixi.com"/>
 <Express-Email _="TAM+49-30-40608444"/>
 <SMS_No _="+49-172-1844751"/>
 <SMS_Provider _="AnnyWay"/>
 <Pager_No _="394000"/>
 <Pager_Provider _="CityRuf"/>
 <Fax _="+49-30-40608438"/>
</Contact_0>

</AddressBook>
</SetConfig>
</D>
```

Please note that these emails have to be sent in plain text format. HTML- or Ritch-Text (RTF) emails are not supported.

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 logging, alarm triggers or remote control.

Hence, replace the I/O port references (e.g. `/Process/MB/IO/I/P0`) inside the examples by such referring to PLC variables (e.g. `/Process/Bus1/D0/Register1`).

The more complex PLC database (register `17-External`) shall be created using TILA2, for the sake of plainness.

The `basis-messages-sps.cnf` sample shows alarm features 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.