User documentation - `xml2database` converter

Version 1.0.0
Status: **final**

CVS: *Revision* : 1.4

Reinhard Alt Unternehmensberatung

Date 08.02.2012

Contents

Chapter 1

Summary

Thank you for your interest in xml2database!

With xml2database you got a flexible, configurable, fast, multicore capable XML/CVS/database converter. xml2database is i.e. used in the telecom industry to refine large amounts of statistic/accounting XML records into formats which can be imported into large statistic databases very easy. xml2database is developed for Unix systems (Solaris, Opensolaris, linux) to offer 24h/7d guaranteed service called typically from unix cron-jobs in regular intervals i.e. 15 minutes.

The converter is designed to be used in a flexible manner and to support a broad range of data export scenarios. xml2database uses a dynamic record definition language to specify the generated records and the data attributes which need to be extracted from XML.

To process a huge amount of data xml2database uses up to 20 CPU cores. By default only 3 parallel running processes are configured.

HINT: The demo version is limited to use only one processor core and after 10000 Records the performance is reduced significantly.

The created records and the attributes in each table were defined in an external configuration file ($HOME/config/xml2database.config). The configuration file allows easy modification/adding of new attributes or records. On top it allows hiding of changes in the XML data source to the database with the advantage that the datbase mode stays untouched.

The input data could be every XML file generated by ie. network elementsor results of SOAP requests like used in the Alcatel Lucent SAM database.

xml2database represents the middleware core of the conversion process between XML generating data sources and a database. Due to performance requirements xml2database is completly written in C/C++ using SAX Parser and bison/flex.

call_xml2database represents the outer calling shell script and is typically called by a cron job. It and calls the conversion process xml2database. call_xml2database does additional jobs like sftp transfer, the deletion of aged files and error signaling. It is written in bash shell language to be easy modified by customer for new subtasks.

Chapter 2

Functional description

This section describes `xml2database` in detail and its functionality.

2.1 Overview

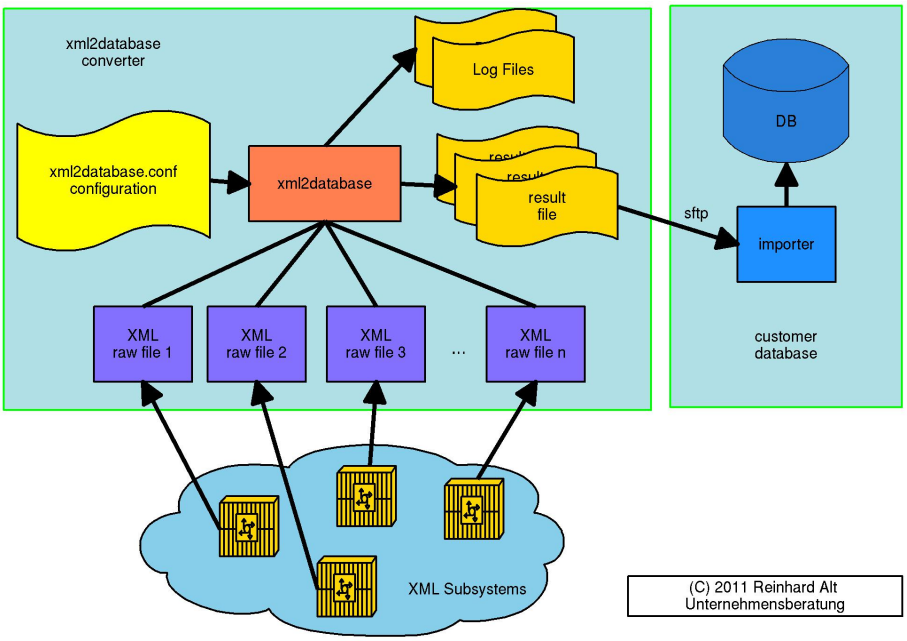

Figure 2.1: `xml2database` overview

The central functionality of `xml2database` is the conversion of XML tree structured data into table structured formats which can be easily imported into databases.

The source of xml raw data could be a statistic monitoring server in the telecom industry like i.e Alcatel Lucent Service Aware Manager or a XML protocol unit in the car maker industry or a quality monitoring system in the silicon chip industry reporting operation parameters in XML.

In all cases the XML generating subsytem transfers the XML files in regular intervals into

the $HOME/input directory for further processing.

XML data typically is structured as a tree. On each level of the tree attribute data can be found. For the creation of a database table the identification of unique database keys is an important issue. Two types of data can be found in XML:

- Normal entities:

```
<StartTime>2011-05-18T07:15:00Z</StartTime>
```

- Attributes inside of entities:

```
<attributeWithArgs arg1="1111" arg2="2222" arg3="3333"/>
```

The configuration language for xml2database contains *operators* to extract this data from the XML files. The most simple *operators* of xml2database are:

- For normal entity extraction:

```
ATTRIBUTE( "StartTime" ) "StartTime"
```

- Example for extraction of attributes inside of an entity:

```
ARGATTRIBUTE( "attributeWithArgs", "arg1" ) "ARGATTR1"
ARGATTRIBUTE( "attributeWithArgs", "arg3" ) "ARGATTR3"
```

All data extraction *operators* of xml2database are bundled in one or more MEASUREMENT definition per INPUTFILE.

```
INPUTFILE <Filename 1> {
    GENERATES MEASUREMENT {
        <operators>
        ...
    }

    GENERATES MEASUREMENT {
        <operators>
        ...
    }
}
```

You can create one or more MEASUREMENT = database tables from the same input file. For more details how to configure xml2database see section 3.5.2.

The configuration file $HOME/config/xml2database.conf contains the complete description of tables to be generated from the XML data.

During the startup xml2database parses the configuration file and forks a configured (option -t <numer of processes>) amount of subprocesses to support multicore CPU's.

IMPORTANT: If two subtrees needs a common root object i.e. a slot or port number, it is very important for the xml2database converter to have these common attributes in the beginning of the extraction statement an BEFORE the subclass data extraction.

Another important issue is the fact that identical named attributes like `operationalState` as in our example may appear at different positions in the XML tree with different meanings. For example `operationalState` for a `Shelf` or a `card` needs to be addresses by the converter together with a so called *pointer* based on the name of the higher ordered class. This will be discussed later in chapter 3.6.

```
<result>
    <attribute>siteId</attribute>
    <attribute>siteName</attribute>
    <attribute>shelfType</attribute>
    <attribute>slotName</attribute>
    <attribute>specificType</attribute>
    <attribute>operationalState</attribute>
    <attribute>administrativeState</attribute>
    <attribute>manufacturerBoardNumber</attribute>
    <attribute>serialNumber</attribute>
    <children>
        <resultFilter class="equipment.PhysicalPort">
            <attribute>portName</attribute>
            <attribute>operationalState</attribute>
            <attribute>administrativeState</attribute>
            <attribute>specificType</attribute>
            <children>
                <resultFilter class="equipment.MediaAdaptor">
                    <attribute>specificType</attribute>
                    <attribute>transceiverCode</attribute>
                    <attribute>sfpOpticalCompliance</attribute>
                    <attribute>laserWaveLength</attribute>
                    <attribute>modelNumber</attribute>
                    <attribute>vendorSerialNumber</attribute>
                    <attribute>vendorPartNumber</attribute>
                </resultFilter>
            </children>
        </resultFilter>
    </children>
</result>
```

2.2 Support

If you find `xml2database` useful or if you have a specific problem which currently cannot be solved we are happy if you contact us. We improve the quality of the software and extend the list of supported features continously.

We also offer on-site support for installation in the European Union and off-site support for the rest of the world. Customization of the `xml2database` configuration file and definition of database records will also be done by us if needed. Contact us for an offer and define a raw set of expected database tables you expect.

Contact address:

Reinhard Alt Unternehmensberatung
Im Mainfeld 42, 60528 Frankfurt
Germany
SIP: +49 69 1730 9140
Fax: 069 677 30 895
e-mail: xml2database@arcor.de

Chapter 3

Configuration

This chapter describes the configuration and the configuration file `xml2database.conf`

3.1 Directory structure

The default installation path is `/opt/xml2database`. The directory contains the following subdirectories:

path	explanation
bin	binaries and scripts
lib	dynamic libaries
config	configuration file
config/xml2database.conf	main configuration fail
logs	Log Files
lock	`call_xml2database` lock files
input	XML raw files
output	results
documentation	user documentation

- The directory `bin` contains all binaries and scripts. A symbolic links points to the newest binary version of the converter `xml2database`.
- `xml2database` needs the libaries from Qt 4.7 from Nokia which were stored in the subdirectory `lib`.
- The subdirectory `config` contains the configuration files. The configuration file name can be set with the option `-c <config file>`.
- The `logs` directory is the default logging path for `xml2database`. The cron-job `call_xml2database` purges all log files older than the retention time `RETENTION_LOGS`.
- All raw files in the directory `input` will be analyzed by `call_xml2database`. The files should contain a timestamp suffix of the format `YYYYMMDD_HHMM.xml`. If the date suffix is not found in the filename, the converter uses the current date for generated date stamps in the output file names. The path to the XML input data can be set with `-i <input path>`.

- All results were created in the directory `$HOME/output`. Using the script `call_xml2database` all files were zipped with gzip.

  The output path of the generated data can be set with `-o <output path>`.

3.2 Dynamic C++ Libraries, Environment variables

HINT: In the following examples the shell bash is used and all examples a developed based on bash.

`xml2database` uses the Qt libaries locaed in `$HOME/lib`. You can use your own Qt libs installed on your system. It should work but all tests are done with the delivered version.

```
xml2database@orgsunny:~/bin$ ldd xml2database
        libQtSql.so.4 =>         /opt/xml2database/lib/libQtSql.so.4
        libQtXml.so.4 =>         /opt/xml2database/lib/libQtXml.so.4
        libQtCore.so.4 =>        /opt/xml2database/lib/libQtCore.so.4
        libpthread.so.1 =>       /usr/lib/libpthread.so.1
        librt.so.1 =>    /usr/lib/librt.so.1
        libstdc++.so.6 =>        /opt/xml2database/lib/libstdc++.so.6
        libm.so.2 =>     /usr/lib/libm.so.2
        libgcc_s.so.1 =>         /opt/xml2database/lib/libgcc_s.so.1
        libc.so.1 =>     /usr/lib/libc.so.1
        libz.so.1 =>     /usr/lib/libz.so.1
        libdl.so.1 =>    /usr/lib/libdl.so.1
        libgthread-2.0.so.0 =>   /usr/lib/libgthread-2.0.so.0
        libthread.so.1 =>        /usr/lib/libthread.so.1
        libglib-2.0.so.0 =>      /usr/lib/libglib-2.0.so.0
```

The environment variables `PATH` and `LD_LIBRARY_PATH` needs to be extended to find the program and the libraries.

For the shell `bash` the typical information holder is the file `.profile` in the `$HOME` directory of the used user `xml2db`. You can use i.e. `vi` or `emacs` to add the following lines:

```
> vi .bashrc
# extend HOME
export PATH=$PATH:/$HOME/bin
# own libs first
export LD_LIBRARY_PATH=$HOME/lib:/usr/lib:/usr/sfw/lib
```

The new environment is available either after log in again or explicit sourcing::

```
> vi .bashrc
> source .profile
```

3.3 Program call

In a 24h/7d environment the cron job `call_xml2database` calls the converter `xml2database` automatically. Several parameters are set in the cron job and forwarded to the converter `xml2database`.

It is recommended to create a new user (i.e. `xml2db`) for the conversion because there is no reason to let it run with `root` permissions.

IMPORTANT: If you use a different user you have to change the permissions on all files and directories in `/opt/xml2database`.

To test the environment and the software call the converter with the option `-v` from a unix shell:

```
> /opt/xml2database/bin/xml2database -v
```

With correctly set environment variables `LD_LIBRARY_PATH` gesetzt sein (see 3.2) the software identifies with a version number:

```
>/opt/xml2database/bin/xml2database -v
Reinhard Alt Unternehmensberatung
xml2database - fast flexible data converter
Version 0.8.1
CVS: Revision:  1.4
CVS: Date:  2011/10/06 14:03:26
Made in Germany
```

3.4 Start options

The converter `xml2database` offers several start options which are typically used in the cron-job `call_xml2database`:

-v program version
The process prints the version and finishes immediatly without starting the conversion process.

-h short help
The process prints a short help and terminates.

-d <level> The debug level can be set between 0=off and 5=max debugging. A higher level includes the messages of the lower level. The higher the debug level the more informations were printed.

On productive systems it is not recommended to use a debug level due to fact of fast growing log files and a huge load.

-c <config> With `-c <configuration file>` the configuration file will be defined. Default: `$HOME/config/xml2database.conf`.

-t <processes> The option `-t <number of processes>` the number od parallel running processes will be defined. The number is limited to max 20.

On the evaluation version this switch has no effect.

-i <path> The option `-i <path>` sets the path to the XML input data (Default: `$HOME/input`).

-o <path> The option `-o <path>` defined the path to the converted output data. (Default: `$HOME/output`).

3.5 Configuration file structure config/xml2database.conf

The configuration file config/xml2database.conf contains all parameter and statements which are necessary for extraction and conversion.

The configuration file structure starts with GLOBALS { PARAMETER ... } block containing global parameter valid for all MEASUREMENTS followed by a repeating list of measurement definitions for each INPUTFILE <file filter>:

```
GLOBALS {
        <PARAMETER 1>       <value>
        <PARAMETER 2>       <value>
        <PARAMETER 3>       <value>
        ...                 ...
}

INPUTFILE <file filter 1> {
    <column statements>
    ...
}

INPUTFILE <file filter 2> {
    <column statements>
    ...
}

...
```

3.5.1 Parameter of GLOBALS block

The configuration block GLOBALS contains parameter which are of general usage for all INPUTFILES definitions. The list of GLOBALS parameter may be extended in future versions of the software.

Current GLOBALS parameter:

- VERSION "<String>"

  Specifies the version string used in the filename of the output files.
  Allowed type: String

- FILE_MASK "%3_%2_%4_%1.data"
  # %1 = <JJMMTTHHMM> date
  # %2 = <granularity> D,Q,E
  # %3 = <measurement name>
  # %4 = <Version>

  Defines the format of the output filename.

- TIME_FORMAT "yyMMdd hh:mm" # i.e. Sybase Format
  #TIME_FORMAT "yyyy.MM.dd hh:mm" # i.e. Sqlite & Postgres
  #TIME_FORMAT "yyyy-MM-dd hh:mm:ss" # i.e. Oracle

  This parameter defines the layout of Date-Time fields in generated data for the operators STIME_FILESTAMP, STRING_TO_LOCAL, STRING_TO_UTC, MS_TO_LOCAL and MS_TO_UTC.
  For format detaults see the Qt date print function Qt 4.7 date print function

- `COLUMN_SEPARATOR ";" # semicolon`

  `COLUMN_SEPARATOR` defines the string used to split the data columns.

- `HEADER_SIZE_MAX 30  # i.e. Sybase`

  `HEADER_SIZE_MAX` defines the size in chars of the largest possible table column name. Data cells are not affected! Background is, that some databases needs limits regarding the number of chars in the table header. If a table column name is longer than the specified `HEADER_SIZE_MAX` the following message will be generated:

  ```
  ERROR: not allowed column size for "<header name>" > <size> chars
  <header name> : will be replaced with the column name
  <size> : will be replaced with the configured HEADER_SIZE_MAX
  ```

3.5.2 Parameter of `INPUTFILE` block

The processing of the XML is defined in `config/xml2database.conf` using a language which is interal using bison parser generator. The defined grammar can be found at the end of this document. (Section A.3)

Because one XML raw file could generate one or more result files, the definition of `INPUTFILE` contains one or many `GENERATES MEASUREMENT{}`[1] statements.

```
INPUTFILE "<file filter>" {
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "<output file 1>"
        ...
    }

    GENERATES MEASUREMENT {
        MEASUREMENTNAME "<output file 2>"
        ...
    }

    GENERATES MEASUREMENT {
        MEASUREMENTNAME "<output file 3>"
        ...
    }

    ...
}
```

The `INPUTFILE` String for `<file filter>` could contain eiter a normal file name or a regular expression like `"^(mueller*.xml|miller*.XML)$"`

The columns of the result file (measurement) were processed and genereated by several *operators*. The `MATCH` subblock specifies the XML subtrees which need to be passed before data is exported. `NEEDS` defines a pointer of the XML subtree under which all *attributes* of the `COLUMNS` block need to be located. The `COLUMNS` Block lists all "data print functions" (*operators*).

Here a simple application example of a `xml2database.conf` definition to extract data of the `"datarecord" -> "shell"` XML subtree of the files `mueller*.xml` or `miller*.XML`.

[1] A `MEASUREMENT` is used as a synonym for a database table

```
INPUTFILE "^(mueller*.xml|miller*.XML)\$" {                    # input file names
  GENERATES MEASUREMENT {
    MEASUREMENTNAME "TESTDATA"                                 # measurement name
    GRANULARITY    <15MIN|24H|EVENT>                           # granularity of the data
    MATCH {                                                    # filter
      NEEDS "datarecord"->"shell"
    }

    COLUMNS {                                                  # column operators
      UTC_FILESTAMP                                            "SYSTEMTIME"
      REGEXPATTRIBUTE( "Port", "Port ([0-9]*)/[0-9]*/[0-9]*" ) "Shelf"
      REGEXPATTRIBUTE( "Port", "Port [0-9]*/([0-9]*)/[0-9]*" ) "Slot"
      REGEXPATTRIBUTE( "Port", "Port [0-9]*/[0-9]*/([0-9]*)" ) "Port"
      CONVERTATTRIBUTE STRING_TO_UTC ( "ISO8601TIME",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )       "ISO_UTC_TIME"    # UTC
      CONVERTATTRIBUTE STRING_TO_LOCAL( "ISO8601TIME",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )       "ISO_LOCAL_TIME"  # localtime
      ATTRIBUTE( "receivedTotalOctets" )                       "RECEIVED"
      ATTRIBUTE( "transmittedTotalOctets" )                    "TRANSMITTED"
      BOOLATTRIBUTE( "bits"->"subattribute", "active" )        "IS_ACTIVE"
    }
  }
}
```

Comments might be added by using the "#"-char. Text to the right will be treated as comment and will be ignored.

Each measurement definition (`GENERATES MEASUREMENT {...}`) need to have the following parameters und substructures:

- `MEASUREMENTNAME`
  The parameter `MEASUREMENTNAME` defines the filename of the resulting file. The full file name is constructed by following parameters:

  ```
  <JJMMTTHHMM>.manyNEs.<granularity>_<measurementname>_<version>
  - <JJMMTTHHMM> date using the format year,month,day,hour,minute
  - <nodename> manyNEs
  - <granularity> the parameter may have the values 24H, 15MIN or EVENT
  - <measurementname> measurementname defined by {\tt MEASUREMENTNAME}
  - <version> version derived from the global parameter VERSION
  ```

  The locations of parameters might be rearranged by using a different `FILE_MASK` in the block `GLOBALS`.

  > Important: <measurementname> may not contain special characters like .,*, ? or _

- `GRANULARITY`
  The parameter `GRANULARITY` defines a time based type of the measurement.

  15MIN For quarterly measurements - the quantifier in the filename will be set to "Q".

  24H For measurements taken once a day - the quantifier in the filename is set to "D".

  EVENT For unspecified measurements For measurements taken once a day - the quantifier in the filename is set to "E".

- `MATCH`
  The filter block `MATCH` selects for all attributes of the `COLUMNS` block the XML subtree. The `MATCH` block must be fulfilled before any attributes of an XML subtree can be analyzed and be written into the generated output file. When reaching the closing statement of an XML block the data is written to the file. The following filters are possible:

  - `VOID`: The expression `VOID` selects all data from every subtree. `VOID` should only be used for the specification of `xml2database` configuration files. Practically at every reached end of a XML subtree data is written to the output.

- NEEDS <pointer>: Limits the data extraction ro the XML subtree specified by <pointer. A pointer is a recursive structure in a recursive form like "subtree1" -> "subtree2" -> "subtree3".

  > Important: A subtree of a <*pointer*> needs quotation marks ("). The pointer chain is linked by a combination of minus sign and greater sign (->).

  Example of a MATCH block:

```
MATCH {
    NEEDS "Card"->"subcard"->"FlashMemory"
}
```

- AND NEEDS <pointer>: In rare environments it might be neccessary to write only data after two XML subtrees were passed. In such cases further limitations can added to the MATCH block by adding a AND NEEDS <pointer> statement.

  The number of number of additional AND NEEDS <pointer> statements is (theoretical) not limited.

  Example readung two blocks :

```
MATCH {
    NEEDS "Card.Type.A"
    AND NEEDS "Card.Type.B"
}
```

  This MATCH example takes the same attributes from the two XML subtrees Card.Type.A and Card.Type.B which belongs to the same common XML tree.

- COLUMNS
  The COLUMNS block defines the columns of the exported data and contains *operators* to select, extract and manipulate the data.

  The sequence of the columns is defined by the sequence of the column operators in the COLUMNS block. The operators are described in detail in the next section 3.6.

3.6 COLUMNS operators

The design goal of xml2database was convert XML data fast & flexible. The processing of XML raw data is defined by *operators* in COLUMNS block in the configuration file xml2database.conf. The concept of operators allows the customer to add new columns or tables very easy. For the future versions of the software this concept allows to add new operators with new data extraction or data manipulation functionality.

The following column *operators* are currently available:

name	usage
`ATTRIBUTE( P )`	Unmodified data extraction
`UTC_FILESTAMP`	Date extraction from file name
`CONVERTATTRIBUTE MS_TO_UTC( P )`	Milliseconds to UTC
`CONVERTATTRIBUTE MS_TO_LOCAL( P )`	Milliseconds to local time
`CONVERTATTRIBUTE STRING_TO_UTC( P, S )`	Date string to UTC
`CONVERTATTRIBUTE STRING_TO_LOCAL( P, S )`	Date string to local time
`REGEXPATTRIBUTE( P, S )`	Extraction by regular expression
`BOOLATTRIBUTE( P, S )`	String comparision to 0 or 1
`ARGATTRIBUTE( P, S )`	XML argument extraction
`P`	Pointer to a XML attribute
`S`	String / regular expression

The typical result consists of a table with a header and following data fields. Both the header and the data fields were filled and arranged by the sequence of column operators.

The operator contains typically the column name as last argument. The data is generated from different sources like XML tags, XML arguments or the filename.

Some *operators* use a *pointer* P to address raw data from the XML tree. If the XML tag requested by the pointer is not available, the related data is set to the empty string "".
The following example shows the XML raw data with two extractable data elements which will be addressed by two *pointers*:

```
<computer>
    <hostName>myCastle</hostName>
    <cards>
        <inet>10.0.0.1</inet>
    </cards>
</computer>
```

The two XML data set can be addressed with the two `xml2database` pointers:

- `"computer" -> "hostName"` und
- `"computer" -> "cards" -> "inet"`

A *pointer* might be right associative shorted as far as it is distinctive. Instead of addressing `"computer" -> "cards" -> "inet"` it is possible to address `"inet"`.

The following example shows XML raw data where full qualified *pointers* are needed:

```
<computer>
    <hostName>myCastle</hostName>
    <status>up</status>
    <cards>
        <inet>10.0.0.1</inet>
        <status>up</status>
    </cards>
</computer>
```

To distinguish the status field a full qualified *pointer* is needed:

- `"computer" -> "status"` und
- `"computer" -> "cards" -> "status"`

Or shorted to still distinctive *pointers*:

- `"computer" -> "status"` und
- `"cards" -> "status"`

By adding the parent XML subtree to the *pointer* the XML attribute can be addressed exactly. Without the pointer prefix the related data of `status` would become messed up and wrong.

> IMPORTANT: For a correct data assignment it is important to specify an exacly matching *pointer*! Otherwise the extracted data could become wrong!

3.6.1 Column operator - `ATTRIBUTE`

The column operator

```
ATTRIBUTE( "<pointer>" ) "<column name>"
```

generates a column with unmodified data from the XML raw data. `<pointer>` describes an element in the XML tree and has to be conclusively.

The following example shows a measurement definition for the operator `ATTRIBUTE()` and generates the column `RECEIVEDBYTES`

```
INPUTFILE "^SAM_TEST.*\.xml$" {
    GENERATES MEASUREMENT {
    MEASUREMENTNAME "DEVELOPMENT_TEST1"
    GRANULARITY    15MIN
    MATCH {
        NEEDS "datarecord"->"shell"
    }

    COLUMNS {
        ATTRIBUTE( receivedBytes )      "RECEIVEDBYTES"
        }
    }
}
```

3.6.2 Column operator - `UTC_FILESTAMP`

The column operator

```
UTC_FILESTAMP "<column name>"
```

generates a column containing a date/time stamp extracted from the date/time string from the filename of the raw file.

The timestamp can only be read if the filename has the following suffix:

```
*_yyyyMMdd_hhmm.xml

yyyy : year
MM : month
dd : day
hh : hour
mm : minute
```

If the format cannnot be recognized, the following ERROR message will be generated and the current time will be taken:

```
"ERROR: could not extract date/time from input file name - using current time"
"INFO: input file extensions is not "_yyyyMMdd_hhmm.xml""
```

The output format of `UTC_FILESTAMP` is defined by the global parameter `TIME_FORMAT` (see section 3.5.1).

The following example generates a column `UTCTIME` using the format `yyyyMMdd hh:mm`

```
GLOBALS {
    TIME_FORMAT "yyyyMMdd hh:mm"
}

INPUTFILE "^SAM_TEST.*\.xml$" {
    GENERATES MEASUREMENT {
    MEASUREMENTNAME "DEVELOPMENT_TEST1"
    GRANULARITY    15MIN
    MATCH {
        NEEDS "datarecord"->"shell"
    }

    COLUMNS {
        UTC_FILESTAMP      "UTCTIME"            # UTC Zeit aus Filenamen
        }
    }
}
```

3.6.3 Column operator - CONVERTATTRIBUTE MS_TO_UTC(P)

The column operator

```
CONVERTATTRIBUTE MS_TO_UTC( "<pointer>" ) "<column name>"
```

generates a UTC timestamp (time zone Greenwich) from a date/time stamp in milliseconds since 1.1.1970 00:00.

`<pointer>` describes an element in the XML tree and has to be conclusively.

The output format of `MS_TO_UTC` is defined by the global parameter `TIME_FORMAT` (see section 3.5.1).

The following example shows a definition for the operator `CONVERTATTRIBUTE MS_TO_UTC(P)`:

```
INPUTFILE "^SAM_TEST.*\.xml$" {
    GENERATES MEASUREMENT {
    MEASUREMENTNAME "DEVELOPMENT_TEST1"
    GRANULARITY    15MIN
    MATCH {
        NEEDS "datarecord"->"shell"
    }

    COLUMNS {
        CONVERTATTRIBUTE MS_TO_UTC( "timeCaptured" )      "UTCTIME"
        }
    }
}
```

3.6.4 Column operator - CONVERTATTRIBUTE MS_TO_LOCAL(P)

The column operator

```
CONVERTATTRIBUTE MS_TO_LOCAL( "<pointer>" ) "<column name>"
```

generates a time date column using the local time zome from a date time representation in milli seconds. The converter `xml2database` uses the `TZ` environment variable to determin the timezone.

`<pointer>` describes the XML element in the XML tree and has to be conclusively.

The output format of `MS_TO_UTC` is defined by the global parameter `TIME_FORMAT` (see section 3.5.1).

The following example shows definition for the operator `CONVERTATTRIBUTE MS_TO_LOCAL( P )` generating the column `LOCALTIME`:

```
INPUTFILE "^SAM_TEST.*\.xml$" {
    GENERATES MEASUREMENT {
    MEASUREMENTNAME "DEVELOPMENT_TEST1"
    GRANULARITY    15MIN
    MATCH {
        NEEDS "datarecord"->"shell"
    }

    COLUMNS {
        CONVERTATTRIBUTE MS_TO_LOCAL( "timeCaptured" )      "LOCALTIME"
        }
    }
}
```

3.6.5 Column operator - `CONVERTATTRIBUTE STRING_TO_UTC(P,S)`

The column operator

```
CONVERTATTRIBUTE STRING_TO_UTC( "<pointer>", "<format>" )
                                "<column name>"
```

generates a UTC timestamp (time zone Greenwich) from a date/time string.

`<pointer>` describes an element in the XML tree and has to be conclusively.

`<format>` describes the input date time format of the XML element. The format has to be compliant to the Qt date time specification Qt 4.7 date print function.

The output format of `STRING_TO_UTC` is defined by the global parameter `TIME_FORMAT` (see section 3.5.1).

The following example shows a definition for the operator `CONVERTATTRIBUTE STRING_TO_UTC( P, S )`:

```
INPUTFILE "^SAM_TEST.*\.xml$" {
    GENERATES MEASUREMENT {
    MEASUREMENTNAME "DEVELOPMENT_TEST1"
    GRANULARITY    15MIN
    MATCH {
        NEEDS "datarecord"->"shell"
    }

    COLUMNS {
        CONVERTATTRIBUTE STRING_TO_UTC ( "ISO8601TIME",
            "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )   "UTCTIME"
        }
    }
}
```

3.6.6 Column operator - CONVERTATTRIBUTE STRING_TO_LOCAL(P)

The column operator

```
CONVERTATTRIBUTE STRING_TO_LOCAL( "<pointer>", "<format>" )
                                  "<column name>"
```

generates a time date column using the local time zome from a date time string.

The converter xml2database uses the TZ environment variable to determin the time-zone.
<pointer> describes the XML element in the XML tree and has to be conclusively.

<format> describes the input date time format of the XML element. The format has to be compliant to the Qt date time specification Qt 4.7 date print function.

The output format of STRING_TO_UTC is defined by the global parameter TIME_FORMAT (see section 3.5.1).

The following example shows definition for the operator CONVERTATTRIBUTE STRING_TO_LOCAL(P, S) generating the column LOCALTIME:

```
INPUTFILE "^SAM_TEST.*\.xml$" {
    GENERATES MEASUREMENT {
    MEASUREMENTNAME "DEVELOPMENT_TEST1"
    GRANULARITY    15MIN
    MATCH {
        NEEDS "datarecord"->"shell"
    }

    COLUMNS {
        CONVERTATTRIBUTE STRING_TO_LOCAL( "ISO8601TIME",
              "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" ) "LOCALTIME"
        }
    }
}
```

3.6.7 Column operator - REGEXPATTRIBUTE(P, R)

The column operator

```
REGEXPATTRIBUTE( "<pointer>", "<RegExp>" ) "<column name>"
```

extracts data with the aid of a *regular expression* <RegExp>.

This operator is extremly powerful to perform complex string extractions like extractions of port/card numbers. The first subexpression is used for the data extraction of the column. In regular expressions the subexpression is marked with round brackets "()".

The following example shows a definition for the operator REGEXPATTRIBUTE(P, R) and generates three attributes SHELF, SLOT and PORT from the same raw element:

```
INPUTFILE "^SAM_TEST.*\.xml$" {
    GENERATES MEASUREMENT {
    MEASUREMENTNAME "DEVELOPMENT_TEST1"
    GRANULARITY    15MIN
    MATCH {
        NEEDS "datarecord"->"shell"
    }
```

```
    COLUMNS {
        REGEXPATTRIBUTE( "Port", "Port ([0-9]*)/[0-9]*/[0-9]*" )  "SHELF"
        REGEXPATTRIBUTE( "Port", "Port [0-9]*/([0-9]*)/[0-9]*" )  "SLOT"
        REGEXPATTRIBUTE( "Port", "Port [0-9]*/[0-9]*/([0-9]*)" )  "PORT"
        }
    }
}
```

3.6.8 Column operator - BOOLATTRIBUTE(P, S)

The operator

```
BOOLATTRIBUTE( "<pointer>", "<string>" ) "<column name>"
```

compares the raw data of the with <pointer> specified XML element with <String> and create "1" in case of equity and "0" in case of different values in the data column.

The following example shows the definition of a measurement fot the *operator* BOOLATTRIBUTE((P, S). For raw data with the value active the data column will be set to 1 otherwise 0:

```
INPUTFILE "^SAM_TEST.*\.xml$" {
    GENERATES MEASUREMENT {
    MEASUREMENTNAME "DEVELOPMENT_TEST1"
    GRANULARITY    15MIN
    MATCH {
        NEEDS "datarecord"->"shell"
    }

    COLUMNS {
        BOOLATTRIBUTE( "bits"->"subattribute", "active" ) "IS_ACTIVE"
        }
    }
}
```

3.6.9 Column operator - ARGATTRIBUTE(P, S)

The column operator

```
ARGATTRIBUTE( "<pointer>", "<argname>" ) "<column name>"
```

searches in the XML attribute <pointer> for the argument <argname> and extracts the data. The resulting column will be named as <column name>.

The example shows a definition for the operator ARGATTRIBUTE((P, S).:

```
INPUTFILE "^SAM_TEST.*\.xml$" {
    GENERATES MEASUREMENT {
    MEASUREMENTNAME "DEVELOPMENT_TEST1"
    GRANULARITY    15MIN
    MATCH {
        NEEDS "datarecord"->"shell"
    }

    COLUMNS {
        ARGATTRIBUTE( "attributeWithArgs", "arg3" )  "ARGATTR3"
        }
    }
}
```

3.7 Format of the output

All result files were stored into the directory `/opt/xml2database/output`. The filename will be constructed based on the `GLOBALS` parameter `FILE_MASK` (see section 3.5.1):

```
FILE_MASK "%3_%2_%4_%1.data"
# %1 = <JJMMTTHHMM> date
# %2 = <granularity> D,Q,E
# %3 = <measurement name>
# %4 = <Version>
```

The `<measurement name>` depend on the `INPUTFILE` definition so every `INPUTFILE` create its own file.

The separator between the data cells is defined by the `COLUMN_SEPARATOR` variable in the `GLOBALS` block (see section 3.5.1).

The records were separated by a single new line (”\n”) and cannot be modified.

The data cells will be filled as follows:

- If the XML raw element was found the data from the XML element or the processed data of the operator will be used.
- The empty string `""` will be set if the XML raw element does not has data or the if the referenced XML element cannot be found. Another reason could be in case of `REGEXPATTRIBTUE` that the regular expression does not match. In this case you will find warnings during processing.

  In most cases an empty field could be an indicator of a misconfiguration and needs to be investigated.

Es folgt ein Beispiel für eine Ergebnisdatei:

```
STIME1  Shelf   Slot    Port    STIME2  LTIME   ISO_UTC_TIME    ISO_LOCAL_TIME  IS_ACTIVE   TRANSMITTED RECEIVED
110401 12:34 1 2 3 110301 14:11 110301 15:11 1 2 123456789
110401 12:34 4 5 6 110301 14:11 110301 15:11 110518 05:30 110518 07:30 0 987654321 123456789
...
```

Chapter 4

Log Files

This chapter describes xml2database logging.

If you use the default xml2database environment, all log files were stored in /opt/xml2database/logs.

4.1 Log Files <date>_xml2database.log

The log files $HOME/logs/YYYY.MM.DD-HH:MM_xml2database.log stores all important hints, errors, warnings, debug information of the main process and its subprocesses.

The output is categorized in four classes:

- INFO: This type of message is only information and nothing serious.
- WARNING: Warnings indicate an unexpected state or result and should be noticed.
- ERROR: This type of message points to a serious problem and needs to be fixed.
- DEBUG: These messages were only generated if the debug switch (option -d <debug level>) was activated.

IMPORTANT: During normal operation the debug mode should not be used! Only the developer or for test analysis the debug switch should be activated on a test or reference system.

During the startup of the converter xml2database the main process writes converter version and used parameters:

```
[596] 2011.11.07 15:52:01     Reinhard Alt Unternehmensberatung
    xml2database - fast flexible data converter
    Version 0.8.1
    CVS: $Revision: 1.4 $
    CVS: $Date: 2012/02/08 15:57:37 $
    Made in Germany
[596] 2011.11.07 15:52:01 INFO: debug_level: 0
[596] 2011.11.07 15:52:01 INFO: mypid: 596
[596] 2011.11.07 15:52:01 INFO: parallel processes: 3
[596] 2011.11.07 15:52:01 INFO: configuration file: /opt/xml2db/config/xml2database.conf
[596] 2011.11.07 15:52:01 INFO: output_path: </opt/xml2db/output/>
[596] 2011.11.07 15:52:01 INFO: input_path: </opt/xml2db/input/>
[596] 2011.11.07 15:52:01 INFO: file mask: <%3_%2_%4_%1.data>
[596] 2011.11.07 15:52:01 INFO: time format: <yyMMdd hh:mm>
```

```
[596] 2011.11.07 15:52:01 INFO: column separator: <     >
[596] 2011.11.07 15:52:01 INFO: max header size: <30>
[596] 2011.11.07 15:52:01 INFO: xml2database will execute 3 processes parallel
```

The used parametes can be configure in the configuration file `$HOME/config/xml2database.conf`.

For every converted file a report summary is printed. The number in box brackets [ ] is the process number of either the main process or subprocess.

```
[597] 2011.11.07 15:52:01 INFO: processing file = TESTDATA_20110401_1234.xml
[597] 2011.11.07 15:52:01 INFO: parsing successful
```

At the end of the conversion process the main programm notes the number of successful converted files/total files.:

```
...
[596] 2011.11.07 15:52:01 Info: Main Program terminated
[596] 2011.11.07 15:52:01 Info (1/1) Prozesse erfolgreich konvertiert
```

Error messages will be marked with the prefix `ERROR` or `WARNING`. In case of problems it is useful to grep for these strings. The following example shows a `WARNING` that the filename does not contain a well formatted date/time file name extension:

```
[645] 2011.08.11 15:00:05 INFO: processing file = SAM_AGGSCHEDULER_DATE.xml
[646] 2011.08.11 15:00:05 ERROR: could not extract date/time from input file name - using current time
[646] 2011.08.11 15:00:05 INFO: input file extensions is not "_yyyyMMdd_hhmm.xml"
[646] 2011.08.11 15:00:05 INFO: parsing successful
[645] 2011.08.11 15:00:05 INFO: processing file = SAM_INVENTORY_DATE.xml
[647] 2011.08.11 15:00:05 ERROR: could not extract date/time from input file name - using current time
[647] 2011.08.11 15:00:05 INFO: input file extensions is not "_yyyyMMdd_hhmm.xml"
```

When using the debug mode (Option `-d <debug_level>`) all logs will be written to the log file. Using debug level 1 the parsing of the configuration file can be analyzed:

```
[656] 2011.11.07 15:08:53 INPUTFILE = ^TESTDATA.*\.xml$
[656] 2011.11.07 15:08:53     MEASUREMENT
[656] 2011.11.07 15:08:53         MEASUREMENTNAME = DEVELOPMENT_TEST1
[656] 2011.11.07 15:08:53         GRANULARITY: MIN15
[656] 2011.11.07 15:08:53         MATCH
[656] 2011.11.07 15:08:53             NEEDS
[656] 2011.11.07 15:08:53                 POINTER = datarecord
[656] 2011.11.07 15:08:53                 POINTER = shell
[656] 2011.11.07 15:08:53         COLUMNS
[656] 2011.11.07 15:08:53             UTC_FILESTAMP STIME1
[656] 2011.11.07 15:08:53             ATTRIBUTE_REGEXP ( POINTER, "Port ([0-9]*)/[0-9]*/[0-9]*" ) Shelf
[656] 2011.11.07 15:08:53                 POINTER = Port
[656] 2011.11.07 15:08:53             ATTRIBUTE_REGEXP ( POINTER, "Port [0-9]*/([0-9]*)/[0-9]*" ) Slot
[656] 2011.11.07 15:08:53                 POINTER = Port
[656] 2011.11.07 15:08:53             ATTRIBUTE_REGEXP ( POINTER, "Port [0-9]*/[0-9]*/([0-9]*)" ) Port
[656] 2011.11.07 15:08:53                 POINTER = Port
[656] 2011.11.07 15:08:53             ATTRIBUTE_CONVERT MS_TO_UTC( POINTER ) STIME2
[656] 2011.11.07 15:08:53                 POINTER = timeCaptured
[656] 2011.11.07 15:08:53             ATTRIBUTE_CONVERT MS_TO_LOCAL( POINTER ) LTIME
[656] 2011.11.07 15:08:53                 POINTER = timeCaptured
[656] 2011.11.07 15:08:53             ATTRIBUTE_CONVERT STRING_TO_UTC( POINTER, yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z' ) STIME
[656] 2011.11.07 15:08:53                 POINTER = StartTime
[656] 2011.11.07 15:08:53             ATTRIBUTE_CONVERT STRING_TO_LOCAL( POINTER, yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z' ) STIME
[656] 2011.11.07 15:08:53                 POINTER = StartTime
[656] 2011.11.07 15:08:53             ATTRIBUTE ( POINTER ) RECEIVED
[656] 2011.11.07 15:08:53                 POINTER = receivedTotalOctets
[656] 2011.11.07 15:08:53             ATTRIBUTE ( POINTER ) TRANSMITTED
[656] 2011.11.07 15:08:53                 POINTER = transmittedTotalOctets
[656] 2011.11.07 15:08:53             ATTRIBUTE_BOOL ( POINTER, "active" ) IS_ACTIVE
[656] 2011.11.07 15:08:53                 POINTER = bits
[656] 2011.11.07 15:08:53                 POINTER = subattribute
[656] 2011.11.07 15:08:53             ATTRIBUTE_ARG ( POINTER, "arg1" ) ARGATTR1
[656] 2011.11.07 15:08:53                 POINTER = attributeWithArgs
[656] 2011.11.07 15:08:53             ATTRIBUTE_ARG ( POINTER, "arg3" ) ARGATTR3
[656] 2011.11.07 15:08:53                 POINTER = attributeWithArgs
```

Chapter 5

Installation

This chapter describes all steps to install the software.

5.1 Preparation / create user / environment variables

For security reasons it is recommended not to use `root` permissions for `xml2db`. You should create a new user. The default installation path for the software is `/opt/xml2db`.

> Change the file ownership and permissions in `/opt/xml2db` and its subdirectories especially `/opt/xml2db/log` and `/opt/xml2db/output` to the created new user.

On Solaris the tool `admintool` might be used to add a new user.

`xml2db` needs several libraries and needs the environment variable `LD_LIBRARY_PATH` to include `/opt/xml2db/lib`. The environment variable `PATH` needs to be extended for `/opt/xml2db/bin` too.

In case of using the shell `bash` the file `/opt/xml2db/.profile` needs to be extended with the environment variables `PATH` and `LD_LIBRARY_PATH`.

```
> vi /opt/xml2db/.profile
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:\
/opt/xml2db/lib
export PATH=$PATH:/opt/xml2db/bin
```

5.2 Extraction of tar file / test start

The complete software, libraries, default configuration, directories will be extracted in `/opt/xml2db` (default). It is expected, that the tar archive of the software `xml2database-1.0.0.tar.gz` is found in `/tmp`.

To install the software for the user `xml2db` in `/opt/xml2db` the following commands needs to be performed:

```
> cd /opt/xml2fb
> gtar xvf /tmp/xml2database-1.0.0.tar.gz
> chown -R xml2db /opt/xml2db
```

To test the software start the converter with the option `-v`

```
xml2db@orgsunny: $ ./bin/xml2database -v
Reinhard Alt Unternehmensberatung
xml2database - fast flexible data converter
Version 0.8.1
CVS: $Revision:  1.4
CVS : $Date : 2012/02/0815 : 57 : 37
Made in Germany
```

5.3 Parameter of `call_xml2database`

The cron job `call_xml2database` allows automatic processing of data started by a cron job.

Befor the cron-job `call_xml2database` can be started several parameter need to be modified. The most important parameter is the `$HOME`.

Parameter description of `call_xml2database`:

- `HOME`: Points to the installation path of `xml2database` and needs to be configured. Default: `/opt/xml2db`
- `PATH`: Path where to find the binary files. Does not need to be modified.
- `LD_LIBRARY_PATH`: Path to the dynamic link libraries used by the converter `xml2database`. Does not need to be modified.
- `LOCKFILE`: To prevent multiple starts of `call_xml2database` a locking file is realized.
- `TZ`: timezone - for Europe MEZ.
- `CONFIG_FILE`: Used configuration file. Default: `$HOME/config/xml2database.conf`
- `LOGS`: Directory for log files. Does not need to be modified.
- `INPUT`: Input directory for XML raw files. Default: `$HOME/input`
- `OUTPUT`: Output directory for the converted results. Default: `$HOME/output`
- `DEBUGLEVEL`: Debug of the converter. 0 = off ... 5 = max.

  Important: In regular productive environment the debug level has to be set to 0 = off.

- `SUBPROCESSES`: Number of parallel running subprocesses. Possible values 1-20. Default 3.
- `FTP_SERVER`: IP Adresse of the database importer. Needs to be adapted.
- `FTP_USER`: sfpt user name. Needs to be adapted.
- `FTP_ROOTPATH`: sftp destination path. The directory needs to be adapted.

- MAILUSER: If the email notofication has been activated within the call_xml2database script the variable MAILUSER needs a fully qualified e-mail address.
- SFTPCOMMANDLIST: sftp uses a command list which is much faster than an iteration over all files to transfer them. The user id of xml2db needs write permissions to this temporary file.
- RETENTION_LOGS: Retention time for log files: Default 14 days
- RETENTION_INPUT: Retention time for raw files: Default 3 days
- RETENTION_OUTPUT: Retention time for results: Default 3 days

The following block shows the default configuration of the call_xml2database cron job.

Important: Edit only the marked area.

```
##########################################
# start here start here start here
##########################################

# specify the USER HOME PATH
export HOME=/opt/xml2database

# define some variables
# extend the bin path
export PATH=$PATH:/$HOME/bin

# extend the library search
#export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/lib
# Solaris 8
#export LD_LIBRARY_PATH=/usr/lib:$HOME/lib
# opensolaris
export LD_LIBRARY_PATH=$HOME/lib:/usr/lib:/usr/sfw/lib

# lock file
export LOCKFILE=$HOME/lock/xml2database-lock

# set the typical timezone
export TZ=MET

# config file
export CONFIG_FILE=$HOME/config/xml2database.conf

# log file prefix
# xml2database will add current date to the log file
export LOGS=$HOME/logs

# Eingangsdaten
export INPUT=$HOME/input

# Ergebnisse
export OUTPUT=$HOME/output

# compressor binary
export ZIPPRG=/usr/bin/gzip

# DEBUG Level
# 0 = off (default)
# 1 = minimal
# 2 = Parameter call
# 3 = Function calls
# 4 = very detailled
# 5 = maximum HANDS OFF !!!
export DEBUGLEVEL=0
```

```
# define the number of subprocesses (MAX = 20)
export SUBPROCESSES=3


# ftp server
export FTP_SERVER=servername

# ftp user
export FTP_USER=ftppassword

# ftp rootpath
export FTP_ROOTPATH=/ftp/input/dir

# mail user
export MAILUSER=xml2database@localhost

# filename containg ftp transfer commands
export SFTPCOMMANDLIST=/tmp/sftpcommands

# retention times for automatic purging in
# log, input or output directories
export RETENTION_LOGS=+14
export RETENTION_INPUT=+3
export RETENTION_OUTPUT=+3

##########################################
# STOP HERE STOP HERE STOP HERE
##########################################
```

5.4 `ssh` key generation

For a secure transfer to the database importer the sftp transfer tool is used.

The public key of the `xml2db` accounts needs to be added to the authorized key ring of the destination platform.

To generate a key enter the following command:

```
ssh-keygen -t dsa
or
ssh-keygen -t rsa
```

Accept the location where to save the key - enter <return>.

```
Generating public/private dsa key pair.
Enter file in which to save the key (/opt/xml2database/.ssh/id_dsa):
Created directory '/opt/xml2database/.ssh'.
```

For automatic transfer enter an empty pass phrase:

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /opt/xml2database/.ssh/id_dsa.
Your public key has been saved in /opt/xml2database/.ssh/id_dsa.pub.
```

In case of `dsa` the public key `id_dsa.pub` will be created and needs to be added to the keyring of the destination system:

```
on the destination server:
cat id_dsa-from-xml2database-server.pub » .ssh/authorized_keys2
```

Anschliessen muss einmalig ein händischer sftp login erfolgen mit dem ein Eintrag in der "'known_hosts"' Datei des Zielsystems erzeugt wird.

Test the connection to the target system:

```
> sftp username@target-system.earth.universe
```

Important: The last step needs to be performed to add the `xml2database` server to the list of known hosts.

Chapter 6

Error handling

This chapter decribes several error handling.

6.1 Program errors

In case of a program error the reason will be documented in the log messages `$HOME/logs/<Datum>_xml2database.log`. The reasons could be:

- not wirteable directories or files
- not readable configuration files
- unset environment variable `LD_LIBRARY_PATH`
- too many polling processes
- unexpected programm error

In case of an error the cron job the cron job `call_xml2database` needs to be commented out to prevent further conficts on the workstation. A first analysis can be done by using grep on the keyword `ERROR` in the most current log file.

6.2 Missing libraries

If the needet Qt libraries are missing or the `LD_LIBRARY_PATH` is not set correctly the following error message will appear

```
> xml2database -v
libc.so.1:  xml2database-OpenSolaris-V0.8.1:
fatal:  libQtXml.so.4:  open failed:  No such file or directory
Killed
```

Analyze the available/missing libraries by using `ldd`

```
> ldd bin/xml2database
```

The output gives you a hint about missing libraries:

```
libQtXml.so.4 => (file not found)
libQtCore.so.4 => (file not found)
libpthread.so.1 => /lib/libpthread.so.1
librt.so.1 => /lib/librt.so.1
libstdc++.so.6 => (file not found)
libm.so.2 => /lib/libm.so.2
libgcc_s.so.1 => (file not found)
libc.so.1 => /lib/libc.so.1
```

To solve this issue set the `LD_LIBRARY_PATH` in the resource file of the used shell. I.e. for `bash` set `LD_LIBRARY_PATH` in `.profile`.

6.3 Debug switches

In some cases it might be helpful to use DEBUG information during program execution. The option `-d <debug_level>` enables debugging. A higher value produces more debug information from the main process and subprocesses.

Basically the debug switch is only intended to by used by the programmer. Debug switches should only be used on test or reference systems.

The debug level depend on each: The higher debug level enable all lower debug levels. The debug enables following output:

Level	output
0	debug off
1	prints state changes and parsed configuration grammar
2	function call arguments
3	tracing functions
4	details function processing
5	low level details

6.4 Return codes

The following table shows the return codes send back after program termination. The return code $! = 0$ is a hint of a possible problem:

```
ERR_OK                    = 0,          // no error
ERR_COMMON                = 1,          // general error
ERR_OPEN_FILE             = 2,          // problem when opening a file
ERR_CLOSE_FILE            = 3,          // problem when closing a file
ERR_WRONG_OPTION          = 4,          // a program option generates a problem
ERR_NO_MEM                = 5,          // problem during memory allocation
ERR_WRONG_CONFIG          = 6,          // a configuration file has a problem
ERR_PIPE                  = 7,          // problem with a pipe() communication
ERR_FORK_FAILED           = 8,          // problem fork()ing a process
ERR_PROCESS               = 9,          // a process had a fatal problem
ERR_API                   = 10,         // a problem with the API occured
ERR_MSG_EMPTY             = 11,         // empty message received
ERR_MSG_LENGTH            = 12,         // the message was too long
ERR_PARSING_CONFIG        = 13,         // problem during parsing of the config file
ERR_OPEN_PIPE             = 14,         // problem when opening the pipe
ERR_CLOSE_PIPE            = 15,         // problem when closing the pipe
ERR_UNEXPECTED            = 16,         // an unexpected programming state reached
ERR_RANGE                 = 17,         // an array or vector range was hit
ERR_COMPONENT             = 18,         // problem adding, changing or deleting a component
ERR_OPENDB                = 19,         // could not opend DB
ERR_DBEXEC                = 20,         // problem during SQL statement execution
ERR_PATHCONSTRUCTION      = 21,         // problem during path construction
ERR_MEASUREMENT_NOT_FOUND = 22          // no measurement definition found
```

Appendix A

Files

A.1 File `xml2database.conf`

The following example configurations shows the possibilities of the converter. Unused measurment definitions should be commented out to reduce overhead during processing.

```
#
#
# (C) 2011 Reinhard Alt Unternehmensberatung
#          xml2database@arcor.de
#
# File:       xml2database.conf
# Funktion:   configuration file for xml2database
#
# Projekt:    xml2database - xml to database converter
# Autor:      Reinhard Alt Unternehmensberatung
#
# CVS:  Id: xml2database.conf,v 1.5 2011/11/10 13:54:29 reinhard Exp $
#       Revision: 1.5 $
#       Date: 2011/11/10 13:54:29 $
#       Author: reinhard $
#       Log: xml2database.conf,v $
#       Revision 1.5  2011/11/10 13:54:29  reinhard
#       - testrecord changed
#
#       Revision 1.4  2011/11/10 12:40:03  reinhard
#       - english version
#
#       Revision 1.3  2011/10/12 16:42:32  reinhard
#       - added some records
#
#       Revision 1.2  2011/10/06 14:04:27  reinhard
#       - more example measurements added
#
#       Revision 1.1  2011/09/23 09:55:51  reinhard
#       - initial version
#
#
#
#

###############################################################
#                                                             #
# tab width = 4 for a perfect layout !                        #
#                                                             #
###############################################################


#
# The GLOBALS block defines general parameters valid for all MEASUREMENTS
#
GLOBALS {
    #
    # The VERSION String is used in the file name of a measurement
    #
    VERSION "VERSION001"

    #
    # FILE_MASK
    # The output file names are generated by default as:
    #  <JJMMTTHHMM>_<granularity>_<measurementname>_<Version>
    #     arg 1          arg 2         arg 3        arg 4
    # %1 = <JJMMTTHHMM> Date year, month, day, hour, month
    # %2 = <granularity> possible values quantifier D=daily, Q=15min , E=event
    # %3 = <measurementname> name of the measurement
    # %4 = <Version> used version
    #
    # You can rearrange the position of the arguments but all arguments needs to be
```

```
    # used. Otherwise a Qt libary warning will appear like
    #
    # QString::arg: Argument missing: ...
    #
    #
    # default: "%1_%2_%3_%4.data"

    FILE_MASK "%3_%2_%4_%1.data"

    #
    # TIME_FORMAT
    #
    # This parameter defines the layout of DateTime fields in generated data
    # for format detauls see the Qt date print function
    # http://doc.qt.nokia.com/stable/qdatetime.html#toString
    #
     TIME_FORMAT "yyMMdd hh:mm"               # i.e. Sybase Format
    #TIME_FORMAT "yyyy.MM.dd hh:mm"           # i.e. Sqlite & Postgres
    #TIME_FORMAT "yyyy-MM-dd hh:mm:ss"       # i.e. Oracle

    #
    # COLUMN_SEPARATOR
    #
    # COLUMN_SEPARATOR is used to split the columns
    #

    COLUMN_SEPARATOR "    "                  # Tab Char
    #COLUMN_SEPARATOR ";"                   # semicolon
    #COLUMN_SEPARATOR " ; "                  # semicolon with spaces

    #
    # HEADER_SIZE_MAX
    #
    # HEADER_SIZE_MAX defines the size in chars of the largest possible table column name.
    # Data cells are not affected! Background is, that some databases allows limits
    # the number of chars in the table header
    #

    HEADER_SIZE_MAX    30         # i.e. Sybase
}


#
# Server Monitoring Records
#

INPUTFILE "^NetworkElementStats.xml$" {
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "GlobalElementStats"                          # Output Dateiname
        GRANULARITY     15MIN                                         # "15MIN"       -> "Q"
                                                                      # "24H"         -> "D"
        MATCH {
            NEEDS "Sample"
        }

        COLUMNS {
            # database keys
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
            "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )                      "STIME" # local time
                    CONVERTATTRIBUTE STRING_TO_LOCAL( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )              "LTIME" # local time
            ATTRIBUTE( "NeId" )                                       "NE_ID" # Nodename

            # Datenspalten
            ATTRIBUTE( "Sample"->"SessionCount" )                     "GLOBALSESSIONCOUNT" # SessionCount (global)
            ATTRIBUTE( "Sample"->"Capacity" )                         "capacityGlobal" # Capacity (global)
            ATTRIBUTE( "Sample"->"BandwidthUpstream" )                "BandwidthUpstreamGlobal" # Capacity (global)
            }
    }

    GENERATES MEASUREMENT {
        MEASUREMENTNAME "LocalElementStats"                           # Output Dateiname
        GRANULARITY     15MIN                                         # "15MIN"       -> "Q"
                                                                      # "24H"         -> "D"
        MATCH {
            NEEDS "Sample"->"Interface"
        }

        COLUMNS {
            # database keys
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )               "STIME"        # local time
            ATTRIBUTE( "NeId" )                                       "NE_ID"        # Nodename
            ATTRIBUTE( "Interface" -> "InterfaceName" )               "OBJINTERFACE" # Interface

            # Datenspalten
            # pro Interface
            ATTRIBUTE( "Interface" -> "SessionCount" )                "SessionCountInterface" # SessionCount interface
            ATTRIBUTE( "Interface" -> "Capacity" )                    "CapacityInterface"
            ATTRIBUTE( "Interface" -> "BandwidthUpstream" )           "BandwidthUpstreamInterface"
        }
    }
}
```

```
INPUTFILE "^DiameterAfPeerStats.xml$" {
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "DiameterAfPeerStats"                          # Output Dateiname
        GRANULARITY     15MIN                                          # "15MIN"       -> "Q"
                                                                       # "24H"         -> "D"
        MATCH {
            NEEDS "DiameterAfPeerStats" -> "Sample"
        }

        COLUMNS {
            # database keys
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )                "STIME" # local time
            ATTRIBUTE( "Name" )                                        "NE_ID" # Nodename

            # Datenspalten
            ATTRIBUTE( "MessagesInCount" )                             "MessagesInCount"
            ATTRIBUTE( "MessagesOutCount" )                            "MessagesOutCount"
            ATTRIBUTE( "ASRMessagesSentCount" )                        "ASRMessagesSentCount"
            ATTRIBUTE( "ASRMessagesReceivedCount" )                    "ASRMessagesReceivedCount"
            ATTRIBUTE( "ASASuccessMessagesReceivedCount" )             "ASASuccessMessagesReceivedCnt"
            ATTRIBUTE( "ASASuccessMessagesSentCount" )                 "ASASuccessMessagesSentCount"
            ATTRIBUTE( "ASAFailureMessagesReceivedCount" )             "ASAFailureMessagesReceivedCnt"
            ATTRIBUTE( "ASRMessagesReceivedCount" )                    "ASRMessagesReceivedCount"
            ATTRIBUTE( "ASAFailureMessagesSentCount" )                 "ASAFailureMessagesSentCount"
            ATTRIBUTE( "RARMessagesReceivedCount" )                    "RARMessagesReceivedCount"
            ATTRIBUTE( "RARMessagesSentCount" )                        "RARMessagesSentCount"
            ATTRIBUTE( "RAASuccessMessagesReceivedCount" )             "RAASuccessMessagesReceivedCnt"
            ATTRIBUTE( "RAASuccessMessagesSentCount" )                 "RAASuccessMessagesSentCount"
            ATTRIBUTE( "RAAFailureMessagesReceivedCount" )             "RAAFailureMessagesReceivedCnt"
            ATTRIBUTE( "RAAFailureMessagesSentCount" )                 "RAAFailureMessagesSentCount"
            ATTRIBUTE( "STRMessagesReceivedCount" )                    "STRMessagesReceivedCount"
            ATTRIBUTE( "STRMessagesSentCount" )                        "STRMessagesSentCount"
            ATTRIBUTE( "STASuccessMessagesReceivedCount" )             "STASuccessMessagesReceivedCnt"
            ATTRIBUTE( "STASuccessMessagesSentCount" )                 "STASuccessMessagesSentCount"
            ATTRIBUTE( "STAFailureMessagesReceivedCount" )             "STAFailureMessagesReceivedCnt"
            ATTRIBUTE( "STAFailureMessagesSentCount" )                 "STAFailureMessagesSentCount"
        }
    }
}


INPUTFILE "^DiameterAfStats.xml$" {
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "DiameterAfStats"                              # Output Dateiname
        GRANULARITY     15MIN                                          # "15MIN"       -> "Q"
                                                                       # "24H"         -> "D"
        MATCH {
            NEEDS "DiameterAfStats" -> "Sample"
        }

        COLUMNS {
            # database keys
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )                "STIME" # local time
            ATTRIBUTE( "PolicyServer" )                                "NE_ID" # Nodename

            # Datenspalten
            ATTRIBUTE( "PendingConnectionsCount" )                     "PendingConnectionsCount"
            ATTRIBUTE( "CurrentConnectionsCount" )                     "CurrentConnectionsCount"
            ATTRIBUTE( "MessagesInCount" )                             "MessagesInCount"
            ATTRIBUTE( "MessagesOutCount" )                            "MessagesOutCount"
            ATTRIBUTE( "ASRMessagesReceivedCount" )                    "ASRMessagesReceivedCount"
            ATTRIBUTE( "ASRMessagesSentCount" )                        "ASRMessagesSentCount"
            ATTRIBUTE( "ASASuccessMessagesReceivedCount" )             "ASASuccessMessagesReceivedCnt"
            ATTRIBUTE( "ASASuccessMessagesSentCount" )                 "ASASuccessMessagesSentCount"
            ATTRIBUTE( "ASAFailureMessagesReceivedCount" )             "ASAFailureMessagesReceivedCnt"
            ATTRIBUTE( "ASAFailureMessagesSentCount" )                 "ASAFailureMessagesSentCount"
            ATTRIBUTE( "RARMessagesReceivedCount" )                    "RARMessagesReceivedCount"
            ATTRIBUTE( "RARMessagesSentCount" )                        "RARMessagesSentCount"
            ATTRIBUTE( "RAASuccessMessagesReceivedCount" )             "RAASuccessMessagesReceivedCnt"
            ATTRIBUTE( "RAASuccessMessagesSentCount" )                 "RAASuccessMessagesSentCount"
            ATTRIBUTE( "RAAFailureMessagesSentCount" )                 "RAAFailureMessagesSentCount"
            ATTRIBUTE( "STRMessagesReceivedCount" )                    "STRMessagesReceivedCount"
            ATTRIBUTE( "STRMessagesSentCount" )                        "STRMessagesSentCount"
            ATTRIBUTE( "STASuccessMessagesReceivedCount" )             "STASuccessMessagesReceivedCnt"
            ATTRIBUTE( "STASuccessMessagesSentCount" )                 "STASuccessMessagesSentCount"
            ATTRIBUTE( "STAFailureMessagesReceivedCount" )             "STAFailureMessagesReceivedCnt"
            ATTRIBUTE( "STAFailureMessagesSentCount" )                 "STAFailureMessagesSentCount"
            ATTRIBUTE( "AARMessagesReceivedCount" )                    "AARMessagesReceivedCount"
            ATTRIBUTE( "AARMessagesSentCount" )                        "AARMessagesSentCount"
            ATTRIBUTE( "AAASuccessMessagesReceivedCount" )             "AAASuccessMessagesReceivedCnt"
            ATTRIBUTE( "AAASuccessMessagesSentCount" )                 "AAASuccessMessagesSentCount"
            ATTRIBUTE( "AAAFailureMessagesSentCount" )                 "AAAFailureMessagesSentCount"
            ATTRIBUTE( "ActiveSessionsCount" )                         "ActiveSessionsCount"
            ATTRIBUTE( "MaximumActiveSessionsCount" )                  "MaximumActiveSessionsCount"
            ATTRIBUTE( "PeerOkayCount" )                               "PeerOkayCount"
            ATTRIBUTE( "PeerDownCount" )                               "PeerDownCount"
            ATTRIBUTE( "PeerSuspectCount" )                            "PeerSuspectCount"
            ATTRIBUTE( "PeerReopenCount" )                             "PeerReopenCount"
        }
    }
}
```

```
INPUTFILE "^DiameterCTFStats.xml$" {
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "DiameterCTFStats"                          # Output Dateiname
        GRANULARITY     15MIN                                       # "15MIN"       -> "Q"
                                                                    # "24H"         -> "D"
        MATCH {
            NEEDS "DiameterCTFStats" -> "Sample"
        }

        COLUMNS {
            # database keys
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )             "STIME" # local time
            ATTRIBUTE( "PolicyServer" )                             "NE_ID" # Nodename

            # Datenspalten
            ATTRIBUTE( "CurrentConnectionsCount" )                  "CurrentConnectionsCount"
            ATTRIBUTE( "PeerOkayCount" )                            "PeerOkayCount"
            ATTRIBUTE( "PeerDownCount" )                            "PeerDownCount"
            ATTRIBUTE( "PeerSuspectCount" )                         "PeerSuspectCount"
            ATTRIBUTE( "PeerReopenCount" )                          "PeerReopenCount"
            ATTRIBUTE( "MessagesInCount" )                          "MessagesInCount"
            ATTRIBUTE( "MessagesOutCount" )                         "MessagesOutCount"
            ATTRIBUTE( "CCRMessagesReceivedCount" )                 "CCRMessagesReceivedCount"
            ATTRIBUTE( "CCRMessagesSentCount" )                     "CCRMessagesSentCount"
            ATTRIBUTE( "CCASuccessMessagesReceivedCount" )          "CCASuccessMessagesReceivedCnt"
            ATTRIBUTE( "CCASuccessMessagesSentCount" )              "CCASuccessMessagesSentCount"
            ATTRIBUTE( "CCAFailureMessagesReceivedCount" )          "CCAFailureMessagesReceivedCnt"
            ATTRIBUTE( "CCAFailureMessagesSentCount" )              "CCAFailureMessagesSentCount"
            ATTRIBUTE( "RARMessagesReceivedCount" )                 "RARMessagesReceivedCount"
            ATTRIBUTE( "RARMessagesSentCount" )                     "RARMessagesSentCount"
            ATTRIBUTE( "RAASuccessMessagesReceivedCount" )          "RAASuccessMessagesReceivedCnt"
            ATTRIBUTE( "RAASuccessMessagesSentCount" )              "RAASuccessMessagesSentCount"
            ATTRIBUTE( "RAAFailureMessagesReceivedCount" )          "RAAFailureMessagesReceivedCnt"
            ATTRIBUTE( "RAAFailureMessagesSentCount" )              "RAAFailureMessagesSentCount"
            ATTRIBUTE( "ASRMessagesReceivedCount" )                 "ASRMessagesReceivedCount"
            ATTRIBUTE( "ASRMessagesSentCount" )                     "ASRMessagesSentCount"
            ATTRIBUTE( "ASASuccessMessagesReceivedCount" )          "ASASuccessMessagesReceivedCnt"
            ATTRIBUTE( "ASASuccessMessagesSentCount" )              "ASASuccessMessagesSentCount"
            ATTRIBUTE( "ASAFailureMessagesReceivedCount" )          "ASAFailureMessagesReceivedCnt"
            ATTRIBUTE( "ASAFailureMessagesSentCount" )              "ASAFailureMessagesSentCount"
            ATTRIBUTE( "ActiveSessionsCount" )                      "ActiveSessionsCount"
            ATTRIBUTE( "MaximumActiveSessionsCount" )               "MaximumActiveSessionsCount"
        }
    }
}

INPUTFILE "^DiameterPcefPeerStats.xml$" {
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "DiameterPcefPeerStats"                     # Output Dateiname
        GRANULARITY     15MIN                                       # "15MIN"       -> "Q"
                                                                    # "24H"         -> "D"
        MATCH {
            NEEDS "DiameterPcefPeerStats" -> "Sample"
        }

        COLUMNS {
            # database keys
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )             "STIME" # local time
            ATTRIBUTE( "Name" )                                     "NE_ID" # Nodename

            # Datenspalten
            ATTRIBUTE( "NetworkElementType" )                       "NetworkElementType"
            ATTRIBUTE( "NetworkElementSubType" )                    "NetworkElementSubType"
            ATTRIBUTE( "ConnectTime" )                              "ConnectTime"
            ATTRIBUTE( "DisconnectTime" )                           "DisconnectTime"
            ATTRIBUTE( "ConnectAddress" )                           "ConnectAddress"
            ATTRIBUTE( "ConnectPort" )                              "ConnectPort"
            ATTRIBUTE( "MessagesInCount" )                          "MessagesInCount"
            ATTRIBUTE( "MessagesOutCount" )                         "MessagesOutCount"
            ATTRIBUTE( "RARMessagesReceivedCount" )                 "RARMessagesReceivedCount"
            ATTRIBUTE( "RARMessagesSentCount" )                     "RARMessagesSentCount"
            ATTRIBUTE( "RAASuccessMessagesReceivedCount" )          "RAASuccessMessagesReceivedCnt"
            ATTRIBUTE( "RAASuccessMessagesSentCount" )              "RAASuccessMessagesSentCount"
            ATTRIBUTE( "RAAFailureMessagesReceivedCount" )              "RAAFailureMessagesReceivedCnt"
            ATTRIBUTE( "RAAFailureMessagesSentCount" )              "RAAFailureMessagesSentCount"
            ATTRIBUTE( "CCRMessagesReceivedCount" )                 "CCRMessagesReceivedCount"
            ATTRIBUTE( "CCRMessagesSentCount" )                     "CCRMessagesSentCount"
            ATTRIBUTE( "CCASuccessMessagesReceivedCount" )          "CCASuccessMessagesReceivedCnt"
            ATTRIBUTE( "CCASuccessMessagesSentCount" )              "CCASuccessMessagesSentCount"
            ATTRIBUTE( "CCAFailureMessagesReceivedCount" )          "CCAFailureMessagesReceivedCnt"
            ATTRIBUTE( "CCAFailureMessagesSentCount" )              "CCAFailureMessagesSentCount"
            ATTRIBUTE( "ActiveSessionsCount" )                      "ActiveSessionsCount"
            ATTRIBUTE( "MaximumActiveSessionsCount" )               "MaximumActiveSessionsCount"
        }
    }
}

INPUTFILE "^DiameterPcefStats.xml$" {
```

```
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "DiameterPcefStats"                          # Output Dateiname
        GRANULARITY     15MIN                                        # "15MIN"       -> "Q"
                                                                     # "24H"         -> "D"
        MATCH {
            NEEDS "DiameterPcefStats" -> "Sample"
        }

        COLUMNS {
            # database keys
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )              "STIME" # local time
            ATTRIBUTE( "PolicyServer" )                              "NE_ID" # Nodename

            # Datenspalten
            ATTRIBUTE( "CurrentConnectionsCount" )                   "CurrentConnectionsCount"
            ATTRIBUTE( "MessagesInCount" )                           "MessagesInCount"
            ATTRIBUTE( "MessagesOutCount" )                          "MessagesOutCount"
            ATTRIBUTE( "RARMessagesReceivedCount" )                  "RARMessagesReceivedCount"
            ATTRIBUTE( "RARMessagesSentCount" )                      "RARMessagesSentCount"
            ATTRIBUTE( "RAASuccessMessagesReceivedCount" )           "RAASuccessMessagesReceivedCnt"
            ATTRIBUTE( "RAASuccessMessagesSentCount" )               "RAASuccessMessagesSentCount"
            ATTRIBUTE( "RAAFailureMessagesReceivedCount" )           "RAAFailureMessagesReceivedCnt"
            ATTRIBUTE( "RAAFailureMessagesSentCount" )               "RAAFailureMessagesSentCount"
            ATTRIBUTE( "CCRMessagesReceivedCount" )                  "CCRMessagesReceivedCount"
            ATTRIBUTE( "CCRMessagesSentCount" )                      "CCRMessagesSentCount"
            ATTRIBUTE( "CCASuccessMessagesReceivedCount" )           "CCASuccessMessagesReceivedCnt"
            ATTRIBUTE( "CCASuccessMessagesSentCount" )               "CCASuccessMessagesSentCount"
            ATTRIBUTE( "CCAFailureMessagesReceivedCount" )           "CCAFailureMessagesReceivedCnt"
            ATTRIBUTE( "CCAFailureMessagesSentCount" )               "CCAFailureMessagesSentCount"
            ATTRIBUTE( "ActiveSessionsCount" )                       "ActiveSessionsCount"
            ATTRIBUTE( "MaximumActiveSessionsCount" )                "MaximumActiveSessionsCount"
            ATTRIBUTE( "PeerOkayCount" )                             "PeerOkayCount"
            ATTRIBUTE( "PeerDownCount" )                             "PeerDownCount"
            ATTRIBUTE( "PeerSuspectCount" )                          "PeerSuspectCount"
            ATTRIBUTE( "PeerReopenCount" )                           "PeerReopenCount"
        }
    }
}


INPUTFILE "^PolicyServerStats.xml$" {
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "PolicyServerStats"                          # Output Dateiname
        GRANULARITY     15MIN                                        # "15MIN"       -> "Q"
                                                                     # "24H"         -> "D"
        MATCH {
            NEEDS "PolicyServerStats" -> "Sample"
        }

        COLUMNS {
            # database keys
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )              "STIME"  # local time
            ATTRIBUTE( "PCRF" )                                      "NE_ID"  # Nodename

            # Datenspalten
            ATTRIBUTE( "TotalNetworkElementCount" )                  "TotalNetworkElementCount"
            ATTRIBUTE( "TotalSubscriberCount" )                      "TotalSubscriberCount"
        }
    }
}



INPUTFILE "^TopologyUpdateStats.xml$" {
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "TopologyUpdateStats"                        # Output Dateiname
        GRANULARITY     15MIN                                        # "15MIN"       -> "Q"
                                                                     # "24H"         -> "D"
        MATCH {
            NEEDS "TopologyUpdateStats" -> "Sample"
        }

        COLUMNS {
            # database keys
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )              "STIME" # local time

            # Datenspalten
            ATTRIBUTE( "TopologyUpdateCount" )                       "TopologyUpdateCount"
            ATTRIBUTE( "TopologyUpdateFailCount" )                   "TopologyUpdateFailCount"
        }
    }
}


#
# Software testing
#
INPUTFILE "^TESTDATA.*\.xml$" {                                      # data source
    GENERATES MEASUREMENT {
        MEASUREMENTNAME "DEVELOPMENT_TEST1"                          # name of the measurement
        GRANULARITY    15MIN                                         # "15MIN"     -> "Q"
```

```
                                                                    # "24H"        -> "D"
                                                                    # "EVENT"   -> "E"
        MATCH {
            NEEDS "datarecord"->"shell"
        }

        COLUMNS {
            UTC_FILESTAMP                                           "UTCTIME" # UTC from filename
            REGEXPATTRIBUTE( "Port", "Port ([0-9]*)/[0-9]*/[0-9]*" )    "Shelf" # extract shelf
            REGEXPATTRIBUTE( "Port", "Port [0-9]*/([0-9]*)/[0-9]*" )    "Slot" # extract slot
            REGEXPATTRIBUTE( "Port", "Port [0-9]*/[0-9]*/([0-9]*)" )    "Port" # extract port
            CONVERTATTRIBUTE MS_TO_UTC(    "timeCaptured" )         "UTCTIME1" # UTC time from milliseconds
            CONVERTATTRIBUTE MS_TO_LOCAL( "timeCaptured" )          "LTIME1" # localtime from milliseconds
            CONVERTATTRIBUTE STRING_TO_UTC( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )            "ISO_UTC_TIME" # UTC time
            CONVERTATTRIBUTE STRING_TO_LOCAL( "StartTime",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )            "ISO_LOCAL_TIME" # local time
            ATTRIBUTE( "receivedTotalOctets" )                      "RECEIVED"    #
            ATTRIBUTE( "transmittedTotalOctets" )                   "TRANSMITTED" #
            BOOLATTRIBUTE( "bits"->"subattribute", "active" )       "IS_ACTIVE" # 1 = if "active" else 0
            ARGATTRIBUTE( "attributeWithArgs", "arg1" )             "ARGATTR1"
            ARGATTRIBUTE( "attributeWithArgs", "arg3" )             "ARGATTR3"
        }
    }

    GENERATES MEASUREMENT {
        MEASUREMENTNAME "DEVELOPMENT_TEST2"                         # Name der Messung
        GRANULARITY    24H                                          # "15MIN"     -> "Q"
                                                                    # "24H"        -> "D"
                                                                    # "EVENT"   -> "E"
        MATCH {
            NEEDS "data"->"shell"->"bits"
        }

        COLUMNS {
            UTC_FILESTAMP                                           "UTCTIME" # UTC from filename
            REGEXPATTRIBUTE( "Port", "Port ([0-9]*)/[0-9]*/[0-9]*" )    "Shelf"   # extract shelf
            REGEXPATTRIBUTE( "Port", "Port [0-9]*/([0-9]*)/[0-9]*" )    "Slot"    # extract slot
            REGEXPATTRIBUTE( "Port", "Port [0-9]*/[0-9]*/([0-9]*)" )    "Port"    # extract port
            CONVERTATTRIBUTE MS_TO_UTC(    "timeCaptured" )         "UTCTIME1" # UTC time from milliseconds
            CONVERTATTRIBUTE MS_TO_LOCAL( "timeCaptured" )          "LTIME1"  # localtime from milliseconds
            CONVERTATTRIBUTE STRING_TO_UTC ( "ISO8601TIME",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )            "ISO_UTC_TIME"  # UTC
            CONVERTATTRIBUTE STRING_TO_LOCAL( "ISO8601TIME",
                    "yyyy'-'MM'-'dd'T'hh':'mm':'ss'Z'" )            "ISO_LOCAL_TIME"  # localtime
            BOOLATTRIBUTE( "bits"->"subattribute", "active" )       "IS_ACTIVE" # 1 = if "active" else 0
            ATTRIBUTE( "transmittedTotalOctets" )                   "TRANSMITTED" #
            ATTRIBUTE( "receivedTotalOctets" )                      "RECEIVED"   #
            #ATTRIBUTE( "nix" )                                     "12345678901234567890123456789012345678901"    # 31 Zeichen --> error message
        }
    }
}


####### End of File
```

A.2 Cron job call_xml2database.conf

The following cron job call_xml2database might be used as a template for own customer specific cron jobs.

```
#!/usr/bin/bash
# File: call_xml2database
# call script to start conversion of xml data into cvs/database format
#
# input files need to be in $HOME/input
# results were stored in $HOME/output
#
# syntax:
#       call_xml2database
#
# CVS:  Id: call_xml2database,v 1.1 2011/11/10 17:48:56 reinhard Exp $
#       Revision: 1.1 $
#       Date: 2011/11/10 17:48:56 $
#       Author: reinhard $
#       Log: call_xml2database,v $
#       Revision 1.1  2011/11/10 17:48:56  reinhard
#       - initial version
#
#
#

##########################################
# start here start here start here
##########################################

# specify the USER HOME PATH
export HOME=/opt/xml2db

# define some variables
```

```
# extend the bin path
export PATH=$PATH:/$HOME/bin

# extend the library search
#export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/lib
# Solaris 8
#export LD_LIBRARY_PATH=/usr/lib:$HOME/lib
# opensolaris
export LD_LIBRARY_PATH=$HOME/lib:/usr/lib:/usr/sfw/lib

# lock file
export LOCKFILE=$HOME/lock/xml2database-lock

# set the typical timezone
export TZ=MET

# config file
export CONFIG_FILE=$HOME/config/xml2database.conf

# log file prefix
# xml2database will add current date to the log file
export LOGS=$HOME/logs

# Input data
export INPUT=$HOME/input

# conversion results
export OUTPUT=$HOME/output

# compressor binary
export ZIPPRG=/usr/bin/gzip

# DEBUG Level
# 0 = off (default)
# 1 = minimal
# 2 = Parameter call
# 3 = Function calls
# 4 = very detailled
# 5 = maximum HANDS OFF !!!
export DEBUGLEVEL=0

# define the number of subprocesses (MAX = 20)
export SUBPROCESSES=3


# ftp server
export FTP_SERVER=servername

# ftp user
export FTP_USER=ftppassword

# ftp rootpath
export FTP_ROOTPATH=/ftp/input/dir

# mail user
export MAILUSER=xml2database@localhost

# filename containg ftp transfer commands
export SFTPCOMMANDLIST=/tmp/sftpcommands

# retention times for automatic purging in
# log, input or output directories
export RETENTION_LOGS=+14
export RETENTION_INPUT=+3
export RETENTION_OUTPUT=+3

##############################################
# STOP HERE STOP HERE STOP HERE
##############################################

if [ -f $LOCKFILE ] ; then \
        echo "the converter is still running"
        echo "or a cron job is still running"
        echo "or you have to clean the lock file $LOCKFILE manually"
        exit 0
fi

touch $LOCKFILE

##############################
export CURRENT_DATE=`date +%Y.%m.%d-%H:%M`

##############################
# define functions
##############################

### check_return()

check_return() {
    if [ $? == 0 ] ; then
            echo " "
            echo "conversion successful"
            echo " "
    else
```

```
            STATUS="NOK"
            echo " "
            echo "conversion FAILED"
            echo "Details in ${LOGS}"
            echo " "
            # exit 1;
    fi
}


process_data () {
    STATUS="OK"

        export CURRENT_DATE=`date +%Y.%m.%d-%H:%M`
        echo "`date` processing all files in ${INPUT}"
        time xml2database -d ${DEBUGLEVEL} \
            -t ${SUBPROCESSES} \
            -i ${INPUT} \
            -o ${OUTPUT} \
            -c ${CONFIG_FILE} \
                    >${LOGS}/${CURRENT_DATE}_xml2database.log 2>&1
        check_return;

        if [ $STATUS == "NOK" ] ; then
                # starte here some other signaling stuff
        echo $CURRENT_DATE
        fi
}


send_negative_mail() {
export DATE=`date`
/usr/bin/mail ${MAILUSER}<<EOF
Subject: xml2database conversion failed

#################### call_xml2database ####################
Achtung!!!

The xml2database conversion of $DATE FAILED
Call the skript

       call_xml2database

manually and check the result.

Further details can be found in the logging directory
Verzeichnis:
${LOGS}

.
EOF
}


send_positive_mail() {
export DATE=`date`
/usr/bin/mail ${MAILUSER} <<EOF
Subject: xml2database conversion successful

#################### call_xml2database ####################

The xml2database conversion of $DATE was successful!

.
EOF
}


####################################
# start conversion
####################################

date
echo "Starte Konvertierung"
process_data ;


####################################
# compress output data
####################################

date
echo "compressing data in ${OUTPUT}"
$ZIPPRG -f ${OUTPUT}/*


####################################
# sftp the stuff
####################################

# use sftp
    echo "now, transferring via sftp"
    cd ${OUTPUT}
    # create transfer command list
```

```
    rm $SFTPCOMMANDLIST
    echo "cd ${FTP_ROOTPATH}" > $SFTPCOMMANDLIST
    for FILE in * ; do \
        if [ -f $FILE ] ; then \
            echo "put $FILE $FILE.tmp" >> $SFTPCOMMANDLIST
        fi
    done
    echo "starting file transfer for ${FTP_SERVER} ..."
    sftp -b $SFTPCOMMANDLIST ${FTP_USER}@${FTP_SERVER} >> ${LOGS}/${CURRENT_DATE}_sftp.log ;\
    if [ $? != 0 ] ; then\
        echo "ERROR could transfer one or more file(s)" >>${LOGS}/${CURRENT_DATE}_sftp.log
    else
        # prepare renaming files
        echo "cd ${FTP_ROOTPATH}" > $SFTPCOMMANDLIST
        for FILE in * ; do \
            if [ -f $FILE ] ; then \
                echo "rename $FILE.tmp $FILE" >> $SFTPCOMMANDLIST
            fi
        done
        echo "starting renaming for ${FTP_SERVER} ..."
        sftp -b $SFTPCOMMANDLIST ${FTP_USER}@${FTP_SERVER} >> ${LOGS}/${CURRENT_DATE}_sftp.log ;\
        if [ $? != 0 ] ; then\
            echo "ERROR could rename one or more file(s)" >>${LOGS}/${CURRENT_DATE}_sftp.log
        else
            # remove files in $OUTPUT
            rm ${OUTPUT}/*
        fi
    fi
date

##########################
# start cleanup
##########################

echo "starting some cleanups"
# clean aged log files
find ${LOGS} -mtime ${RETENTION_LOGS} -exec rm {} \;

# clean aged input files
find ${INPUT} -mtime ${RETENTION_INPUT} -exec rm {} \;

# clean aged output files
find ${OUTPUT} -mtime ${RETENTION_OUTPUT} -exec rm {} \;

# clear lock file
rm $LOCKFILE

echo "finished ..."

#EOF
```

A.3 `xml2database` configuration grammar

Excerpt from the bison grammer of `xml2database` configuration system.

```
Grammar

    0 $accept: input $end

    1 input: GLOBALS '{' list_gl_attribute '}' list_inputfile

    2 list_gl_attribute: list_gl_attribute gl_attribute
    3                  | /* empty */

    4 gl_attribute: VERSION STRING
    5             | FILE_MASK STRING
    6             | TIME_FORMAT STRING
    7             | COLUMN_SEPARATOR STRING
    8             | HEADER_SIZE_MAX INTEGER

    9 list_inputfile: list_inputfile inputfile
   10               | /* empty */

   11 inputfile: INPUTFILE STRING '{' list_measurement '}'

   12 list_measurement: list_measurement measurement
   13                 | /* empty */

   14 measurement: GENERATES MEASUREMENT '{' list_measurement_element '}'

   15 list_measurement_element: list_measurement_element measurement_element
   16                         | /* empty */

   17 measurement_element: MEASUREMENTNAME STRING
   18                    | GRANULARITY H24
   19                    | GRANULARITY MIN15
   20                    | GRANULARITY EVENT
   21                    | MATCH '{' list_match_elements '}'
   22                    | COLUMNS '{' list_columns '}'
```

```
23 list_match_elements: list_match_elements match_element
24                    | /* empty */

25 match_element: NEEDS list_hierarchy
26              | AND NEEDS list_hierarchy
27              | VOID

28 list_hierarchy: list_hierarchy POINTER hierarchy_element
29               | hierarchy_element
30               | /* empty */

31 hierarchy_element: STRING

32 list_columns: list_columns column
33             | /* empty */

34 column: UTC_FILESTAMP STRING
35       | ATTRIBUTE_CONVERT MS_TO_UTC '(' list_hierarchy ')' STRING
36       | ATTRIBUTE_CONVERT MS_TO_LOCAL '(' list_hierarchy ')' STRING
37       | ATTRIBUTE_CONVERT STRING_TO_LOCAL '(' list_hierarchy ',' STRING ')' STRING
38       | ATTRIBUTE_CONVERT STRING_TO_UTC '(' list_hierarchy ',' STRING ')' STRING
39       | ATTRIBUTE_NORMAL '(' list_hierarchy ')' STRING
40       | ATTRIBUTE_BOOL '(' list_hierarchy ',' STRING ')' STRING
41       | ATTRIBUTE_REGEXP '(' list_hierarchy ',' STRING ')' STRING
42       | ATTRIBUTE_ARG '(' list_hierarchy ',' STRING ')' STRING
```

List of Figures