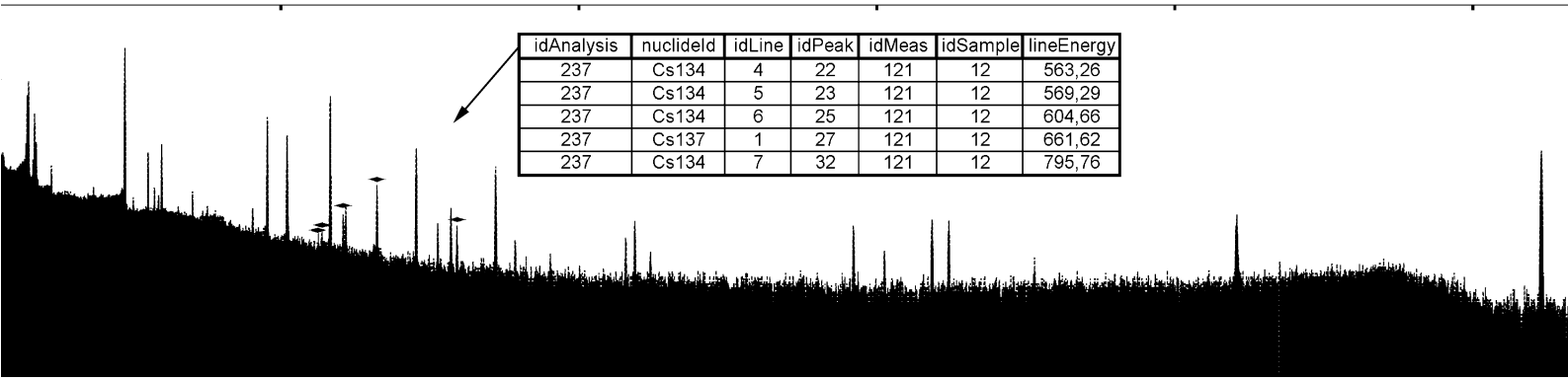
Helsinki University of Technology Publications in Engineering Physics. A
Teknillisen korkeakoulun teknillisen fysiikan julkaisuja. A
Espoo 2006 TKK-F-A842

LINSSI
SQL DATABASE FOR GAMMA-RAY SPECTROMETRY
PART II: SCRIPTS AND INTERFACES
Version 1.1

Jarmo Ala-Heikkilä



TEKNILLINEN KORKEAKOULU
HELSINKI UNIVERSITY OF TECHNOLOGY

Helsinki University of Technology Publications in Engineering Physics. A
Teknillisen korkeakoulun teknillisen fysiikan julkaisuja. A
Espoo 2006 TKK-F-A842

LINSSI
SQL DATABASE FOR GAMMA-RAY SPECTROMETRY
PART II: SCRIPTS AND INTERFACES
Version 1.1

Jarmo Ala-Heikkilä

Helsinki University of Technology
Department of Engineering Physics and Mathematics
Laboratory of Advanced Energy Systems

Teknillinen korkeakoulu
Teknillisen fysiikan ja matematiikan osasto
Energiateknologiat

Database and Software

Pertti Aarnio[1], Jarmo Ala-Heikkilä[1], Arto Isolankila[2], Antero Kuusi[2], Mikael Moring[2], Mika Nikkinen[2], Tommi Salonen[2], Teemu Siiskonen[2], Harri Toivonen[2], Kurt Ungar[3], Weihua Zhang[3]

[1]Helsinki University of Technology, Radiation Physics Group
[2]Finnish Radiation and Nuclear Safety Authority
[3]Health Canada, Radiation Protection Bureau

Manual

Jarmo Ala-Heikkilä

Distribution:
Helsinki University of Technology
Laboratory of Advanced Energy Systems
P.O. Box 4100
FI-02015 TKK

ISBN 951-22-8185-6
ISSN 1456-3320

For the latest version of this manual see:
http://linssi.hut.fi/radphys/linssi

Information in this document is subject to change without notice and does not represent any commitment on the part of the authors. The software described in this document is furnished under a license agreement. The user may not copy the software on magnetic or optical tape, disk or any other medium, for any other purpose than the license holder's personal use.

Copyright

License

License conditions are defined in a separate document that must be consulted.

Disclaimer of Responsibility for the Software

Any data may be defined in one place only.
If data are defined in more places, they will diverge (it will not stay the same, if it ever was). If data are changed while not in one place only, you never know whether you changed every instance. However, you need a method (document control) that assures that all places where the changed data is referenced from are informed of any change. Relational databases use this principle. The same applies to software: every function should be defined only once (it would have made the millennium problem a piece of cake!).
Niels R. Malotaux

Contents

Chapter 1

Introduction

This document is targeted to people who know what the SQL database for gamma-ray spectrometry called *Linssi* is. *Linssi* is documented in a comprehensive manual [1]. Please read at least the chapter "Introduction" of the database manual carefully before proceeding with this document, as well as its appendix on naming conventions.

The purpose of this manual is to document the scripts available in the *Linssi* package and how *Linssi* can be interfaced with software packages for gamma-ray spectrometry. The scripts are self-documenting: they are requested to start with a comment block that explains their purpose, syntax, history and relevant information about their action. These comments have been utilized in generating this manual, with the aim to present a general view of the complete package.

All *Linssi* scripts in the package are not necessarily documented here, because new scripts may be added to the distribution more frequently than this document is updated. An up-to-date list of the scripts will be maintained on the *Linssi* home page
`http://linssi.hut.fi/radphys/linssi/`.

The UniSampo–Shaman (USS) analysis package is documented in comprehensive manuals [2, 3]. It has been interfaced with *Linssi*, UniSampo since the first database version and Shaman since *Linssi* v.0.9 in July 2003. As a consequence, functionality of the analysis result tables of *Linssi* has been tested most thoroughly.

Nevertheless, it should be stressed that the development goal of *Linssi* has been to be as generic as possible so that it can be connected with gamma-ray spectrum analysis software packages other than UniSampo and Shaman. As an illustration, the spectrum analysis tool *Aatami* developed by the Evaluation Section of the CTBTO was interfaced with *Linssi* in December 2004. The occurrences of UniSampo and Shaman in this document shall mainly be understood as examples. Additionally, this document is meant to serve as a documentation of the functional UniSampo-Shaman-*Linssi*-system at STUK.

We have chosen MySQL as the database engine. However, a *Linssi* database can be implemented with any SQL engine (PostgreSQL, Oracle, etc.) and its documentation is independent of the choice. The same is true for the adopted interface between *Linssi* and any analysis software package. On the other hand, the database scripts written in SQL and Perl may be engine-dependent, so their applicability is to be evaluated with other database engines but MySQL. However, we do not expect any major difficulties in porting the scripts to other engines.

Chapter 2

Linssi Interface, Configuration and Documentation

This chapter presents basic information of *Linssi* user interface, configuration principles and script documentation instructions. These are the basics that help the reader to understand the following chapters.

2.1 Basic *Linssi* Interface: `LinssiWorld`

Experienced users may use *Linssi* through SQL-queries on the MySQL command prompt or through scripts written in Perl and SQL. For an ordinary user, the main *Linssi* interface is a *Linssi* home page known as `LinssiWorld` in a WWW browser, by default the one shown in Fig. 2.1. The exact contents of the home page may vary from one organization to another, but the distributed version has the following functionality in this interface:

- Queries for sample, measurement and analysis data.
- Display of analysis data in different formats of text reports.
- Display of analysis data in graphical format with zooming capability.
- Display of measurement systems: stations, detectors, setups.
- State-of-health and Quality Control displays.
- Display of calibration data in text and graph, with an update option.

A `LinssiWorld` installation is available in the WWW at `http://linssi.hut.fi/linssi.html`. This demonstration installation has all features of the distributed script package, but it lacks all update features, for obvious reasons.

The wide variety of *Linssi* scripts available in the distribution, illustrated by the demonstration version, are presented in detail in the following chapters of this document. The presentation is organized according to the timeline of their usage, denoted as entry points 0 . . . 3 in accordance with *Linssi* manual Part I [1].

Another perspective on the basic *Linssi* scripts is presented in Ch. 10. Its purpose is to work as a reference when one is looking for a *Linssi* script for a specific purpose.

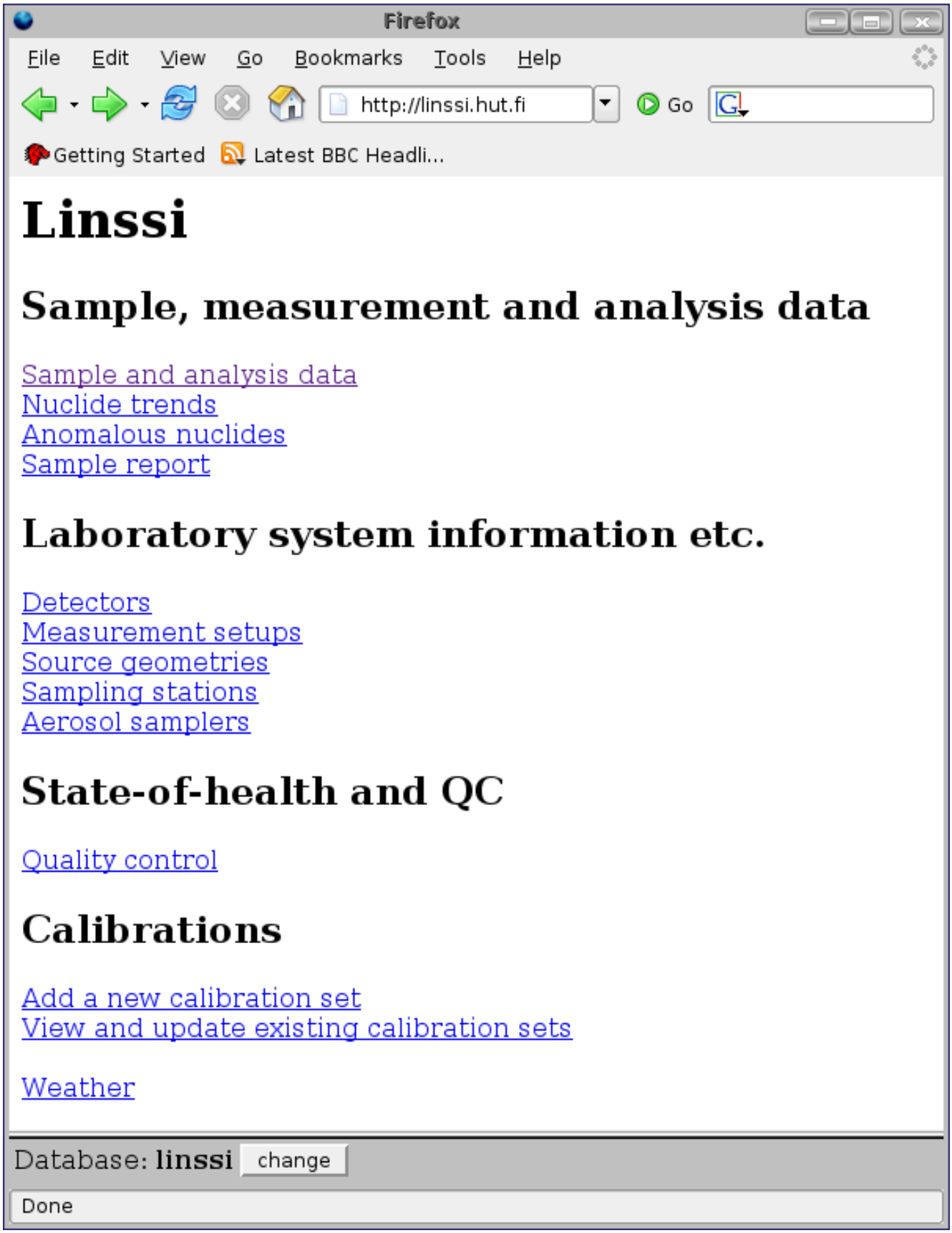


Figure 2.1: The distributed version of `LinssiWorld`.

The distribution package of *Linssi* scripts includes two alternative `LinssiWorld` pages: a standard one and one for a CTBT laboratory. The files connected with them are listed here.

linssi.html

Purpose: An example of a `LinssiWorld` web page, calls linssi_11.html

Package: *Linssi* core

Created: 6.2.2006

linssi_11.html

Purpose: An example of a `LinssiWorld` web page

Package: *Linssi* core

Created: 23.2.2006

linssi.css

Purpose: Style sheet for `LinssiWorld`

Package: *Linssi* core

Created: 30.1.2006

ctbtLab.html

Purpose: An example of a `LinssiWorld` web page for a CTBT laboratory

Package: *Linssi* CTBT scripts

Created: 8.6.2006

ctbtLabSamples.css

Purpose: Style sheet for `LinssiWorld` of a CTBT laboratory

Package: *Linssi* CTBT scripts

Created: 8.6.2006

2.2 Basic *Linssi* Configuration

For using *Linssi*, configuration on two levels is needed: on the administrator level and on the user level. The basic configuration files are `linssiConfig.php` and `.linssirc`, respectively. The former is utilized in WWW-based usage of *Linssi* i.e., with PHP-scripts, and the latter in all other *Linssi* usage but PHP-scripts.

Step-by-step installation and configuration instructions for the *Linssi* administrator are presented in App. A. The basic configuration like creating the database and its users is done on the command prompt or using the scripts of Ch. 3.

linssiConfig.php

The configuration file `linssiConfig.php` shall be scrutinized before the WWW interface `LinssiWorld` can be utilized. This PHP configuration file contains various settings needed in the PHP scripts like:

- paths to external libraries,
- basic database settings,
- option settings,
- directory settings.

The `linssiConfig.php` file in the distribution contains the default settings, but they must naturally be adjusted if a non-default installation is made. The distributed `linssiConfig.php` is extensively commented, making it self-documented.

.linssirc

The configuration file for a *Linssi* user is `.linssirc` and the *Linssi* script distribution includes a template file for it, named `linssirc-template`. This template file must be checked by the administrator and modified to reflect the *Linssi* environment in use. The updated template file is then copied to the Unix home directory of each user allowed to use *Linssi*.

If there are several databases available, a separate file shall be made for each of them, named as `.linssirc*` where `*` can be any string allowed by the operating system. A symbolic link named `.linssirc` may be made to the default one.

The administrator shall define the user-specific usage permissions in the updated `.linssirc` file. For example, if there are users only allowed to read a database, their `.linssirc` should not contain a `[write]`-section with write username and password.

The structure of `.linssirc` is such that various **options** are defined and organized in separate **sections**. The file is always started with a default section that contains the following obligatory options (see for additional explanations in the comments of `linssirc-template`):

- `databaseVersion`:
  version of *Linssi* database schema
- `databaseScriptDir`:
  directory where *Linssi* Perl scripts reside
- `databaseName`:
  full name of the database as given to Perl scripts (may include host and port)
- `databasePlain`:
  plain name of the database without host or port
- `databaseHost`:
  name of the database server (def. localhost)
- `databasePort`:
  server port number for MySQL connections (def. 3306)

- `odbc`:
  selection of ODBC drivers (0 or 1) [5]

Then come the optional sections, most often `[read]` and `[write]`. As shown, section names are in brackets and they contain a list of section-specific options. In the named sections, the following options are to be specified:

- `username`:
  MySQL username for read or write access
- `password`:
  password corresponding to the MySQL username for read or write access

The options and sections defined in the `linssirc-template` file are reserved words. It is allowed to add own sections and options to `.linssirc` and they can be reserved globally by an announcement to Linssi Administrative Body. All programs and scripts reading `.linssirc` shall ignore the options they do not recognize.

There are two Perl function packages in the *Linssi* script distribution for interpreting the `.linssirc` files, named `linssiConfig.pm` and `linssiGetoptStd.pm`. They are used by current Perl scripts and also recommended for any new scripts.

2.3 Documentation of *Linssi* Scripts

The script descriptions in Ch. 3–10 of this document have been made semi-automatically by the `collectInfo` script included in the *Linssi* package. This is possible when the script authors have obeyed a commenting practice developed at STUK/ASL. In this practice, all scripts are started with a **documentation block** behind double comment signs (e.g., `##` in Perl scripts), utilizing fixed XML-type tags for different features of the scripts. This practice is recommended to all *Linssi* script authors and it is in fact enforced on any new scripts to be added to the official distribution.

The following documentation items are defined as compulsory for each script by the *Linssi* Administrative Body (LAB):

1. Name of the script followed by **space-hyphen-space** and the purpose of the script in max. 80 characters. This line must be before all tagged information in the documentation block.
2. Author(s) of the script within tags `<author>...</author>`.
3. Original creation date of the script within tags `<created>...</created>`.
4. Package that the script belongs to within tags `<package>...</package>`. Current package alternatives are "*Linssi* core" and "*Linssi* CTBT scripts".
5. Version number of the script within tags `<version>...</version>`. The current practice is to use three numbers for scripts: the first two are the *Linssi* database schema version and the third one the actual version number of the script. Example: 1.1.13 means the 13th released version of a script for *Linssi* v.1.1. **The actual version number of the script shall be upgraded when any modifications are made in the script and released.**

When appropriate, the following additional tags can be used:

- `<bugs>...</bugs>`:
  known bugs or deficiencies
- `<configuration>...</configuration>`:
  configuration files and their contents used
- `<description>...</description>` (alias `<module>...</module>`):
  detailed documentation of the script functionality
- `<functionList>...</functionList>` (alias `<subroutine>...</subroutine>`):
  list of functions or subroutines included in the file
- `<modules>...</modules>` (alias `<uses>...</uses>`):
  list of external modules used by the script
- `<options>...</options>`:
  documentation of the options supported by the script
- `<parameters>...</parameters>` (alias `<param>...</param>`):
  documentation of parameters needed by the script
- `<readsTables>...</readsTables>`:
  list of *Linssi* tables read by the script
- `<requires>...</requires>`:
  external modules required by the script
- `<return>...</return>`:
  return value of the script
- `<synopsis>...</synopsis>`:
  synopsis of the script
- `<syntax>...</syntax>`:
  script/module call syntax
- `<updated>...</updated>`:
  date of an update; may also include a description of the update and the author
- `<writesTables>...</writesTables>`:
  list of *Linssi* tables written by the script

Anything in the documentation block between lines like "`------`" is output to this document verbatim. An example is seen in the *Description* of script `stufftodb` on p. 18 (11 lines starting from `#table_name`).

The current *Linssi* scripts can be used as a model for documentation.

Chapter 3

Database Management Scripts

The most basic database scripts are the management scripts to create the database, to list its table contents, and to check its integrity. These scripts are presented in this chapter and their usage is illustrated in App. A.

Some scripts refer to ODBC that is clarified in Ref. [5].

3.1 preparedb

Purpose: Create database and user accounts

Version: 1.1.1

Author: Andreas Pelikan

Package: *Linssi* core

Created: 18.11.2005

Updated: 5.7.2006 JAH : Documentation

Description: This script creates a database instance and user accounts with standard privileges for a *Linssi* MySQL database, i.e. it prepares the database. The *Linssi* database tables have to be created separately with a subsequent call to maketables. If the database instance already exists, only the user accounts are created/updated.
The script connects to the database with the database account information in the [create] section of the configuration, which may be overwritten by options. Database creator needs CREATE and GRANT OPTION privileges!

Syntax:
```
preparedb [-d database_name] [-u username] [-p password] [-r readuser]
[-R readpwd] [-w writeuser] [-w writepwd] [-C configFile]
```

Options:
```
-C name of configuration file.  If not specified, ~/.linssirc is used.
For options that read their default from the config file, these are
indicated by (section.key)
-d name of database instance to generate (dbname)
-u db creator user name (create.username)
-p db creator password (create.password)
-r read-account user name (read.username)
-R read-account password (read.password)
```

```
-w write-account user name (write.username)
-W write-account password (write.password)
```

3.2 maketables

Purpose: Creates tables in *Linssi* database

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 15.7.2003

Updated: 30.11.2004
Documentation updated.
9.12.2004
Table specification changed to v.1.1. (b.4.7)
21.12.2004
Some bugfixes
19.10.2005
version 1.1.1: ODBC support added. New version numbering. Uses linssiConfig.pm and supports configuration files. -g
-option added. Supports empty strings as password (of course you shouldn't do this with your write user, but if you want to leave gaping security holes why should we stop you).
24.5.2006
CTBT specific tables modified -tos
5.7.2006 JAH : Documentation

Description: This script automatically creates tables for *Linssi* database. The table specifications are according to the database definition 1.1.
The script uses database name, username and password given as arguments. If database name, username, password and/or odbc-paramter are not given, the values in configuration file will be used (username and password are those of write-user). Note that you need to have the linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script or add 'use lib "linssiConfigPath"' at the beginning of file. If the name of configuration file is not given, the default file ($ENV{HOME}/.linssirc) is used.
If a table with the same name already exists in database the script continues with next table. The script does NOT check if the existing table contains the same fields that the script is trying to create.

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax:
```
maketables [-d database_name] [-u username] [-p password] [-g configFile]
[- c] [-odbc|-noodbc]
```

Options:
```
-c Generate CTBT tables in addition to default tables.

-g configFile Use configFile as configuration file.  If database name,
```

```
username, password and/or odbc-option are not given, the values
read from configFile instead (see linssiConfig.pm for details).
If this option is not given, $ENV{HOME}/.linssirc is used as
configuration file.

-odbc Use ODBC instead of DBI's internal MySQL drivers.  In that case
DSN is used instead of database name (see ODBC documentation).

-noodbc Use DBI's internal MySQL drivers instead of ODBC. If neither
odbc nor noodbc is given, the value in configuration file is
used.  If there there is no odbc paramater in configuration file
internal MySQL drivers are used.
```

Modules: linssiConfig.pm v.1.1.1 or newer

3.3 desctables

Purpose: Shows descriptions of all tables in a *Linssi* database

Version: 1.1.6

Author: Jarmo Ala-Heikkila

Package: *Linssi* core

Created: 11.2.2004

Updated: 13.2.2004
22.3.2004
27.1.2006
8.6.2006
5.7.2006

Description: This script shows descriptions of all tables in a *Linssi* database. It is invoked from Unix command line using the given syntax. The MySQL username (def. webbi) shall be one for reading without a password. The obsolete configuration file ~/.my.cnf shall not exist.

Syntax: `desctables [-d database] [-u username]`

3.4 checkdb

Purpose: Checks *Linssi* database for errors and inconsistencies in data

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 4.6.2004

Updated: 8.6.2005
Updated for current database specification.
15.6.2005

Modified the system for checking links to analyses table. Functionality is unchanged but the script should run much faster if there are a lot of analyses in the system.
17.6.2005
Added -t -option.
11.7.2005
Few minor bugs fixed and error messages clarified.
10.10.2005
Version 1.1.1: Uses linssiConfig.pm and supports configuration files. -g -option added. Supports empty strings as password (of course you shouldn't do this with your write user, but if you want to leave gaping security holes why should we stop you). Added -l -option. Added –noodbc -option.
9.11.2005
Version 1.1.1a: Fixed a bug in removal of data from lineAssociations.
5.7.2006 JAH : Documentation

Description: This script checks *Linssi* database for errors and inconsistencies. The table specifications are according to the database definition 1.1.
The database name and username and password can be given as arguments for the script. If you wish to use no password give empty string as password (-p ''). If username, database name, password or odbc-parameter are not given the values in the configuration file are used instead (username and password are those of write- user). See below for more details. Note that you need to have the linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script or add 'use lib "linssiConfigPath"' at the beginning of file. If - g -option is not given, the default file ($ENV{HOME}/.linssirc) is used. Script can be run either in normal interactive mode, in batch mode (-b option) or in list mode (-l option). List mode does not make any changes to database, it only lists found errors. In interactive mode the user is prompted for confirmation before doing any changes to database other than removing data failing critical checks. If you want only to reset the invalid links to NULL or automatically add dummy entries instead of removing those links run fixdb afterwards. In batch mode only changes done to database are those that don't need user confirmation. In all other cases the script only prints out a warning. Thus batch mode corresponds to interactive mode but answering "no" to all questions.
The script does following checks (! = check for critical reference, data failing this check will be removed without prompting user for confirmation, * = Warning only, no modification even in interactive mode). When checking foreign keys in non-critical references a NULL value in foreign key is considered to be intentionally left blank and is not reported as an error. In critical references a NULL foreign key is considered an error and entry is removed. To fix errors in warning-only checks see fixdb. NOTE: With a lot of content in database the running time of this script may be tens of minutes. Whole database will be locked during that time. It is recommended to ensure that no time-critical database access will be attempted in the near future before running this script.
-For every wheathers.stationId foreign key there is corresponding stations.stationId primary key.
*For every samplers.stationName foreign key there is corresponding stations.stationId primary key.

-For every mobileCoordinates.stationId foreign key there is corresponding stations.stationId primary key.
-For every mobileCoordinates.idSample foreign key there is corresponding samples.idSample primary key.
-For every mobileCoordinates.idMeas foreign key there is corresponding measurements.idMeas primary key.
-For every airFilterSOH.samplerId foreign key there is corresponding samplers.samplerId primary key.
!For every airFilterSOH.idSample foreign key there is corresponding sampleData.idSample primary key.
!For every airFilterSamples.idSample foreign key there is corresponding samples.idSample primary key.
*For every airFilterSamples.samplerId foreign key there is corresponding samplers.samplerId primary key.
*For every airFilterSamples.stationId foreign key there is corresponding station.stationId primary key.
-For every calibrationSamples.idSample foreign key there is corresponding samples.idSample primary key.
!For every calibrationNuclides.idSample foreign key there is corresponding samples.idSample primary key.
*For every calibrationNuclides.idCalibrationLibrary foreign key there is corresponding calibrationLibraries.idCalibrationLibrary primary key.
!For every sampleSplitsCombines.parentIdSample foreign key there is corresponding samples.idSample primary key.
!For every sampleSplitsCombines.daughterIdSample foreign key there is corresponding samples.idSample primary key.
*For every sources.idSample foreign key there is corresponding samples.idSample primary key.
-For every measurementSetups.detectorId foreign key there is corresponding detectors.detectorId primary key.
-For every measurementSetups.sourceId foreign key there is corresponding sources.sourceId primary key.
-For every measurementSetups.idCal foreign key there is corresponding calibrations.idCal primary key.
-For every measurementSetups.blankIdAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
-For every measurementSetups.backgroundIdAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
!For every measurements.idSample foreign key there is corresponding samples.idSample primary key.
-For every measurements.measSetupId foreign key there is corresponding measurementSetups.measSetupId primary key.
-For every measurements.blankIdMeas foreign key there is corresponding measurements.idMeas primary key.
-For every measurements.backgroundIdMeas foreign key there is corresponding measurements.idMeas primary key.
-For every calibrations.idInputCal foreign key there is corresponding calibrations.idCal primary key.
-For every calibrations.measSetupId foreign key there is corresponding measure-

mentSetups.measSetupId.idMeas primary key.
!For every calPoints.idCal, calPoints.calTypeId foreign key pait there is corresponding alPoints.idCal, calPoints.calTypeId primary key pair.
-For every calPoints.idAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
!For every calPreferences.idAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
!For every calPreferences.idCal foreign key there is corresponding calibrations.idCal primary key.
!For every analyses.idMeas foreign key there is corresponding measurements.idMeas primary key.
!Analyses.idSample is same as measurements.idSample for the measurement used in this analysis.
-For every analyses.blankIdAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
-For every analyses.backgroundIdAnalysis foreign key there is corresponding analysesData.idAnalysis primary key.
-For every analyses.inputIdAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
!For every peaks.idAnalysis foreign key there is corresponding analysiss.idAnalysis primary key.
!Peaks.idSample and peaks.idMeas is same as analyses.idSample and analyses.idMeas for the analysis used in this analysis.
!For every lineAssociations.idAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
!LineAssociations.idSample and peakData.idMeas is same as analyses.idSample and analyses.idMeas for the analysis used in this analysis.
!For every activities.idAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
!activities.idSample and activities.idMeas is same as analyses.idSample and analyses.idMeas for the analysis used in this analysis.
!For every activityLimits.idAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
!activityLimits.idSample and acitivityLimits.idMeas is same as analyses.idSample and analyses.idMeas for the analysis used in this analysis.
!For every nuclideRatios.idAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
!nuclideRatios.idSample and nuclideRatios.idMeas is same as analyses.idSample and analyses.idMeas for the analysis used in this analysis.
!For every finalResults.idAnalysis foreign key there is corresponding analysisData.idAnalysis primary key.
!finalResults.idSample and finalResults.idMeas is same as analyses.idSample and analyses.idMeas for the analysis used in this analysis.

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax: `checkdb [-b|-l] [-t] [-d databaseName/DSN] [-u username] [-p password] [-g configFile] [-odbc|-noodbc]`

Options: `-g configFile Use configFile as configuration file. If database name,`

```
username, password and/or odbc-option are not given, the values
read from configFile will be used instead (see linssiConfig.pm for details).
If this option is not given, $ENV{HOME}/.linssirc is used as
configuration file.

-odbc Use ODBC instead of DBI's internal MySQL drivers.  In that case
DSN is used instead of database name (see ODBC documentation).

-noodbc Use DBI's internal MySQL drivers instead of ODBC. If neither
odbc nor noodbc is given, the value in configuration file is
used.  If there there is no odbc paramater in configuration file
internal MySQL drivers are used.

-d databaseName Name of the database that contains the wanted information.
If not given the name in the configuration file is used.

-u username Username to be used.  User need to have select rights to
the database.  If not given the name in the configuration
file is used.

-p password Password for the user.  If not given the one in the
configuration file is used.

-b Use batch mode instead of interactive mode.

-l Use list mode

-t Print starting and ending times.
```

Reads tables: weathers, stations, samplers, mobileCoordinates, samples, measurements, airFilterSOH, airFilterSamples, calibrationSamples, calibrationNuclides, calibrationLibraries, sampleSplitsCombines, sources, measurementSetups. detectors. calibrations, analyses, calPoints, calPreferences, peaks, lineAssociations, activities, activityLimits, nuclideRatios, finalResults

Writes tables: weathers, stations, samplers, mobileCoordinates, samples, measurements, airFilterSOH, airFilterSamples, calibrationSamples, calibrationNuclides, calibrationLibraries, sampleSplitsCombines, sources, measurementSetups. detectors. calibrations, analyses, calPoints, calPreferences, peaks, lineAssociations, activities, activityLimits, nuclideRatios, finalResults

Modules: linssiConfig.pm v.1.1.1 or newer

3.5 fixdb

Purpose: Fixes errors and inconsistencies in *Linssi* database

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 9.10.2005
Updated: 5.7.2006 JAH : Documentation

Description: This script fixes those errors and inconsistencies in *Linssi* database that checkdb does not.
The database name, username and password can be given as arguments for the script. If you wish to use no password give an empty string as password (-p '') If username, database name, password or odbc-parameter are not given the values in the configuration file are used instead (username and password are those of write-user). See below for more details. Note that you need to have the linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script or add 'use lib "linssiConfigPath"' at the beginning of file. If -g -option is not given, the default file ($ENV{HOME}/.linssirc) is used. Script can be run either in normal interactive mode or in batch mode (-b option). In interactive mode user is prompted for confirmation before doing any changes to database. Possible actions are usually resetting the invalid links to NULL or automatically adding dummy entries. In batch mode "add dummy" action is selected where applicable, otherwise "null" action is used.
The script does following checks. NOTE: With a lot of content in database the running time of script may be tens of minutes. The whole database will be locked during that time. It is recommended to ensure that no time-critical database access will be attemped in the near future before running this script.
-For every wheathers.stationId foreign key there is corresponding stations.stationId primary key.
-For every mobileCoordinates.stationId foreign key there is corresponding stations.stationId primary key.
-For every mobileCoordinates.idSample foreign key there is corresponding samples.idSample primary key.
-For every mobileCoordinates.idMeas foreign key there is corresponding measurements.idMeas primary key.
-For every airFilterSOH.samplerId foreign key there is corresponding samplers.samplerId primary key.
-For every airFilterSamples.samplerId foreign key there is corresponding samplers.samplerId primary key.
-For every airFilterSamples.stationId foreign key there is corresponding station.stationId primary key.
-For every calibrationSamples.idSample foreign key there is corresponding samples.idSample primary key.
-For every calibrationNuclides.idCalibrationLibrary foreign key there is corresponding calibrationLibraries.idCalibrationLibrary primary key.
-For every sources.idSample foreign key there is corresponding samples.idSample primary key.
-For every measurementSetups.detectorId foreign key there is corresponding detectors.detectorId primary key.
-For every measurementSetups.sourceId foreign key there is corresponding sources.sourceId primary key.
-For every measurementSetups.idCal foreign key there is corresponding calibrations.idCal primary key.
-For every measurementSetups.blankIdAnalysis foreign key there is correspond-

ing analyses.idAnalysis primary key.
-For every measurementSetups.backgroundIdAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
-For every measurements.measSetupId foreign key there is corresponding measurementSetups.measSetupId primary key.
-For every measurements.blankIdMeas foreign key there is corresponding measurements.idMeas primary key.
-For every measurements.backgroundIdMeas foreign key there is corresponding measurements.idMeas primary key.
-For every calibrations.idInputCal foreign key there is corresponding calibrations.idCal primary key.
-For every calibrations.measSetupId foreign key there is corresponding measurementSetups.measSetupId.idMeas primary key.
-For every calPoints.idAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
-For every analyses.blankIdAnalysis foreign key there is corresponding analyses.idAnalysis primary key.
-For every analyses.backgroundIdAnalysis foreign key there is corresponding analysesData.idAnalysis primary key.
-For every analyses.inputIdAnalysis foreign key there is corresponding analyses.idAnalysis primary key.

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax:

```
fixdb [-b] [-d databaseName/DSN] [-u username] [-p password]
[-g configFile] [-odbc|-noodbc]
```

Options:

```
-g configFile Use configFile as configuration file.  If database name,
username, password and/or odbc-option are not given, the values
read from configFile instead (see linssiConfig.pm for details).
If this option is not given, $ENV{HOME}/.linssirc is used as
configuration file.

-odbc Use ODBC instead of DBI's internal MySQL drivers.  In that case
DSN is used instead of database name (see ODBC documentation).

-noodbc Use DBI's internal MySQL drivers instead of ODBC. If neither
odbc nor noodbc is given, the value in configuration file is
used.  If there there is no odbc paramater in configuration file
inter MySQL drivers are used.

-d databaseName Name of the database that contains the wanted information.
If not given the 'write' username in the configuration file is used.

-u username Username to be used.  User need to have select rights to
the database.  If not given the 'write' uesrname in the
configuration file.

-p password Password for the user.  If not given the one in user's
.linssirc will be used.
```

```
-b Uses batch mode instead of interactive mode.
```

Reads tables: weathers, stations, samplers, mobileCoordinates, samples, measurements, airFilterSOH, airFilterSamples, calibrationSamples, calibrationNuclides, calibrationLibraries, sampleSplitsCombines, sources, measurementSetups. detectors. calibrations, analyses, calPoints, calPreferences, peaks, lineAssociations, activities, activityLimits, nuclideRatios, finalResults

Writes tables: weathers, stations, samplers, mobileCoordinates, samples, measurements, airFilterSOH, airFilterSamples, calibrationSamples, calibrationNuclides, calibrationLibraries, sampleSplitsCombines, sources, measurementSetups. detectors. calibrations, analyses, calPoints, calPreferences, peaks, lineAssociations, activities, activityLimits, nuclideRatios, finalResults

Modules: linssiConfig.pm v.1.1.1 or newer

Chapter 4

Entry Point Zero

In *Linssi* manual Part I [1] entry point 0 is defined as denoting the station group of database tables, i.e., tables `stations`, `weathers`, and `mobileCoordinates`. These tables can be updated with the scripts of this chapter and this can be done relatively independent from the other table groups.

Some of these scripts refer to AKu input file format that is documented in Ch. 14. Some scripts refer to ODBC that is clarified in Ref. [5].

4.1 stufftodb

Purpose: Script for reading miscellaneous data into *Linssi* database
Version: 1.1.2
Author: Antero Kuusi
Package: *Linssi* core (v.1.1)
Created: 30.11.2005
Updated: 5.7.2006 JAH : Documentation

Description: Script for importing miscellaneous table data into *Linssi* database. This script reads those tables that other input scripts don't into database. Data is read from stdin unless -f -option is used. The AKu input format (see *Linssi* manual Part II) is the same as type 1 format of analysistodb:

```
#table_name
firstValue
secondValue
<blobName>
BLOB
BLOB
</blobName>
thirdValue

#another_table_name
...
```

Empty line alone (except inside a blob) is assumed to end the current table. All tables can exist more than once in the input report, each instance of same table is treated as a separate entry into database. None of the tables are mandatory. Lines containing only "&" are treated as NULL value when entering it into table.

The database name and username and password can be given as arguments for the script. If you wish to use no password give empty string as password (-p ''). If username, database name, password or odbc-parameter are not given the values in the configuration file are used instead (username and password are those of write-user). See below for more details. Note that you need to have the linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script or add 'use lib "linssiConfigPath"' at the beginning of file. If -g -option is not given, the default file ($ENV{HOME}/.linssirc) is used. Following tables are currently supported by stufftodb: stations, weathers, samplers, calibrationLibraries, detectors, shields, attenuators, measurementSetups (for "standard" setups that are not related to one specific sample).

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax:
```
stufftodb [-f reportfile] [-d database_name/DSN] [-u username]
[-p password] [-g configFile] [-o] [-odbc|-noodbc]
```

Options:

```
-f Data is read from a given file.  If this option is not given, the
report is read from stdin.

-g configFile Use configFile as configuration file.  If database name,
username, password and/or odbc-option are not given, the values
read from configFile instead (see linssiConfig.pm for details).
If this option is not given, $ENV{HOME}/.linssirc is used as
configuration file.

-odbc Use ODBC instead of DBI's internal MySQL drivers.  In that case
DSN is used instead of database name (see ODBC documentation).

-noodbc Use DBI's internal MySQL drivers instead of ODBC. If neither
odbc nor noodbc is given, the value in configuration file is
used.  If there there is no odbc paramater in configuration file
inter MySQL drivers are used.

-o If some of tables contain data with the same primary key, data
of those reacords already in database is overwritten.  If this
option is not specified, all records with duplicate primary
keys are skipped
```

Writes tables: stations, weathers, samplers, calibrationLibraries, detectors, shields, attenuators, measurementSetups

Modules: linssiConfig.pm v.1.1.1 or newer

4.2 showStations.php

Purpose: Shows the stations from the database

Version: 1.1.1

Author: Teemu Siiskonen, Tommi Salonen

Package: *Linssi* core

Created: 1.5.2006

Updated: 5.7.2006 JAH : Documentation

Description: Shows the stations from the database.

Reads tables: stations

Modules: linssiConfig.php, htmlTags.php, linssiPHPLib.php

4.3 showWeather.php

Purpose: Shows selected weather data from selected weather stations

Version: 1.1.2

Author: Tommi Salonen, Satu Kuukankorpi

Package: *Linssi* core

Created: 8.11.2005

Updated: 2.1.2006 Updated to show additional weather data
Satu Kuukankorpi
27.1.2006 Changed linePlot to scatterPlot, fixed bugs concerning
handling of missing data. Weather stations are fetched from the database
station table. Only stations with stationType = weather are selected
Satu Kuukankorpi
5.7.2006 JAH : Documentation

Description: This script shows selected weather data from selected weather stations in database in a chosen time period

Reads tables: stations, weathers

Modules: linssi version 1.1, JpGraph graph-library

4.4 deleteStation

Not implemented yet.

4.5 showCoordinates

Not implemented yet.

4.6 deleteTempCoordinates

Not implemented yet.

4.7 deleteCoordinates

Not implemented yet.

4.8 deleteWeather

Not implemented yet.

Chapter 5

Entry Point One

In *Linssi* manual Part I [1] entry point 1 is defined as the procedures that start from the production or collection of activity to from the sample. In *Linssi* v.1.1 we have defined three sample types that can apply entry point 1: air filter samples, calibration samples and CTBT laboratory samples. The scripts, or their proposed names, in this chapter deal with air samplers and filter samples, but similar scripts should also be developed for other sample types.

5.1 `recordSample.php`, `selectSampler.php`, `startSampler.php`, etc.

Not implemented yet.

5.2 `showAirFilterSamples`

Not implemented yet.

5.3 `updateAirFilterSamples`

Not implemented yet.

Chapter 6

Entry Point Two

In *Linssi* manual Part I [1] entry point 2 is defined as the procedures that start from receiving a sample to be measured. The scripts presented in this chapter deal with displaying and modifying sample data.

Some of these scripts refer to AKu input file format that is documented in Ch. 14. Some scripts refer to ODBC that is clarified in Ref. [5].

6.1 sampletodb

Purpose: Script for reading sample data into *Linssi* database

Version: 1.1.3

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 19.12.2005

Updated: 17.1.2006
Version 1.1.2: Changed the type of sampleSplitsCombines block to type 2.
5.7.2006 JAH : Documentation

Description: Script for importing sample table data into *Linssi* database. Data is read from stdin unless -f -option is used. The AKu input format (see *Linssi* manual Part II) is either of type 1 or type 2 depending on the table.

Type 1:

```
#table_name
firstValue
secondValue
<blobName>
BLOB
BLOB
</blobName>
thirdValue

#another_table_name
...
```

Type 2:

```
#table_name
firstValue1 secondValue1 "BLOB BLOB" thirdValue1
```

```
firstValue2 secondValue2 "BLOB BLOB" thirdValue2
...
```

Empty line alone (except inside a blob) is assumed to end the current table. Each table name should exist only once in the file. Following tables use the type 2 format: sampleSplitsCombines, mobileCoordinates and airFilterSOH.
Lines containing only "&" are treated as NULL value when entering it into table. The database name and username and password can be given as arguments for the script. If you wish to use no password give empty string as password (-p ''). If username, database name, password or odbc-parameter are not given the values in the configuration file are used instead (username and password are those of write-user). See below for more details. Note that you need to have the linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script or add 'use lib "linssiConfigPath"' at the beginning of file. If -g -option is not given, the default file ($ENV{HOME}/.linssirc) is used.

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax:

```
sampletodb [-f reportfile] [-d database_name/DSN] [-u username]
[-p password] [-g configFile] [-i] [-c|-o|-s] [-odbc|-noodbc]
```

Options:

```
-f Data is read from a given file.  If this option is not given, the
report is read from stdin.

-i Script reports assigned idSample after processing.

-g configFile Use configFile as configuration file.  If database name,
username, password and/or odbc-option are not given, the values
read from configFile instead (see linssiConfig.pm for details).
If this option is not given, $ENV{HOME}/.linssirc is used as
configuration file.

-odbc Use ODBC instead of DBI's internal MySQL drivers.  In that case
DSN is used instead of database name (see ODBC documentation).

-noodbc Use DBI's internal MySQL drivers instead of ODBC. If neither
odbc nor noodbc is given, the value in configuration file is
used.  If there there is no odbc paramater in configuration file
inter MySQL drivers are used.

-c If database contains data with the same sampleId, data in
samples, is compared with the data in file.
Fields that have NULL in file or database are ignored.  If some of
the compared fields does not match, the script is aborted with
error message.  Useful for checking consistency of data when it is
inputted piecemeal.

-o If database contains data with the same sampleId, data in
samples with the same sampleId/measurementId is overwritten.
```

```
Useful when file contains newer or more accurate information
than in database.
NOTE: Requires MySQL v.4.1.0 or newer.

-s If database contains data with the same sampleId
the input is aborted.  Useful when combining data from two
different sources that may have assigned same
sampleId/measurementId for different samples/measurements.
```

Writes tables: mobileCoordinates, airFilterSOH, airFilterSamples, calibrationSamples, calibrationNuclides, samples, sampleSplitsCombines, sources, ctbtLabSamples

Modules: linssiConfig.pm v.1.1.1 or newer

6.2 meastodb

Purpose: Script for reading measurement data into *Linssi* database

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 19.12.2005

Updated: 5.7.2006 JAH : Documentation

Description: Script for importing measurement table data into *Linssi* database. Data is read from stdin unless -f -option is used. The AKu input format (see *Linssi* manual Part II) is of type 1:

```
#table_name
firstValue
secondValue
<blobName>
BLOB
BLOB
</blobName>
thirdValue

#another_table_name
...
```

Empty line alone (except inside a blob) is assumed to end the current table. Each table name should exist only once in the file. Sample-block is a special case in the script. The sample-block can be either of normal form or it can contain only the the two first values of normal form. In either case only sampleId-field is used from the block. That field is used to correlate rest of the data in the file to the correct sample already in database.
Lines containing only "&" are treated as NULL value when entering it into table. The database name and username and password can be given as arguments for the script. If you wish to use no password give empty string as password (-p ''). If username, database name, password or odbc-parameter are not given the values in the configuration file are used instead (username and password are those of write-user). See below for more details. Note that you need to have the

linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script or add 'use lib "linssiConfigPath"' at the beginning of file. If -g -option is not given, the default file ($ENV{HOME}/.linssirc) is used. Following tables are currently supported by meastodb: measurements, measurementSetups. In addition samples block is needed in normal or shortened format for sampleId.

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax:

```
meastodb [-f reportfile] [-d database_name/DSN] [-u username]
[-p password] [-g configFile] [-i] [-c|-o|-s] [-odbc|-noodbc]
```

Options:

```
-f Data is read from a given file.  If this option is not given, the
report is read from stdin.

-i Script reports assigned idMeas after processing.

-g configFile Use configFile as configuration file.  If database name,
username, password and/or odbc-option are not given, the values
read from configFile instead (see linssiConfig.pm for details).
If this option is not given, $ENV{HOME}/.linssirc is used as
configuration file.

-odbc Use ODBC instead of DBI's internal MySQL drivers.  In that case
DSN is used instead of database name (see ODBC documentation).

-noodbc Use DBI's internal MySQL drivers instead of ODBC. If neither
odbc nor noodbc is given, the value in configuration file is
used.  If there there is no odbc paramater in configuration file
inter MySQL drivers are used.

-c If database contains data with the same measId, data in
measurements and samplingData with the same
measurementId is compared with the data in file.
Fields that have NULL in file or database are ignored.  If some of
the compared fields does not match, the script is aborted with
error message.  Useful for checking consistency of data when it is
inputted piecemeal.

-o If database contains data with the same measurementId, data in
measurements with the same
measId is overwritten.  Useful when file contains
newer or more accurate information than in datbase.
NOTE: Requires MySQL v.4.1.0 or newer.

- s If database contains data with the same measId
the input is aborted.  Useful when combining data from two
different sources that may have assigned same
measurementId for different measurements.
```

Writes tables: measurements, measurementSetups

Modules: linssiConfig.pm v.1.1.1 or newer

6.3 showSamples.php

Purpose: Fetches sample data from chosen stations in a time interval

Version: 1.1.6

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.10.2005

Updated: Status backgroundcolor now changes if anomalous nuclides were detected in the final analysis. - tos
Added status explanations - tos
Added ”samples without status” count - tos
Changed the order of the software in the list - tos
Added ”include background measurements”-option - tos
5.7.2006 JAH : Documentation

Description: This script fetches samples / measurements from the chosen stations in the given time interval. Information about found samples / measurements is printed in a table, and in the end of each row a form is printed with all the software that has produced an analysis from the corresponding sample / measurement. Form is used as a link to the analysis results (analysisReport.php). Analysis status field indicates if anomalies have been found in the final analysis of the corresponding sample / measurement. Nuclides that are not considered as anomalies (and that are defined in the array in anomalousNuclides.php) are defined in the beginning of the script.

Reads tables: airFilterSamples, ctbtLabSamples, finalResults, measurements, analyses, samples, activities

Modules: linssiConfig.php, htmlTags.php, linssiFormTools.php, linssiPHPLib.php, anomalousNuclides.php

6.4 sampleReport.php

Purpose: Creates sampling reports in html and excel format

Version: 1.1.2

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.10.2005

Updated: Added acqStart, acqEnd, completionTime and dbEntryTime to the report - tos
5.7.2006 JAH : Documentation

Description: This script creates sampling reports in html and excel format (PEAR::OLE and PEAR::SpreadsheetExcelWriter packages must be installed on the server to get the reports in excel format).

Reads tables: stations, airFilterSamples, finalResults, samples, activities, activityLimits

Modules: PEAR::OLE, PEAR::SpreadsheetExcelWriter, linssiConfig.php, htmlTags.php, linssiPHPLib.php, linssiFormTools.php

6.5 deleteSample

Purpose: Deletes a sample and all associated data from *Linssi*

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 23.7.2003

Updated: 18.1.2004
5.5.2005
Updated for v.1.1 table specification.
16.11.2005
Version 1.1.1: Uses linssiConfig.pm and supports configuration files. -g -option added. Supports empty strings as password. Added –noodbc - option. Syntax slightly modified.
5.7.2006 JAH : Documentation

Description: Script for deleting a sample and all associated data from *Linssi* database v.1.1. The database name and username and password can be given as arguments for the script. If you wish to use no password give an empty string as password (-p ''). If username, database name, password or odbc-parameter are not given the values in the configuration file are used instead (username and password are those of write-user). See below for more details. Note that you need to have the linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script or add 'use lib "linssiConfigPath"' at the beginning of file. If -g -option is not given, the default file ($ENV{HOME}/.linssirc) is used.

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax: `deleteSample -s sampleId [-d database_name] [-u username] [-p password] [-g optionFile] [-odbc|-noodbc]`

Writes tables: samples, airFilterSamples, airFilterSOH, sources, measurements, analyses, calPreferences, peaks, lineAssociations, activities, activityLimits, nuclideRatios, finalResults

Modules: linssiConfig.pm v.1.1.1 or newer

6.6 deleteMeas

Purpose: Deletes a measurement and all associated data from *Linssi*

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 23.7.2003

Updated: 18.1.2004
5.5.2005
Updated for v.1.1 table specification.
16.11.2005
Version 1.1.1: Uses linssiConfig.pm and supports configuration files. -g -option added. Supports empty strings as password. Added –noodbc - option. Syntax slightly modified.
5.7.2006 JAH : Documentation

Description: Script for deleting a measurement and all associated data from *Linssi* database v.1.1.
The database name and username and password can be given as arguments for the script. If you wish to use no password give an empty string as password (-p ’’).
If username, database name, password or odbc-parameter are not given the values in the configuration file are used instead (username and password are those of write-user). See below for more details. Note that you need to have the linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script or add ’use lib ”linssiConfigPath”’ at the beginning of file. If -g -option is not given, the default file ($ENV{HOME}/.linssirc) is used.

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax: `deleteSample -m measId [-d database_name] [-u username] [-p password] [-g optionFile] [-odbc|-noodbc]`

Writes tables: measurements, analyses, calPreferences, peaks, lineAssociations, activities, activityLimits, nuclideRatios, finalResults

Modules: linssiConfig.pm v.1.1.1 or newer

Chapter 7

Calibration Management

Calibration management typically belongs to the basic tasks under `LinssiWorld`. The scripts presented in this chapter are linked to it.

7.1 calibrations.php

Purpose: Tool for viewing calibrations

Version: 1.1.2

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.6.2005

Updated: 20.6.2006 : calPoints are converted to double values before creating the graph (this created problems in some installations of PHP/JpGraph) - tos
5.7.2006 JAH : Documentation

Description: Tool for viewing calibrations. Shows general information (comments, creation time etc.), calDataPairs and line graph images of calibrations. This script can also be used to edit/add comments in calibrations and to set calibration as a default in the measurementSetup.

Reads tables: calPreferences, measurementSetups, calibrations, analyses, calPoints

Writes tables: measurementSetups, calibrations

Modules: JpGraph, linssiConfig.php, htmlTags.php, linssiPHPLib.php, linssiFormTools.php

7.2 addCalibration.php

Purpose: Adds a new calibration set to *Linssi* database

Version: 1.1.2

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.6.2005

Updated: 12.5.2006 : Removed the requirement that x values must be higher than y values -tos
5.7.2006 JAH : Documentation

Description: Adds a new calibration set to *Linssi* database.

Reads tables: calibrations, measurementSetups

Writes tables: calibrations, calPoints

Modules: linssiConfig.php, htmlTags.php, linssiPHPLib.php, linssiFormTools.php

7.3 deleteCalibration

Not implemented yet.

Chapter 8

Entry Point Three

In *Linssi* manual Part I [1] entry point 3 is defined as procedures related with analysing a gamma-ray spectrum. The spectrum may come from earlier entry points, from outside, or from a *Linssi* database.

Some of these scripts refer to AKu input file format that is documented in Ch. 14. Some scripts refer to ODBC that is clarified in Ref. [5].

8.1 analysistodb

Purpose: Script for reading analysis data into *Linssi* database

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 23.12.2005

Updated: 23.2.2006
Version 1.1.1a: Fixed comments field in analysis-tables.
5.7.2006 JAH : Documentation

Description: Script for importing data into analysis tables of *Linssi* database. Data is read from stdin unless -f -option is used. The AKu input format (see *Linssi* manual Part II) is either type 1 or type 2 depending on the table:
Type 1:

```
#table_name
firstValue
secondValue
<blobName>
BLOB
BLOB
</blobName>
thirdValue

#another_table_name
...
```

Type 2:

```
#table_name
firstValue1 secondValue1 "BLOB BLOB" thirdValue1
```

```
 firstValue2 secondValue2 "BLOB BLOB" thirdValue2
 ...
```

Empty line alone (except inside a blob) is assumed to end the listing of fields for the current table. Most table names should exist only once in the file. Exception to this are the calibrations, calPreferences and calPoints tables. In these blocks idCal field for the same calibration appearing in different blocks should be the same. Preferably it should start from one, so first calibration in the file will have idCal 1, next idCal 2 and so on. Note however that this script assigns different idCals to calibrations during input process to prevent multiple calibrations in database from having the same idCal.

Following tables use the type 2 format: peaks, lineAssociations, activities, nuclideRatios, activityLimits and calPoints. Sample and measurements-tables are a special case in the script. These blocks can either follow the normal form of the table or alternatively they can contain only the first two (samples) or seven (measurements) values of normal block. In either case only the sampleId/measId-field is used from these blocks. That field is used to correlate the rest of the data in file to the correct sample already in database.

Lines containing only "&" are treated as NULL value when entering it into table.

The database name and username and password can be given as arguments for the script. If you wish to use no password give empty string as password (-p ''). If username, database name, password or odbc-parameter are not given, the values in the configuration file are used instead (username and password are those of write-user). See below for more details. Note that you need to have the linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script, or add 'use lib "linssiConfigPath"' at the beginning of file. If -g -option is not given, the default file ($ENV{HOME}/.linssirc) is used.

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax:

```
analysistodb [-f reportfile] [-d database_name/DSN] [-u username]
[-p password] [-g configFile] [-i] [-odbc|-noodbc]
```

Options:

```
-f Data is read from a given file.  If this option is not given, the
report is read from stdin.

-i Script reports assigned idAnalysis and idCal after processing.

-g configFile Use configFile as configuration file.  If database name,
username, password and/or odbc-option are not given, the values
read from configFile instead (see linssiConfig.pm for details).
If this option is not given, $ENV{HOME}/.linssirc is used as
configuration file.

-odbc Use ODBC instead of DBI's internal MySQL drivers.  In that case
DSN is used instead of database name (see ODBC documentation).

-noodbc Use DBI's internal MySQL drivers instead of ODBC. If neither
odbc nor noodbc is given, the value in configuration file is
used.  If there there is no odbc paramater in configuration file
```

`inter MySQL drivers are used.`

Writes tables: calibrations, analyses, calPreferences, calPoints, peaks, lineAssociations, activities, nuclideRatios, activityLimits, finalResults

Modules: linssiConfig.pm v.1.1.1 or newer

8.2 deleteAnalysis

Purpose: Deletes an analysis and all associated data from *Linssi*

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 23.7.2003

Updated: 18.1.2004
5.5.2005
Updated for v.1.1 table specification.
16.11.2005
Version 1.1.1: Uses linssiConfig.pm and supports configuration files. -g -option added. Supports empty strings as password. Added –noodbc - option. Syntax slightly modified.
5.7.2006 JAH : Documentation

Description: Script for deleting an analysis and all associated data from *Linssi* database v.1.1.
The database name, username and password can be given as arguments for the script. If you wish to use no password give empty string for password (-p ''). If username, database name, password or odbc-parameter are not given the values in the configuration file are used instead (username and password are those of write-user). See below for more details. Note that you need to have the linssiConfig.pm file either in one of the perl library directories or in the the same directory as this script or add 'use lib "linssiConfigPath"' at the beginning of file. If -g -option is not given, the default file ($ENV{HOME}/.linssirc) is used.

Configuration: Script supports the following options of .linssirc: databaseName, odbc, [write] username, [write] password

Syntax: `deleteSample -a idAnalysis [-d database_name] [-u username] [-p password] [-g optionFile] [-odbc|-noodbc]`

Writes tables: analyses, calPreferences, peaks, lineAssociations, activities, activityLimits, nuclideRatios, finalResults

Modules: linssiConfig.pm v.1.1.1 or newer

8.3 analysisReport.php

Purpose: Prints different kinds of analysis reports

Version: 1.1.6

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.10.2005

Updated: Added link to showCorrFactors.php - tos
4.5.2006 : If sampleProductionTable is ctbtLabSamples and stationSplitAirVolume is set, use that instead of airVolumeTotal - tos
8.5.2006 : list of other measurements from the same sample can be set off from linssiconfig.php - tos
12.5.2006 : fixed a bug related to activities and ctbtLab samples - tos
20.6.2006 - changed the hard coded port 80 to $_SERVER['SERVER_PORT'] in sendToHost function -tos
5.7.2006 JAH : Documentation

Description: Prints different kinds of analysis reports (short, long, peaks, associations). It's also possible to get analysis reports in PDF format (modified HTML2FPDF library must be installed on the server). Parameters can be idAnalysis OR idMeas and software OR idSample and software OR idMeas OR idSample. In cases when there is more than one analysis matching the parameters, the one with final status (or with highest idAnalysis) is printed. Links in the report toolbar can be removed by changing the settings in the linssiConfig.php configuration file.

Reads tables: analyses, finalResults, activities, peaks, lineAssociations, measurements, samples, airfilterSamples, ctbtLabSamples, stations, measurementSetups, sources, calPreferences, calibrations, activityLimits

Modules: linssiConfig.php, linssiPHPLib.php, htmlTags.php, HTML2FPDF, showSpectrum.php, spectrumPart.php, peakGraph.php, editAnalysis.php, eurdep.php, showCorrFactors.php

8.4 dbtophd

Purpose: Creates PHD file of given measurement from *Linssi* database

Version: 1.1.4

Author: Antero Kuusi

Package: *Linssi* core

Created: 3.6.2004

Updated: 30.8. 2004
Added -c -option.
31.8.2004
Added -e -option
30.11.2004
Linssi v.1.01 to v.1.1 upgrade
Dropped -e option (always used)
14.03.2005 AP
Reading defaults from config file (~/.linssirc)
23.11.2005 AP

Version 1.1.3b
DataType and SystemType decided by samples.sampleType
Option -i idCal added
21.02.2006 AP
Version 1.1.3b2
Bug Fixes: airVolume for split samples, background MID, spacing
23.02.2006 AP
Version 1.1.3b4
g_TotalEfficiency to TotalEff, Linssi:sourceId from measurements.sourceId,
GasBackgroundMID (0) added. -f option bug solved.
5.7.2006 JAH : Documentation

Description: This script creates PHD file from information in *Linssi* database. The table specifications are according to the database definition 1.1.
The script uses database name, username and password given as arguments. If database name or username are not given, they are read from configuration file (~/.linssirc)
The PHD file is not probably exactly as the one used to create the data in database. This depends completely on the input system used.

Syntax:
```
dbtophd [-d databaseName] [-u username] [-p password] -f outfile
-m measId|-h phdMeasName [-i idCal] [-C configFile] [-c]
```

Options:
```
-d databaseName Name of the database that contains the information.
If not given the name in the beginning of the script is used.

-u username Username to be used.  User need to have select rights to
the database.  If not given the name in the beginning of the
script is used.

-p password Password for the user.  If not given the one in user's
.linssirc will be used.

-f outfile Filename for the output file.

-m measId MeasId of the measurement.

-h phdMeasName PhdMeasName of the measurement.

-i idCal idCal of the calibration set to be reported, default is
idCal in measurementSetups table.

-C configFile Read configuration from configFile instead of default
config file ~/.linssirc.

-c Add sampleId, measId, stationId, measSetupId, idMeas,
idCal and sampleType to #Comments block.
```

Chapter 9

Reporting and Displaying Scripts

The basic reporting and graphical display scripts linked to `LinssiWorld` are presented in this chapter.

9.1 showSpectrum.php

Purpose: Prints a line graph image of a spectrum with identifications

Version: 1.1.3

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.6.2005

Updated: Don't fetch nuclides if their energy is null - tos
8.6.2006 : two for loops optimized - tos
5.7.2006 JAH : Documentation

Description: This script prints a line graph image of a spectrum with nuclide identifications given the idAnalysis number. HTML-page also includes a javaScript function which is used to zoom into the certain part of the spectrum (zoomSpectrum.php required). Script also fetches the strippedSpectrum and baseline from the database and inserts them into PHP session variables, which are later used by the zoomSpectrum.php.

Reads tables: samples, airFilterSamples, measurements, analyses, calPreferences, calibrations, peaks, lineAssociations

Modules: JpGraph, linssiConfig.php, linssiPHPLib.php

9.2 zoomSpectrum.php

Purpose: Zoom feature for showSpectum.php

Version: 1.1.4

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.6.2005

Updated: modified energy shifts functionality - tos
To improve speed, script modified to handle only those channels which are inside the zoomed area - Sakari Ihantola
Fixed a bug which occurred if register_globals setting in PHP was on - Tommi Salonen & Sakari Ihantola
5.7.2006 JAH : Documentation

Description: A zoom feature for showSpectum.php. A 100 keV wide portion of the spectrum is shown in a separate browser window, together with strippedSpectrum, baseline, fitted peak and nuclide identifications (if available). User can select either linear or logarithmic vertical scale. Further zooming in/out and scrolling are also implemented.

Modules: JpGraph, linssiConfig.php

9.3 editAnalysis.php

Purpose: Script for manual editing of analysis results

Version: 1.1.2

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.6.2005

Updated: 4.5.2006 - If sampleProductionTable is ctbtLabSamples and stationSplitAirVolume is set, use that instead of airVolumeTotal - tos
5.7.2006 JAH : Documentation

Description: Script for manual editing of analysis results. This script can be used to change the calculation method for activity concentrations, change analysis status or to create a new manual analysis from existing one by deleting some nuclides.

Reads tables: analyses, peaks, lineAssociations, activities, activityLimits, nuclideRatios, finalResults, calPreferences, samples, airFilterSamples, ctbtLabSamples

Writes tables: analyses, peaks, lineAssociations, activities, activityLimits, nuclideRatios, finalResults, calPreferences

Modules: linssiConfig.php, linssiPHPLib.php, htmlTags.php

Chapter 10

Linssi Script Reference List

This chapter presents the list of *Linssi* scripts in the distribution package as of July 2006. A majority of these scripts were presented in more detail in the previous chapters, but there are a number of scripts that are listed only here. The categorization of scripts differs slightly from the previous chapters.

New scripts are probably added to the distribution more frequently than this document is updated. An up-to-date list of the scripts will be maintained on the *Linssi* home page http://linssi.hut.fi/radphys/linssi/.

10.1 Configuration Scripts and Libraries

linssirc-template

Purpose: Template for the configuration file .linssirc
Version: 1.1.3
Author: Andreas Pelikan
Package: *Linssi* core (v.1.1)
Created: 7.3.2005

linssiConfig.pm

Purpose: Package for handling *Linssi* configuration files
Version: 1.1.3
Author: Antero Kuusi
Package: *Linssi* core (v.1.1)
Created: 10.10.2005

linssiGetoptStd.pm

Purpose: Package for parsing command line options from *Linssi* config file
Version: 1.1.2
Author: Andreas Pelikan
Package: *Linssi* core
Created: 18.11.2005

linssiConfig.php

Purpose: *Linssi* PHP scripts configuration file

Version: 1.1.2

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.10.2005

linssiFormTools.php

Purpose: Functions for printing of forms and checking parameters

Version: 1.1.3

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.10.2005

linssiPHPLib.php

Purpose: PHP functions

Version: 1.1.3

Author: Tommi Salonen

Package: *Linssi* core

Created: 1.10.2005

htmlTags.php

Purpose: Perl CGI-like library, with functions that generate html tags

Version: 1.1.1

Author: Andreas Pelikan

Package: *Linssi* core

Created: 27.1.2006

collectInfo

Purpose: Collects tagged comments from code and exports them to LaTeX

Version: 1.1.2

Author: Sakari Ihantola

Package: *Linssi* core

Created: 13.6.2006

10.2 Database Creation Scripts

preparedb

Purpose: Create database and user accounts
Version: 1.1.1
Author: Andreas Pelikan
Package: *Linssi* core
Created: 18.11.2005

maketables

Purpose: Creates tables in *Linssi* database
Version: 1.1.2
Author: Antero Kuusi
Package: *Linssi* core (v.1.1)
Created: 15.7.2003

desctables

Purpose: Shows descriptions of all tables in a *Linssi* database
Version: 1.1.6
Author: Jarmo Ala-Heikkila
Package: *Linssi* core
Created: 11.2.2004

10.3 Basic Housekeeping Scripts

checkdb

Purpose: Checks *Linssi* database for errors and inconsistencies in data
Version: 1.1.2
Author: Antero Kuusi
Package: *Linssi* core (v.1.1)
Created: 4.6.2004

fixdb

Purpose: Fixes errors and inconsistencies in *Linssi* database
Version: 1.1.2
Author: Antero Kuusi
Package: *Linssi* core (v.1.1)
Created: 9.10.2005

deleteSample

Purpose: Deletes a sample and all associated data from *Linssi*

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 23.7.2003

deleteMeas

Purpose: Deletes a measurement and all associated data from *Linssi*

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 23.7.2003

deleteAnalysis

Purpose: Deletes an analysis and all associated data from *Linssi*

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 23.7.2003

10.4 Basic Database Input Scripts

filetodb

Purpose: Imports sample, measurement and analysis data into *Linssi* database

Version: 1.1.1

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 23.12.2005

analysistodb

Purpose: Script for reading analysis data into *Linssi* database

Version: 1.1.2

Author: Antero Kuusi

Package: *Linssi* core (v.1.1)

Created: 23.12.2005

meastodb

Purpose: Script for reading measurement data into *Linssi* database
Version: 1.1.2
Author: Antero Kuusi
Package: *Linssi* core (v.1.1)
Created: 19.12.2005

sampletodb

Purpose: Script for reading sample data into *Linssi* database
Version: 1.1.3
Author: Antero Kuusi
Package: *Linssi* core (v.1.1)
Created: 19.12.2005

stufftodb

Purpose: Script for reading miscellaneous data into *Linssi* database
Version: 1.1.2
Author: Antero Kuusi
Package: *Linssi* core (v.1.1)
Created: 30.11.2005

10.5 Data Extraction Scripts

dbtofile

Purpose: Generates database reports from *Linssi* database
Version: 1.1.4
Author: Antero Kuusi
Package: *Linssi* core (v.1.1)
Created: 20.3.2005

dbtophd

Purpose: Creates PHD file of given measurement from *Linssi* database
Version: 1.1.4
Author: Antero Kuusi
Package: *Linssi* core
Created: 3.6.2004

dbtorlr

Purpose: Creates an rlr report in IMS2.0_IDCR6 format
Version: 1.1.1
Author: Andreas Pelikan
Package: *Linssi* core
Created: 03/23/05

createPhd.php

Purpose: Creates a PHD file from the chosen analysis
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.3.2006

eurdep.php

Purpose: Creates an analysis report in Eurdep 2.0 format
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.12.2005

10.6 Scripts for Handling Calibrations

calibrations.php

Purpose: Tool for viewing calibrations
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

addCalibration.php

Purpose: Adds a new calibration set to *Linssi* database
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

10.7 Scripts for Interactive Data Browsing and Analysis

changeDb.php

Purpose: Prints a database selection form
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

showSamples.php

Purpose: Fetches sample data from chosen stations in a time interval
Version: 1.1.6
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.10.2005

analysisReport.php

Purpose: Prints different kinds of analysis reports
Version: 1.1.6
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.10.2005

showSpectrum.php

Purpose: Prints a line graph image of a spectrum with identifications
Version: 1.1.3
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

showCorrFactors.php

Purpose: Shows correction factors and raw activities for nuclides
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* core
Created: 2.3.2006

zoomSpectrum.php

Purpose: Zoom feature for showSpectum.php
Version: 1.1.4
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

editAnalysis.php

Purpose: Script for manual editing of analysis results
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

peakGraph.php

Purpose: Shows the chosen peak as a line graph image
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

spectrumPart.php

Purpose: Creates a linegraph image of certain part of a spectrum
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

anomalousNuclides.php

Purpose: Defines an array that contains all anomalous nuclides
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.10.2005

showWeather.php

Purpose: Shows selected weather data from selected weather stations
Version: 1.1.2
Author: Tommi Salonen, Satu Kuukankorpi
Package: *Linssi* core
Created: 8.11.2005

10.8 Report Generating Scripts

sampleReport.php

Purpose: Creates sampling reports in html and excel format
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.10.2005

showDetectors.php

Purpose: Shows the detectors from the database
Version: 1.1.1
Author: Teemu Siiskonen, Tommi Salonen
Package: *Linssi* core
Created: 1.5.2006

showMeasurementSetups.php

Purpose: Shows the measurementSetups from the database
Version: 1.1.1
Author: Teemu Siiskonen, Tommi Salonen
Package: *Linssi* core
Created: 1.5.2006

showSources.php

Purpose: Shows the sources from the database
Version: 1.1.1
Author: Teemu Siiskonen, Tommi Salonen
Package: *Linssi* core
Created: 1.5.2006

showStations.php

Purpose: Shows the stations from the database
Version: 1.1.1
Author: Teemu Siiskonen, Tommi Salonen
Package: *Linssi* core
Created: 1.5.2006

showSamplers.php

Purpose: Shows the samplers from the database
Version: 1.1.1
Author: Teemu Siiskonen, Tommi Salonen
Package: *Linssi* core
Created: 1.5.2006

trends.php

Purpose: Creates a line graph image showing activity concentrations
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

anomalies.php

Purpose: Prints a report that shows all detected anomalous nuclides
Version: 1.1.4
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

qc.php

Purpose: Shows the resolution for certain nuclides for QC purposes
Version: 1.1.3
Author: Tommi Salonen
Package: *Linssi* core
Created: 1.6.2005

10.9 CTBT Laboratory Scripts

ctbtConfig.php

Purpose: CTBT script configuration file
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtWebConfig.php

Purpose: CTBT webscript configuration file
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtCheckMail.php

Purpose: Checks new mail and inserts the messages to database
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtLabSamples.php

Purpose: User interface to the CTBT scripts
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtMsgLib.php

Purpose: Functions for CTBT scripts
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

labDataMsg.php

Purpose: Class representing a LabDataMsg
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtMessage.php

Purpose: Functions to handle LabData messages
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtPHD.php

Purpose: Functions to handle PHD messages
Version: 1.1.2
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtRecipient.php

Purpose: Functions to handle recipients
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtRLR.php

Purpose: Functions to handle RLR messages
Version: 1.1.0
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtSampleTrackings.php

Purpose: Functions to handle sample tracking
Version: 1.1.3
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtSource.php

Purpose: Functions to handle sources
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

ctbtChangeDb.php

Purpose: Prints a database selection form
Version: 1.1.1
Author: Tommi Salonen
Package: *Linssi* CTBT scripts
Created: 1.4.2006

Chapter 11

Interface between *Linssi* and Analysis Software

This chapter starts the second topic of this manual, i.e., interfacing *Linssi* with analysis software. The UniSampo–Shaman software package can be understood as an illustrating example, but this chapter also serves as a documentation of the functional UniSampo-Shaman-*Linssi*-system at STUK.

11.1 *Linssi* with UniSampo–Shaman

The connection between UniSampo–Shaman (USS) and the *Linssi* database is based on temporary result files and scripts that import the file contents to database. The database support has been built on the file-based USS-system, so everything else in the analysis package works as without *Linssi* [3].

The operation of the UniSampo–Shaman system is mainly handled by a basic analysis script called `shaman_run`. It takes care of the settings required by the analysis software packages UniSampo and Shaman prior to calling them. The management of database insertion is also controlled by `shaman_run`.[a]

The `shaman_run` script supports three basic operational modes: a pipeline mode, a non-pipeline batch mode and an interactive mode. The support for storing data to *Linssi* is available in all three modes. In each operational mode, `shaman_run` calls the binaries UniSampo and Shaman in due order and thus generates the temporary database reports `.uda` and `.udb` from UniSampo and `.sdb` from Shaman. Whether or not these files are processed to the database depends on the configuration (pipeline mode) or user actions (batch and interactive mode).

It should be noted that even though the `shaman_run` script supports analysis of several file formats (`phd, asc, ids, mac, xml`), database insertion is currently possible only for `phd`-files.[b] This is because the other formats do not support the essential keys required by *Linssi* (Ch. 12). Generating unique and compatible database keys is **the key issue** when

[a] This means that if UniSampo or Shaman is invoked any other way than by `shaman_run`, their results cannot be stored to *Linssi*. There may also be problems with the database connection if the spectrum is acquired using UniSampo's MCA connection, i.e., not given to it as input.

[b] If `shaman_run` is invoked with an `asc`, `ids` or `mac`-file **that was previously created during `phd`-file analysis**, the results can be stored in database. However, this is not possible for `asc`, `ids` or `mac`-files created any other way.

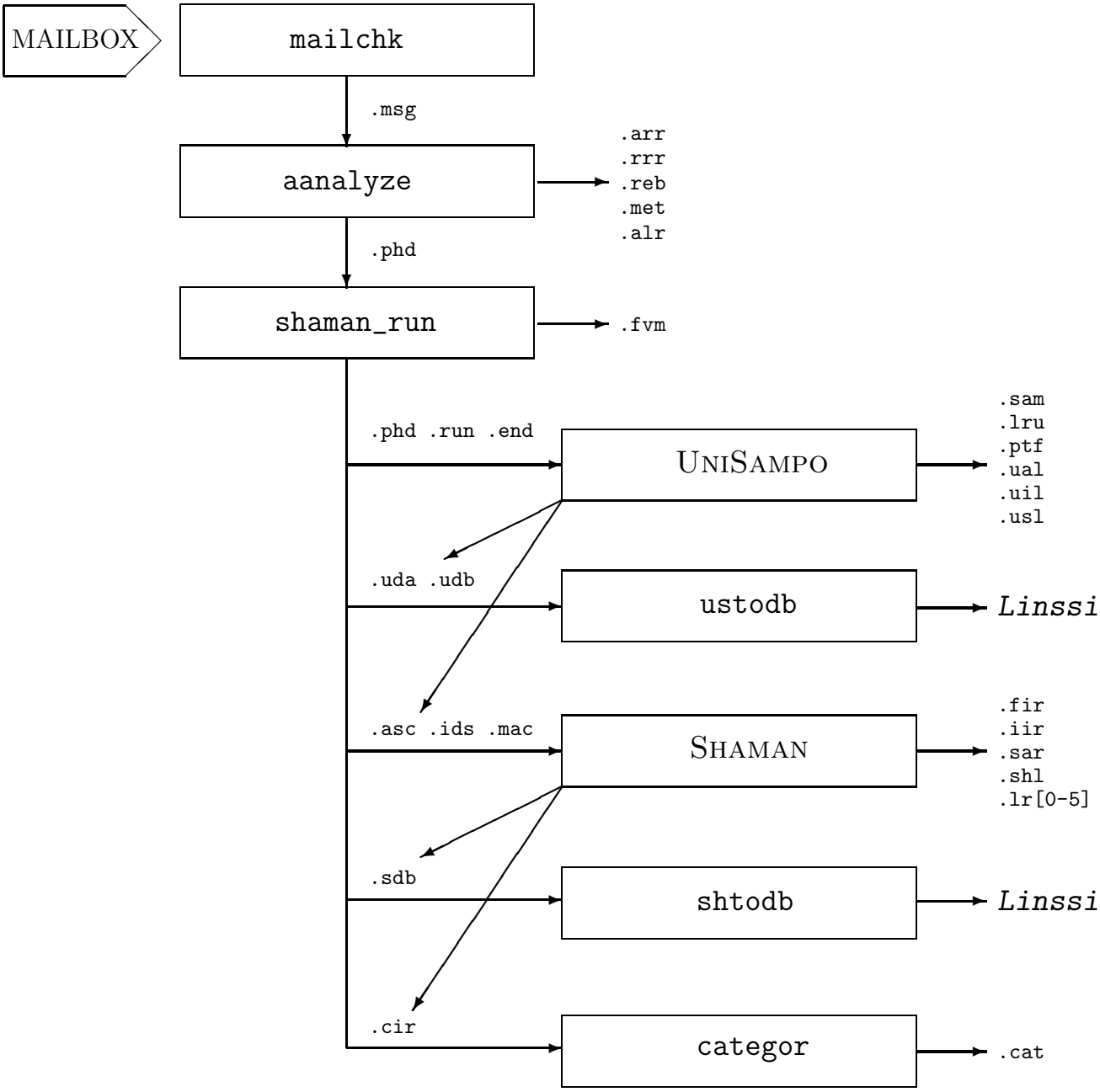


Figure 11.1: Flow chart of the UniSampo–Shaman pipeline with *Linssi*.

interfacing analysis software with *Linssi*.

If the database insertion scripts are invoked in any operational mode of `shaman_run`, this is done in the foreground. If the database is blocked, this will also interrupt the analysis progress. Database insertion in the background would solve this problem and it was experimented with earlier versions of *Linssi*. However, it is impossible to make correct database links between UniSampo and Shaman results in that operational mode, so it had to be abandoned. This means that any database jams need to be resolved promptly, making usage of alarming scripts necessary.

11.1.1 Database Insertion in the Pipeline Mode

The UniSampo–Shaman analysis pipeline is illustrated in Fig. 11.1. It is operated by the `mailchk` script that polls a dedicated mailbox at constant intervals. All mail messages are given to the script `aanalyze` script that investigates their contents. If an incoming mail message contains a spectrum, `aanalyze` calls the `shaman_run` script that makes the initial settings and then calls UniSampo and Shaman in due order.

The USS-pipeline saves analysis results under the directory `/home/shaman/dbroot` (or similar) in the reporting formats produced by UniSampo and Shaman. The different reports can be distinguished by their filename extensions that are presented in Fig. 11.1. The reports include temporary database reports with extensions `.uda`, `.udb` and `.sdb` that are inserted to the database with the scripts `ustodb` and `shtodb`. Their functionality is explained on p. 57.

Storing of analysis results to a *Linssi* database can be turned on by setting the variable `linssidb=y` in Shaman's configuration file `shaman.config`. Additionally, the pipeline script checks that the scripts for database insertion `ustodb` and `shtodb` are available.

Indirectly, a successful database connection also requires the file `~/.linssirc`, since it is consulted by the `*todb` scripts in the *Linssi* package that are used by `ustodb` and `shtodb`. The usernames and passwords in `~/.linssirc` must naturally be valid. If there are several databases available, there should be a separate `~/.linssirc*` file for each of them. The USS system utilizes environmental variables `LINSSIRC_READ` and `LINSSIRC_WRITE` for selecting the configuration file for the input and output database, respectively.

Data for all spectra processed by the pipeline are stored in the database by default if database name and username have been defined. However, spectra can be excluded from the database on the basis of spectrum type (`FULL, PREL, BLNK, BACK, CALI, QCSP, XXXX`), sampling station (e.g., `XXX00`) and detector (e.g., `XXX00-001`). The exclusion rules are configured in the `shaman.config` file.

Note that these exclusion rules are only applied in pipeline operation. In other operational modes full control is given to the user, provided that he/she has permissions for database updating.

11.1.2 Database Insertion in the Batch Mode

Non-pipeline batch mode means that `shaman_run` is invoked from the Unix prompt with the `-b` option. The graphical user interfaces of UniSampo and Shaman are not displayed in this mode. The mode is very similar to the pipeline mode, but there are some differences. **The difference in storing of analysis results in the database is that in the batch mode, this is done only when the command line option `-l` is used, whereas in the pipeline mode all results are stored in the database by default.** Furthermore, the exclusion rules that are applied in the pipeline mode are not applied in the batch mode.

Note that the database username and password need to be defined in the configuration file `~/.linssirc` of the user running the `shaman_run` script. The environment variables `LINSSIRC_READ` and `LINSSIRC_WRITE` can be used to select the input and output database configuration file, respectively.

11.1.3 Database Insertion in the Interactive Mode

In the full interactive mode with graphical user interfaces, analysis results from

UniSampo and Shaman are not stored in the database by default, but only on user's request.

In UniSampo's case, the user is prompted for storing its final results in database upon exiting. The default reply to the prompt, given by pressing the `Enter` key or anything else but `y` and `Enter`, is not to save the results.

UniSampo's intermediate analysis results can be stored to database upon selecting the command "Store Analysis Results to Linssi" available in the Macro-menu. Please note that it is the user's responsibilty to ensure that the calibrations and analysis results are consistent upon storing.

In Shaman's case, there are no intermediate results, only final results. They are stored to database upon pressing the "Store to Database"-button that is available in the standard button window of Shaman if a database has been configured. The button opens a confirmation dialogue prior to invoking the database storing script.

Note that the database username and password need to be defined in the configuration file `~/.linssirc` of the user running the `shaman_run` script. The environment variables `LINSSIRC_READ` and `LINSSIRC_WRITE` can be used to select the input and output database configuration file, respectively.

11.2 Generation of Database Keys and Identifiers

Since analysis software packages run on entry point 3 of *Linssi*, they require knowledge of some essential database keys as an input in order to provide consistent output to *Linssi*. The adopted method to define these essential database keys, listed in Ch. 12, and some additional string identifiers of *Linssi* is to utilize the `#Comment` block of the PHD-spectrum as explained in Ch. 13.

The keys and identifiers that are not input using this method are generated by the `shaman_run` script during runtime. The script extracts them from the other blocks of the PHD-spectrum as described below. In this process, it is essential that the spectrum conforms to the PHD-format definition by the CTBTO [4]. The blocks that are utilized are `#Header`, `#Collection` and `#Acquisition`.

a. `sampleId` [c]

- sampling station (site), `samsta`, from the second line of `#Header` block
- sampling start date, `samstartdate`, and time, `samstarttime`, from the second line of `#Collection` block
  - if the source is not sampled, take acquisition start date and time from the second line of `#Acquisition` block
- split symbol is the last two characters of the `phdSampleName` (SRID) in `#Header` block
  - if the source is not sampled, `splitsymbol=00`
- sequential integer, `seqint`, is generated from sampling start date: it is the 3-digit day of year 001 . . . 366

[c] Please note that the `sampleId` generated from standard PHD fields, like presented here, does not conform to the definition of Ch. 12. This is because the components sampler code and filter type are not defined in the PHD format.

- if the source is not sampled, `seqint=000`

```
sampleid=${samsta}_${samstartdate}_${samstarttime}_${splitsymbol}_${seqint}
```

b. `measId`

- MID, `phdmid`, from the fourth line of `#Header` block
- live aqcuisition time, `livetime`, from the second line of `#Acquisition` block

```
measid=${phdmid}_${livetime}
```

c. `extSampleName`

- if this key is not given in `#Comment` block, its value is `null`.

d. `extMeasName`

- if this key is not given in `#Comment` block, its value is `null`.

e. `stationId`

- sampling station (site), `samsta`, from the second line of `#Header` block

```
stationid=${samsta}
```

f. `samplerId`

- sampling station (site), `samsta`, from the second line of `#Header` block

```
samplerid=${samsta}
```

g. `sourceId`

- if this key is not given in `#Comment` block, its value is `null`.

h. `detectorId`

- detector code consisting of measuring station and detector, `measta` and `meadet`, from the second line of `#Header` block

```
detectorid = ${measta}${meadet}
```

i. `measSetupId`

- detector code consisting of measuring station and detector, `measta` and `meadet`, from the second line of `#Header` block
- measuring geometry, `measgeom`, from the second line of `#Header` block

```
meassetupid = ${detectorid}_${measgeom}
```

j. `blankIdMeas`

– if this key is not given in `#Comment` block, its value is `null`.

k. `backgroundIdMeas`

– if this key is not given in `#Comment` block, its value is `null`.

l. `blankIdAnalysis`

– if this key is not given in `#Comment` block, its value is `null`.

m. `backgroundIdAnalysis`

– if this key is not given in `#Comment` block, its value is `null`.

n. `inputIdAnalysis`

– if this key is not given in `#Comment` block, its value is `null`.

o. `inputIdCal`

– if this key is not given in `#Comment` block, its value is `null`.

p. `calibrations.class`

– if this key is not given in `#Comment` block, its value is `null`.

q. `measurementSetups.idCal`

– if this key is not given in `#Comment` block, its value is `null`.

r. `combined`

– if this key is not given in `#Comment` block, its value is 0.

s. `sampleType`

The line `DATA_TYPE` specifying the PHD-spectrum type and the system type (`P/B/G`) on the second line of the `#Header` block are mapped to the following `sampleType`'s:

– spectrum type `SAMPLEPHD` and system type `B` or `G` ⇒ `gassample`

– spectrum type `GASBKPHD` and system type `B` or `G` ⇒ `gasbackground`

– spectrum type `DETBKPHD` and system type `P` ⇒ `background`

– spectrum type `BLANKPHD` and system type `P` ⇒ `blank`

– spectrum type `CALIBPHD` and system type `P` ⇒ `calibration`

– spectrum type `QCPHD` and system type `P` ⇒ `qcspectrum`

– spectrum type `SAMPLEPHD` and system type `P` ⇒ `airfilter` or `environmental` depending on the command line option given to `shaman_run` (`-a` or `-e` correspondingly)

This list needs to be updated when support for new sample types is added to *Linssi* and the analysis software.

t. `sourceDensity` [d]

– if this parameter is not given in `#Comment` block, its value is `null`.

u. `sourceThickness` [d]

– if this parameter is not given in `#Comment` block, its value is `null`.

11.2.1 Technical Implementation

`shaman_run` is a Bourne shell script where the different keys are defined as variables. These variables are input to UniSampo's and Shaman's database reports similarly in principle, but the technical implementation differs slightly:

1. UniSampo generates preliminary database reports (`.uda`, `.udb`) where database fields containing strings irrelevant for the analysis are presented as placeholders separated by two at-characters, e.g., `@sampleId@`. These placeholders are replaced afterwards by the `ustodb` script using the Unix `sed`-command and the shell script variables. This is because the RGL report generator of UniSampo does not support command line arguments.

2. Shaman generates directly the final database report (`.sdb`), i.e., including correct values also for the database fields irrelevant for the analysis. This is because the RGL report generator of Shaman supports command line arguments that are written verbatim to the correct places in the database report. The `shtodb` script is therefore only a symbolic link to the generic `filetodb` script.

An additional complication arises from the need to output the calibration data pairs input to UniSampo, in addition to those that were possibly updated by it during processing. This is implemented so that the same RGL report is produced by UniSampo with original calibrations (`.uda`) and with updated calibrations (`.udb`). The tables `calibrations`, `calPoints` and `calPreferences` are extracted from the `.uda` file, their `idCal` and `usedInAnalysis` fields are modified and finally appended to the `.udb` file. Then the placeholder replacement described above takes place.

The `ustodb` script also extracts the `#Certificate`-block from the PHD-file, if available, and inserts it into its proper place in the `calibrations` tables of the `.uda` report. In `calibrations` tables of the `.udb` report, on the other hand, no certificate information is available, since they are usually internal calibrations.

[d] The source parameters are included on the list, because they may be used by Shaman for self-absorption correction. For this purpose, they must be accompanied by a setting of `calibrations.class` to `SOURCE`.

11.3 Tables Updated by UniSampo and Shaman

The following tables are updated by UniSampo (U) and Shaman (S):

3	airFilterSamples	U	S
3	calibrationSamples	U	S
7	samples	U	S
12	measurementSetups	U	S
13	measurements	U	S
---	---	---	---
14	calibrations	U	
14.1	calPreferences	U	S
15	calPoints	U	
16	analyses	U	S
17	peaks	U	S
18	lineAssociations		S
19	activities	U	S
20	activityLimits	U	S
21	nuclideRatios		S

The numbering refers to that used in Fig. 1.3 of the *Linssi* manual Part I [1].

The last 9 tables are clearly analysis results from USS. Every new analysis is given a unique `analysisId` and saved in the database as such. The possibilities for manual modification of USS-results in the database should be kept at minimum for traceability reasons. Of course, the results may be later removed from the database if they are grossly erroneous and if this is the practice of the organization. Some organizations may keep all analysis results in the database for the record and only flag the grossly erroneous results in the `comments` field of each table, for example. The practice should be decided by the organization itself and an administration script should be written for removing or flagging analysis result with a given `analysisId` when found necessary.

The first 5 tables on the list mainly contain input for USS or fields that are not needed by them. They are always output by USS, but they should be ignored in cases when the sample has been collected or at least measured by the organization itself (**entry points 0, 1 and 2** in Fig. 1.2 of the *Linssi* manual Part I [1]). In this case, the tables should contain the correct information already prior to running UniSampo and Shaman, i.e., the software packages cannot add any relevant information to these tables.

The first 5 tables are needed in cases when the organization receives a measured spectrum from outside, e.g., a spectrum measured by the IMS network of the CTBTO (**entry point 3** in Fig. 1.2 of the *Linssi* manual Part I [1]). In this case the sample and measurement data are missing from the database prior to running UniSampo and Shaman. By using the first 5 tables output by the USS, the essential information can be retrieved from the PHD-spectrum to the database, possibly for manual completion later.

Chapter 12

Adopted Syntax for Unique Keys

The database includes a number of unique keys. Some of these keys are auto-increment integers that are maintained by the database itself. However, there are also some string keys (type varchar) whose uniqueness must be assured by the user. This means that naming practices need to be defined for these keys by the organization using *Linssi*.

The most essential keys in *Linssi*, and in gamma spectrometry in general, are the sample and measurement identifiers, for which character strings are used in *Linssi*. The spectrum file format that is understood by the UniSampo–Shaman package is the PHD-format defined by the CTBTO. In connection with the definition of the PHD-format, the formats for the sample and measurement identifiers (Sample Reference Identification, SRID, and Measurement Identification, MID, in the CTBTO jargon) as well as their positions in the PHD-file are defined in the document "Formats and Protocols for Messages – Revision 6" (IDC-3.4.1Rev6) [4].

However, the definitions in IDC-3.4.1Rev6 are quite limited and actually, the definition of the MID fails to be unique: it only includes the acquisition start time that is identical if spectra are measured in slices ($N\times$PREL-measurements and a FULL-measurement without acquisition restart in between). Another complication arises in the situation where a spectrum or a sample is given to analysis from an outside customer that has a different naming practice than the analyzing organization.

In order to have degrees of freedom, there are four different fields for both the sample and measurement identifier in *Linssi*:

sample: `idSample` (int), `sampleId` (varchar), `phdSampleName` (varchar), i.e., SRID, and `extSampleName` (varchar) in table `samples`

measurement: `idMeas` (int), `measId` (varchar), `phdMeasName` (varchar), i.e., MID, and `extMeasName` (varchar) in table `measurements`

In both cases, the integer key is made unique by the database itself and the uniqueness of `sampleId` and `measId` must be secured by the user. The fields with prefix `phd` and `ext` are not required but only recommended to be unique by the database specifications.

The current USS-implementation defines `phdSampleName` and `phdMeasName` to obey the naming practices of IDC-3.4.1Rev6. The `sampleId` and `measId` fields use another definition by the Finnish Radiation and Nuclear Safety Authority (STUK). These format definitions are presented below.

12.1 idSample

An auto-increment integer key managed by the database.

12.2 sampleId

The key `sampleId` is defined by STUK/ASL for a sampled source (with a collection start time) as:

```
cccccccccc_yyyy/mm/dd_hh:mm:ss_Pp_xxx

  cccccccccc    station code + sampler code + filter type
  yyyy/mm/dd    collection start date
  hh:mm:ss      collection start time
  Pp            split identifier: P = split number, p = total nr of splits
  xxx           sequential integer

Example: HEP02CI01F_2004/02/17_08:04:00_11_18
```

For a source that is not sampled (blank, background, QC, calibration etc.), `sampleId` has the same format but with the following field contents:

```
  cccccccccc    station code / detector code
  yyyy/mm/dd    acquisition start date
  hh:mm:ss      acquisition start time
  Pp            split identifier, always "00"
  xxx           sequential integer, always "000"

Example: FI001-D01_2004/02/05_10:05:03_00_000
```

Please note that a `sampleId` generated from standard PHD spectrum fields cannot conform to this definition. This is because the components sampler code and filter type are not defined in the PHD format. This definition can be applied only when specifying the keys in the `#Comment` block like documented in Ch. 13.

12.3 phdSampleName

The key `phdSampleName` obeys the definition of SRID of the CTBTO:

1. In the case of air filters with pre-printed barcodes where the SRID format below is impossible to follow, any unique SRID for every air filter will be considered valid.
2. The format of the SRID for filter samples is a 14 or 15-character code:

```
ccyyyymmddhhPpT

  cc            CTBT station number (e.g., CKP23 -> 23)
  yyyymmddhh    collection start
```

```
  Pp            split identifier: P = split number, p = total nr of splits
  T             station type: G for noble gas, blank for particulate

Example: 23200305230711
```

3. The format of the SRID for other than filter samples is a 14 or 15-character code:

```
ccttttttttxxxxT

  cc            CTBT station or laboratory number
  tttttttt      sample type identifier:
      00000000      blank filter identifier
      11111111      detector background measurement identifier
      77777777      special IMS sample identifier
      88888888      check source identifier
      99999999      calibration source identifier
  xxxx          a sequential number
  T             station type: G for noble gas, blank for particulate

Example: 35111111110006G
```

12.4 extSampleName

A freely settable character string, preferably unique.

12.5 idMeas

An auto-increment integer key managed by the database.

12.6 measId

The key `measId` is defined by STUK/ASL to be identical to `phdMeasName`, but the acquisition live time is appended to make the key unique:

```
ccccc_ddd-yyyy/mm/dd-hh:mm:ss_aaaaaa

  ccccc         station code
  ddd           detector code
  yyyy/mm/dd    acquisition start date
  hh:mm:ss      acquisition start time
  aaaaaa        acquisition live time

Example: HEP02_D01_2004/02/19_08:25:00.0_81871.5
```

12.7 phdMeasName

The key `phdMeasName` obeys the definition of MID by the CTBTO: the first nine characters are the station+detector code, the tenth character is a dash, and the remaining characters are the date and time of the acquisition start.

```
ccccc_ddd-yyyy/mm/dd-hh:mm:ss

  ccccc         station code
  ddd           detector code
  yyyy/mm/dd    acquisition start date
  hh:mm:ss      acquisition start time

Example: BRG11_001-2000/02/06-20:00:00
```

12.8 extMeasName

A freely settable character string, preferably unique.

Chapter 13

PHD-File Format Extension

There are two ways to construct keys and identifiers for *Linssi* in entry point 3 — spectrum analysis:

1. Write the keys to the `#Comment` block of a PHD-spectrum (see the *Linssi* core script `dbtophd`). A PHD-spectrum modified in this way conforms to the PHD-spectrum definition of the CTBTO [4]. The extensions are documented below.

2. Let the spectrum analysis script define the keys on the basis of the data in the PHD-spectrum blocks `#Header`, `#Collection`, and `#Acquisition` (see Sec. 11.2).

The first alternative is recommended and it shall be given priority by the analysis system, i.e., a key or identifier given in a `#Comment` block must not be changed by the analysis script. The following keys and identifiers in the `#Comment` block are currently supported:

a. `sampleId`

b. `measId`

c. `extSampleName`

d. `extMeasName`

e. `stationId`

f. `samplerId`

g. `sourceId`

h. `detectorId`

i. `measSetupId`

j. `blankIdMeas`

k. `backgroundIdMeas`

l. `blankIdAnalysis`

m. `backgroundIdAnalysis`

n. `inputIdAnalysis`

o. `inputIdCal`

p. `calibrations.class`

q. `measurementSetups.idCal`

r. `combined`

s. `sampleType`

t. `sourceDensity` [a]

u. `sourceThickness` [a]

For the definitions of the keys see the *Linssi* manual Part I [1]. It is important to note that a PHD-spectrum that contains any of the integer database keys **j.**–**o.** is tied to one *Linssi* implementation, i.e., the integer keys cannot be exported to another *Linssi* implementation.

The syntax for defining each of these keys/identifiers in the `#Comment` block is:

1. one key/identifier per line
2. the key/identifier name prefixed by "`Linssi:`" and no space
3. the key/identifier name followed by a space
4. the value for the key/identifier

Example: `Linssi:calibrations.class SETUP`

13.1 Example of an Extended PHD-File

Below we give an example of a PHD-file that contains *Linssi* extensions in the `#Comment` block. The extensions are framed for clarity.

```
BEGIN RMS2.0
MSG_TYPE DATA
MSG_ID 00003662
DATA_TYPE SAMPLEPHD
#Header
HEP02 FI001-D01 P A9                FULL
HEP02CI01P_2004/12/30_08:02:00_11
HEP02_FI001-D01_2005/01/01_08:12:00.0             -FI001-D01-1999/07/02-08:24:00.0

2005/01/02 08:01:52.0
#Comment
Other comments

Linssi:sampleId HEP02CI01P_2004/12/30_08:02:00_11_203
Linssi:samplerId CI01
Linssi:stationId HEP02
Linssi:measId HEP02_FI001-D01_2005/01/01_08:12:00.0_79777

Still more comments
#Collection
2004/12/30 08:02:00.0 2004/12/31 08:01:00.0      13471
#Sample
 9.80  0.15  9.00
#Acquisition
2005/01/01 08:12:00.0          85777          79777
#Energy
46.539001        137.893585       0.100000
77.108002        230.136398       0.100000
87.180000        260.206879       0.100000
238.632004       714.635803       0.100000
```

[a] The source parameters are included on the list, because they may be used by SHAMAN for self-absorption correction. For this purpose, they must be accompanied by a setting of `calibrations.class` to `SOURCE`.

```
477.612000       1431.399048      0.100000
583.190979       1748.041992      0.100000
727.330017       2180.433105      0.100000
860.564026       2580.149902      0.100000
1093.900024      3280.095215      0.100000
1460.800049      4380.392578      0.100000
1764.494019      5291.525391      0.100000
2103.532959      6308.622070      0.100000
2614.532959      7841.349121      0.100000
#Resolution
185.873032       1.687073         0.162545
238.677277       1.599915         0.019327
277.402283       1.207947         0.196168
351.989990       2.054047         0.317100
477.615326       1.797760         0.026302
583.177673       1.962036         0.021768
609.345398       1.498779         0.159257
661.534546       1.668091         0.179939
727.351318       2.055054         0.054498
785.438232       1.761597         0.095406
860.608032       2.005737         0.091816
911.139832       2.023499         0.092679
968.791321       2.143982         0.184401
1093.880127      1.696411         0.274368
1120.146240      2.477661         0.159326
1460.686768      2.610596         0.389208
1620.982910      1.852783         0.491402
1764.506104      2.545166         0.358929
2103.510010      3.485352         0.099305
2614.471924      3.136230         0.063993
#Efficiency
       5.000E+01        2.450E-02        1.225E-03
       8.000E+01        9.350E-02        4.667E-03
       9.000E+01        1.152E-01        5.833E-03
       1.000E+02        1.325E-01        6.667E-03
       1.100E+02        1.455E-01        7.333E-03
       1.350E+02        1.645E-01        8.167E-03
       1.500E+02        1.673E-01        8.333E-03
       2.000E+02        1.597E-01        8.000E-03
       3.000E+02        1.272E-01        6.333E-03
       5.000E+02        8.833E-02        4.500E-03
       7.000E+02        6.983E-02        3.500E-03
       1.200E+03        4.783E-02        2.333E-03
       2.000E+03        3.333E-02        1.667E-03
       3.000E+03        2.517E-02        1.333E-03
       3.600E+03        2.200E-02        1.167E-03
#Spectrum
 8192 2700
    0          0          0          0          0          0
    5          0          0          0          0          0
   10          0          0          0          0          0
   15          0          0          0          0          0
   20          0          0          0          0          0
   25          0          0          0          0          0
   30          0          0          0          0          0
   35          0          0          0          0          0
   40          0          0          0          0          0
   45          0          0          0          0       2658
   50       2298       1998       1758       1572       1467
...
 8140          5          6          1          2          5
 8145          4          1          2          3          1
 8150          4          5          2          4          3
 8155          2          4          3          4          1
 8160          1          4          4          2          3
 8165          5          5          3          1          4
 8170          2          6          5          4          2
 8175          4          8          3          2          3
 8180          4          6          6          7          3
 8185          1          2          3          2          3
 8190          1          0          0          0          0
STOP
```

Chapter 14

AKu File Format for Database Import

Analysis results can be stored to a *Linssi* database with the script `analysistodb` that is a basic building block for entry point three in the *Linssi* package. It understands text files in a simple blocked format called AKu after its developer Antero Kuusi of FINDC. This format is also used in other `*todb` scripts of the *Linssi* package as documented earlier in this manual.

The AKu-format is obeyed by the temporary database reports `.uda`, `.udb` and `.sdb`, generated by UniSampo and Shaman using their report generating language (RGL). Its use is also recommended for other analysis software packages. The AKu file format has the following principles:

1. Each block corresponds to a database table. A block starts with a line containing the table name preceded by a hash sign, e.g., `#samples`. The block ends with an empty line or a line including the next table name.

2. Blocks with names that do not correspond to *Linssi* table names are neglected, but a warning is issued by the `*todb` scripts by default. This feature can be used for commenting or for applying the analysis result files for other purposes.

3. The fields of the table are written in their correct order, separated either with a newline or a space. The former separation method is used in tables with a single entry for each field (e.g., `analyses`) and referred to as **AKu format Type 1**. The latter is used in tables with several entries for each field (e.g., `peaks`) and referred to as **AKu format Type 2**.

4. Fields containing simple data types (integer, double, char, etc.) are presented as such. Floating point numbers are output with the format `%-22.17g` that guarantees a sufficient number of significant digits for double precision numbers.

5. String fields (varchar, text) are within double quotes if they contain space, otherwise without quotes.

6. The fields of type text are mostly within XML-type separators (e.g., `<inputParam>...</inputParam>`).

7. The (long)blob fields are also within XML-type separators, using the most generic ASCII text format for the data in the block. Usually this means one number per row, e.g., the longblob field `spectrum` in table `measurements`.

8. The sign for a missing field is an ampersand `&`. These fields are given the value `null` in the database insertion phase.

This blocked format is understood by the `analysistodb` script that takes the database reports from an analysis software package as input and feeds their contents to the database after checking that database interrelations work correctly, e.g., database keys are unique. In the USS-package, the `analysistodb` script is used through `filetodb` script in the *Linssi* package. The script `filetodb`, in turn, is used by the `ustodb` and `shtodb` scripts that are delivered with the USS-package. The `shtodb` script is actually a symbolic link to the `filetodb` script, whereas `ustodb` performs some string manipulation tasks before calling the `filetodb` script (see p. 57). Basically, data from UniSampo and Shaman are fed to the database identically.

Any other analysis software package can utilize the database insertion scripts `analysistodb` in a similar manner, provided that it supports tailorable reports and the essential database keys like the USS-package does.

14.1 Example Report in AKu Format

An example of a database report produced by UniSampo and processed by the `ustodb` script is presented below. Some repetitive parts have been deleted and replaced with "`...`" and long lines have been split to fit to the page. A split line is indicated with backslash.

```
#airFilterSamples
&
CI01
HEP02
20041230080200
20041231080100
86340
86340
13471
13471
&
&
&
&
&
20041231080100
&
<comments>
</comments>

#samples
&
HEP02CI01P_2004/12/30_08:02:00_11_203
HEP02CI01P_2004/12/30_08:02:00_11
&
11
&
0
airfilter
airFilterSamples
&
<barcode>
HEP02CI01P_2004/12/30_08:02:00_11
</barcode>
&
<sampleConditionArrival>
</sampleConditionArrival>
<packConditionArrival>
</packConditionArrival>
<sealConditionArrival>
```

```
</sealConditionArrival>
&
&
&
&
<comments>
</comments>

#measurementSetups
HEP02FI001-D01
FI001-D01
&
&
&
&
&
&
&
&
&
&
&
&
&
&
&
<comments>
</comments>

#measurements
&
&
&
HEP02FI001-D01
&
&
HEP02_FI001-D01_2005/01/01_08:12:00.0_79777
HEP02_FI001-D01_2005/01/01_08:12:00.0
&
&
&
&
&
&
&
87060
20050101081200
20050102080137
85777
79777
FULL
1
1
8192
8192
<spectrum>
0
0
0
0
0
...
3
2
3
1
0
</spectrum>
20050102080152
<comments>
</comments>

#calibrations
2
```

```
energy
HEP02FI001-D01
&
1
SOH
&
20050101081200
<calInfo>
</calInfo>
<calCertificate>
</calCertificate>
2
<functionDef>
</functionDef>
0
8192
1.5689100325108E-01
3.3334475755692E-01
&
&
&
&
&
&
&
&
4.0824156254530E-02
1.1395708497730E-05
&
&
&
&
&
&
&
&
<comments>
</comments>

#calPoints
1     2 energy              & & 138.52696228027344      46.53900146484375       0.10000000149011612     &
2     2 energy              & & 231.19894409179688      77.107002258300781      0.10000000149011612     &
3     2 energy              & & 261.09396362304688      87.3489990234375        0.10000000149011612     &
4     2 energy              & & 715.6475830078125       238.63200378417969      0.10000000149011612     &
5     2 energy              & & 1055.520263671875       351.9320068359375       0.10000000149011612     &
6     2 energy              & & 1432.475830078125       477.59500122070312      0.10000000149011612     &
7     2 energy              & & 1749.1329345703125      583.19097900390625      0.10000000149011612     &
8     2 energy              & & 2181.37255859375        727.33001708984375      0.10000000149011612     &
9     2 energy              & & 2581.535888671875       860.56402587890625      0.10000000149011612     &
10    2 energy              & & 4381.84814453125        1460.822021484375       0.10000000149011612     &
11    2 energy              & & 5292.2724609375         1764.4940185546875      0.10000000149011612     &
12    2 energy              & & 6309.96533203125        2103.532958984375       0.10000000149011612     &
13    2 energy              & & 7842.982421875          2614.532958984375       0.10000000149011612     &

...

#calPreferences
2
&
1

#analyses
&
&
&
&
&
&
&
&
&
<inputParam>
Library lookup tolerance: 0.60 keV.
</inputParam>
```

```
<interactiveLog>
</interactiveLog>
shaman_run/b
UniSampo
"UniSAMPO 2.24 (22 December 2004) "
jarmo
<baseline>
0
0
0
0
0
...
3
2
3
1
0
</baseline>
<strippedSpectrum>
0
0
0
0
0
...
3
2
3
1
0
</strippedSpectrum>
<peakSearchSignificance>
0
0
0
0
0
...
0
0
0
0
0
</peakSearchSignificance>
&
&
&
&
<refConstants>
</refConstants>
<baselineMethod>
</baselineMethod>
<peaksMethod>
The peak analysis software uses Mariscotti's generalized second differences
method for peak search. After initial search, peak candidates are fitted and
tested for their significance against Currie's decision limit and candidates
below the limit are discarded from the peak list.
Peak areas are determined by fitting a Gaussian function with exponential
lower and upper tails to spectrum data, with peak parameters obtained from
peak shape calibration (fixed-width fitting).
</peaksMethod>
<nuclideMethod>
Nuclide identification is based on LSQ solution on candidate matrix.
UniSampo used a library of 150 nuclides and 726 gamma-ray and X-ray lines.
Its methods and parameters are presented in a comprehensive manual.
</nuclideMethod>
<uncCalcMethod>
</uncCalcMethod>
<lcMethod>
</lcMethod>
0.050000011920928955
0.050000011920928955
98
```

```
8177
2.4000000953674316
41
417644
<comments>
</comments>

#peaks
1     & & & 138.27375793457031       0.26241952180862427      46.249721527099609       0.10376090556383133                  \
  242.77622985839844       40.667675018310547       55.600063323974609       1.7290549278259277       4.0716133117675781     \
  22.485544204711914       3.5115795135498047       3.5115795135498047       &                        &                      \
  0                        0.0030431856866925955    0.00050976692000404      0.017300082370638847     0.0043250205926597118 \
  3.3276784420013428       1.428852915763855        & 83.602760314941406       169.90988159179688       0      "M Q0Q1C0"     \
  117   160   &                        &     1310.4656982421875       188.42481994628906       13.726792335510254           \
  117   160   189.4771728515625        -0.63915663957595825     0                        &                                    \
  0.17590585350990295      0.030750799924135208     &                        &                        &            &        \
  &                        &            &
2     & & & 189.16575622558594       0.22575381398200989      63.214305877685547       0.09333985298871994                  \
  408.0308837890625        47.844329833984375       91.94451904296875        1.7565633058547974       4.136390209197998      \
  4.2087516784667969       3.5393800735473633       3.5393800735473633       &                        &                      \
  0                        0.0051146429032087326    0.00059972587041556835   0.05511065199971199      0.013756892643868923  \
  6.4090538024902344       2.322054386138916        & 86.507720947265625       175.71978759765625       0      "M Q0Q1C0"     \
  164   199   &                        &     1407.3572998046875       199.18736267089844       14.113375663757324           \
  164   199   200.29981994628906       0.86513090133666992      0                        &                                    \
  0.092806793749332428     0.011828898452222347     &                        &                        &            &        \
  &                        &            &
...

#activities
Be-7       & & & 0.99898308515548706      4604256                  s &                        & 14.594400405883789          \
  0.75564807653427124      1 &                        &                        & &                        &                  \
  &                        &                        1.0064705610275269       &                        &                  \
  &                        1.0131926536560059       &                        77.43572998046875        &                  \
  1.0065131187438965       &                        &                        &                        &
K-40       & & & 0.99999153614044189      39761724894609408        s &                        & 15.657556533813477          \
  0.80023986101150513      1 &                        &                        & &                        &                  \
  &                        &                        1                        &                        &                  \
  &                        1                        &                        664397086720             &                  \
  1                        &                        &                        &                        &
...

#activityLimits
Np-239     & & & 105.99699401855469       62.008010864257812       63.720748901367188       1366.6767578125              \
  36.968589782714844       &                        &                        &                        &                  \
  0.35588937997817993      0 &
Ce-144     & & & 133.43045043945312       73.3912353515625         56.363945007324219       1360.5181884765625           \
  36.885204315185547       &                        &                        0.13667310774326324      1.0183001904806588e-05\
  0.42216187715530396      0 &
...

#calibrations
1
energy
HEP02FI001-D01
&
0
PHD
&
20050101081200
<calInfo>
</calInfo>
<calCertificate>
</calCertificate>
1
<functionDef>
</functionDef>
0
8192
0.0000000000000E+00
&
&
&
&
&
```

```
&
&
&
&
0.0000000000000E+00
&
&
&
&
&
&
&
&
&
<comments>
</comments>

#calPoints
1     1 energy              & & 138.89358520507812      46.53900146484375       0.10000000149011612     &
2     1 energy              & & 231.13639831542969      77.108001708984375      0.10000000149011612     &
3     1 energy              & & 261.20687866210938      87.180000305175781      0.10000000149011612     &
4     1 energy              & & 715.63580322265625      238.63200378417969      0.10000000149011612     &
5     1 energy              & & 1432.3990478515625      477.61199951171875      0.10000000149011612     &
6     1 energy              & & 1749.0419921875         583.19097900390625      0.10000000149011612     &
7     1 energy              & & 2181.43310546875        727.33001708984375      0.10000000149011612     &
8     1 energy              & & 2581.14990234375        860.56402587890625      0.10000000149011612     &
9     1 energy              & & 3281.09521484375        1093.9000244140625      0.10000000149011612     &
10    1 energy              & & 4381.392578125          1460.800048828125       0.10000000149011612     &
11    1 energy              & & 5292.525390625          1764.4940185546875      0.10000000149011612     &
12    1 energy              & & 6309.6220703125         2103.532958984375       0.10000000149011612     &
13    1 energy              & & 7842.34912109375        2614.532958984375       0.10000000149011612     &

...

#calPreferences
1
&
0
```

Chapter 15

Administrative Issues

During the testing phase of *Linssi*, the following important lessons were learnt:

1. The organization utilizing *Linssi* shall define its procedures prior to introducing *Linssi* as an operational database. The design goal of *Linssi* was to keep it sufficiently flexible for different users, but it is likely that old procedures need to be adjusted for utilizing *Linssi* in the most efficient way.

2. There should be at least two databases running at the same time: an operational database for strict business and a test database for making all kinds of experiments, testing new queries etc. This secures the integrity of the operational database and still enables further development of the system.

3. Running a database requires an administrator. His/her major tasks are to constantly monitor the functionality and integrity of the database and to maintain and develop efficient database scripts for the basic users. An experienced administrator could write scripts that alarm him/her automatically by e-mail or text message if anything out of the ordinary is happening in the database.

4. Basic database users should not make any complicated SQL queries, since they can mess up the database or at least block its usage. Model scripts and queries should be made available by the administrator and collected to a place available to all users. Database access through well-designed web forms would be preferable.

5. The ability of an SQL database for parallel processing is very limited. If you make a large query, it is likely to block the database from all other queries, especially those inserting new data to the database. Housekeeping of the database should be scheduled to out-of-office hours.

6. If you run so large a query in MySQL that the results do not fit into memory, MySQL will start writing them under its temporary data directory on the `/var`-disk. Therefore, the hard disk of the database server should be partitioned cleverly. It is possible that even making the `/var`-disk a separate partition is not sufficient, as it is also used by many other processes than MySQL. If MySQL fills the disk, probably nothing will work.

7. If the design of a query is not optimal, it may start stealing computing resources (see point 6). In MySQL, a query can be exited by pressing `Ctrl-C`, but it will keep running in the background until explicitly killed. The kill procedure is the following:

a. find out the process `idNumber` inside MySQL: `show processlist;`

b. kill the process: `kill idNumber;`

8. MySQL has a very informative operational manual available on the web. All database users should at least have a glance at the manual prior to using the database. The manual is likely to be needed in any troubleshooting situation. A printed version may also be useful sometimes.

9. If there are 16k spectra to be analyzed, the default limits of MySQL are exceeded by the database reports. To allow processing of these large reports, the MySQL daemon shall be started with an option increasing the maximum packet size:

```
% mysqld --max_allowed_packet=10M
```

10. The `analyses` table is by far the largest one due to the three longblob's (`baseline`, `strippedSpectrum`, `peakSearchSignificance`) stored in the table. If large amounts (tens of analyses per day) of data are stored to *Linssi*, the size may grow larger than the maximum table size allowed by default by MySQL, more exactly its default table format MyISAM. There are two alternative solutions:

    a. Increase the maximum MyISAM table size by MySQL command line arguments. One analysis takes an average of 400 kB of space in the `analyses` table, and by default, the maximum size of a MyISAM table is 4 GB. Thus, the `analyses` table in MyISAM format accepts about 10,000 analyses with default settings.

    b. Convert the `analyses` table to InnoDB format that has no size limitations (`http://dev.mysql.com/doc/mysql/en/innodb.html`). This can be done right after creating the *Linssi* database or when the database has been operating any period of time. Detailed instructions are available from the author upon request.

Bibliography

[1] P.A. Aarnio, *Linssi— SQL Database for Gamma-Ray Spectrometry.* Part I: Database, Version 1.1. Helsinki University of Technology Publications in Engineering Physics. A, TKK-F-A841, Espoo 2006.
http://linssi.hut.fi/radphys/linssi/linssi.pdf

[2] UniSampo — *Advanced Gamma Spectrum Analysis Software.* User's Guide, Version 2.4. Doletum Oy, Ltd., Helsinki, August 2006.

[3] Shaman — *Expert System for Radionuclide Identification, Version 1.13.* User's Guide Version 1.8.1. Baryon Oy, Ltd., Espoo, June 2005.

[4] Preparatory Commission for the Comprehensive Nuclear-Test-Ban Treaty Organization, *IDC Documentation: Formats and Protocols for Messages.* IDC-3.4.1Rev6, Vienna 2004.

[5] Open Database Connectivity (ODBC) Portal.
http://www.sqlsummit.com/ODBCPORT.HTM

Appendix A

Installation Instructions for *Linssi*

A.1 Database

A *Linssi* database can be implemented with any SQL engine (MySQL, PostgreSQL, Oracle, etc.). However, scripts written in Perl and PHP may be engine-dependent. We have chosen MySQL as the database engine and present installation instructions for *Linssi* under MySQL.

MySQL database is a commercial package that is included in many Linux distributions, including RedHat, SuSE and Mandrake. It is free for non-commercial use as can be seen from the license on the MySQL web page http://www.mysql.com. The newest versions of the database can be found on these pages, as well as a comprehensive online manual.

Linssi v.1.1 requires MySQL version 4.0 or newer. The scripts for *Linssi* take advantage of Perl and PHP. For example, the following packages were installed under Mandrake 10.1 from its distribution CD's for *Linssi* testing purposes:

```
MySQL-common-4.0.20-3.1.101mdk
MySQL-4.0.20-3.1.101mdk
MySQL-client-4.0.20-3.1.101mdk
libmysql12-4.0.20-3.1.101mdk

perl-base-5.8.5-3.1.101mdk
perl-Mysql-1.22_19-9mdk
perl-CGI-3.05-1mdk
perl-DBI-1.43-2mdk
zlib1-1.2.1.1-3mdk

php-mysql-4.3.8-1mdk
php-ini-4.3.8-1mdk
php-gd-4.3.8-2mdk
php-cgi-4.3.8-3.2.101mdk
libphp_common432-4.3.8-3.2.101mdk
```

This list is only meant for illustration, not to recommend Mandrake 10.1 that is actually an obsolete, unsupported version since March 2006. The *Linssi* v.1.1 scripts have been in production use under CentOS4 and SuSE 9.3.

The package names vary across different Linux distributions and in some cases they may need to be downloaded as compressed TAR-files. The compressed TAR-files shall be unpacked,

compiled and installed according to the instructions that follow in the package. The RPM-files, on the other hand, can be installed in a simple manner as root:

```
% rpm -i name.rpm
```

Additionally, the system is to be set up using a suitable Linux system configuration tool to start MySQL automatically at boot time.

When everything is installed, log in to the MySQL database to change the MySQL root password (Note: different from the root password of the Linux system):

```
% mysql -u root
```

Change the password to `new_password` (or something else):

```
mysql> use mysql;
mysql> update user set password=password('new_password') where user='root';
mysql> flush privileges;
```

Now is time to generate the *Linssi* database and some users for the database. In the example below, the user `webbi` may only view the results and the user `xunil` can do nearly anything but delete the database. The former is to be used by database viewing scripts and the latter by scripts that modify the contents of the database.

```
mysql> create database linssi;
mysql> grant select on linssi.* to webbi@localhost;
mysql> grant create,lock tables,insert,select,update on linssi.* to \
       xunil@localhost identified by 'some_password';
```

The *Linssi* database was created, but it does not contain any tables yet. A perl script called `maketables` is provided in the *Linssi* package to install the basic tables for result storage. If the *Linssi* tables related to CTBT-laboratory samples are to be created, `maketables` should be invoked with option `-c`. Its success can be checked with the script `desctables` that also belongs to the *Linssi* package.

```
% maketables -d linssi -u xunil [-c] -p some_password
% desctables -d linssi -u webbi | less
```

These and other database scripts should reside under directory `/usr/local/gamma/linssi` or a similar directory defined in each user's *Linssi* configuration file `.linssirc` (see Sec. 2.2). The most important script is `analysistodb` that imports analysis results to *Linssi*. When using UniSampo and Shaman with *Linssi*, there should be a symbolic link named `shtodb` pointing to `filetodb` (that calls `analysistodb`) and another link called `ustodb` pointing to `../shaman/bin/ustodb` that calls `filetodb` after making some manipulations in UniSampo's result files.[a]

To log in the database as the user `webbi`, a password is not required:

```
% mysql linssi -u webbi
```

[a] Other necessary configuration for connecting UniSampo and Shaman with *Linssi* is documented in the USS installation instructions.

User `xunil` needs password, so the system requires that the password is given either on the command line (not recommended), at a separate prompt, or in the *Linssi* configuration file `.linssirc` in the home directory (see Sec. 2.2). Assuming the second alternative:

```
% mysql linssi -u xunil -p
```

Only MySQL root user may delete the whole database. This is done with command `drop` inside MySQL database. The following example deletes the whole database and its contents:

```
% mysql -u root -p
mysql> drop database linssi;
```

When some data has been imported to the database, its contents can be browsed on MySQL command line using the `select`-command or using the database browsing scripts provided in the *Linssi* package. These scripts are presented elsewhere in this document.

A.2 Installation of the *Linssi* PHP Scripts

In order to use PHP scripts and web features there should be a webserver, like Apache, installed and running. These instructions are written for Apache, but any other webserver can be used with only minor modifications.

Installation of the main *Linssi* PHP scripts is very simple. Extract the script package into a directory within the webserver's document root and edit `linssiConfig.php` configuration file (see Sec. 2.2) to set available databases, paths to external libraries and other configuration directives. Directory where files are extracted is here expected to be `/var/www/html/`, but it may vary depending on the system used.

Instructions on setting the database access credentials and installing the required (and optional) external packages are given in the following sections. Every command in this section should be given as the Unix root user.

A.2.1 Database Access Credentials

A good method and the one recommended by the PHP security consortium (`http://phpsec.org/projects/guide/`) to protect database access credentials is to store them into Apache's environment variables. Create a file that only root can read with the following lines:

```
SetEnv read_username "webbi"
SetEnv read_password ""
SetEnv write_username "xunil"
SetEnv write_password "some_password"
```

Usernames and passwords should be equivalent to those used when creating the database. Since there was no password for user `webbi`, the value of `read_password` is also left empty. We have chosen to save this file as `dbConfig` into directory `/var/includes/`. Filename and location can be chosen freely, but for security reasons, it should not be saved in webserver document root among the PHP scripts.

Include this file within Apache's configuration file `httpd.conf` by adding the line:

```
Include  /var/includes/dbConfig
```

In some Linux distributions the `httpd.conf` file is located in directory `/etc/httpd/conf`. The filename can also be something else, like `httpd2.conf`.

Restart Apache. Now the access credentials can be found from the superglobal `$_SERVER[]` array. Just be careful not to expose these variables with something like `phpinfo()` or `print_r($_SERVER)`.

By default *Linssi* PHP scripts expect that the database access credentials are stored in Apache's environment variables as decribed above, but the credentials can also be set directly in `linssiConfig.php` (see Sec. 2.2) by replacing the `$_SERVER["read_username"]` etc. variables with the actual usernames and passwords. This is not recommended on public servers.

A.2.2 Installing JpGraph

The graphics in these scripts requires a PHP package named `JpGraph` that is available from http://www.aditus.nu. Extract the `JpGraph` package into any directory readable by PHP. By default, the path `/var/www/html/php/jpgraph/src` is defined in the `linssiConfig.php` file. When using some other installation directory, edit `linssiConfig.php` configuration file to set the `JpGraph` path variable pointing to the `src`-directory of your `JpGraph` installation.

A.2.3 Getting Analysis Reports in PDF Format

A modified version of the `html2fpdf` library is required to get the analysis reports in PDF format. The original library is available from http://html2fpdf.sourceforge.net, but T. Salonen has modified it in order to be able to use png-figures generated dynamically by PHP in *Linssi* reports. The modified library is distributed with *Linssi* scripts and it should be used in this context.

Installation of the `html2fpdf` is similar to the installation of `JpGraph`. Extract the package into any directory readable by PHP and set the installation path in the `linssiConfig.php` configuration file. By default, the directory `/var/www/html/php/pdf` is assumed.

A.2.4 Getting Sample Reports in XLS Format

The `Pear::OLE` and `Pear::Spreadsheet_Excel_Writer` packages are required to get sample reports in XLS format (MS-Excel spreadsheet). `Pear` is a PHP extension and application repository and it is usually installed by default with all recent PHP distributions.

Installation of the required `Pear` packages:

```
% pear install OLE
% pear install Spreadsheet_Excel_Writer
```

If there are only beta versions of these packages available, the installation has to be performed with the `-d` option:

```
% pear -d preferred_state='beta' install OLE
% pear -d preferred_state='beta' install Spreadsheet_Excel_Writer
```

The `Pear` packages are usually installed in the directory `/usr/share/pear`, which is also the default path in `linssiConfig.php`.

A.2.5 HTTP Authentication

Updates to the database are controlled with HTTP authentication when done through a web interface. For this to work, a password file must be created with Apache's `htpasswd` program in the following way:

```
% htpasswd [-c] /var/includes/.htpasswd some_username
```

`htpasswd` will prompt you for the password. This creates a new password file and adds the user `some_username` to the file. The `-c` flag is used only when you are creating a new file. After the first time, you shall omit the `-c` flag. If using some other directory or filename, edit `linssiConfig.php` to define the path to the created password file.

A.3 Installation of the CTBT Laboratory Scripts

The CTBT laboratory scripts are meant for CTBT laboratories. They handle the complicated message traffic between the CTBTO and the laboratory in addition to basic sample, measurement and analysis functions. They are most probably useful for CTBT laboratories only.

The CTBT laboratory sample tables and CTBT scripts have been designed and implemented at STUK that runs the CTBT laboratory FIL07. Please contact `tommi.salonen@stuk.fi` or `mikael.moring@stuk.fi` for closer details.

A.3.1 Basic Installation and Configuration

CTBT laboratory scripts shall be copied to their own directory somewhere under the webserver's document root, typically `/var/www/html/ctbt/`. After this the address to the user interface is `http://SERVER/ctbt/index.html` or just `http://SERVER/ctbt/`.

CTBT scripts read messages from and write messages and spectrum files to the directories that are defined in `ctbtConfig.php`. This file contains also some other configurations. Some of them affect the handling of mail (e.g., the directory where invalid mail is moved to), some of them are site specific (e.g., site code). These definitions are commented in the file itself.

There is also another configuration file, `ctbtWebConfig.php`. In this file, the user can set available databases etc. These definitions are similar to `linssiConfig.php` that is used for configuring the basic Linssi PHP scripts.

A.3.2 Storing Mails to the Database

New mail is stored to a *Linssi* database with the script `ctbtCheckMail.php`. This script is meant to be executed from the command line. In this way the directory from where the new mail is checked can be set readable only by the user (not webserver or others). The `ctbtCheckMail.php` script reads database access credentials and database name from the user's `.linssirc` file.

Sender of the mail is authenticated with `openssl` if the authentication directory defined in `ctbtConfig.php` exists.

A.3.3 Automated Mail Processing

At STUK the handling of mail is automated with the mail processing software `procmail` and executing the `ctbtCheckMail.php` as a cronjob. `Procmail` is instructed to move all mail messages from a certain sender to the directory defined in `ctbtConfig.php`. After this `ctbtCheckMail.php` stores the mail message, if valid, automatically to a *Linssi* database.

Sending of mail created at the CTBT laboratory is also automated. Additionally, null PHD-files are automatically copied with `scp` to a remote server where a new measurement is started. The command that sends the mail is another cronjob. If these operations are not automated, the created mail and null PHD-files are just saved in the corresponding directories.

ISBN 951-22-8185-6
ISSN 1456-3320