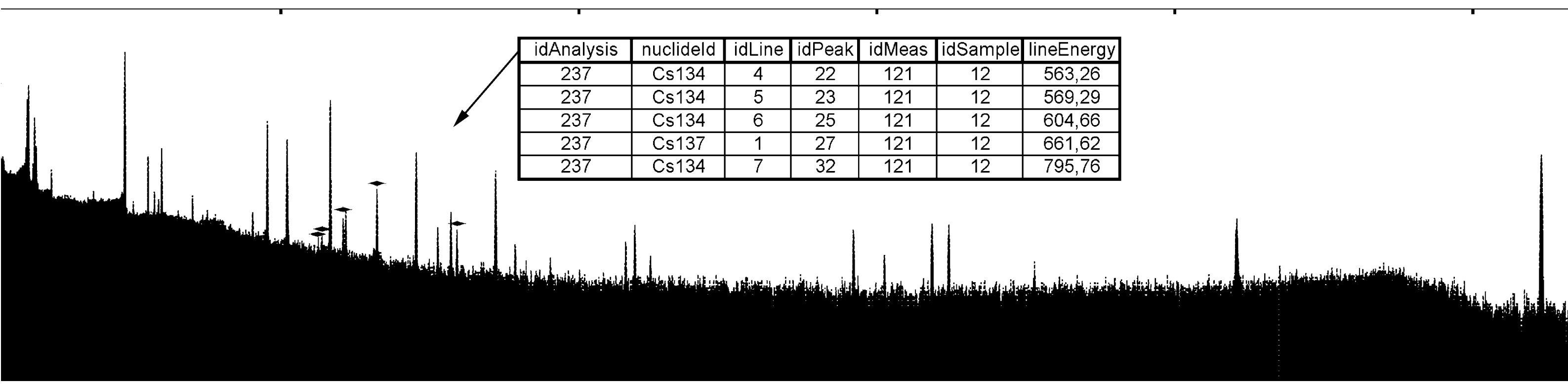
Helsinki University of Technology Publications in Engineering Physics. A
Teknillisen korkeakoulun teknillisen fysiikan julkaisuja. A
Espoo 2011 TKK-F-A862

LINSSI
SQL DATABASE FOR GAMMA-RAY SPECTROMETRY
PART II: SCRIPTS AND INTERFACES
Version 2.3

Pertti Aarnio, Jarmo Ala-Heikkilä, Ian Hoffman, Tarja Ilander, Seppo Klemola, Aleksi Mattila, Antero Kuusi, Mikael Moring, Mika Nikkinen, Andreas Pelikan, Samu Ristkari, Tommi Salonen, Teemu Siiskonen, Petri Smolander, Harri Toivonen, Kurt Ungar, Kaj Vesterbacka, Weihua Zhang



idAnalysis	nuclideId	idLine	idPeak	idMeas	idSample	lineEnergy
237	Cs134	4	22	121	12	563,26
237	Cs134	5	23	121	12	569,29
237	Cs134	6	25	121	12	604,66
237	Cs137	1	27	121	12	661,62
237	Cs134	7	32	121	12	795,76

Aalto University
School of Science

Helsinki University of Technology Publications in Engineering Physics. A
Teknillisen korkeakoulun teknillisen fysiikan julkaisuja. A
Espoo 2011 TKK-F-A862

LINSSI
SQL DATABASE FOR GAMMA-RAY SPECTROMETRY
PART II: SCRIPTS AND INTERFACES
Version 2.3

Pertti Aarnio, Jarmo Ala-Heikkilä, Ian Hoffman, Tarja Ilander, Seppo Klemola, Aleksi Mattila, Antero Kuusi, Mikael Moring, Mika Nikkinen, Andreas Pelikan, Samu Ristkari, Tommi Salonen, Teemu Siiskonen, Petri Smolander, Harri Toivonen, Kurt Ungar, Kaj Vesterbacka, Weihua Zhang

Aalto University School of Science
Department of Applied Physics
Fission and Radiation Physics Group

Database Design, Software and Manuals

Pertti Aarnio[1], Jarmo Ala-Heikkilä[1], Ian Hoffman[3], Tarja Ilander[2], Seppo Klemola[2], Aleksi Mattila[2], Antero Kuusi[2], Mikael Moring[2], Mika Nikkinen[2], Andreas Pelikan[4], Samu Ristkari[2], Tommi Salonen[2], Teemu Siiskonen[2], Petri Smolander[2], Harri Toivonen[2], Kurt Ungar[3], Kaj Vesterbacka[2], Weihua Zhang[3]

[1]Aalto University School of Science, Fission and Radiation Physics Group
[2]Finnish Radiation and Nuclear Safety Authority
[3]Health Canada, Radiation Protection Bureau
[4]Dienstleistungen in der automatischen Datenverarbeitung und Informationstechnik

Distribution:
Aalto University School of Science
Fission and Radiation Physics Group
P.O. Box 14100
FI-00076 Aalto
Finland

ISBN 978-952-60-3315-0 (printed)
ISBN 978-952-60-3616-8 (pdf)
ISSN 1456-3320

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

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. Knowledge of *Linssi* version 1.1 [2, 3] is helpful but not necessary.

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 radiation 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/`.

The UniSampo–Shaman (USS) analysis package is documented in comprehensive manuals [4, 5]. 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 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 (Comprehensive Nuclear-Test-Ban Treaty Organization) was interfaced with *Linssi* in December 2004. Later, STUK (Radiation and Nuclear Safety Authority of Finland) has interfaced many of their in-house analysis tools with *Linssi*. 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 operated at STUK, Health Canada, and other organizations.

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, Perl and PHP 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`

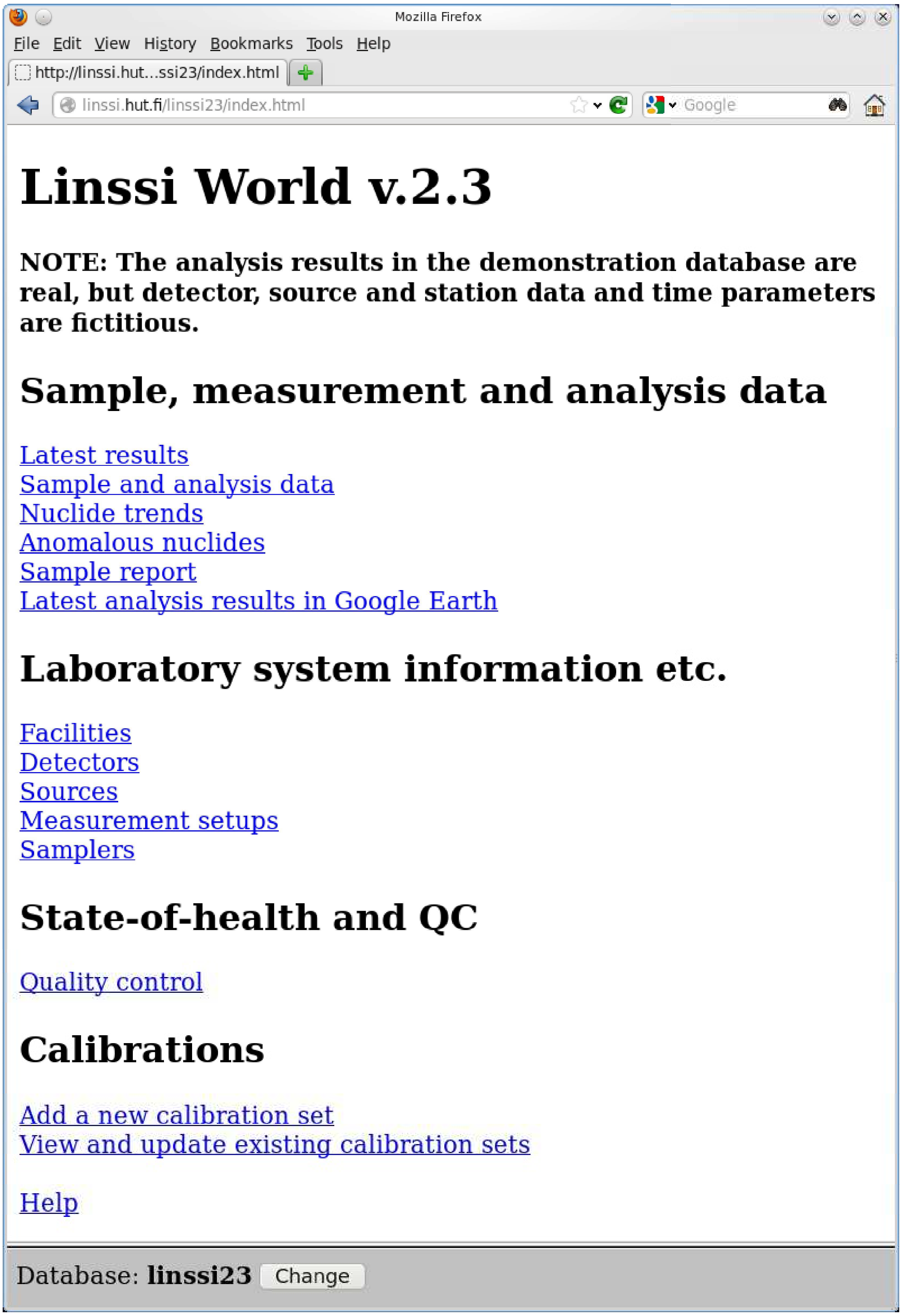
Experienced users may use *Linssi* through SQL-queries on the MySQL command prompt or through scripts written in Perl and SQL. For an ordinary user, the main *Linssi* interface is a *Linssi* home page known as `LinssiWorld` in a WWW browser, by default the one shown in Fig. 2.1. The exact contents of 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/`. 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. Only the most essential scripts are included. A comprehensive reference list of the basic *Linssi* scripts is presented in Ch. 7, useful when one is looking for a *Linssi* script for a specific purpose.

The files connected with the distribution package of *Linssi* scripts include the following `LinssiWorld` pages:



Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://linssi.hut...ssi23/index.html

linssi.hut.fi/linssi23/index.html

Linssi World v.2.3

NOTE: The analysis results in the demonstration database are real, but detector, source and station data and time parameters are fictitious.

Sample, measurement and analysis data

Latest results
Sample and analysis data
Nuclide trends
Anomalous nuclides
Sample report
Latest analysis results in Google Earth

Laboratory system information etc.

Facilities
Detectors
Sources
Measurement setups
Samplers

State-of-health and QC

Quality control

Calibrations

Add a new calibration set
View and update existing calibration sets

Help

Database: **linssi23** Change

Figure 2.1: The distributed version of `LinssiWorld` v.2.3.

linssi.html

Purpose:	An example of a `LinssiWorld` web page, calls linssi_23.html
Package:	*Linssi* core (v.2.3)
Created:	18.1.2012

linssi_23.html

Purpose:	An example of a `LinssiWorld` web page
Package:	*Linssi* core (v.2.3)
Created:	18.1.2012

linssi.css

Purpose:	Style sheet for `LinssiWorld`
Package:	*Linssi* core (v.2.3)
Created:	18.1.2012

help.html

Purpose:	Help page for `LinssiWorld`
Package:	*Linssi* core (v.2.3)
Created:	18.1.2012

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) [7]

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 is a Perl function package in the *Linssi* script distribution for interpreting the `.linssirc` files, named `linssiConfig.pm`. It is used by current Perl scripts and also recommended for any new scripts.[a]

2.3 Documentation of *Linssi* Scripts

The script descriptions in Ch. 3–7 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>`. The current default package is "*Linssi* core (v.2.3)".

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: 2.3.13 means the 13th released version of a script for *Linssi* v.2.3. **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

[a]Some scripts originally from *Linssi* v.1.1 may use the Perl function package `linssiGetoptStd.pm`. The package is available in the v.1.1 script release but not in v.2.3.

- `<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

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

Chapter 3

Database Management

The most basic database scripts are the management scripts to create the database, to list its table contents, and to delete data related with a given sample, measurement, or analysis. These scripts are presented in this chapter and their usage is illustrated in App. A.

In *Linssi* v.1.1 there are two maintenance scripts, checkdb and fixdb [3, Ch. 3]. These scripts have not been upgraded to v.2.3, since the default database engine has been changed to InnoDB, while the named scripts are needed for MyISAM only. If running *Linssi* v.2.3 on the MyISAM database engine, it is recommended to upgrade the maintenance scripts from v.1.1 to v.2.3.

Some scripts refer to ODBC that is clarified in Ref. [7].

3.1 maketables

Purpose:	Creates tables in *Linssi* database
Version:	2.3.1
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
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 *Linssi* 2.0 28.1.2009 AKu : Fixed ActionSites table definition

13.1.2010
Pertti Aarnio: Linssi2.2 table definitions
3.2.2010
til : linssiInfo-table
3.3.2010
x-ape : Updated version number to 2.2 in documentation
bug-fix DBI::errstr -> $DBI::errstr
2.7.2010 JAH : Documentation
11.3.2011
Samu Ristkari : mobileCoordinates PRIMARY KEY is now idPosition.
mobileCoordinates and measurements are connected via
relation measurementCoordinates(idPosition, idMeas).
14.4.2011 Samu Ristkari : missionId VARCHAR(20) => missionId VARCHAR(80)
22.8.2011 til :
SRS-table:
date=> dateId
Primary Key: facilityId=>facilityId, dateId
samples table:
sampleProductionTable lenght varchar(20)=>varchar(80)
20.12.2011 til
Updated version number to 2.3 in documentation
2.8.2012 JAH : Corrected version number in linssiInfo- table

Description: This script automatically creates tables for *Linssi* database. The table specifications are according to the database definition 2.3.
The script uses database name, username and password given as arguments. If database name, username, password and/or odbc-parameter 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]
[-i databaseId] [-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.

- databaseId writes databaseId into linssiInfo-table.
```

```
-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

3.2 fillfunctions

Purpose: Fills in function codes and names into functions table

Version: 2.3.0

Author: Jarmo Ala-Heikkila

Package: *Linssi* core (v.2.3)

Created: 5.6.2008

Updated: 11.9.2008 JAH
12.3.2010 JAH: Documentation
20.12.2011 til
Updated version number to 2.3 in documentation

Description: This script fills in function codes (idFunction) and names (functionNameId), as specified in *Linssi* v.2.3 manual, into functions table. This is necessary, because the idFunction field is referred to from many other tables (foreign key).
The Perl code before the first insert-command was copied verbatim from the maketables script by Antero Kuusi. The insert-commands could actually be integrated into maketables.

Syntax:

```
fillfunctions [-d database_name] [-u username] [-p password] [-g config-
File]
[-odbc|-noodbc]
```

3.3 desctables

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

Version: 2.3.0

Author: Jarmo Ala-Heikkila

Package: *Linssi* core (v.2.3)

Created: 5.6.2008

Updated: 12.3.2010 JAH: Documentation
20.12.2011 til
Updated version number to 2.3 in documentation

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 configuration file ~/.my.cnf shall not exist.
The script is portable across *Linssi* v.1.1 and v.2.3.

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

Reads tables: none
Writes tables: none
Modules: none

3.4 listtables

Purpose: List tables in a *Linssi* database instance
Version: 2.3.0
Author: Andreas Pelikan
Package: *Linssi* core (v.2.3)
Created: 29.6.2009
Updated: 06.04.2010 AP
added option –fields
2.7.2010 JAH: Documentation
20.12.2011 til
Updated version number to 2.3 in documentation

Description: Create a listing of tables in a *Linssi* database.
The listing is intended to make it easier to find out if the table structure of a particular database conforms to a certain *Linssi* version.
It provides similar functionality as desctables, but makes comparison easier. Pipe the output to a file and use 'diff' to see differences between two output files.
Two types of listings are supported: a full listing (fields), and a short listing (tables).
The full (field) listing lists all database fields in the format Table.Field:Type:Default:Null:Key:E
with the following elements: Table - the database table name Field - the field name Type - the field's data type Default - the default value Null - YES for fields that may be NULL, empty else Key - one of PRI(mary); MUL(ti), UNI(que), or empty Extra - extra info, such as auto_increment The option –fields can be used to change the order of the last 5 fields, or to include only some of them.
The short (table) listing contains lines of the format table fieldcount with the following elements: table - the database table name fieldcount - the number of fields in the table
Both reports contain a summary line: xx tables, yy fields

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

Syntax:
```
listtables [-d database_name] [-u username] [-p password] [-g configFile]
[- odbc|-noodbc] [-s] [-fields fieldList]
```

Options:
```
-u username Connect to database as user 'username'.  Default is read from
[read] section of configFile.
-p password Password to be used in database connection.  Default is read
from
[read] section of configFile.

-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.

-s Create short output, listing only tables.

-fields list
In the long report, only include the fields given in the
comma separated list.  Default is Type,Default,Null,Key,Extra ;
The records always start with table and field name.
```

Modules: linssiConfig.pm v.1.1.1 or newer

3.5 deleteSample

Purpose:	Deletes a sample and all associated data from *Linssi*
Version:	2.3.0
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	21.7.2010 (for v.2.2)
Updated:	20.12.2011 til Updated version number to 2.3 in documentation

Description: Script for deleting a sample and all associated data from *Linssi* database v.2.3. 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, calibrationSamples, calibrationNuclides, airFilterSamples, basicSamples, ctbtLabSamples, airFilterSOH, sampleTransports, sampleTrackings, sources, measurements, spectra, alphaMeasurements, analyses, analysisBlocks, spectrumComponents, componentsUsed, peaks, lineAssociations, activities, activityLimits, nuclideRatios, rois, finalResults

Modules: linssiConfig.pm v.1.1.1 or newer

3.6 deleteMeas

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

Version: 2.3.0

Author: Antero Kuusi

Package: *Linssi* core (v.2.3)

Created: 21.7.2010 (for version 2.2)

Updated: 20.12.2011 til
Updated version number to 2.3 in documentation

Description: Script for deleting a measurement and all associated data from *Linssi* database v.2.3.
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, spectra, alphaMeasurements, analyses, analysisBlocks, spectrumComponents, componentsUsed, peaks, lineAssociations, activities, activityLimits, nuclideRatios, rois, finalResults

Modules: linssiConfig.pm v.1.1.1 or newer

3.7 deleteAnalysis

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

Version: 2.3.0

Author: Antero Kuusi

Package: *Linssi* core (v.2.3)

Created: 21.7.2010 (for version 2.2)

Updated: 20.12.2011 til
Updated version number to 2.3 in documentation

Description: Script for deleting an analysis and all associated data from *Linssi* database v.2.3.
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 analysisId [-d database_name] [-u username] [-p password]
[-g optionFile] [-odbc|-noodbc]
```

Writes tables: analyses, calsUsed, analysisBlocks, spectrumComponents, componentsUsed, peaks, lineAssociations, activities, activityLimits, nuclideRatios, rois, finalResults

Modules: linssiConfig.pm v.1.1.1 or newer

Chapter 4

Database Communication Using LML

A major upgrade in *Linssi* v.2.3, in contrast to v.1.1, is the introduction of an XML schema called *Linssi* Markup Language or LML for database communication. All data import into and export from *Linssi* is recommended to be done using LML, although some support exists for the AKu format used in v.1.1 (see [3, Ch. 14]) for backward compatibility.

According to the `readme.txt` file included in the script package, the XML-schema of LML is defined using the following files:

lml.xsd Master-level file, contains the definition of the root node. All other files contain only blocks used by this file, start reading here.

general.xsd Contains definitions for blocks that describe parameters that are typically non-changing, such as specification of detectors. These blocks go directly under the main LML-node. Also contains definitions for some simple fields that are used by all of the other files.

sample.xsd Contains characterisation sample or other object being measured. In field conditions this could also mean specification of the location being measured. These blocks go under the main LML-node.

measurement.xsd Contains the actual measurement result (spectra or dose rate) and information about the measurement (such as measurement time). These blocks go inside the sample block, meaning that at least the basic indentifying information described in sample.xsd must be filled to use these blocks.

analysis.xsd Contains the actual analysis result (for example found peaks, nuclide information) and information about analysis (such as parameters used and methods). These blocks go inside the measurement block, meaning that at least the basic information described in measurement.xsd must be filled to use these blocks.

calibration.xsd Contains calibrations. As calibrations can be associated with one of several situations (such as default calibration for a given measurement setup or calibration used with a given analysis), these blocks go directly under the main LML-node. Other parts of the file then refer to these calibrations by their ID.

The LML schema does not need to be documented in more detail, since it has a one-to-one correspondence with the database schema defined in *Linssi* manual Part I [1]. LML examples are presented in Ch. 11.

Using LML, the data entry points 0 . . . 3 defined in *Linssi* manual Part I [1, Ch. 1] need not to be separated on the script side. The scripts defined in this chapter can be used for all data import and export.

4.1 lmltodb

Purpose: Script for reading analysis and other data in LML format into *Linssi* database

Version: 2.3.1

Author: Antero Kuusi

Package: *Linssi* core (v.2.3)

Created: 4.9.2007

Updated: 30.10.2007
Updated to match table specification 2.0b5.9. Main difference is support for spectrumComponents, rois, and roiRatios inside both measurementData and analysisData.
27.11.2007
Updated to match table specification 2.0b5.10.
11.12.2007
Version 2.0.0.1: Added handling of analysisBlobs.
13.12.2007
Version 2.0.0.2: Fixed the handling of roi- subtables (roiLimits, roiComponents, and roiRatios).
29.2.2008
Version 2.0.0.3: Enchanced the handling of boolean fields, true/false now understood in addition to 1/0.
28.3.2008
Version 2.0.0.4: Fixed handiling of analysisBlocks when it appears more than once in a file.
1.4.2008
Version 2.0.0.5: Expanded error message information.
6.6.2008
Version 2.0.0.6: Removed potential double definition of *IdAnalysis-fields in Analysis-table.
6.6.2008
Version 2.0.0.7: Fixed behavior when user inputs an empty element. Script no longer inputs anything to calibration-related tables if user enters calId that already exists. Changed output of -i option to reflect this.
11.6.2008
Version 2.0.0.8: Added a check to guarantee that the completionTime in finalResults table will be written even if there is no finalResults block in the input file.
4.7.2008
Version 2.0.0.9: Improved the handling of calsRecommended blocks.
25.9.2008
Version 2.0.0.10: Removed idSample insertion into alphaMeasurements table (the table doesn't have such a field).
17.10.2008

Version 2.0.0.11: Corrected argument binding for updated values on duplicate key in table calsRecommended.
19.1.2009
Version 2.0.0.12: Added support for actionSites table.
23.1.2009
Version 2.0.0.13: Added support for SMACK (externalKeys table).
19.3.2009
Version 2.0.0.14: Added -a option.
26.6.2009 AP
Version 2.1.0.1: added options -h and –(no)keychecks; fixed bugs;
changed version to 2.1
26.11.2009 AKu
Version 2.1.0.2: Modified the behavior of -o option to work more realiably
(no more ”ghost entries”)
9.12.2009
Version 2.1.0.3: Added -q option
2.3.2010
Version 2.2.0.0: Updated to *Linssi* 2.2 tables specification.
2.7.2010 JAH: Documentation
20.12.2011 til
Updated version number to 2.3 in documentation
2.8.2012 JAH: Documentation

Description: Script for reading LML files into *Linssi* database. Data is read from stdin unless -f -option is used. The LML format is specifed in lml.xsd file and refers to components defined in sample.xsd, measurement.xsd, analysis.xsd, calibration.xsd and general.xsd.
The script assumes that the input file is well-formed and follows the LML schema. Well-formedness of input file may be checked during parsing using the -v -option. However this option does NOT check if the file follows the LML schema.
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:
```
lmltodb [-f reportfile] [-d database_name/DSN] [-u username] [-p password]
[- g configFile] [-i] [-c] [-o] [-q] [-r] [-a] [-s] [-v] [-h]
[-odbc|-noodbc] [-keychecks|-nokeychecks]
```

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.  The ids are report
in the tree travelsal order which is the order the entries appear
in the file practically every time.

-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.

-keychecks Tell the database engine to perform foreign key checks.  This
is useful to guarantee that the lml file contains all referenced
data, or the referenced data were loaded earlier.

-nokeychecks Tell the database engine not to perform foreign key checks.
This is the default behaviour.

-c If database contains data with the same sampleId, data in
samples table is compared with the data in file.
Fields that have NULL in file or database are ignored.  If some of
the compared fields do not match, the script is aborted with
error message.  Useful for checking consistency of data when it is
input piecemeal.  (NOTE: Not currently implemented).

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

-q Same as -o option, but for tables inside generalData block.

-r Same as -o -option, but for calsRecommended table.

-a Same as -o -option, but applies only for those tables that are
in analysisTables block except for analyses and calsUsed.

-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 the same
sampleId/measurementId for different samples/measurements.

-v Check well-formedness of the input during parsing.

-h Show a usage (help) message and exit.
```

Modules:	linssiConfig.pm v.1.1.1 or newer, DBI package, GetOpt package, XML::LibXML package

4.2 dbtolml

Purpose:	Generates LML file from *Linssi* database
Version:	2.3.1
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	25.9.2007 (for v.2.0)
Updated:	12.12.2007 Updated to match table specification 2.0b5.10 13.2.2008 Version 2.0.0.1. Fixed behavior of calsRecommended table. 28.3.2008 Version 2.0.0.2. Multiple analysisBlocks entries for one analysis now supported. 23.4.2008 Version 2.0.0.3. Corrected the name of the attribute in calsUsed block (was callndex, should be callId). 19.5.2008 AP Version 2.0.0.4 corrected database error variable name $DBI::errstr, suppressing empty alphaMeasurements record. 25.9.2008 Version 2.0.0.5: Corrected the sources and calsRecommended blocks to match the schema. Improved facility-table handling with the -l option. 23.1.2009 Version 2.0.0.6: Added support for SMACK (externalKeys table). Added -e and -n options. 19.2.2009 Version 2.0.0.7: -m -option now prints the applicable calibration entries (earlier only the -a -option did so). 16.4.2009 Version 2.0.0.8: -x -option now added, use with extreme caution. 1.7.2009 AP Version 2.1.0.9 relocated sources, documented actual version (2.1) linssiConfig field name changes. 2.3.2010 Version 2.2.0.0: Updated to *Linssi* 2.2 (no material changes). 2.7.2010 JAH: Documentation 20.12.2011 til

Updated version number to 2.3 in documentation
2.8.2012 JAH: Documentation

Description: Script for generating LML files from *Linssi* database. These files can be read into another database using lmltodb.

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

Syntax:
```
dbtolml [-d database_name] [-u username] [-p password] [-f filename]
[-dl label] [-l] [-g configFile] -s|-m|-mn|-a|-an|-af [-n|-e]
-is <sampleId>|- im <measId>|-ia <analysisId>|-x [-odbc|-noodbc]
```

Options:
```
-d databaseName Name of the database that contains the information.
If not given the name in the configuration file will be
used.

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

-p password Password for the user.  If not given the 'read' password in
configuration file will be used.

-f filename Output filename.  If not given 'lml.xml' is used.

-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.

-dl databaseLabel Defines database label that is used to add to comments
section in tables.  This option is not mandatory but
strongly recommended.

-l Write out also information to generalData block that is related
to sample and measurement(s).  Writes the information from stations,
samplers, detectors and sources tables.  NOTE: Those entries from
sources table that belong to sampleData-block (=those referred in
samples-table) are written out even without this switch.

-is sampleId Generate file for any samples/measurements/analyses
that have given sampleId.

-im measId Generate file for any samples/measurements/analyses
that have given measId.

-ia analysisID Generate file for any samples/measurements/analyses
that have given analysisId.
```

```
-x Generate file for the whole contents of the database.
WARNING! This parses everything in the database to one XML file.
If you try this with a sizeable database, you will overload your
database engine, hard drive, and memory.  Unless you know otherwise,
assume that using this will cause your server to crash, burn, and
explode.

-s Write out data in sample tables for matching data.  Sample tables
are tables in sampleData block.

-m Write out data in sample tables (as with -s) and measurement
tables.  Measurement tables are tables in measurementData block
and possibly in calibrationData block.

-mn As with -m -option but if more that one measurement matches
given id, print only the newest one.

-a Write out data in sample tables, measurement tables (as with -m)
and analysis table.  Analysis tables are those in analysis and
calibration blocks.

-an As with -a -option but if more that one measurement and/or analysis
matches given id, print only the newest one.

-af As with -a -option but print only those analysis that are marked
as 'final' in finalResults.analysisStatus.

-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.

-n Do not generate externalKeys data even if the linssiInfo table
exists (if the table does not exist, that data is never
generated).

-e If the externalKeys data is generated, add all applicable data
from the existing externalKeys table to it.
```

Reads tables: samples, sampleSplitsCombines, sampleTransports, sampleTrackings, basicSamples, ctbtLabSamples, airFilterSamples, calibrationSamples, sources, airFilterSOH, calibrationNuclides, messages, measurements, alphaMeasurements, analyses, calsUsed, spectrumComponents, componentsUsed, peaks, lineAssociations, activities, activityLimits, nuclideRatios, rois, roiRatios, finalResults, calibrations, calTypes, calPoints, facilities, samplers, detectors, measurementSetups, sources, calsRecommended

Modules: linssiConfig.pm v.1.1.1 or newer, DBI package, GetOpt package, XML::LibXML package

4.3 akutolml

Purpose: Converts AKu format written by USS to LML format

Version: 2.3.0

Author: Antero Kuusi

Package: *Linssi* core (v.2.3)

Created: 2.5.2008

Updated: 11.6.2008
Version 2.0.0.1: Fixes behavior of measurementSetups table.
19.6.2008 JAH
Version 2.0.0.2: Adds detectors table with detectorId that is foreign key in measurementSetups table.
17.3.2010 JAH
Version 2.2.1: Upgraded to *Linssi* v.2.2 (relatively small additions in tables analysisBlocks, measurements, samples and spectra).
2.7.2010 JAH: Documentation
20.12.2011 til
Updated version number to 2.3 in documentation

Description: Script for converting analysis results in AKu format (block-based text format output by USS at least) to LML format (XML-like format).

Syntax:

```
akutolml <inputFile> [<outputFile>]
```

Modules: XML::LibXML package

Chapter 5

Calibration and Spectrum Management

Calibration management typically belongs to the basic tasks under `LinssiWorld`. The scripts presented in this chapter are linked to it. Additionally, support for the PHD spectrum file format defined by the CTBTO [6] and other spectrum formats is available using the scripts presented in this chapter.

5.1 calibrations.php

Purpose:	Tool for viewing calibrations
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
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 *Linssi* 2.0 - tos 2.7.2010 JAH: Documentation 20.12.2011 - til Updated version number to 2.3 in 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: measurementSetups, calibrations, calsRecommended, calsUsed, calTypes, calPoints, analyses

Writes tables: measurementSetups, calibrations

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

5.2 addCalibration.php

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

Version: 2.3.1

Author: Tommi Salonen

Package: *Linssi* core (v.2.3)

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
Linssi 2.0 - tos
2.7.2010 JAH: Documentation
20.12.2011 - til
Updated version number to 2.3 in documentation
2.8.2012 JAH: Version number corrected

Description: This script can be used to add a new calibration set to *Linssi* database. Calibration can also be imported from LML file.

Reads tables: calibrations

Writes tables: calibrations, calTypes, calPoints

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

5.3 dbtophd

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

Version: 2.3.0

Author: Andreas Pelikan

Package: *Linssi* core (v.2.3)

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
3.3.2008 JAH : Option dbname in configFile changed to databaseName

3.3.2008 AJM : Converted the macro to work with Linssi2.0 dbase
1.1.2010 AJM
Converted the macro to work with Linssi2.2 dbase. Preliminary version, correct execution of all options not guaranteed
2.7.2010 JAH: Documentation
20.12.2011 til
Updated version number to 2.3 in documentation

Description: This script creates PHD file from information in *Linssi* database. The table specifications are according to the database definition 2.3.
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.
```

5.4 dbtocalblocks

Purpose: Outputs given calibration from *Linssi* database in PHD-format
Version: 2.3.1
Author: Antero Kuusi
Package: *Linssi* core (v.2.3)
Created: 6.6.2008
Updated: 19.6.2008 JAH
Version 2.0.0.1: Added option -v to output idCal, added a check for missing -i and -m options, changed from write to read username, fixed documentation.
12.3.2010 JAH: Documentation
2.7.2010 JAH: Documentation
20.12.2011 til
Updated version number to 2.3 in documentation
2.8.2012 JAH: Small change to allow for incomplete calibration sets, e.g., efficiency calibration only

Description: Script for generating files containing calibration of given ID or the default calibration for one measurement setup. Output file format is the same as the one used in PHD-spectrum files for calibrations.

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

Syntax:
```
dbtocalblocks [-d database_name] [-u username] [-p password]
[-g configFile] -i <idCal>|-m <measSetupId> -o <calModelId>
[-f filename] [-v] [-odbc|- noodbc]
```

Options:
```
-d databaseName Name of the database that contains the information.
If not given the name in the configuration file will be
used.

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

-p password Password for the user.  If not given the 'read' password in
configuration file will be used.

-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.

-i idCal Writes out the calibration with given idCal.  Only one
of -i and -m can be given.

-m measSetupId Writes out the default calibration for
```

```
measurement setup measSetupId and calibration model
calModelId (given with option -o).  Only one
of -i and -m can be given.

-o calModelId The calibration model in measurement setup measSetupId
(given with option - m).

-f filename Writes the output to given file.  If this option
is not given the output is written to STDOUT.

-v Output idCal (useful with option -m).

-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.
```

Reads tables: measurementSetups, calibrations, calPoints

Modules: linssiConfig.pm v.1.1.1 or newer Perl libraries DBI and GetOpt

5.5 lmlSpecConv

Purpose: Converts a spectrum in LML-file to another format

Version: 2.3.0

Author: Antero Kuusi

Package: *Linssi* core (v.2.3)

Created: 15.9.2009

Updated: 2.7.2010 JAH: Documentation
20.12.2011 til
Updated version number to 2.3 in documentation

Description: Script for converting a spectrum in LML-file to another format. The following formats are currently recognized (as both input and output format):

- newline (Separator between counts is newline - default separator)
- whitespace (Separator between counts is whitespace)
- symbol (Any character but not 0,1,...,9 may be a separator)
- phdFormat (The spectrum is a combination of newline and whitespace separated formats, see PHD-specification. Conversion function not yet implemented)
- zipBase64 (Zipped spectrum, separator may be any character. Conversion function not yet implemented)

- perlPacked (Packed with Perl's pack function, separator is whitespace. Conversion function not yet implemented)
- countedZeros (See in ANSI42.42. Conversion function not yet implemented)

Syntax: `lmlSpecConv -i <inputFile> -f <outFormat> [-o <outputFile>]`

Options:

```
 -i <inputFile> Name of the file being converted.

-f <outFormat> Format of output spectrum.  See spectrumConverter package
for defined spectrum types.

-o <outputFile> Name for output file.  If no output file is specified,
the input file will be overwritten.
```

Modules: GetOpt package XML::LibXML package spectrumConverter package

Chapter 6

Reporting and Displaying

The most essential reporting and graphical display scripts linked to `LinssiWorld` are presented in this chapter. A full list of PHP scripts in the distribution package is given in Ch. 7.

6.1 analysisReport.php

Purpose:	Prints analysis reports
Version:	2.3.1
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
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 7.8.2006 : If there is no analyses found with the given parameters and idMeas parameter is given, a measurement report is printed - tos 8.2.2007 : Printing of peaks and line associations moved to separate scripts; peaks.php and lineAssociations.php - tos 21.2.2007 : added link to createLml.php -tos *Linssi* 2.0 - tos 2.7.2010 JAH: Documentation 5.8.2010 JAH: function printOrganization() upgraded, associationCount corrected, explainCount added 20.12.2011 - til Updated version number to 2.3 in documentation 2.8.2012 JAH: Documentation

Description: Prints analysis reports (short or long). It's also possible to get the 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 the 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, createEurdep.php, createPhd.php, createLml.php, showCorrFactors.php

6.2 editAnalysis.php

Purpose: Script for manual editing of analysis results

Version: 2.3.0

Author: Tommi Salonen

Package: *Linssi* core (v.2.3)

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
21.9.2007 : Automatic creation of XML messages to Lims (configured in linssiConfig.php) - tos
Linssi 2.0 - tos
2.7.2010 JAH: Documentation
20.12.2011 - til
Updated version number to 2.3 in 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, analysisBlocks, componentsUsed, samples

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

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

6.3 latestResults.php

Purpose: Creates and shows newest results of every facility in *Linssi*

Version: 2.3.1

Author: Antero Kuusi

Package: *Linssi* core (v.2.3)
Created: 25.7.2003
Updated: 14.6.2004
13.1.2005
11.9.2005/AI Original perl-script sar_update_new.cgi
was converted to php and renamed as latest_results.php
1.6.2006/AI Script was renamed as latestResults.php and
phd output was changed to analysisReport
11.6.2007/RR Anomalous nuclides and old spectra are
now printed in different colors.
27.8.2007/MMo Accepts also preliminary analysisStatus in addition to
NULL. This is required for NG Aatami analysis.
03.12.2007/MMo Ordering changed so that laboratories show last
19.9.2008/MMo Several modifications for *Linssi* 2.0 adaptation
30.9.2009/MMo Case insensitive search for anomalous nuclides
2.7.2010 JAH: Documentation
2.8.2012 JAH: Documentation

Description: This script creates and shows newest results of every station in *Linssi* database version 2.3. Color coding is used to point the user's attention to anomalous cases.

6.4 sampleReport.php

Purpose: Creates sampling reports in html and excel format
Version: 2.3.1
Author: Tommi Salonen
Package: *Linssi* core (v.2.3)
Created: 1.10.2005
Updated: Added acqStart, acqEnd, completionTime and dbEntryTime to the
report - tos
5.7.2006 JAH : Documentation
Moved mdc sign "<" to it's own column in excel report and
added a possibility to select any nuclides - tos
Linssi 2.0 - tos
2.7.2010 JAH: Documentation
20.12.2011 - til
Updated version number to 2.3 in documentation
2.8.2012 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, samples, airFilterSamples, ctbtLabSamples, measurements, finalResults, activities, activityLimits activityLimits
Modules: linssiConfig.php, htmlTags.php, linssiPHPLib.php, linssiFormTools.php, PEAR::OLE, PEAR::SpreadsheetExcelWriter

6.5 showSamples.php

Purpose:	Print the samples or measurements from chosen facilities
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
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 Fields that are shown can be set from linssiConfig.php + some other modifications - tos Added a possibility to add a sampler selection from linssiConfig.php - tos *Linssi* 2.0 - tos 2.7.2010 JAH: Documentation 20.12.2011 - til Updated version number to 2.3 in documentation

Description: This script prints the samples or measurements from the chosen stations in the chosen time interval. Samples / measurements are printed in a table and in the end of each row is printed a software selection form with all the software that has produced an analysis from the corresponding sample / measurement. The form is used as a link to the analysis results (analysisReport.php).
Status field indicates if there is a final, preliminary or invalid analysis from the corresponding sample / measurement or if anomalies have been found in the final analysis.
Samples that have not been measured can be included in the results by checking the corresponding checkbox.
Data sanity check option checks if any of the channels of the spectra in the result set has a value less than zero.Only the first spectrum in each measurement is checked (idSpectrum = 1).
There are some options in linssiConfig.php configuration file that affect the behaviour of this script. For example some of the fields that are shown for each sample / measurement can be configured. Also, station, sampler, sample type and sample production table selections can be switched on and off.

Reads tables: samples, measurements, measurementSetups, spectra, finalResults, analyses

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

6.6 showSpectrum.php

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

Version: 2.3.0
Author: Tommi Salonen
Package: *Linssi* core (v.2.3)
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
21.7.2006 : Script extended to handle the printing of multiple
spectra given their idMeas numbers + some other modifications - tos
Linssi 2.0 - tos
2.7.2010 JAH: Documentation
20.12.2011 - til
Updated version number to 2.3 in documentation

Description: This script prints line graph image of spectrum given the idAnalysis or idMeas number. If idAnalysis is given, the nuclide identifications are also printed if they are found from the database. If one or more idMeas parameters are given, the script creates a line graph for each spectra with different color. HTML-page also includes a javaScript function which is used to zoom into the certain part of the spectrum (zoomSpectrum.php required).
Reads tables: samples, measurements, analyses, calsUsed
Modules: JpGraph, linssiConfig.php, linssiPHPLib.php

6.7 zoomSpectrum.php

Purpose: Zoom feature for showSpectum.php
Version: 2.3.0
Author: Tommi Salonen
Package: *Linssi* core (v.2.3)
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
21.7.2006 : Extended to handle the zooming into a graph with
multiple spectra - tos
Linssi 2.0 - tos
2.7.2010 JAH: Documentation
6.8.2010 JAH: Introduced parameter showStrippedSpectrum
20.12.2011 - til
Updated version number to 2.3 in documentation

Description: A zoom feature for showSpectum.php. A 100 keV wide portion of the graph 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

Chapter 7

Linssi Script Reference List

This chapter presents the list of *Linssi* scripts in the distribution package as of August 2012. Many 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/`.

7.1 Configuration Scripts and Libraries

linssirc-template

Purpose:	Template for the configuration file .linssirc
Version:	2.3.1
Author:	Andreas Pelikan
Package:	*Linssi* core (v.2.3)
Created:	7.3.2005

linssiConfig.pm

Purpose:	Package for handling *Linssi* configuration files
Version:	2.3.1
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	10.10.2005

linssiConfig.php

Purpose:	*Linssi* PHP scripts configuration file
Version:	2.3.1
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.10.2005

linssiFormTools.php

Purpose:	Functions for printing forms element and getting selected parameters
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)

linssiPHPLib.php

Purpose:	PHP functions
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.10.2005

htmlTags.php

Purpose:	Perl CGI-like library, with functions that generate html tags
Version:	2.3.1
Author:	Andreas Pelikan
Package:	*Linssi* core (v.2.3)
Created:	27.1.2006

7.2 Database Creation Scripts

maketables

Purpose:	Creates tables in *Linssi* database
Version:	2.3.1
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	15.7.2003

fillfunctions

Purpose:	Fills in function codes and names into functions table
Version:	2.3.0
Author:	Jarmo Ala-Heikkila
Package:	*Linssi* core (v.2.3)
Created:	5.6.2008

desctables

Purpose:	Shows descriptions of all tables in a *Linssi* database
Version:	2.3.0
Author:	Jarmo Ala-Heikkila
Package:	*Linssi* core (v.2.3)
Created:	5.6.2008

listtables

Purpose:	List tables in a *Linssi* database instance
Version:	2.3.0
Author:	Andreas Pelikan
Package:	*Linssi* core (v.2.3)
Created:	29.6.2009

linssiFacilities.php

Purpose:	Populates the facilities-table in *Linssi* v.2.3
Version:	2.3.2
Author:	Ryan Lawrie
Package:	*Linssi* core (v.2.3)
Created:	October 20, 2011

alterTableNames

Purpose:	Changes table names case sensitive
Version:	2.3.0
Author:	til
Package:	*Linssi* core (v.2.3)
Created:	15.6.2011

linssi22to23

Purpose:	Script for converting *Linssi* 2.2 database to 2.3
Version:	2.3.0
Author:	Antero Kuusi
Package:	*Linssi* (v.2.3)
Created:	16.1.2011

7.3 Basic Housekeeping Scripts

deleteSample

Purpose:	Deletes a sample and all associated data from *Linssi*
Version:	2.3.0
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	21.7.2010 (for v.2.2)

deleteMeas

Purpose:	Deletes a measurement and all associated data from *Linssi*
Version:	2.3.0
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	21.7.2010 (for version 2.2)

deleteAnalysis

Purpose:	Deletes an analysis and all associated data from *Linssi*
Version:	2.3.0
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	21.7.2010 (for version 2.2)

7.4 Basic Database Input Scripts

lmltodb

Purpose:	Script for reading analysis and other data in LML format into *Linssi* database
Version:	2.3.1
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	4.9.2007

akutolml

Purpose:	Converts AKu format written by USS to LML format
Version:	2.3.0
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	2.5.2008

7.5 Data Extraction Scripts

dbtolml

Purpose:	Generates LML file from *Linssi* database
Version:	2.3.1
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	25.9.2007 (for v.2.0)

dbtophd

Purpose:	Creates PHD file of given measurement from *Linssi* database
Version:	2.3.0
Author:	Andreas Pelikan
Package:	*Linssi* core (v.2.3)
Created:	3.6.2004

dbtocalblocks

Purpose:	Outputs given calibration from *Linssi* database in PHD-format
Version:	2.3.1
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	6.6.2008

lmlSpecConv

Purpose:	Converts a spectrum in LML-file to another format
Version:	2.3.0
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	15.9.2009

spectrumConverter.pm

Purpose:	Library for converting spectrum in LML-file to another format
Version:	2.3.0
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	15.9.2009

7.6 Scripts for Handling Calibrations

calibrations.php

Purpose:	Tool for viewing calibrations
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

addCalibration.php

Purpose:	Add a new calibration set to *Linssi* database
Version:	2.3.1
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

7.7 Scripts for Interactive Data Browsing, Analysis, and Report Generation

addComments.php

Purpose:	A tool to add comments to data
Version:	2.3.0
Author:	NN
Package:	*Linssi* core (v.2.3)
Created:	1.1.2010

addMeasurementSetup.php

Purpose:	Add a new measurementSetup to *Linssi* database
Version:	2.3.0
Author:	NN
Package:	*Linssi* core (v.2.3)
Created:	1.1.2010

analysisReport.php

Purpose:	Prints analysis reports
Version:	2.3.1
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.10.2005

analysisResultsKML.php

Purpose:	View the latest analysis results in Google Earth
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	8.6.2007

anomalies.php

Purpose:	Prints a report that shows all detected anomalous nuclides
Version:	2.3.1
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

anomalousNuclides.php

Purpose:	Defines an array that contains all anomalous nuclides
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.10.2005

changeDb.php

Purpose:	Print a database selection form
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

createLml.php

Purpose:	Creates LML file from the chosen analysis
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	21.2.2006

createPhd.php

Purpose:	Creates a PHD file from the chosen analysis
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.3.2006

editAnalysis.php

Purpose:	Script for manual editing of analysis results
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

latestResults.php

Purpose:	Creates and shows newest results of every facility in *Linssi*
Version:	2.3.1
Author:	Antero Kuusi
Package:	*Linssi* core (v.2.3)
Created:	25.7.2003

lineAssociations.php

Purpose:	Prints the lineAssociations from the chosen analysis
Version:	2.3.1
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	8.2.2007

peakGraph.php

Purpose:	Shows the chosen peak as a line graph image
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

peaks.php

Purpose:	Prints the peaks from the chosen analysis
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	8.2.2007

qc.php

Purpose:	Shows the resolution for certain nuclides for QC purposes
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

ratioCalculus.php

Purpose:	Count ratios of Pu and U nuclides in the chosen analysis
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	4.1.2008

sampleReport.php

Purpose:	Creates sampling reports in html and excel format
Version:	2.3.1
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.10.2005

selectTimes.php

Purpose:	Prints form fields for time selection
Version:	2.3.0
Author:	NN
Package:	*Linssi* core (v.2.3)
Created:	1.1.2010

showActionSites.php

Purpose:	Prints rows from actionSites table for given idSample or idMeas
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	8.5.2009

showCorrFactors.php

Purpose:	Shows correction factors and raw activities for nuclides
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	2.3.2006

showDetectors.php

Purpose:	Shows the detectors in the database
Version:	2.3.0
Author:	Teemu Siiskonen, Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.5.2006

showFacilities.php

Purpose:	Shows the facilities in the database
Version:	2.3.0
Author:	Teemu Siiskonen, Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.5.2006

showMeasurementSetups.php

Purpose:	Shows the measurementSetups in the database
Version:	2.3.0
Author:	Teemu Siiskonen, Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.5.2006

showSampleInfo.php

Purpose:	Prints sample info
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	16.11.2007

showSamplers.php

Purpose:	Shows the samplers in the database
Version:	2.3.0
Author:	Teemu Siiskonen, Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.5.2006

showSamples.php

Purpose:	Print the samples or measurements from chosen facilities
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.10.2005

showSources.php

Purpose:	Shows the sources in the database
Version:	2.3.0
Author:	Teemu Siiskonen, Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.5.2006

showSpectrum.php

Purpose:	Prints a line graph image of a spectrum with nuclide identifications
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

spectrumPart.php

Purpose:	Creates a linegraph image of a certain part of a spectrum
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

trendsImage.php

Purpose:	Outputs image for trends.php
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	23.11.2007

trends.php

Purpose:	Creates a line graph image showing activity concentrations
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

zoomSpectrum.php

Purpose:	Zoom feature for showSpectum.php
Version:	2.3.0
Author:	Tommi Salonen
Package:	*Linssi* core (v.2.3)
Created:	1.6.2005

Chapter 8

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, Health Canada, and other organizations.

8.1 *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* [5].

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. 9). Generating unique and compatible database keys is **the key issue** when interfacing

[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.

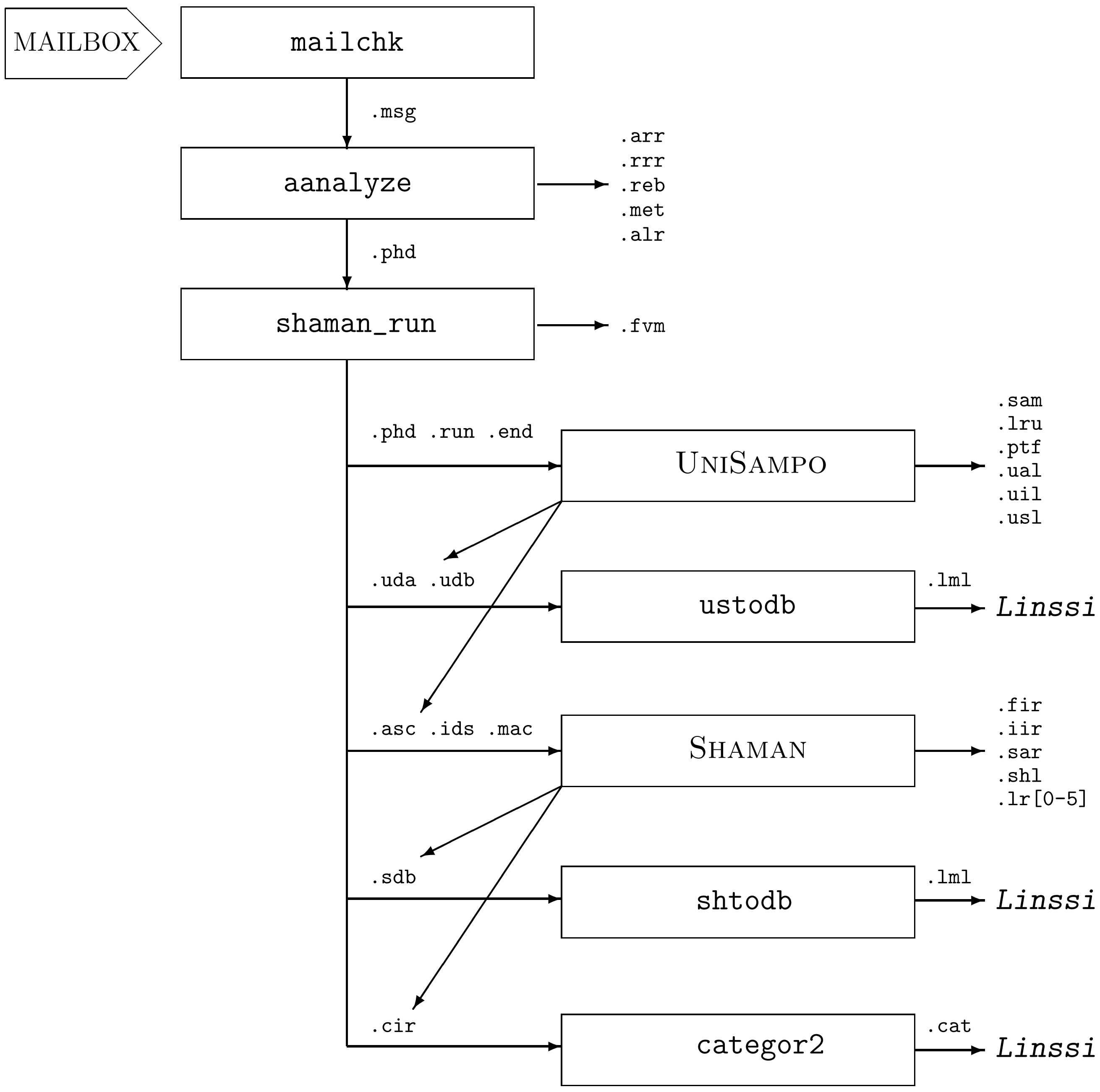


Figure 8.1: Flow chart of the UniSampo–Shaman pipeline with *Linssi*.

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.

8.1.1 Database Insertion in the Pipeline Mode

The UniSampo–Shaman analysis pipeline is illustrated in Fig. 8.1. 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. 8.1. 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. 52.

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 `lmltodb` script in the *Linssi* package that is 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.

8.1.2 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.

8.1.3 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. Additionally, 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.

8.2 Generation of Database Keys and Identifiers

Since analysis software packages run on entry point 3 of *Linssi* (see *Linssi* manual Part I [1]), 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. 9, and some additional string identifiers of *Linssi*, is to utilize the `#Comment` block of the PHD-spectrum as explained in Ch. 10.

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** [6]. 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 characters 13–14 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
  - if the source is not sampled, `seqint=000`

[c] Please note that the `sampleId` generated from standard PHD fields, like presented here, does not conform to the definition of Ch. 9. This is because the components sampler code and filter type are not defined in the PHD format.

```
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`.

v. `barcode`

– if this parameter is not given in `#Comment` block, its value is MID, `phdmid`, from the fourth line of `#Header` block

8.2.1 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.

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`, `calTypes`, `calPoints` and `calsUsed` are extracted from the `.uda` file, their `idCal`, `calId` 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 `calTypes` tables of the `.uda` report. In `calTypes` tables of the `.udb` report, on the other hand, no certificate information is available, since they are usually internal calibrations.

The UniSampo and Shaman reports are converted to the LML format (see Ch. 4) with the `akutolml` script prior to calling the `lmltodb` script.

[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`.

8.3 Tables Updated by UniSampo and Shaman

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

`airFilterSamples`	U	S
`calibrationSamples`	U	S
`samples`	U	S
`measurementSetups`	U	S
`measurements`	U	S
`spectra`	U	S

`calibrations`	U	
`calTypes`	U	
`calPoints`	U	
`calsUsed`	U	S
`analyses`	U	S
`analysisBlocks`	U	S
`peaks`	U	S
`lineAssociations`		S
`activities`	U	S
`activityLimits`	U	S
`nuclideRatios`		S

The last 11 tables are clearly analysis results from USS. Every new analysis is given a unique `idAnalysis/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 `idAnalysis/analysisId` when found necessary.

The first 6 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 6 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 6 tables output by the USS, the essential information can be retrieved from the PHD-spectrum to the database, possibly for manual completion later.

Chapter 9

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 radiation 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) [6].

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.

New in *Linssi* v.2.3 in contrast to v.1.1, there is a character string key also for analysis named `analysisId` in parallel with the integer key `idAnalysis`. Similarly, a character string key for calibration `calId` has been added in parallel with the integer key `idCal`. Their format definitions are also presented below.

9.1 idSample

An auto-increment integer key managed by the database.

9.2 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. 10.

9.3 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:

```
cctttttttxxxxT

  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
```

9.4 extSampleName

A freely settable character string, preferably unique.

9.5 idMeas

An auto-increment integer key managed by the database.

9.6 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
```

9.7 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
```

9.8 extMeasName

A freely settable character string, preferably unique.

9.9 idAnalysis

An auto-increment integer key managed by the database.

9.10 analysisId

The key `analysisId` is defined to be unique by using the software name, computer name (hostname) and the timestamp of analysis start.

```
pppppppp@hhhhhhhh:yyyy/mm/dd-hh:mm:ss

  pppppppp      software name
  hhhhhhhh      computer hostname
  yyyy/mm/dd    analysis start date
  hh:mm:ss      analysis start time

Example: Shaman@localhost:2010/08/04-16:47:04
```

9.11 idCal

An auto-increment integer key managed by the database.

9.12 calId

The key `calId` is defined to be unique in two different manners:

1. For calibrations included in a PHD-spectrum file, the `calId` is equal to `phdMeasName` (MID of the CTBTO).

2. For calibrations updated during the analysis and stored to *Linssi*, the `calId` is equal to `analysisId`.

```
Example 1: HEP02_FI001-D01_2004/12/13_08:12:00
Example 2: UniSampo@localhost:2010/08/04-16:47:04
```

Chapter 10

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 [6]. 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. 8.2).

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]

v. `barcode`

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`

10.1 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
```

[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`.

```
87.180000        260.206879       0.100000
238.632004       714.635803       0.100000
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 11

LML File Format for Database Import and Export

The XML-type tagged format named *Linssi* Markup Language or LML is the recommended file format for importing data to *Linssi* v.2.3. The scripts of Ch. 4 also support using it for data export.

The LML schema is defined with the xsd-files presented on p. 15. The purpose of this chapter is to give some hints for implementing LML in analysis software. We start with the simplest possible LML-file that imports only a `sampleId` key to the `samples` table:

```
<?xml version="1.0"?>
<LML xmlns="http://www.stuk.fi/linssi" linssiVersion="2.3">
  <sampleData>
    <samples>
      <sampleId>HEP02CI01F_2004/02/17_08:04:00_11_18</sampleId>
    </samples>
  </sampleData>
</LML>
```

During import with the `lmltodb` script, MySQL also generates the corresponding auto-increment key `idSample`, but no other data are added to *Linssi*. If we name the LML-file as `test.lml` and if the *Linssi* scripts reside in the directory `/usr/local/gamma/linssi2`, the LML-file is imported to *Linssi* with the following two commands:

```
$ export PERL5LIB=${PERL5LIB}:/usr/local/gamma/linssi2
$ /usr/local/gamma/linssi2/lmltodb -f test.lml -g ~/.linssirc
```

Here we assume that database username and password are defined in the `.linssirc` file in the user's home directory (see p. 5).

Below we present more realistic LML-files. The fields have a one-to-one correspondence with the *Linssi* database schema. The only complication is that the tables have been grouped to `sampleData`, `measurementData`, `analysisData`, `calibrationData`, `generalData`, and `otherData`. The *Linssi* tables that belong to each group have been listed in the xsd-files defining the LML schema. The `analysisData` group is hierarchically below the `measurementData` group that in turn is below the `sampleData` group, as can be seen in the example below. The last group is meant for communicating additional information to other programs, not for importing data into *Linssi*.

What is the minimum set of information that can be imported to *Linssi* using LML? The answer is relatively simple:

1. If any sample-related data is imported, the `samples` table data with at least the `sampleId` needs to be included in the LML-file. (See the example above.)
2. If any measurement-related data is imported, the `samples` table data with at least the `sampleId` and the `measurements` table data with at least the `measId` need to be included in the LML-file.
3. If any analysis-related data is imported, the `samples` table data with at least the `sampleId`, the `measurements` table data with at least the `measId` and the `analyses` table data with at least the `analysisId` need to be included in the LML-file.
4. If any calibration-related data is imported, the `calibrations` table data with at least the `calibrationId` needs to be included in the LML-file.
5. In each *Linssi* table there is a primary key, sometimes consisting of various components. The primary key component(s) must be defined in the LML-file, if anything is to be stored in the corresponding table in *Linssi*. However, if auto-increment integer keys like `idSample`, `idMeas`, `idAnalysis` and `idCal` are components of the primary key, they are not to be written in the LML-file.

11.1 Examples of an LML-File

The following kind of an LML file is used to initialize a new detector in *Linssi*. Note that the facility information must be included (at least the `facilityId` key) and the default measurement setup.

```
<?xml version="1.0"?>
<LML xmlns="http://www.stuk.fi/linssi" linssiVersion="2.3">
  <generalData>
    <facilities>
      <facilityId>stuk-zorro-01</facilityId>
      <facilityName>PikkuMusta-01</facilityName>
      <isMobile>1</isMobile>
      <organization>stuk</organization>
      <facilityType>backpack</facilityType>
      <isSampling>0</isSampling>
      <isMeasuring>1</isMeasuring>
    </facilities>

    <detectors>
      <detectorId>g-nai-5x4-55109</detectorId>
      <biasVoltage>670</biasVoltage>
      <diameter>127</diameter>
      <thickness>102</thickness>
      <detectorType>g-nai-5x4</detectorType>
      <detectorModel>55109</detectorModel>
      <facilityId>stuk-zorro-01</facilityId>
    </detectors>

    <sources>
      <sourceId>stuk-zorro-01-mobile</sourceId>
      <sourceGeometry>pointSource-unshielded</sourceGeometry>
    </sources>

    <measurementSetups>
      <measSetupId>stuk-zorro-01-g-nai-55109-3200</measSetupId>
      <detectorId>g-nai-5x4-55109</detectorId>
      <sourceId>stuk-zorro-01-mobile</sourceId>
      <sampleDistx>5000</sampleDistx>
    </measurementSetups>
  </generalData>
</LML>
```

The following kind of an LML file is used to store measurements in *Linssi*. Note that the pointers to the corresponding sample and calibration data must be included in the LML.

```
<?xml version="1.0"?>
<LML xmlns="http://www.stuk.fi/linssi" linssiVersion="2.3">
  <sampleData>
    <samples>
      <comments>Original idSample : 24</comments>
      <dbLastUpdate>2010-08-24 13:47:50</dbLastUpdate>
      <sampleId>STUK-RanidPro200-1-INSITU</sampleId>
      <facilityId>STUK-RanidPro200-1</facilityId>
    </samples>
    <measurementData>
      <measurements calId="LaBr_38x38_Q690_INSITU">
        <comments>Original idMeas :1523</comments>
        <acqRealTime>32.019775</acqRealTime>
        <measId>LaBr_38x38_Q690_1282741237941</measId>
        <acqEnd>2010-08-25 13:00:37</acqEnd>
        <measSetupId>STUK-RanidPro200-1_LaBr_38x38_Q690_INSITU</measSetupId>
        <acqStart>2010-08-25 13:00:05</acqStart>
        <acqStartNanoSeconds>652000000</acqStartNanoSeconds>
        <acqLiveTime>31.900146</acqLiveTime>
        <facilityId>STUK-RanidPro200-1</facilityId>
      </measurements>
      <spectra>
        <spectrum>0
0 0 0 0 0 0 0 0 0 0 0 19 32 49 48 72 117 207 2.3.055 89 56 44 38 54 69 56 76 74 63 64 87 88 80 77 86 97 126
141 91 120 97 95 111 114 116 121 116 86 105 100 85 102 84 74 83 86 90 104 92 75 73 72 81 76 106 101 79 86 78
75 91 83 83 82 84 75 77 74 77 72 75 81 76 64 54 77 60 76 52 55 61 68 62 49 72 65 56 68 54 49 61 55 55 61 68
...
2 1 1 2 1 1 0 1 0 0 2 1 2 1 3 1 2 0 1 0 2 3 0 0 0 0 0 1 0 1 1 1 0 1 1 1 0 0 1 0 0 1 2 1 1 0 0 1 1 0 2 0 1 0
2 2 1 0 2 2 1 1 0 0 0 2 0 0 0 1 1 2 0 1 0 1 2 1 0 2 2 0 0 2 0 0 0 1 1 0 0 1 2 1 1 1 0 0 1 1 0 1 1 2 0 2 0 2
0 0 2 0 0 1 2 1 0 1 0 0 1 0 0 1 0 0 3 0 1 2 3 0 0 0 4 2 0 0 1 0 2 0 0 0 2 2 1 0 0 0 0 0</spectrum>
        <spectrumFormat>symbol</spectrumFormat>
        <spectrumType>SUM</spectrumType>
        <lastChannel>1023</lastChannel>
        <idSpectrum>1</idSpectrum>
        <firstChannel>0</firstChannel>
        <firstValidChannel>0</firstValidChannel>
        <lastValidChannel>1023</lastValidChannel>
      </spectra>
    </measurementData>
  </sampleData>
  <calibrationData>
    <calibrations>
      <calId>LaBr_38x38_Q690_INSITU</calId>
      <comments>Copied from another database, original idCal 5 .
</comments>
      <measSetupId>fi.stuk.RANID-200-1_LaBr_38x38_Q690_INSITU</measSetupId>
    </calibrations>
    <calTypes>
      <creationTime>0000-00-00 00:00:00</creationTime>
      <calTypeId>energy</calTypeId>
      <parameters>2.0 0.01
1.89 0.01
8.7407E-5 0.01
7.5587E-9 0.01
</parameters>
      <idFunction>2</idFunction>
    </calTypes>
    <calTypes>
      <creationTime>0000-00-00 00:00:00</creationTime>
      <calTypeId>resolution</calTypeId>
      <parameters>38.8115 0.01
0.3819 0.01
7.599E-5 0.01
</parameters>
      <idFunction>4</idFunction>
    </calTypes>
    <calsRecommended>
      <calModelId>aatamiV1</calModelId>
      <measSetupId>STUK-RanidPro200-1_LaBr_38x38_Q690_INSITU</measSetupId>
    </calsRecommended>
  </calibrationData>
  <generalData>
    <measurementSetups>
      <detectorId>LaBr_38x38_Q690</detectorId>
      <comments>Copy of measurementSetup.
</comments>
      <measSetupId>STUK-RanidPro200-1_LaBr_38x38_Q690_INSITU</measSetupId>
    </measurementSetups>
  </generalData>
  <otherData>
    <externalKeys>
      <keyType>sample</keyType>
      <keyId>STUK-RanidPro200-1-INSITU</keyId>
      <idKey>24</idKey>
      <databaseId>snitch22@kemi.stuk.fi</databaseId>
    </externalKeys>
    <externalKeys>
      <keyType>meas</keyType>
      <keyId>LaBr_38x38_Q690_1282741237941</keyId>
      <idKey>1523</idKey>
      <databaseId>snitch22@kemi.stuk.fi</databaseId>
    </externalKeys>
    <externalKeys>
      <keyType>cal</keyType>
      <keyId>LaBr_38x38_Q690_INSITU</keyId>
      <idKey>5</idKey>
      <databaseId>snitch22@kemi.stuk.fi</databaseId>
    </externalKeys>
  </otherData>
</LML>
```

Chapter 12

Administrative Issues and Lessons Learnt

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 may be 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
   ```

   Alternatively, this setting can be made in the `[mysqld]` section of the configuration file `/etc/mysql/my.cnf`.

10. The `analysisBlocks` table is by far the largest one due to the three longblob's (baseline, stripped spectrum, peak search significance) stored in the table in a single analysis by default. 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 engine MyISAM. There are two alternative solutions:

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

    b. Convert the `analyses` table (or all tables) to InnoDB format that has no size limitations (http://dev.mysql.com/doc/refman/5.5/en/innodb-storage-engine.html). This can be done right after creating the *Linssi* database or when the database has been operating any period of time.

    In *Linssi* v.2.3, the default database engine defined in the `maketables` script is InnoDB for all database tables.

11. Indexing of fields makes a difference in query run times. The `maketables` script defines the most important indexes, but if you experience excessive run times, you should check that the relational key in your query is indexed. If not, indexing can be requested at any time, although in a large database it takes time. The time is paid back when running the queries.

12. There is a bug in the auto-increment feature of InnoDB in some MySQL versions. It makes the auto-increment keys to skip over integers. For most users this is only a nuisance, but it can be prevented by using the "traditional" auto-increment mode. This can be specified in the `[mysqld]` section of the configuration file `/etc/mysql/my.cnf`: `innodb_autoinc_lock_mode = 0`.

[a]The space requirement depends on the analysis software and methods.

13. We have experience of *Linssi* databases that contain of the order of 10 million analyses. In this case the disk space requirement is measured in hundreds of gigabytes or even a few terabytes. This amount of data can still be managed by MySQL in a single server. However, the query times are clearly starting to increase to a level that is inconvenient for the user. This is illustrated by the following tables where simple queries of type `select count (distinct keyX) from tableY;` were tried on variously sized *Linssi* databases. These results were obtained with the `rois` table:

Nr of records	Size of table (GB)	Size of database (GB)	Query time (s)
872,000	0.093	5.4	1.96
7,400,000	1.0	185	25.15
10,100,000	1.4	249	37.98

These results were obtained with the `analyses` table:

Nr of records	Size of table (GB)	Size of database (GB)	Query time (s)
282,000	0.162	5.4	0.36
2,700,000	1.3	185	29.39
3,500,000	1.8	249	38.65

On the basis of these data and our experience in practice, our recommendation is to start a new *Linssi* at the latest when the number of analyses is measured in millions. Depending on the organization and its activities, this may mean yearly databases, or even monthly databases. (Other database engines may be more efficient with huge databases, but probably the hardware requirements are also more demanding.)

Bibliography

[1] Pertti Aarnio, Jarmo Ala-Heikkilä, Ian Hoffman, Tarja Ilander, Seppo Klemola, Aleksi Mattila, Antero Kuusi, Mikael Moring, Mika Nikkinen, Andreas Pelikan, Samu Ristkari, Tommi Salonen, Teemu Siiskonen, Petri Smolander, Harri Toivonen, Kurt Ungar, Kaj Vesterbacka, Weihua Zhang, *Linssi — SQL Database for Gamma-Ray Spectrometry.* Part I: Database, Version 2.3. Helsinki University of Technology Publications in Engineering Physics. A, TKK-F-A861, Espoo 2011.
http://linssi.hut.fi/linssi_23.pdf

[2] 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/linssi.pdf

[3] J. Ala-Heikkilä, *Linssi — SQL Database for Gamma-Ray Spectrometry.* Part II: Scripts and Interfaces, Version 1.1. Helsinki University of Technology Publications in Engineering Physics. A, TKK-F-A842, Espoo 2006.
http://linssi.hut.fi/scripts.pdf

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

[5] Shaman — *Expert System for Radionuclide Identification, Version 1.16.* User's Guide Version 1.9. Baryon Oy, Ltd., Espoo, July 2007.

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

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

Appendix A

Installation Instructions for MySQL-*Linssi*

A.1 MySQL 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. Here we present installation instructions for *Linssi* under MySQL.

MySQL database is a commercial package that is included in many Linux distributions, including RedHat, SuSE, and Ubuntu. 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.2.3 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 Ubuntu 10.04 for *Linssi* testing purposes:

```
mysql-client-5.1
mysql-client-core-5.1
mysql-common
mysql-server-5.1
mysql-server-core-5.1

libdbd-mysql-perl
libmysqlclient16
libqt4-sql-mysql

perl
perl-base
perl-modules

php5
php5-cli
php5-common
php5-gd
php5-mysql
php-pear
```

```
libapache2-mod-php5
libphp-jpgraph
```

This list is only meant for illustration, since the naming of packages varies from distribution to another. Some of the listed packages come in the default installation, some must be selected, and some others come as dependencies of the selected packages. In addition to Ubuntu, the *Linssi* v.2.3 scripts have been in production use under CentOS and SuSE.

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';
mysql> show grants for xunil@localhost;
```

If you are running the SHAMAN spectrum categorizer `categor2`, the following grant that allows loading of files into *Linssi* must also be made:

```
mysql> grant file on *.* to xunil@localhost;
```

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. The script `fillfunctions` shall be invoked right after in order to fulfill the foreign key requirements when storing analysis results to *Linssi*. The success can be checked with the scripts `desctables` and `listtables` that also belong to the *Linssi* package.

In order to use these scripts, the environmental variable `PERL5LIB` probably needs to be defined to point to the directory where the Perl library of *Linssi* `linssiConfig.pm` resides. Let us assume that the directory is `/usr/local/gamma/linssi2` and the bash shell is used. The chain of commands is:

```
% export PERL5LIB=${PERL5LIB}:/usr/local/gamma/linssi2
% cd /usr/local/gamma/linssi2
% maketables -d linssi -u xunil -p some_password
% fillfunctions -d linssi -u xunil -p some_password
% desctables -d linssi -u webbi | less
% listtables -d linssi -u webbi | less
```

The directory for database scripts shall be defined in each user's *Linssi* configuration file `.linssirc` (see Sec. 2.2). The most important script is `lmltodb` that imports analysis results to *Linssi*. When using UniSampo and Shaman with *Linssi*, there should be a symbolic link named `shtodb` pointing to `../shaman/bin/shtodb2` and another link called `ustodb` pointing to `../shaman/bin/ustodb2`.[a]

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

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

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 as 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:

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

```
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://jpgraph.net/`. 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.

Appendix B

Installation Instructions for PostgreSQL-*Linssi*

B.1 PostgreSQL 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. Here we present installation instructions for *Linssi* under PostgreSQL.

PostgreSQL is an open source object-relational database system available for all major operating systems including Linux, many flavors of Unix, and Windows. In Linux it is packaged in many distributions including, but not limited to, Debian, CentOS, Fedora, Red Hat, and Ubuntu. It is released under the PostgreSQL License similar to the BSD or MIT licenses, and is free to use for any purpose.

PostgreSQL version 9.1 or newer is recommended with *Linssi* v 2.3. Mixed installation of 9.1 server on the database server and 8.4 client libraries on a separate webserver is known to work. Using 8.4 server is untested and is not recommended.

Here we give instructions on how to install a PostgreSQL database server version 9.3 in Ubuntu 14.04.02 LTS desktop.The installation provides remote access through http-server giving access to local *Linssi* scripts. The scripts are written in PHP and perl. For PHP version 5.3 or newer is recommended. 5.2 will probably work, but has not been tested. The perl scripts require version 5.10.1 or newer

It should be noted that by default the Ubuntu desktop software package does not include a http-server. We have installed Apache2.

At least the following (sub)packages provided by Ubuntu software center are needed to run *Linssi* scripts:

```
postgresql-9.3
postgresql-client-9.3
postgresql-client-common
postgresql-common

libpq5
libdbd-pg-perl
libdbi-perl
```

```
perl
perl-base
perl-modules

php5
php5-cli
php5-common
php5-gd
php5-pgsql
php-pear

apache2
libapache2-mod-php5

libxml-libxml-perl
```

This list is only meant for illustration, since the naming of packages varies from distribution to another. Some of the listed packages come in the default installation, some must be selected, and some others come as dependencies of the selected packages.

In addition to the above, there are *Linssi* scripts needing software not supported by the Ubuntu software repository. This software is provided with the *Linssi* scripts package.

B.2 Installing PostgreSQL

Package installation in Ubuntu is done either using the command `sudo apt-get install` or the Ubuntu Software Center (USC) to find and install the required packages. Use the USC search window to find the relevant packages. The following search words can be used to find the corresponding packages: `apache2, postgresql, php5, php-pear, libdbi-perl, libdbd-pg-perl`, `php5-pgsql`, and `libxml-libxml-perl`. Install all of them by clicking the package name and selecting 'Install'. You can find out the already installed software by selecting 'Installed' in the USC. Note that you have to click the 'Show nnnn technical items' text to see all the packages.

The system is to be set up using a suitable Linux system configuration tool to start PostgreSQL automatically at boot time. In Ubuntu this is done automatically when installing the `postgresql` package.

B.3 Creating PostgreSQL Users

In a typical configuration of PostgreSQL in Linux systems, the user `postgres` can connect to databases with the database super user privileges without a password, i.e., peer authentication. Thus it is recommended to create Linux user `postgres` and do all database administration with that identity.

To create other users first login to Linux as user `postgres`.

Best way to create users with password is with the `createuser` tool using -P option for password prompt. This avoids clear text password being visible on command line or recorded in database log. In our example we create two users, one for reading and another with write privileges. Answer no to all questions about granting various super user privileges to the new users. Below $-sign marks the Linux prompt.

```
$ createuser -P xunil
$ createuser -P webbi
```

Access to a database is controlled, not only by user definitions alone, but also in the config file `pg_hba.conf` in PostgreSQL config directory `/etc/postgresql/9.3/maín/`. To allow the users created above to connect to database `linssi` with password over Unix domain socket, the following row or a broader definition has to added in the config file.

```
local linssi xunil,webbi md5
```

For network based access use

```
host linssi xunil,webbi 192.168.0.0/24 md5
```

adjusting the allowed IP address range as necessary or use keyword `all` to allow any address.
After editing the config of a running database, forced reload of config is required for the changes to take effect.

```
$ /etc/init.d/postgresql reload
```

B.4 Creating Database *Linssi*

Still as the user `postgres` connect to PostgreSQL

```
$ psql
```

Note that commands given when using `psql`, the command line interface to PostgreSQL, are below identified by the prompt `#`.
In this example we create both the new database `linssi` and the schema `linssi23` in it and grant privileges to its users. On an existing server setup it may be necessary to only create one or more schemas in an already existing database. The user `xunil` is made the owner of the new schema, giving all the privileges in the new schema, including dropping the schema and granting privileges within the schema.

```
# create database linssi;
# \c linssi
# create schema linssi23 authorization xunil;
```

The user `webbi` has no privileges by default. There are two approaches to giving the necessary select privilege.

```
# alter default privileges for user xunil grant select on tables to webbi;
```

will grant user `webbi` select privileges on any tables created by user `xunil` in the future. Alternatively privileges can be granted with standard grant statement

```
# grant select on all tables in schema linssi23 to webbi;
# grant usage on schema linssi23 to webbi;
```

but that will only apply to objects that exist at the time, so it would do nothing right now when used before creating the `linssi23` tables.

Linssi core scripts will always set session default to value given in the configuration, but setting user default in database is useful if direct usage with the `psql` client is needed. This can be done by the commands

```
#alter user webbi set search_path to linssi23;
#alter user xunil set search_path to linssi23;
#\q
```

B.5 Using PostgreSQL Commands on *Linssi*

To log in the database *Linssi* as the user `webbi` or `xunil`,

```
$ psql -d linssi -U webbi
```

Password is required for both accounts and will be prompted for, unless password is given in config file `/.pgpass`

When some data has been imported to the database, its contents can be browsed on PostgreSQL command line using the `select`-command. This is, however, not recommended. In order to make efficient use of the data some expertise is needed and that should be aimed at writing useful scripts. Some of them are available in the core distribution. See below.

B.6 Installing *Linssi* Perl Scripts

In order to create and manipulate *Linssi* tables and data a set of around 20 Perl scripts is provided in the *Linssi* core scripts. These database scripts need to be copied to the directory defined in each user's *Linssi* configuration file `.linssirc`, and in *Linssi* PHP scripts configuration file `linssiConfig.php` (see Sec. 2.2).

In addition, in order to use these scripts, the environmental variable `PERL5LIB` needs to be set to point the Perl directory. Let us assume that the directory is `/usr/local/gamma/linssi2` and the bash shell is used. The chain of commands is:

```
$ export PERL5LIB=${PERL5LIB}:/usr/local/gamma/linssi2
$ cd /usr/local/gamma/linssi2
$ chmod u+x *
```

where also the execution rights have been given to the user. Users should also include the above export command to their `.bashrc` scripts.

B.7 Creating and Filling *Linssi* Tables

A perl script called `maketables_pgsql` is provided in the *Linssi* package to install the tables for results storage. The success can be checked with the script `desctables`, which also belongs to the *Linssi* package.

```
$ maketables_pgsql -d linssi -sc linssi23 -u xunil -p some_password
$ desctables -d linssi -u webbi | less
```

The filling of *Linssi* tables is normally provided by the measurement and analysis software used in your laboratory. For example, when using UniSampo and Shaman with *Linssi*, you need a symbolic link named `shtodb` pointing to `../shaman/bin/shtodb2` and another link called `ustodb` pointing to `../shaman/bin/ustodb2`. [a]

Your existing analysis results can be imported to the database using the script `lmltodb`. It takes an LML file as input and stores the data to *Linssi*. LML stands for *Linssi* Markup Language, which is written in XML to facilitate data import to *Linssi*.

B.8 Installing *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.

If running *Linssi* databases under both MySQL and PostgreSQL, eg., for testing purposes, separate installations of PHP scripts are needed for MySQL and PostgreSQL. On the other hand, single installation of these scripts can access several *Linssi* database instances running on the same server provided these can be accessed with the same user credentials.

B.8.1 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 environment variables. How to set these variables depends on the installation. In Ubuntu 14.04.02 LTS the typical configuration file, `httpd.conf` or `httpd2.conf`, does not exist by default.

Create the file `/etc/apache2/httpd2.conf`, insert there the following lines:

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

and add the command `Include /etc/apache2/httpd2.conf` as the last line of the file `/etc/apache2/apache2.conf`. Reload Apache configuration

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

```
$ /etc/init.d/apache2 reload
```

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)`.

The function `phpinfo()` can be disabled by editing the file `/etc/php5/apache2/phpi.ini`. Add `phpinfo` to the list disabled functions in the line `disable_functions = ...`. In addition, in order to stop printing of the credentials unauthorized users must not have rights to write PHP (or other) scripts for Apache to execute.

By default *Linssi* PHP scripts expect that the database access credentials are stored in Apache environment variables as described 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.

B.8.2 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 -d /var/www/includes/.htpasswd some_username
```

and `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 should omit the `-c` flag. If using some other directory or filename, edit `linssiConfig.php` to define the path to the created password file. The `-d` flag is used to match with the `crypt()` encryption method of the *Linssi* `php` scripts.

B.8.3 Installing JpGraph

The graphics in these scripts requires a PHP package named `JpGraph` that is available from `http://jpgraph.net/`. The latest free release is Ver: 3.07 from January 2010. This version is distributed with *Linssi* scripts and should be used in this context.

Extract the PHP source code of the `JpGraph` package, i.e., the contents of the `src` directory, into any directory readable by PHP and set the installation path in the `linssiConfig.php` configuration file. By default, the directory `/var/www/html/jpgraph` is assumed.

B.8.4 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.

Extract the `html2fpdf` 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/pdf` is assumed.

B.8.5 Getting Sample Reports in XLS Format

The `Pear::OLE` and `Pear::Spreadsheet_Excel_Writer` packages are required to get sample reports in XLS format. `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 should go like this:

```
$ pear install OLE
$ pear install Spreadsheet_Excel_Writer
```

Unfortunately, there seems to be only beta versions of these packages available. Their installation has to be performed with the `-d` option as:

```
$ 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/php`, which is also the default path in `linssiConfig.php`.

In the beta version, at least, there is an error on the line 2490 of the script `/usr/share/php/Spreadsheet/Excel/Writer/Worksheet.php`. Remove `&` from the line: `$this -> _append(&$string,true);`. In addition, the produced XLS format is not readable with all the versions of Microsoft Excel. However, it will be readable when saved from Libre Office.

Appendix C

Installation Instructions for MariaDB-*Linssi*

C.1 MySQL Database

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

MariaDB is open source relational database. It is an enhanced, drop-in replacement for MySQL. It is available for many Linux distributions, including RedHat, SuSE, and Ubuntu. The newest versions of the database can be found on `https://mariadb.org`.

In addition to MariaDB software *Linssi* distribution takes advantage of Perl and PHP. For example, the following packages were installed under Ubuntu 14.04 for *Linssi* testing purposes:

```
MariaDB common metapapackage
MariaDB database client binaries
MariaDB database client library
MariaDB database core client binaries
MariaDB database core server files
MariaDB database server binaries

libhtml-template-perl
libdb-mysql-perl
libdbi-perl
perl
perl-base
perl-modules

php5
php5-cli
php5-common
php5-gd
php5-mysql
php-pear
```

```
apache2
libapache2-mod-php5

libphp-jpgraph
libxml-libxml-perl
```

This list is only meant for illustration, since the naming of packages varies from distribution to another. Some of the listed packages come in the default installation, some must be selected, and some others come as dependencies of the selected packages. In addition to Ubuntu, the *Linssi* v.2.3 scripts have been in production use under CentOS and SuSE.

In addition to the above, there are *Linssi* scripts needing software not supported by the Ubuntu software repository. This software is provided with the *Linssi* scripts package.

C.2 Installing MariaDB

Package installation in Ubuntu is done either using the command `sudo apt-get install` or the Ubuntu Software Center (USC) to find and install the required packages. Use the USC search window to find the relevant packages. The following search words can be used to find the corresponding packages: `apache2`, `mariadb`, `php5`, `php-pear`, `libdbi-perl`, `php5-mysql`, and `libxml-libxml-perl`. Install all of them by clicking the package name and selecting 'Install'. You can find out the already installed software by selecting 'Installed' in the USC. Note that you have to click the 'Show nnnn technical items' text to see all the packages.

The system is to be set up using a suitable Linux system configuration tool to start MariaDB automatically at boot time. In Ubuntu this is done automatically when installing the `MariaDB common metapackage` package.

C.3 Creating *Linssi* Database And Its Users

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 -p
```

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

```
MariaDB[(none)]> use mysql;
MariaDB[(mysql)]> update user set password=password('new_password') where
                  user='root';
MariaDB[(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.

```
MariaDB[(mysql)]> create database linssi;
MariaDB[(mysql)]> grant select on linssi.* to webbi@localhost identified by
```

```
                  'some_read_password';
MariaDB[(mysql)]> show grants for webbil@localhost;
MariaDB[(mysql)]> grant create,lock tables,insert,select,update on linssi.*
                  to xunil@localhost identified by 'some_write_password';
MariaDB[(mysql)]> show grants for xunil@localhost;
```

If you are running the SHAMAN spectrum categorizer `categor2`, the following grant that allows loading of files into *Linssi* must also be made:

```
MariaDB[(mysql)]> grant file on *.* to xunil@localhost identified by
                  'some_write_password';
MariaDB[(mysql)]> \q
```

C.4 Installing *Linssi* Perl Scripts

In order to create and manipulate *Linssi* tables and data a set of around 20 Perl scripts is provided in the *Linssi* core scripts. These database scripts need to be copied to the directory defined in each user's *Linssi* configuration file `.linssirc`, and in *Linssi* PHP scripts configuration file `linssiConfig.php` (see Sec. 2.2).

In addition, in order to use these scripts, the environmental variable `PERL5LIB` needs to be set to point the Perl directory. Let us assume that the directory is `/usr/local/gamma/linssi2` and the bash shell is used. The chain of commands is:

```
$ export PERL5LIB=${PERL5LIB}:/usr/local/gamma/linssi2
$ cd /usr/local/gamma/linssi2
$ chmod u+x *
```

where also the execution rights have been given to the user. Users should also include the above export command to their `.bashrc` scripts.

C.5 Creating and Filling *Linssi* Tables

The *Linssi* database was created, but it does not contain any tables yet. A perl script called `maketables_mysql` is provided in the *Linssi* package to install the basic tables for result storage.The success can be checked with the scripts `desctables` and `listtables` that also belong to the *Linssi* package.

The chain of commands is:

```
$ cd /usr/local/gamma/linssi2
$ maketables_mysql -d linssi -u xunil -p some_write_password
$ desctables -d linssi -u webbi | less
$ listtables -d linssi -u webbi | less
```

The filling of *Linssi* tables is normally provided by the measurement and analysis software used in your laboratory. For example, when using UNISAMPO and SHAMAN with *Linssi*, you need a symbolic link named `shtodb` pointing to `../shaman/bin/shtodb2` and another link called `ustodb` pointing to `../shaman/bin/ustodb2`. [a]

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

Your existing analysis results can be imported to the database using the script `lmltodb`. It takes an LML file as input and stores the data to *Linssi*. LML stands for *Linssi* Markup Language, which is written in XML to facilitate data import to *Linssi*.

C.6 Using MySQL Commands on *Linssi*

To login to database 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 `/.my.cnf` in the home directory (see Sec. 2.2). Assuming the second alternative: To log in the database as the user `webbi`, a read password is required:

```
$ mysql linssi -u webbi -p
```

To log in the database as the user `xunil`, a write password is required:

```
$ 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
MariaDB[(none)]> 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. This is, however, not recommended. In order to make efficient use of the data some expertise is needed and that should be aimed at writing useful scripts. Some of them are available in the core distribution.

C.7 Installing *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.

If running *Linssi* databases under both MySQL and PostgreSQL, eg., for testing purposes, separate installations of PHP scripts are needed for MySQL and PostgreSQL. On the other hand, single installation of these scripts can access several *Linssi* database instances running on the same server provided these can be accessed with the same user credentials.

C.7.1 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 environment variables. How to set these variables depends on the installation. In Ubuntu 14.04.02 LTS the typical configuration file, `httpd.conf` or `httpd2.conf`, does not exist by default.

Create the file `/etc/apache2/httpd2.conf`, insert there the following lines:

```
SetEnv read_username "webbi"
SetEnv read_password "some_read_password"
SetEnv write_username "xunil"
SetEnv write_password "some_write_password"
```

Usernames and passwords must be equivalent to those used when creating the database.

Add the command `Include /etc/apache2/httpd2.conf` as the last line of the file `/etc/apache2/apache2.conf`. Reload Apache configuration

```
$ /etc/init.d/apache2 reload
```

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)`.

The function `phpinfo()` can be disabled by editing the file `/etc/php5/apache2/phpi.ini`. Add `phpinfo` to the list disabled functions in the line `disable_functions = ...`. In addition, in order to stop printing of the credentials unauthorized users must not have rights to write PHP (or other) scripts for Apache to execute.

By default *Linssi* PHP scripts expect that the database access credentials are stored in Apache environment variables as described 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.

C.7.2 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 -d /var/www/includes/.htpasswd some_username
```

and `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 should omit the `-c` flag. If using some other directory or filename, edit `linssiConfig.php` to define the path to the created password file. The `-d` flag is used to match with the `crypt()` encryption method of the *Linssi* `php` scripts.

C.7.3 Installing JpGraph

The graphics in these scripts requires a PHP package named `JpGraph` that is available from http://jpgraph.net/. The latest free release is Ver: 3.07 from January 2010. This version is distributed with *Linssi* scripts and should be used in this context.

Extract the PHP source code of the JpGraph package, i.e., the contents of the src directory, into any directory readable by PHP and set the installation path in the linssiConfig.php configuration file. By default, the directory /var/www/html/jpgraph is assumed.

C.7.4 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.

Extract the html2fpdf 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/pdf is assumed.

C.7.5 Getting Sample Reports in XLS Format

The Pear::OLE and Pear::Spreadsheet_Excel_Writer packages are required to get sample reports in XLS format. 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 should go like this:

```
$ pear install OLE
$ pear install Spreadsheet_Excel_Writer
```

Unfortunately, there seems to be only beta versions of these packages available. Their installation has to be performed with the -d option as:

```
$ 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/php, which is also the default path in linssiConfig.php.

In the beta version, at least, there is an error on the line 2490 of the script /usr/share/php/Spreadsheet/Excel/Writer/Worksheet.php. Remove & from the line: `$this -> _append(&$string,true);`. In addition, the produced XLS format is not readable with all the versions of Microsoft Excel. However, it will be readable when saved from Libre Office.

ISBN 978-952-60-3315-0 (printed)
ISBN 978-952-60-3616-8 (pdf)
ISSN 1456-3320