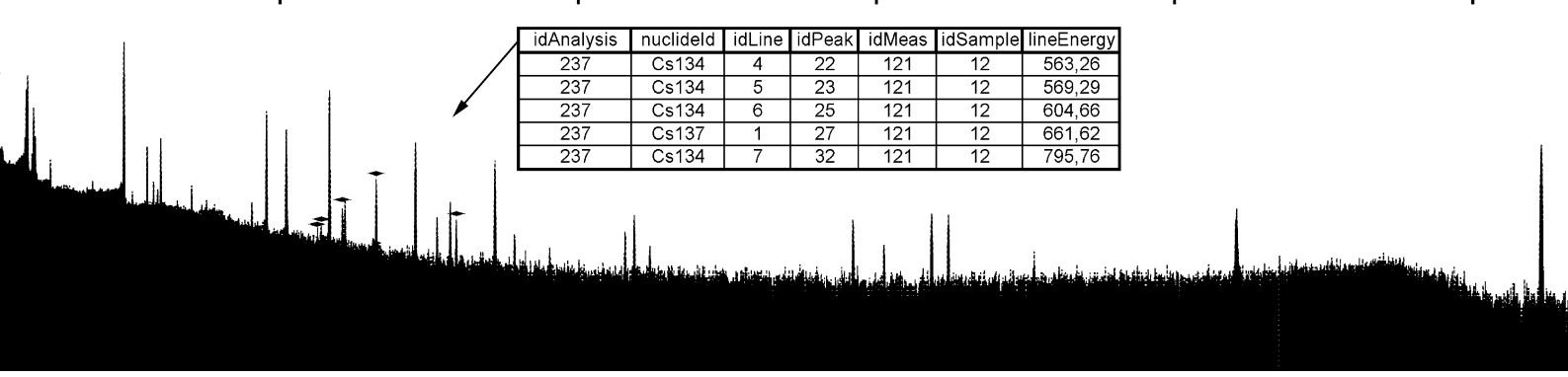


**LINSSI**  
**SQL DATABASE FOR GAMMA-RAY SPECTROMETRY**  
PART I: DATABASE  
Version 2.3

**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	nuclideld	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



**LINSSI**  
**SQL DATABASE FOR GAMMA-RAY SPECTROMETRY**  
PART I: DATABASE  
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**

## Database Design and Manual

© 2005, 2006, 2007, 2009, 2010, 2011

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

<sup>1</sup>Aalto University School of Science, Fission and Radiation Physics Group

<sup>2</sup>Finnish Radiation and Nuclear Safety Authority

<sup>3</sup>Health Canada, Radiation Protection Bureau

<sup>4</sup>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 978-952-60-3262-7 (printed)

ISBN 978-952-60-3581-9 (pdf)

ISSN 1456-3320

For the latest version of this manual see:

<http://linssi.hut.fi/>

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

## **Copyright**

This database design and accompanying written materials are products of copyright © owners and thereby protected by international copyright laws and treaties. You must keep the database package in strict confidence and treat it like any other copyrighted material. You may not copy the database or the written materials accompanying the database package except as explicitly allowed by the license. The use of the database package must be in strict adherence with the license.

## **License**

License conditions that are applicable for database scripts are defined in a separate document that must be consulted.

## **Disclaimer of Responsibility for the Software**

The database design is provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. The authors do not warrant that the functions contained in the design will meet any requirements or that the operation of the database will be error free.

In no event will the authors be liable for any damages, including any lost profits, lost savings, or other incidental or consequential damages arising out of the use or inability to use the database, even if authors' representative has been advised of the possibility of such damages, or for any claim by any other party.

MySQL is a trademark of MySQL AB. SHAMAN is a trademark of Baryon Oy. SAMPO, MICROSAMPO and SAMPO 90 are trademarks of Logion Oy. Other trademarks are the property of their respective owners.

**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

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	The Database System . . . . .	1
1.2	A Stroll between the Tables . . . . .	3
1.3	Tables . . . . .	4
1.4	Units and Default Values . . . . .	6
1.5	Naming Conventions . . . . .	6
<b>2</b>	<b>Facilities and Action Sites</b>	<b>9</b>
2.1	Facilities . . . . .	9
2.2	Mobile Coordinates . . . . .	12
2.3	Weather . . . . .	14
2.4	Weather Parameters . . . . .	15
2.5	Source Receptor Sensitivity (SRS) . . . . .	16
2.6	Missions . . . . .	17
2.7	Mission Devices . . . . .	18
2.8	Action Sites . . . . .	19
<b>3</b>	<b>Sample Types</b>	<b>21</b>
3.1	Calibration Samples . . . . .	21
3.1.1	Calibration Samples . . . . .	21
3.1.2	Calibration Nuclides . . . . .	23
3.1.3	Calibration Libraries . . . . .	24
3.2	Basic Samples . . . . .	25
3.3	Air Filter Samples . . . . .	27
3.3.1	Air Filter Samples . . . . .	27
3.3.2	Samplers . . . . .	29
3.3.3	Sampling State-of-Health Data . . . . .	30
3.4	CTBT Laboratory Samples . . . . .	31
<b>4</b>	<b>Common Sample Properties</b>	<b>35</b>
4.1	Samples . . . . .	35
4.2	Splitting and Combining Samples . . . . .	38
4.3	Sample Transport . . . . .	39
4.4	Sample Tracking . . . . .	41
<b>5</b>	<b>Calibrations</b>	<b>43</b>
5.1	Calibrations . . . . .	43
5.2	Calibration Types . . . . .	45
5.2.1	Peak Shape Calibrations . . . . .	47

5.2.2	Energy Calibration . . . . .	47
5.2.3	Efficiency Calibration . . . . .	47
5.2.4	Calibration Functions . . . . .	47
5.3	Calibration Points . . . . .	48
5.4	Recommended Calibrations . . . . .	49
5.5	Used Calibrations . . . . .	50
5.6	Recommended Backgrounds . . . . .	51
5.7	Recommended Blanks . . . . .	52
<b>6</b>	<b>Measurements</b>	<b>53</b>
6.1	Sources . . . . .	53
6.2	Detectors . . . . .	56
6.3	Shields . . . . .	57
6.4	Attenuators . . . . .	58
6.5	Measurement Setups . . . . .	59
6.6	Measurements . . . . .	61
6.7	Spectra . . . . .	64
6.8	Alpha Measurements . . . . .	66
<b>7</b>	<b>Regions of Interest and Spectrum Components</b>	<b>69</b>
7.1	Regions of Interest (ROIs) . . . . .	69
7.2	ROI Limits . . . . .	71
7.3	ROI Components . . . . .	72
7.4	ROI Ratios . . . . .	73
7.5	Spectrum Components . . . . .	74
7.6	Spectrum Components Used . . . . .	75
<b>8</b>	<b>Analysis</b>	<b>77</b>
8.1	Analyses . . . . .	77
8.2	Analysis Blocks . . . . .	81
8.3	Peaks . . . . .	83
8.4	Line Associations . . . . .	88
8.5	Nuclides and Their Activities . . . . .	91
8.6	Activity Limits . . . . .	96
8.7	Nuclide Ratios . . . . .	99
8.8	Final Analysis Results . . . . .	101
<b>9</b>	<b>Functions</b>	<b>103</b>
9.1	Functions . . . . .	103
9.2	Function Definitions . . . . .	104
9.3	Function List . . . . .	105
9.3.1	MathML presentation of the functions . . . . .	109
9.3.2	MathML support in SHAMAN . . . . .	110
<b>10</b>	<b>Connections</b>	<b>113</b>
10.1	Database Information . . . . .	113
10.2	Messages . . . . .	114
10.3	External Keys . . . . .	116
	<b>Bibliography</b>	<b>117</b>



# Chapter 1

## Introduction

*Linssi* database for gamma-ray spectrometry is developed in an international collaboration between Health Canada (HC), Aalto University School of Science, and Radiation and Nuclear Safety Authority (STUK).

The previous version of *Linssi* database system, version 1.1, was published on July 7, 2006. It is documented in two reports, one describing the database[1], and other the scripts and interfaces[2]. The database report also contains a brief history of *Linssi*. The version 1.1 contained 32 tables and 557 fields. The current version 2.3 was frozen on August 11, 2011 and contains 54 tables and 740 fields. The major changes in 2.3 include support for alpha and coincidence spectrometry and restructuring the sample and calibration tables for better applicability. The differences between the versions 1.1 and 2.3 are further described in the release notes.

### 1.1 The Database System

The *Linssi* database system consists of the database and the necessary scripts needed to do the updates and administer the database. An increasing number of query scripts is also provided. The database is described in detail in this manual. The scripts provided with the database are mostly self documenting and are briefly described in the accompanying Scripts and Interfaces manual [3].

A typical user environment is displayed in Fig. 1.1, where the database and script layers form the *Linssi* distribution and the upper layers are commercial or in-house software. User interfaces with the system via a graphical user interface (GUI). The GUI to scripts is web-based and the analysis software has its own GUI. Most of the reports can be directly created using SQL queries but the analysis software has also a report generating possibility independent from *Linssi*. The analysis software - in our case UNISAMPO[4], SHAMAN[5], Aatami[6] and others - is interfaced with the update and query scripts either using temporary files, which are documented in the script manual, or calling the database directly via ODBC calls.

The amount of data processed in a laboratory performing gamma-ray spectrum analysis can be quite large and very heterogenous. It contains everything from the measured spectra to obscure notes on analysts' log books. Our intention has not been to store everything in the *Linssi* database. However, the idea has been to include all the *relevant* information starting from the collection of radioactivity to a sample all the way to the final analysis results and conclusions made by the laboratory experts. Clear emphasis has been on the spectrum analysis results and on the information directly affecting the quality of these results. The

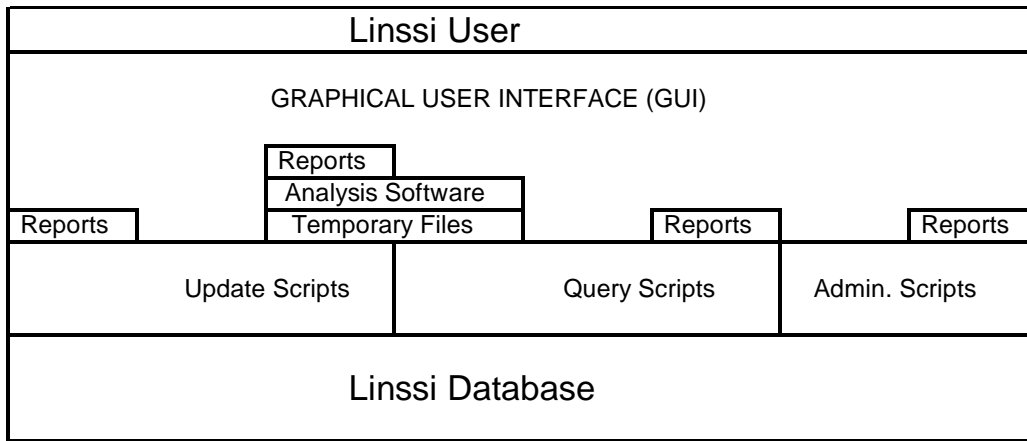


Figure 1.1: How *Linssi* interfaces with the user and analysis software.

information is meant to be complete enough to allow laboratory certification and thus also to facilitate outside review on the quality of the results based on the information available in the *Linssi* database. In case information not directly available in the database is needed, there should be enough pointers in *Linssi* to identify where the information might be available. In most cases that means information on the facility responsible for sample manufacturing and, possibly, responsible for its measurement.

The number of fields in the analysis related tables is more than 300. That is quite a lot and we do not expect that they are all needed in every application. However, these are the generic fields readily available from our analysis software and with modern computers the overhead due to unused database fields is negligible.

We have made every effort to provide a database with information that is generic, i.e., not depending on the specific software used. An example of this type of information is nuclide activity. The activity itself does not depend on the analysis software used, even though different software provides different values. On the other hand, there exist tens of different peak shape models, for example. We have provided a relatively flexible model, but there certainly exist software for which our definitions do not apply without user modifications.

The *Linssi* design assumes three main entry points to the system (Fig. 1.2). They are sample production (**entry point 1**), sample measurement (**entry point 2**), and spectrum analysis (**entry point 3**). In addition there is the **entry point 0** denoting the facility group of tables (Ch. 2), which can be updated relatively independently from the other table groups. In the time line of the analysis we assume that the facilities have existed forever.

In **entry point 1** we start from the production or collection of activity to form the sample. That can be done in a multitude of ways. Currently (version 2.3) we have defined four different sample types, calibration samples (Ch. 3.1), basic samples (Ch. 3.2), air filter samples (Ch. 3.3), and CTBT laboratory samples (Ch. 3.4). The CTBT samples are quite specific and the calibration samples are just calibration samples. Basic samples, on the other hand, are quite generic and can be applied to many sample collection modes. If that is not sufficient new production table groups can be defined. This is illustrated with the yet non-existent irradiation group of tables in Fig. 1.2. Properties common to all sample types are defined in Ch. 4.

If we receive a sample to be measured we start from **entry point 2**, i.e., from preparing the source and measuring it (Ch. 6).

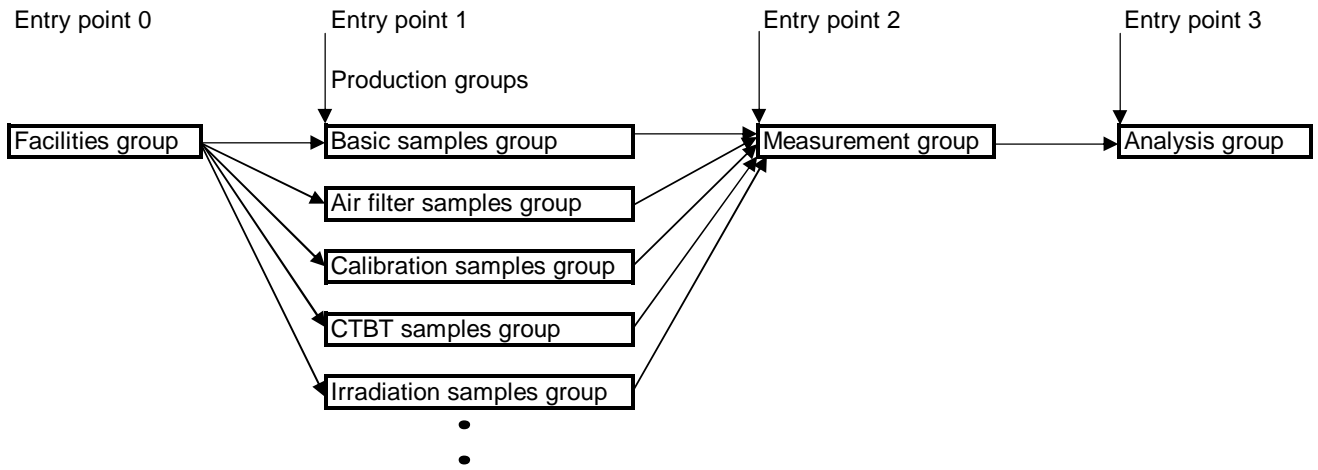


Figure 1.2: *Linssi* entry points and major table groups.

Finally, if we receive a gamma-ray spectrum for analysis, we start from **entry point 3**, i.e., send the spectrum directly to the analysis pipeline (Ch. 8). At this stage, in order to do proper peak analysis and nuclide identification, calibrations must be available (Ch. 5)

If our laboratory controls the whole chain of events from **entry point 0** to **entry point 3** the database is updated as we go along. On the other hand, if the later entry points are applied the necessary data must be provided from outside and stored to the previous tables of the chain. Scripts are provided for each of these entry points. The entry points also define the most important keys in the *Linssi* database: `idSample` in table 4.1 `samples`, `idMeas` in table 6.6 `measurements`, and `idAnalysis` in table 8.1 `analyses` identifying the sample, its measurements and its analyses, respectively. Due to the database design at least dummy samples and measurements must be provided in order to store the analysis results in the analysis group of tables.

## 1.2 A Stroll between the Tables

Let's take an introductory stroll between 31 of the 54 tables of *Linssi*. Their lay-out is shown in Fig. 1.3. In the figure only the table names and key persons sitting in the table are shown. The arrows are drawn from the foreign keys to the primary keys they are referencing. Note that a key can simultaneously be both primary and foreign.

As can be seen, almost half of the tables (21–27, 42–49)<sup>a</sup> concentrate on gamma-ray spectrum analysis. The number of fields is also greatest in these tables. Spectrum measurement is covered by tables 28–33 and different facilities, including sample production, by tables 1–8. The different sample production processes are covered in the tables 9–12, i.e., in basic samples group (Ch. 3.2), air filter samples group (Ch. 3.3), calibration samples group (Ch. 3.1), and CTBT lab samples group (Ch. 3.4).

Starting from table 3 we can find the weather reigning outside the facility, which itself is defined in table 1. The facility may, for example, be the place where the radioactivity is collected to the sample or where the sample manufacturing takes place. If the facility is mobile its positions are stored in table 2.

Obviously a sample can be produced in many different methods. Currently we have defined tables for basic samples, calibration samples, air filter samples and CTBT laboratory samples.

<sup>a</sup>All these tables are not shown in the figure.

These groups will be described in the subsequent chapters.

After sampling the sample arrives to a measurement facility, which may, or may not, be situated at the sampling facility. Table 17 characterizes the sample at this point. The sample is further processed into the actual source geometry, table 28, to be put on the spectrometer. From the sample it is possible to produce different sources, for example by using different containers. As long as the radionuclide content is not changed we are dealing with the same sample. It has just been transformed to a new source. Now the source is measured in a measurement setup described in table 32. The setup uses a detector, shield and attenuator described in tables 29, 30 and 31, respectively. Important information of the measurements themselves and, most importantly, the measurement results, i.e., gamma-ray-spectra are stored in tables 33–34.

The measurements are followed by analyses. General information on analyses is stored in table 42. Analysis itself relies on calibrations in tables 21–27. The traditional gamma-ray spectrum peak analysis results are stored in table 44. They are followed by the identification and activity calculation results, tables 45 and 46, respectively.

Minimum detectable activities and other activity limits are stored in table 47 and activity ratios for relevant nuclides in table 48. Finally the reviewed final results are stored in table 49. This table contains analysts' comments and pointers to the best analysis results.

Other tables not shown in the figure include support for alpha and coincidence spectrometry, Regions-of-Interest measurements and analysis, spectral component analysis, function definitions and metadata, among others.

It should be emphasized that a sample can be measured multiple times and that the resulting spectra can be also analyzed any number of times. The results of all measurements and analyses can be stored in the database.

## 1.3 Tables

In the following chapters each database table is described in detail. This is done in the form of a table followed by explanation of the independent fields. In the table header we show first the name of the database table. The column **Field** gives the name of the database column. The column **Type** defines the data type of the field i.e. whether it is character, integer, float, etc. The column **Length** gives the maximum size of the field in bytes. For variable size fields a zero is printed. The column **Flags** gives information of the relational status of the field. The available flags are:

- P Primary key or a component of the primary key.
- A The field is automatically incremented every time a new entry is made.
- F Foreign key or a component of the foreign key.
- # Number (1,2,...). Combined fields are numbered. Numbers are used to mark combined foreign keys, combined indices and combined unique fields.
- S Self-referencing key that points to a record in the same table.
- U The field content must be unique within this table.
- I The field is indexed for faster reference.
- N The field must contain a value, i.e., it must not be NULL.
- R,r The field contains only reserved alphanumeric values. The reserved values are described for the field marked R. In fields marked r pointers to fields R are provided.
- C,c The field has a defined naming convention. The naming convention is documented in the fields marked C. In fields marked c pointers to fields C are provided.

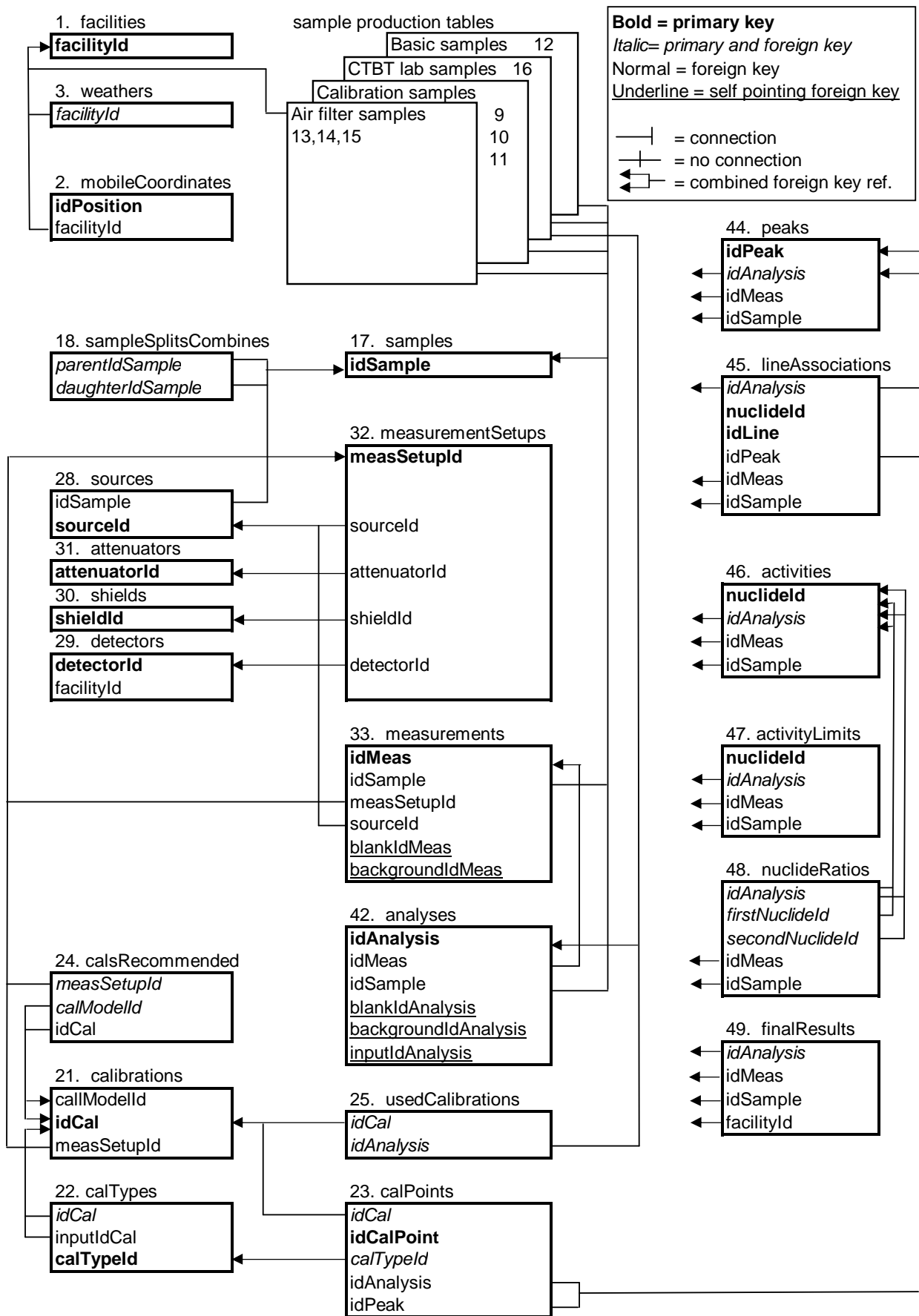


Figure 1.3: Table lay-out of the *Linssi* restaurant.

Since the reserved fields are continuously updated, an up-to-date list is maintained at <http://linssi.hut.fi/>.

Primary keys may be composed of multiple fields. In that case, all included fields are marked as primary. A set forming the complete primary key must be unique. Primary keys are always indexed and that is not explicitly marked. If, however, any individual component of the primary key is indexed, it is marked with I.

Since there may be many foreign keys in a table, flag F is not enough to define the components of combined foreign keys. For that purpose each combined foreign key is numbered using the flag #, i.e., F1, F2, ... The same applies for combined indices and combined unique fields.

## 1.4 Units and Default Values

We use only SI units and units outside the SI that are accepted for use with the SI, and thus consistent with the recommendations of the International Committee for Weights and Measures (CIPM, Comité International des Poids et Mesures) [8]. It is strongly recommended that users of *Linssi* follow the unit conventions of this manual. If different units are needed locally, the conversion should be performed in the interfacing scripts. In this manual the unit to be used is given in square brackets after the field definition, e.g., [mm].

All dates and times (`datetime`, `timestamp`) are in Universal Time Coordinated (UTC).

The uncertainties in *Linssi* are given as one sigma absolute uncertainties. If, for example, uncertainty per cent is needed, it should be calculated from the value and its absolute uncertainty when reading the database.

*Linssi* implementation does not include default values to fields. The action to be taken due to unknown field values should always be left to the user, i.e., to the software or analyst inputting or outputting data to or from *Linssi*. To facilitate this practice unknown values should always be set NULL. That is automatically true for fields that are not filled, i.e., have been left empty. Note, however, that there are fields, which must contain a value. They are marked with flag N.

## 1.5 Naming Conventions

The table and field (column) names are written in lower case, e.g. `measurements`. If a name is composed of multiple words the words are separated by capitalizing their first letters, e.g. `sampleType`. To avoid excessive length the names may be abbreviated, e.g. `sampleCond-FlagArrival`.

There are special conventions applicable to keys<sup>b</sup> and fields with one to one correspondence to keys. Other fields are not allowed to use the syntax of keys. The rules are:

1. The keys of type integer have the prefix `id`, e.g. `idSample`.
2. The keys of non-integer type end with `Id`, e.g. `sampleId`.
3. The name of the foreign keys must be identical to the corresponding primary keys. There are some exceptions where this is not feasible:

---

<sup>b</sup>A key here denotes a field that is a primary or foreign key, or a part of a combined primary or foreign key.

- Self-reference. A record may contain a key pointing to another record of the same table. In this case the key name is formed from the primary key by adding a prefix. E.g., the key `blankIdMeas` is this way formed from the primary key `idMeas`.
  - Multiple foreign keys. A record may have multiple foreign keys pointing to the same primary key. The prefixing is again used, e.g., two nuclides pointing to the primary key `nuclideId` are `firstNuclideId` and `secondNuclideId`.
4. The name of a unique non-integer field, with one-to-one correspondence with the primary integer key of the table, is formed by moving the prefix to the end of the name, e.g. from the primary `idSample` we get the name `sampleId` for the unique field. If the primary key and the unique field are both integers, or non-integers, the name of the unique field must be formed by prefixing. We see no reason for them being of the same type, however.

Note that in MySQL the names of the tables are case sensitive whereas the names of the fields are not. In *Linssi* the field names are unique regardless of case sensitivity. However, in order to be safe all the field and table names used must be typed following the conventions of this manual. In that way readability of the names is also enhanced.





# Chapter 2

## Facilities and Action Sites

The facility group of tables of *Linssi* establishes the **entry point 0** of the database. Sample production, for example, takes place at the facility, where the sample manufacturing equipments are situated. Other types of facilities may include a moving laboratory with samplers and measuring facilities, or even a weather station. Facilities may also be laboratories performing measurements or analysis.

Since the group aims to support any kind of facility, it is quite generic. Basically a facility is defined by its name (`facilityId`), purpose (`facilityType`) and its contact information. It may be mobile and then its position is given in table `mobileCoordinates`. For support of weather stations we have tables `weathers` and `weatherParameters`.

Action sites (`actionSites`) are less permanent than facilities and cannot be mobile.

Tables `Missions` and `MissionDevices`, which define *in situ* measuring or sampling campaigns, also belong to this group of tables.

### 2.1 Facilities

Table 2.1: Facilities

facilities			
Field	Type	Length	Flags
<code>facilityId</code>	<code>varchar</code>	80	PN
<code>facilityName</code>	<code>varchar</code>	80	U
<code>facilityType</code>	<code>varchar</code>	20	R
<code>isMobile</code>	<code>Boolean</code>	1	
<code>isWeather</code>	<code>Boolean</code>	1	
<code>isSampling</code>	<code>Boolean</code>	1	
<code>isMeasuring</code>	<code>Boolean</code>	1	
<code>isAnalyzing</code>	<code>Boolean</code>	1	
<code>sendsSamples</code>	<code>Boolean</code>	1	
<code>receivesSamples</code>	<code>Boolean</code>	1	
<code>sendsMessages</code>	<code>Boolean</code>	1	
<code>receivesMessages</code>	<code>Boolean</code>	1	
<code>organization</code>	<code>varchar</code>	40	

*continued on next page ...*

... continued from previous page

Field	Type	Length	Flags
department	varchar	40	
street	varchar	40	
town	varchar	40	
state	varchar	40	
zip	varchar	40	
country	varchar	40	
location	varchar	40	
contactPerson	varchar	40	
telephone	varchar	40	
fax	varchar	40	
email	varchar	40	
longitude	double	8	
latitude	double	8	
altitude	double	8	
comments	text	0	

end of table.

It is advisable to create a new facility with new `facilityId` after important changes in facility properties take place. In this way old samples, for example, can still be associated with the facility with original properties that may be significant for interpretation of the analysis results.

<code>facilityId</code>	Unique name of the facility.
<code>facilityName</code>	Common name of the facility.
<code>facilityType</code>	Type of the facility. This is very application dependent. The defined values are: P: particulate sampling facility. WEATHER: weather station.
<code>isMobile</code>	TRUE if the facility is mobile. In that case the coordinates of the facility can be found in table 2.2 <code>mobileCoordinates</code> .
<code>isWeather</code>	TRUE if the facility is providing weather information to <i>Linssi</i> .
<code>isSampling</code>	TRUE if the facility is providing sampling information to <i>Linssi</i> .
<code>isMeasuring</code>	TRUE if the facility is providing measurement results to <i>Linssi</i> .
<code>isAnalyzing</code>	TRUE if the facility is providing analysis results to <i>Linssi</i> .
<code>sendsSamples</code>	TRUE if the facility is sending samples to <i>Linssi</i> facility. See tables 4.3 <code>sampleTransports</code> and 4.4 <code>sampleTrackings</code> .
<code>receivesSamples</code>	TRUE if the facility is receiving samples from <i>Linssi</i> facility. See tables 4.3 <code>sampleTransports</code> and 4.4 <code>sampleTrackings</code> .
<code>sendsMessages</code>	TRUE if the facility is sending messages to <i>Linssi</i> facility. See table 10.2 <code>messages</code> .
<code>receivesMessages</code>	TRUE if the facility is receiving messages from <i>Linssi</i> facility. See table 10.2 <code>messages</code> .
<code>organization</code>	Organization running the facility.
<code>department</code>	Department running the facility.
<code>street</code>	Facility street address.
<code>town</code>	Town where the facility is situated.

state	State where the facility is situated.
zip	Facility zip code.
country	Country where the facility is situated.
location	General location of the facility.
contactPerson	Name of the contact person of the facility.
telephone	International telephone number of the facility.
fax	International fax number of the facility.
email	Email address of the facility.
longitude	Longitude of the stationary facility.
latitude	Latitude of the stationary facility.
altitude	Altitude of the stationary facility.
comments	Comments in English [en].

## 2.2 Mobile Coordinates

Table 2.2: Mobile coordinates of facilities

mobileCoordinates			
Field	Type	Length	Flags
<code>idPosition</code>	int	4	PA
<code>idSeconds</code>	int	4	U1N
<code>idNanoSeconds</code>	int	4	U1N
<code>positionSource</code>	varchar	20	U1NR
<code>facilityId</code>	varchar	80	F
<code>positionType</code>	varchar	10	R
<code>project</code>	varchar	40	
<code>positionTime</code>	datetime	8	I
<code>coordSystem</code>	varchar	20	R
<code>speed</code>	double	8	
<code>heading</code>	double	8	
<code>numSatellites</code>	int	4	
<code>xCoordinate</code>	double	8	
<code>yCoordinate</code>	double	8	
<code>altitude</code>	double	8	
<code>hDOP</code>	double	8	R
<code>vDOP</code>	double	8	R
<code>pDOP</code>	double	8	R
<code>age</code>	double	8	
<code>fix</code>	int	4	R
<code>comments</code>	text	0	
(facilityId ) → facilities(facilityId)			

*end of table.*

This table tracks spacetime coordinates of mobile facilities, `facilityId`. In a single mission, for example, we may have many mobile facilities having multiple devices providing continuous stream of coordinates. In addition to primary key `idPosition`, we need to have `UNIQUE(idSeconds, idNanoSeconds, positionSource)` in order to facilitate tracking of the facilities in the case of non-synchronous arrival of the coordinates.

<code>idPosition</code>	Auto incrementing position identifier. Note that, in addition to providing a unique primary key, it also provides the storage time order of the positions.
<code>idSeconds</code>	Unix second timestamp ( <code>tv_sec</code> of the <code>struct timespec</code> ). Provides the <b>time identifier</b> together with <code>idNanoSeconds</code> below.
<code>idNanoSeconds</code>	Unix nanosecond timestamp ( <code>tv_nsec</code> of the <code>struct timespec</code> ), Provides extra precision to the position identifier. Needed mainly in cases with rapid succession of short measurements in sub-second scale. Can be set to zero in most applications.
<code>positionSource</code>	Name of the source providing the coordinates. The facility may have multiple GPS navigators, for example.
<code>facilityId</code>	See table <a href="#">2.1 facilities</a> .

<code>positionType</code>	The following position types are defined: <b>NULL</b> : This is a raw unprocessed position. <b>ROUTE</b> : A processed and validated point on facility's route. <b>GAP</b> : Before this position there is a gap in position information. GPS may have been off or lost its connection with satellites, for example. <b>KEEP</b> : Keep this position information. If the position is continuously updated this table may grow very large. In order to be able to discard unimportant positions the <code>positionType</code> of the important data can be set to <b>KEEP</b> . This could be done during sampling or measuring, for example.
<code>project</code>	Name of the project.
<code>positionTime</code>	Date and time of the coordinate information. Unlike ( <code>idSeconds</code> , <code>idNanoSeconds</code> ) this is human readable.
<code>coordSystem</code>	Name of the coordinate system in use. It is strongly recommended to use only the default <code>WGS84_LLA</code> -coordinate system, i.e., to leave this field <b>NULL</b> .  <b>NULL</b> or <code>WGS84_LLA</code> : WGS84 datum geographical coordinates. <code>WGS84_UTM_35V</code> : WGS84 datum projected coordinates to Universal Transverse Mercator grid zone 35V. <code>FINNISH_KKJ3</code> : Finnish national datum projected coordinates to Finnish national grid (KKJ) zone 3. <code>FINNISH_LLA</code> : Finnish national datum geographical coordinates. <code>EUREF_XYZ</code> : European reference ellipsoid datum cartesian ECEF coordinates.  ECEF above refers to Cartesian Earth-Centered Earth-Fixed coordinates in x, y, z in meters [m]. The z-axis defined as being parallel to the earth rotational axis, pointing towards north and the x-axis intersects the earth at the 0° latitude, 0° longitude; LLA to geographical coordinates in x = longitude and y = latitude in degrees [°], and z = altitude from mean sea level in meters [m]; KKJ to projected coordinates in x = northing, y = easting and z = altitude from mean sea level in meters [m]; and UTM to projected coordinates in x = easting, y = northing and z = altitude from mean sea level in meters [m]:
<code>speed</code>	Facility speed in meters per second [m/s].
<code>heading</code>	Direction the facility is heading in degrees [°](0...360), 0° = North.
<code>numSatellites</code>	Number of satellites the GPS navigator sees.
<code>xCoordinate</code>	x-coordinate or longitude of the mobile facility.
<code>yCoordinate</code>	y-coordinate or latitude of the mobile facility.
<code>altitude</code>	z-coordinate or altitude of the mobile facility.
<code>hDOP</code>	Precision class of dilution of precision of horizontal coordinates.
<code>vDOP</code>	Precision class of dilution of precision of vertical coordinate.
<code>pDOP</code>	Precision class of dilution of precision of position.
<code>age</code>	Age of the information in seconds [s].
<code>fix</code>	The method of fixing the position: 0 = invalid, 1 = GPS, 2 = DGPS, 3 = PPS, 4 = RTK, 5 = float RTK, 6 = estimated, 7 = manual, 8 = simulation.
<code>comments</code>	Comments in English [en].

## 2.3 Weather

Table 2.3: Weather at the facility

weathers			
Field	Type	Length	Flags
facilityId	varchar	80	PFN
weatherSource	varchar	20	PN
idPar	int	4	PFN
weatherStartId	datetime	8	PIN
contactPerson	varchar	40	
weatherEnd	datetime	8	
parValue	double	8	
comments	text	0	

(facilityId ) → facilities(facilityId)  
(idPar ) → weatherParameters(idPar)

*end of table.*

Table **weathers** contains information about the weather conditions at the facility. The most important weather parameters can be continuously monitored. The table is preferably filled on-line even though that is not required. The values are the averages over the report period.

<b>facilityId</b>	See table <a href="#">2.1 facilities</a> .
<b>weatherSource</b>	Origin of the weather information. For example, weather station model name or weather model name for forecasts
<b>idPar</b>	Weather parameter identifier. See table <a href="#">2.4 weatherParameters</a> .
<b>weatherStartId</b>	The starting datetime of the weather report period.
<b>contactPerson</b>	Contact person for the weather station.
<b>weatherEnd</b>	The ending datetime of the weather report period.
<b>parValue</b>	Value of the measured/modelled weather parameter during the reporting period.
<b>comments</b>	Comments in English [en].

## 2.4 Weather Parameters

Table 2.4: Weather Parameters

weatherParameters			
Field	Type	Length	Flags
idPar	int	4	PAN
parName	varchar	40	R
parDescription	text	0	
parUnit	varchar	20	
parForm	varchar	20	R
parType	varchar	40	R
period	double	8	
periodUnit	varchar	20	
comments	text	0	

*end of table.*

Table `weatherParameters` defines the parameters used for reporting the weather conditions at the facility. Due to varying ways to represent the parameters we have chosen a generic representation of the parameters where the quantities and their units can be freely defined. This makes the input to *Linssi* easy, but, on the other hand, the interpretation of the retrieved parameter values more difficult.

NOTE: The representation of the fields and their units is against the standard database design practices. In this case we have followed the practice of Vaisala weather stations.

<code>idPar</code>	Auto-incrementing identification number for the weather parameter definition.
<code>parName</code>	Weather parameter name, e.g. temperature, humidity, precipitation, wind speed, wind direction.
<code>parDescription</code>	Freehand description of the parameter.
<code>parUnit</code>	Unit the parameter is given in, e.g. C for degrees Celsius.
<code>parForm</code>	Additional definition for the measured/modelled parameter, e.g. <code>parForm = snow</code> for <code>parName = precipitation</code> .
<code>parType</code>	Method of determining the value of the parameter, e.g. maximum, average (during reporting period), cumulative
<code>period</code>	Elapsed measuring or model time used to obtain the cumulative value for cumulative weather parameter type ( <code>parType = cumulative</code> ). The period usually starts at midnight.
<code>periodUnit</code>	Unit in which the period is given in.
<code>comments</code>	Comments in English [en].

## 2.5 Source Receptor Sensitivity (SRS)

Table 2.5: Source Receptor Sensitivity (SRS)

SRS			
Field	Type	Length	Flags
<b>facilityId</b>	varchar	80	PF
<b>dateId</b>	datetime	8	P
<b>file</b>	varchar	255	
<b>model</b>	varchar	20	
<b>comments</b>	text	0	
(facilityId ) → facilities(facilityId)			

*end of table.*

The results for the SRS-model for the monitoring station **facilityId** are stored in the file **file**. For every measurement taken at a monitoring station SRS-fields can be computed with backward runs of a Lagrangian Particle Diffusion Model (LPDM). These SRS-fields can be combined with emission inventory yielding backward reconstructed concentration values pertaining to these measurements. Source Receptor Sensitivity (SRS) files contain data in a standardized (World Meteorological Organisation) format. The data portion of the file contains information such as: the location of grid points (latitude and longitude), the time period and the station sensitivity values at the grid points. A complete description of the contents of an SRS file is located at <http://ctbto4.ctbto.org/atm/communication+procedures.html>. SRS files can be interpreted and visualised by using a program such as the CTBTO's Web-grape.

<b>facilityId</b>	Name of the facility for which a SRS (source receptor sensitivity) file is calculated
<b>date</b>	Date and time at end of sample collection
<b>file</b>	Pointer to location of associated SRS file in filesystem
<b>model</b>	Meteorological model used for calculation of SRS file
<b>comments</b>	Comments in English [en].



## 2.6 Missions

Table 2.6: Missions

missions			
Field	Type	Length	Flags
<code>missionId</code>	<code>varchar</code>	80	P
<code>missionStart</code>	<code>datetime</code>	8	I
<code>missionEnd</code>	<code>datetime</code>	8	
<code>comments</code>	<code>text</code>	0	

*end of table.*

Table `Missions` is used in initializing a named mission. A mission has in its use a set of facilities and devices. See table [2.7 missionDevices](#).

<code>missionId</code>	Name of the mission.
<code>missionStart</code>	Start date and time of the mission (UTC).
<code>missionEnd</code>	End date and time of the mission (UTC).
<code>comments</code>	Comments in English [en].

## 2.7 Mission Devices

Table 2.7: Mission devices

missionDevices			
Field	Type	Length	Flags
<code>missionId</code>	<code>varchar</code>	80	PF
<code>facilityId</code>	<code>varchar</code>	80	PIF
<code>deviceId</code>	<code>varchar</code>	80	PI
<code>deviceType</code>	<code>varchar</code>	80	R
<code>comments</code>	<code>text</code>	0	
<code>(missionId ) → missions(missionId)</code>			
<code>(facilityId ) → facilities(facilityId)</code>			

*end of table.*

Table `missionDevices` contains a list of devices used during the mission. Each mission can have in its use one or more facilities. Typically they are mobile units. Each unit can have in its disposal multiple devices - typically detectors, samplers, etc. Each unit and device has to have a name, so the measurements of each detector, for example, can be traced afterwards.

`missionId` Name of the mission.  
`facilityId` Name of the facility. The facility used in missions is typically mobile. See table [2.1 facilities](#).  
`deviceId` Name of a device used in this facility during this mission.  
`deviceType` Type of the device.  
`comments` Comments in English [en].

## 2.8 Action Sites

Table 2.8: Action sites

actionSites			
Field	Type	Length	Flags
<code>idActionSite</code>	int	4	PA
<code>idSample</code>	int	4	IF
<code>idMeas</code>	int	4	IF
<code>siteName</code>	varchar	80	
<code>street</code>	varchar	40	
<code>town</code>	varchar	40	
<code>state</code>	varchar	40	
<code>zip</code>	varchar	40	
<code>country</code>	varchar	2	
<code>floor</code>	varchar	8	
<code>room</code>	varchar	40	
<code>location</code>	text	0	
<code>coordSystem</code>	varchar	20	r
<code>xCoordinate</code>	double	8	
<code>yCoordinate</code>	double	8	
<code>zCoordinate</code>	double	8	
<code>comments</code>	text	0	

(`idSample` ) → `samples(idSample)`  
 (`idMeas` ) → `measurements(idMeas)`

*end of table.*

Action site is a site where sample production, e.g., collection or measurements are done. Facilities are normally used for these purposes. However, they are more or less permanent, large and few in number, while action sites are typically temporary locations where *in situ* sample collection or measurements are performed, and can be large in number.

<code>idActionSite</code>	A unique auto incrementing identifier of the action site.
<code>idSample</code>	Identification number of the sample. See table 4.1 <code>samples</code> . NULL, if site is a measurement site.
<code>idMeas</code>	Identification number of the measurement. See table 6.6 <code>measurements</code> . NULL, if site is a sample production site.
<code>siteName</code>	Name of the action site.
<code>street</code>	Action site street address.
<code>town</code>	Town where the action site is situated.
<code>state</code>	State where the action site is situated.
<code>zip</code>	Action site zip code.
<code>country</code>	Country where the action site is situated.
<code>floor</code>	Floor level of the action site according to the local practice (e.g. P).
<code>room</code>	Room number or name of the action site.
<code>location</code>	Detailed description of the location of the action site.

<code>coordSystem</code>	Name of the coordinate system system in use. See field <code>coordSystem</code> in table <a href="#">2.2 mobileCoordinates</a> .
<code>xCoordinate</code>	x-coordinate of the action site.
<code>yCoordinate</code>	y-coordinate of the action site.
<code>zCoordinate</code>	z-coordinate of the action site.
<code>comments</code>	Comments in English [en].

# Chapter 3

## Sample Types

Gamma-ray spectrum analysis can be based on a measured spectrum alone, and requiring only rudimentary information of the sample. However, it is most useful to have further information of the sample production or manufacturing process, i.e., the process of creating or collecting radioactive material so as to form a radioactive sample. The manufacturing process itself can be quite different for different samples. The tables of this chapter describe the available manufacturing processes producing different types of samples in *Linssi*, namely calibration samples, basic samples, air filter samples, and CTBT laboratory samples.

Producing a sample into *Linssi* requires creating the primary key `idSample` in table `4.1 samples`, i.e., for the **entry point 1** in *Linssi*. No measurements or analyses can be made without this entry. However, if the sample production process is not known, or if the user finds the rudimentary sample information provided in table `samples` sufficient for his purposes, the detailed production steps described in this chapter can be skipped. The production process of the sample can then be added afterwards, if a need arises.

### 3.1 Calibration Samples

Calibration samples are identified with `sampleProductionTable` equal to `calibrationSamples` in table `4.1 samples`. Also the selection of the specific sample production process is described in more detail there.

Calibration samples are samples with known activity and they are preferably certified. The certification is given in the table `calibrationSamples`, the information of the actual nuclides in the sample is given in the table `calibrationNuclides`, and the specific nuclear data in the table `calibrationLibraries`.

#### 3.1.1 Calibration Samples

Table 3.1: Calibration samples

calibrationSamples			
Field	Type	Length	Flags
<code>idSample</code>	int	4	PFN
<code>certificateNumber</code>	varchar	20	

*continued on next page ...*

... continued from previous page

Field	Type	Length	Flags
<code>productCode</code>	<code>varchar</code>	20	
<code>sourceNumber</code>	<code>varchar</code>	20	
<code>manufacturer</code>	<code>varchar</code>	40	
<code>receiptTime</code>	<code>datetime</code>	8	
<code>overallAct</code>	<code>double</code>	8	
<code>comments</code>	<code>text</code>	0	
<code>(idSample ) → samples(idSample)</code>			

*end of table.*

This table gives the basic identification data of calibration samples.

<code>idSample</code>	Identification number of the calibration sample. See table 4.1 samples.
<code>certificateNumber</code>	Certificate number of the calibration sample manufacturer.
<code>productCode</code>	Manufacturer product code of the calibration sample.
<code>sourceNumber</code>	Manufacturer serial number of the calibration sample.
<code>manufacturer</code>	Name of the manufacturer.
<code>receiptTime</code>	Receipt date and time of the calibration sample.
<code>overallAct</code>	Overall activity of the calibration sample at <code>receiptTime</code> . This value is NOT certified.
<code>comments</code>	Comments in English [en].

### 3.1.2 Calibration Nuclides

Table 3.2: Calibration nuclides

calibrationNuclides			
Field	Type	Length	Flags
idSample	int	4	PFN
calNuclideId	varchar	10	PIF1Nc
idCalibrationLibrary	int	4	F1
activity	double	8	
uncActivity	double	8	
assayTime	datetime	8	
comments	text	0	

(idSample ) → samples(idSample)  
(calNuclideId ,idCalibrationLibrary ) → calibrationLibraries(calNuclideId,idCalibrationLibrary)

*end of table.*

This table contains information of the activity content of each nuclide of the calibration sample.

idSample	Identification number of the calibration sample. See table 4.1 samples.
calNuclideId	Name of the calibration nuclide. See table 3.3 calibrationLibraries.
idCalibrationLibrary	Identification number of the library of nuclear data for calibration nuclides. See table 3.3 calibrationLibraries.
activity	Certified activity of the nuclide in becquerels [Bq].
uncActivity	One sigma absolute uncertainty of activity in becquerels [Bq].
assayTime	The assay date and time of the nuclide, i.e. the date and time of the activity.
comments	Comments in English [en].

### 3.1.3 Calibration Libraries

Table 3.3: Calibration libraries

calibrationLibraries			
Field	Type	Length	Flags
idCalibrationLibrary	int	4	PN
calNuclideId	varchar	10	PINC
idCalLine	int	4	PN
calLineEnergy	double	8	
uncCalLineEnergy	double	8	
emissionProb	double	8	
uncEmissionProb	double	8	
halflife	double	8	
uncHalflife	double	8	
calibrationLibraryId	varchar	20	
comments	text	0	

*end of table.*

This table contains gamma line data for calibration samples. Even though these data are fundamental constants, and thus one library should suffice, the table includes a possibility for multiple libraries. This may be advantageous for different calibration purposes where the useful gamma lines may differ.

Due to limited number of calibration sources available in a laboratory this table should not become too big. This is further emphasized by the fact that, unlike identification libraries, a calibration library usually contains data of the prominent gamma lines only.

idCalibrationLibrary	Identification number of the library.
calNuclideId	Name of the calibration nuclide. The adopted syntax for <code>nuclideId</code> is <code>Ba-137m</code> , where <code>m</code> , <code>n</code> , <code>o</code> , ... denote metastable states of increasing energy. If metastable state energies are not known <code>a</code> , <code>b</code> , <code>c</code> , ... may be used. Note that analysis software may use different naming conventions. It is up to the database administrator to ascertain that consistent syntax is used. <i>Linssi</i> does not include a comprehensive nuclide reference library.
idCalLine	Index of the gamma line of the calibration nuclide. Sorted in an ascending order of the line energy starting from 1.
calLineEnergy	Energy of the gamma line of the calibration nuclide in kiloelectronvolts [keV].
uncCalLineEnergy	One sigma absolute uncertainty of <code>calLineEnergy</code> in kiloelectronvolts [keV].
emissionProb	Absolute emission probability of the gamma line, i.e., number of gammas emitted per decay.
uncEmissionProb	Absolute one sigma uncertainty of <code>emissionProb</code> .
halflife	Half-life of the nuclide in seconds [s].
uncHalflife	Absolute one sigma uncertainty of half-life seconds[s].
calibrationLibraryId	Name of the calibration library.
comments	Comments in English [en].



## 3.2 Basic Samples

Basic samples are identified with `sampleProductionTable` equal to `basicSamples` in table 4.1 `samples`. Also the selection of sample production model is described in some detail there. This table is aimed for defining generic basic samples. This sample production model is used for the samples that require more information than is available in table 4.1 `samples`, but do not have their own specific production model defined. For this general purpose production model it has been necessary to relax database rules stating that all the fields should be exactly defined. Thus the table contains unit definitions, which we generally have tried to avoid. Also the field names are flexible and generic. This makes the semantics of the table very dependent on the conventions used by each organization filling this table.

Table 3.4: Basic samples

basicSamples			
Field	Type	Length	Flags
<code>idSample</code>	int	4	PF
<code>sampleCommission</code>	varchar	20	R
<code>sampleClass</code>	varchar	20	R
<code>sampleGroup</code>	varchar	20	R
<code>sampleSubGroup</code>	varchar	20	R
<code>sampleDefinition</code>	varchar	40	
<code>samplingType</code>	varchar	8	R
<code>sampleAmount1</code>	double	8	
<code>uncSampleAmount1</code>	double	8	
<code>sampleAmount1Unit</code>	varchar	8	
<code>sampleAmount2</code>	double	8	
<code>uncSampleAmount2</code>	double	8	
<code>sampleAmount2Unit</code>	varchar	8	
<code>yield</code>	double	8	
<code>uncYield</code>	double	8	
<code>samplingLocation</code>	varchar	40	
<code>samplingLocationDesc</code>	varchar	40	
<code>samplingZ1coord</code>	double	8	
<code>samplingZ2coord</code>	double	8	
<code>alternateQuantity</code>	double	8	
<code>alternateQuantityUnit</code>	varchar	8	
<code>refTime</code>	datetime	8	
<code>resultsDestination</code>	varchar	80	
<code>comments</code>	text	0	
(idSample ) → samples(idSample)			

*end of table.*

`idSample` A unique identification number of the sample. See table 4.1 `samples`.

`sampleCommission` Sample context type (e.g. research, surveillance, service). Naming convention as used in the facility operating the database.

sampleClass	Name of the sample class according to sample grouping hierarchy used in the facility.
sampleGroup	Name of the sample group according to sample grouping hierarchy used in the facility.
sampleSubGroup	Name of the sample subgroup according to sample grouping hierarchy used in the facility.
sampleDefinition	Specific sample definition or description.
samplingType	Type of the sampling.
sampleAmount1	Total amount of sample received or collected.
uncSampleAmount1	Uncertainty of sample amount.
sampleAmount1Unit	Unit of sample amount.
sampleAmount2	Alternate or complementary sample amount, e.g. dry weight or sample collector area.
uncSampleAmount2	Uncertainty of alternate sample amount.
sampleAmount2Unit	Unit of alternate sample amount.
yield	Collection yield of the sample, e.g. for CTBT NG sampling, Xe gas in sample/total Xe gas sampled (fraction).
uncYield	uncertainty of the yield.
samplingLocation	Name of the sampling point or sampling area.
samplingLocationDesc	More specific definition or description of the sampling point or area.
samplingZ1coord	Horizontal coordinate of the sampling point, e.g. minimum sampling height or the minimum depth of sample slice.
samplingZ2coord	Horizontal coordinate of the sampling point, e.g. max. sampling height or max. depth the sample slice.
alternateQuantity	Alternate or complementary amount of measured sample (source), e.g. dry weight or sample collector area.
alternateQuantityUnit	Unit of alternate or complementary amount of measured sample (source).
refTime	Reference date and time at which the activities are calculated.
resultsDestination	Forwarding address in data transfer between databases.
comments	Comments in English [en].

### 3.3 Air Filter Samples

Air filter samples are identified with `sampleProductionTable` equal to `airFilterSamples` in table 4.1 `samples`. Also the selection of sample production model is described in some detail there.

Air filter sampling is covered by the three tables of the air filter samples group. The table `samplers` defines the ‘static’ properties of the sampler. The table `airFilterSamples` gives the information of the sampling specific to a given sample. Finally, the table `airFilterSOH` contains information of the State-Of-Health of the sampler as a function of time. This information is continuous in nature, i.e., not specific to a sample even though `idSample` is provided in the table in order to facilitate on-line following of its sampling.

#### 3.3.1 Air Filter Samples

Table 3.5: Air filter samples

airFilterSamples			
Field	Type	Length	Flags
<code>idSample</code>	int	4	PFN
<code>samplerId</code>	varchar	80	F
<code>samplerOnTime</code>	double	8	
<code>airVolume1</code>	double	8	
<code>presDiff1Start</code>	double	8	
<code>presDiff1End</code>	double	8	
<code>airVolume2</code>	double	8	
<code>presDiff2Start</code>	double	8	
<code>presDiff2End</code>	double	8	
<code>sampleSent</code>	datetime	8	
<code>correctedToNTP</code>	Boolean	1	
<code>comments</code>	text	0	
(idSample ) → samples(idSample)			
(samplerId ) → samplers(samplerId)			

*end of table.*

In this table, fields have been reserved for two channels in a sampler. This gives a possibility to use two types of filters simultaneously, typically active carbon and fiber. See also table 3.6 `samplers`.

<code>idSample</code>	See table 4.1 <code>samples</code> .
<code>samplerId</code>	This is the identifier of the sampler used. See table 3.6 <code>samplers</code> .
<code>samplerOnTime</code>	Time the sampler was on during collection in seconds [s].
<code>airVolume1</code>	Air volume sampled through channel 1 in cubic meters [m <sup>3</sup> ].
<code>presDiff1Start</code>	Air pressure difference in pascals [Pa] over the flow rate meter in channel 1 at <code>collStart</code> . For <code>collStart</code> see table 4.1 <code>samples</code> .
<code>presDiff1End</code>	Air pressure difference in pascals [Pa] over the flow rate meter in channel 1 at <code>collEnd</code> . For <code>collEnd</code> see table 4.1 <code>samples</code> .
<code>airVolume2</code>	Air volume sampled through channel 2 in cubic meters [m <sup>3</sup> ].

<code>presDiff2Start</code>	Air pressure difference in pascals [Pa] over the flow rate meter in channel 2 at <code>collStart</code> . For <code>collStart</code> see table 4.1 samples.
<code>presDiff2End</code>	Air pressure difference in pascals [Pa] over the flow rate meter in channel 2 at <code>collEnd</code> . For <code>collEnd</code> see table 4.1 samples.
<code>sampleSent</code>	Date and time the sample was sent for measurement.
<code>correctedToNTP</code>	If TRUE, air volumes are corrected to Normal Temperature and Pressure (NTP), i.e., corrected to 20°C and 101,325.0 Pa.
<code>comments</code>	Comments in English [en].

### 3.3.2 Samplers

Table 3.6: Samplers

samplers			
Field	Type	Length	Flags
samplerId	varchar	80	PN
facilityId	varchar	80	F
contactPerson	varchar	40	
samplerType	varchar	20	R
flowFactor1	double	8	
flowPower1	double	8	
flowFactor2	double	8	
flowPower2	double	8	
lastMaintenance	datetime	8	
nextMaintenance	datetime	8	
documentDir	varchar	255	
comments	text	0	
(facilityId ) → facilities(facilityId)			

*end of table.*

Samplers consist of two air channels, 1 and 2, where total flow is obtained as a sum of these channels. In both channels there is a flow meter. The flow meters are described by two parameters flow factor  $F$  and flow power  $c$ . These parameters can be used for orifice plates, flow-nozzles or multi-orifice probes, or with any device for which two parameters are sufficient, only the method of flow rate calculation will differ. In our application the flow rate,  $f$ , is calculated from  $f = F \times p^c$ , where  $F$  is the flow factor in cubic meters per second [m<sup>3</sup>/s],  $c$  is the flow power in watts [W], and  $p$  is the pressure difference over the measuring device in pascals [Pa]. For  $p$  see also table 3.5 `airFilterSamples`.

<code>samplerId</code>	Name of the sampler.
<code>facilityId</code>	Identifier of the facility where the sampler is situated. See table 2.1 <code>facilities</code> .
<code>contactPerson</code>	Contact person for the sampling. See table 2.1 <code>facilities</code> .
<code>samplerType</code>	Name of the sampler type.
<code>flowFactor1</code>	Flow factor in cubic meters per second [m <sup>3</sup> /s] for flow meter in channel 1.
<code>flowPower1</code>	Flow power in channel 1 in watts [W].
<code>flowFactor2</code>	Flow factor in cubic meters per second [m <sup>3</sup> /s] for flow meter in channel 2.
<code>flowPower2</code>	Flow power in channel 2 in watts [W].
<code>lastMaintenance</code>	Date and time of the last maintenance of this sampler.
<code>nextMaintenance</code>	Scheduled date and time of the next maintenance of this sampler.
<code>documentDir</code>	The directory, or URI (Universal Resource Identifier), where the documentation of the sampler can be found.
<code>comments</code>	Comments in English [en].

### 3.3.3 Sampling State-of-Health Data

Table 3.7: State-of-health data (SOH) for air filter sampling process

airFilterSOH			
Field	Type	Length	Flags
samplerId	varchar	80	PF
samplingHealthTimeId	datetime	8	PN
idSample	int	4	PN
pressDiff1	double	8	
airVolume1	double	8	
flowRate1	double	8	
pressDiff2	double	8	
airVolume2	double	8	
flowRate2	double	8	
correctedToNTP	Boolean	1	
comments	text	0	
(samplerId ) → samplers(samplerId)			

*end of table.*

samplerId	See table <a href="#">3.6 samplers</a> .
samplingHealthTimeId	Date and time of SOH-data.
idSample	See table <a href="#">4.1 samples</a> .
pressDiff1	Air pressure difference in pascals [Pa] over the flow rate meter in channel 1.
airVolume1	Total volume of air sampled in channel 1 since <code>collStart</code> in cubic meters [m <sup>3</sup> ]. For <code>collStart</code> see table <a href="#">4.1 samples</a> .
flowRate1	Air flow rate in channel 1 in [m <sup>3</sup> /s] at this moment.
pressDiff2	Air pressure difference in pascals [Pa] over the flow rate meter in channel 2.
airVolume2	Total volume of air sampled in channel 2 since <code>collStart</code> in cubic meters [m <sup>3</sup> ]. For <code>collStart</code> see table <a href="#">4.1 samples</a> .
flowRate2	Air flow rate in channel 2 in [m <sup>3</sup> /s] at this moment.
correctedToNTP	If TRUE, air volumes and flow rate are corrected to Normal Temperature and Pressure (NTP), i.e., corrected to 20 °C and 101,325.0 Pa.
comments	Comments in English [en].

### 3.4 CTBT Laboratory Samples

CTBT laboratory samples are identified with `sampleProductionTable` equal to `ctbtLabSamples` in table 4.1 `samples`. Also the selection of sample production model is described in some detail there.

This table stores information about CTBT samples sent to radionuclide laboratories within the framework of the Comprehensive Nuclear-Test-Ban Treaty. The table is very specific and for the full details of all the fields, except `idSample`, the user is asked to consult the report IDC3.4.1Rev6 [9] to which the descriptions below are referencing. The reference RLR/#xxx refers to the data block #xxx of the radionuclide laboratory report (RLR). RLR and the related data blocks are described in Chapter ‘Radionuclide Laboratory Reports’ of the report IDC3.4.1Rev6 [9].

Table 3.8: CTBT laboratory samples

ctbtLabSamples			
Field	Type	Length	Flags
<code>idSample</code>	int	4	PFN
<code>siteCode</code>	varchar	20	R
<code>priority</code>	varchar	10	R
<code>sampleCategory</code>	char	1	R
<code>analysisPurpose</code>	text	0	R
<code>testsAuthorized</code>	text	0	
<code>specialInstructions</code>	text	0	
<code>facilityActCat</code>	varchar	40	R
<code>facilitySamDiam</code>	double	8	
<code>facilitySamThick</code>	double	8	
<code>facilitySamWidth</code>	double	8	
<code>facilitySamMass</code>	double	8	
<code>facilityContDens</code>	double	8	
<code>facilityContThick</code>	double	8	
<code>facilityContMat</code>	varchar	80	R
<code>facilitySamGeomDescr</code>	varchar	80	R
<code>facilitySplitMass</code>	double	8	
<code>idcNuclNotQuant</code>	text	0	
<code>idcNaturalNucl</code>	text	0	
<code>idcActProd</code>	text	0	
<code>idcFissProd</code>	text	0	
<code>idcEventActProdPresent</code>	char	1	
<code>idcEventDaysSinceLastAct</code>	varchar	10	
<code>idcEventOnlyOneFP</code>	char	1	
<code>idcEventDaysSinceLastFP</code>	varchar	10	
<code>idcEventMultFP</code>	char	1	
<code>idcEventDaysSinceLastMultFP</code>	varchar	10	
<code>idcEventCs137Present</code>	char	1	
<code>idcEventTimesCs137InMonth</code>	varchar	10	
<code>test</code>	varchar	40	

*continued on next page ...*

... continued from previous page

Field	Type	Length	Flags
comments	text	0	
(idSample ) → samples(idSample)			

end of table.

idSample	Identification number of the sample measured. See table 4.1 samples.
siteCode	Site code of the laboratory selected for analysis. See RLR/#Header.
priority	Priority level: Urgent or Routine. See RLR/#Header.
sampleCategory	Sample category: A - network quality control sample, B - sample from IMS network categorized by IDC as level 5, C - proficiency test sample, D - station backup sample (measured by a lab), or, E - other. See RLR/#Header.
analysisPurpose	Purpose of the analysis. See RLR/#Objective.
testsAuthorized	Free text describing the tests authorized. See RLR/#Objective.
specialInstructions	Free text describing any special instructions. See RLR/#Objective.
facilityActCat	Activity category of the sample according to international shipping regulations. See RLR/#StationSample.
facilitySamDiam	Sample diameter or length in millimeters [mm]. See RLR/#StationSample.
facilitySamThick	Sample thickness in millimeters [mm]. See RLR/#StationSample.
facilitySamWidth	Sample width in millimeters [mm]. See RLR/#StationSample.
facilitySamMass	Sample mass in kilograms [g]. See RLR/#StationSample.
facilityContDens	Sample container density in kilograms per cubic meter [kg/m <sup>3</sup> ]. See RLR/#StationSample.
facilityContThick	Sample container thickness in millimeters [mm]. See RLR/#StationSample.
facilityContMat	Sample container material. See RLR/#StationSample.
facilitySamGeomDescr	Sample geometry description. See RLR/#StationSample.
facilitySplitMass	Mass of the split that gets to the lab in kilograms [g]. See RLR/#Split.
idcNuclNotQuant	List of nuclides identified but not quantified by the IDC. See RLR/#g_IDCActivitySummary.
idcNaturalNucl	List of natural nuclides quantified by the IDC including their half-lives, concentrations [Bq/m <sup>3</sup> ] and absolute one sigma of uncertainties of concentrations. For the exact format see RLR/#g_IDCActivitySummary.
idcActProd	List of activation products quantified by the IDC including their half-lives, concentrations [Bq/m <sup>3</sup> ] and absolute one sigma of uncertainties of concentrations. For the exact format see RLR/#g_IDCActivitySummary.
idcFissProd	List of fission products quantified by the IDC including their half-lives, concentrations [Bq/m <sup>3</sup> ] and absolute one sigma of uncertainties of concentrations. For the exact format see RLR/#g_IDCActivitySummary.



<code>idcEventActProdPresent</code>	Activation products present in the sample [Y/N]. See RLR/#g_IDCEventScreeningFlags.
<code>idcEventDaysSinceLastAct</code>	Number of days since last activation product seen. See RLR/#g_IDCEventScreeningFlags.
<code>idcEventOnlyOneFP</code>	Only one fission product in the sample [Y/N] See RLR/#g_IDCEventScreeningFlags.
<code>idcEventDaysSinceLastFP</code>	Number of days since last fission product seen. See RLR/#g_IDCEventScreeningFlags.
<code>idcEventMultFP</code>	Two or more fission products in the sample [Y/N]. See RLR/#g_IDCEventScreeningFlags.
<code>idcEventDaysSinceLastMult</code>	Number of days since two or more fission products seen. See RLR/#g_IDCEventScreeningFlags.
<code>idcEventCs137Present</code>	Cs-137 present in the sample [Y/N]. See RLR/#g_IDCEventScreeningFlags.
<code>idcEventTimesCs137InMonth</code>	Number of times Cs-137 was seen in the last 30 days. See RLR/#g_IDCEventScreeningFlags.
<code>test</code>	Type of test performed. See RLR/#Test.
<code>comments</code>	Comments in English [en].



# Chapter 4

## Common Sample Properties

One of the core tables of *Linssi* is the table `samples`, which gives the basic information of the sample. This is the **entry point 1** to *Linssi*. No measurements or analyses can be made without corresponding entries of `idSample` in table 4.1 `samples`. It gives sufficient information of a sample for most measuring and analysis purposes. More complete data on samples, including their production and collection processes, are available in Ch. 3. A specific sample type, i.e., production process is selected with the field `sampleProductionTable` equal to the name of the sample type. See table 4.1 `samples` below.

Transport of samples between facilities can be followed using table `sampleTransports`. Correspondingly their movements inside a facility can be tracked in table `sampleTrackings`.

Samples can also be formed by splitting and combining other samples. The split and combined sources can be tracked to the original sources using the information in table `sampleSplitsCombines`.

### 4.1 Samples

Table 4.1: Sample description at arrival

samples			
Field	Type	Length	Flags
<code>idSample</code>	int	4	PA
<code>sampleId</code>	varchar	80	UNC
<code>phdSampleName</code>	varchar	80	C
<code>extSampleName</code>	varchar	80	
<code>sampleDescription</code>	varchar	80	
<code>facilityId</code>	varchar	80	F
<code>contactPerson</code>	varchar	40	
<code>splitSymbol</code>	varchar	8	R
<code>split</code>	Boolean	1	
<code>combined</code>	Boolean	1	
<code>sampleType</code>	varchar	20	R
<code>sampleProductionTable</code>	varchar	80	
<code>collStart</code>	datetime	8	I
<code>collEnd</code>	datetime	8	

*continued on next page ...*

... continued from previous page

Field	Type	Length	Flags
collTime	double	8	
quantity	double	8	
unit	varchar	8	
overallAct	double	8	
barcode	text	0	
RFID	text	0	
dbEntryTime	datetime	4	
dbLastUpdate	timestamp	4	
comments	text	0	
(facilityId ) → facilities(facilityId)			

end of table.

This table describes a sample when it arrives for measurement. This is also the first entry point of the sample to the database, i.e., this record must always be the first record created for the sample. In many cases the only information at the time of creation may be the `sampleId`.

<code>idSample</code>	A unique auto incrementing identification number of the sample.
<code>sampleId</code>	A unique sample identifier. For the <i>Linssi</i> naming convention see scripts and interfaces manual [3].
<code>phdSampleName</code>	A sample reference identifier (SRID) string following the naming convention of the CTBTO/PTS [9]. Even though not required by the database this string is likely to be globally unique. The naming convention is also described in <i>Linssi</i> scripts and interfaces manual [3].
<code>extSampleName</code>	A sample identifier string following the naming convention of a customer for the analysis. Even though not required by the database this string is meant to be unique.
<code>sampleDescription</code>	Short text describing the sample.
<code>facilityId</code>	Identifier of the facility that produced the sample. See table 2.1 facilities.
<code>contactPerson</code>	Contact person for the sample production. See table 2.1 facilities.
<code>splitSymbol</code>	A two part (Pp) split symbol of the sample. The symbol is used to identify the descendants of the original sample when it is split for multiple counting and analysis purposes. The first half, P, gives the split number, the second half, p, gives the total number of splits. The split identifier is thus 11 for all radionuclide samples before any splitting is performed. If, for example, a sample is split into three parts, the parts are assigned the following split identifiers: 13 for the first piece, 23 for the second piece, and 33 for the third piece. If all the sample splits are combined together, the split number P is set equal to p+1. Therefore, if the three sample splits from the previous example are combined together the split identifier reported is 43.

This definition is a generalization of the one described in Ref. [9]. The split symbol is useful for tagging simple splits. For more complex cases of splitting and combining samples table 4.2 `sampleSplitsCombines` should be used. If both the table `sampleSplitsCombines` and `splitSymbol` are used, it is up to the user to make sure that they are consistent.

<code>split</code>	TRUE if the sample is formed by splitting other samples. For further information see table 4.2 <code>sampleSplitsCombines</code> .
<code>combined</code>	TRUE if the sample is formed by combining other samples. For further information see table 4.2 <code>sampleSplitsCombines</code> .
<code>sampleType</code>	Type of the sample. The following types are reserved: <code>air-filter</code> , <code>calibration</code> , <code>environmental</code> , <code>qcspectrum</code> , <code>blank</code> , <code>background</code> , <code>gassample</code> , <code>gasbackground</code> .
<code>sampleProductionTable</code>	Name of the table containing the production information of this sample. Currently tables <code>basicSamples</code> , <code>airFilterSamples</code> , <code>calibrationSamples</code> , and <code>ctbtLabSamples</code> have been defined (Ch. 3). In generic terms these tables are used to describe the sample manufacturing process, i.e., the process whose result is a radioactive sample. If the specific tables for your application are not available, it is possible to set <code>sampleProductionTable</code> to NULL. In that case the sample is described by the tables described in this chapter (Ch. 4). However, if more information of the sample production is required table <code>basicSamples</code> is the next obvious choice. Finally, a new production model could be implemented for your application.
<code>collStart</code>	Date and time of the start of the sample collection.
<code>collEnd</code>	Date and time of the end of the sample collection.
<code>collTime</code>	Sample collection time in seconds [s].
<code>quantity</code>	The amount sampled. The specific activity of the sample can be obtained by dividing its activity in [Bq] by <code>quantity×unit</code> . See also note on specific activity on p. 98.
<code>unit</code>	The unit of <code>quantity</code> .
<code>overallAct</code>	Approximate overall activity in becquerels [Bq]. Knowledge is useful when deciding the measurement setup and for radioprotection purposes.
<code>barcode</code>	Bar code on the sample.
<code>RFID</code>	Radio Frequency IDentification a.k.a. RFID-tag on the sample.
<code>dbEntryTime</code>	Date and time of entering this record to the database.
<code>dbLastUpdate</code>	Date and time of last update of this record.
<code>comments</code>	Comments in English [en].

## 4.2 Splitting and Combining Samples

Table 4.2: Splitting and combining samples

sampleSplitsCombines			
Field	Type	Length	Flags
parentIdSample	int	4	PFN
daughterIdSample	int	4	PIFN
activityBranching	double	8	
method	vchar	20	R
comments	text	0	
(parentIdSample ) → samples(idSample)			
(daughterIdSample ) → samples(idSample)			

*end of table.*

A sample may be formed by splitting and combining other samples. In many cases it is enough to know simple sample properties, i.e., its history is irrelevant. However, if we want to know how the sample has been formed from other samples, it can be tracked with table `sampleSplitsCombines`.

This table is used together with table 4.1 `samples` where Boolean fields `split` and `combine` tell how the sample is formed. Those fields are used to inform that the sample has been formed by splitting/combining, but for tracking parents and daughters this table is the only thing needed.

<code>parentIdSample</code>	Identification number of the parent sample. See table 4.1 <code>samples</code> .
<code>daughterIdSample</code>	Identification number of the daughter sample. See table 4.1 <code>samples</code> .
<code>activityBranching</code>	Denotes the fraction of activity inherited from the parent to the daughter. If we sum <code>activityBranching</code> over all <code>daughterIdSample:s</code> while keeping <code>parentIdSample</code> fixed the result must be 1.0. <i>Note:</i> <code>activityBranching</code> is only informative and should not be needed to obtain the activity of the daughters. The metrics of each sample shall be available in its production table.
<code>method</code>	Method of splitting or combining used to form the daughter from the parent(s).
<code>comments</code>	Comments in English [en].

## 4.3 Sample Transport

Table 4.3: Transport record

sampleTransports			
Field	Type	Length	Flags
idSample	int	4	PFN
transportOutId	Boolean	1	PN
facilityId	varchar	80	F
contactPerson	varchar	40	
courierCompany	varchar	40	
dateTimeOfHandover	datetime	8	
eta	datetime	8	
airwayBill	varchar	40	
sealNumber	varchar	20	
sampleCondition	text	0	R
packCondition	text	0	R
sealCondition	text	0	R
sampleCondFlag	char	1	R
comments	text	0	
(idSample ) → samples(idSample)			
(facilityId ) → facilities(facilityId)			

*end of table.*

This table tracks the movement of samples between facilities. It is adopted from the practices of the CTBTO. The short description of the fields below, however, should be self explanatory for any facility designing its own practices.

The references to IDC3.4.1Rev6 [9] below are only needed by the certified laboratories of the CTBTO. The reference LABSDN/#Transport refers to the #Transport data block of the LABSDN message giving a notification that a sample has been sent to a laboratory. Same data blocks are also used in the TECSDN message giving a notification that a sample has been sent from the laboratory. LABSDN and TECSDN messages together with their data blocks are described in Chapter ‘Other Laboratory Messages’ of the report IDC3.4.1Rev6 [9].

idSample	Identification number of the sample measured. See table 4.1 samples.
transportOutId	True if this is a transport out of the facility logging the transport, i.e. for CTBT laboratories, message TECSDN. See LABSDN/#Transport.
facilityId	Name of the remote facility. See table 2.1 facilities.
contactPerson	Contact person at the remote facility. See table 2.1 facilities.
courierCompany	Agent handling the transport.
dateTimeOfHandover	Date and time of handover (UTC).
eta	Estimated date and time of arrival (UTC).
airwayBill	Airway bill or freight declaration identifier.
sealNumber	Seal number or identification.
sampleCondition	Condition of the sample at the time of handover.

<code>packCondition</code>	Condition of the package at the time of handover.
<code>sealCondition</code>	Condition of the seal at the time of handover.
<code>sampleCondFlag</code>	Flag indicating if there are any problems with the sample at the time of handover (facility dependent).
<code>comments</code>	Comments in English [en].



## 4.4 Sample Tracking

Table 4.4: Sample tracking

sampleTrackings			
Field	Type	Length	Flags
idSample	int	4	PFN
dateTimeInId	datetime	8	PN
dateTimeOut	datetime	8	
barcode	text	0	
RFID	text	0	
location	varchar	40	
handledBy	varchar	20	
comments	text	0	

(idSample ) → samples(idSample)

*end of table.*

This table is used to track the movements of samples inside a facility.

idSample	See table <a href="#">4.1</a> samples.
dateTimeInId	Date and time of the sample arrival to the location (UTC).
dateTimeOut	Date and time of the sample departure from the location (UTC).
barcode	Barcode of the sample.
RFID	Radio Frequency IDentification a.k.a. RFID-tag on the sample.
location	Name of the location.
handledBy	Name of the person who transferred the sample to the location or is responsible for the sample at the location.
comments	Comments in English [en].



# Chapter 5

## Calibrations

Calibration tables `calibrations`, `calTypes`, `calPoints`, `calsRecommended` and `calsUsed` give the full calibration of each measurement setup. The full calibration is based on sets of calibration points (table `calPoints`) traceable to calibration results of possibly multiple analyses (table `analyses`), which in turn can be based on multiple measurements (table `measurements`) of multiple calibration samples (table `calibrationSamples`). Each set of calibration points defines a type of sub calibration defined in the corresponding table `calTypes`. Each sample analysis may use one or more full calibrations (table `calibrations`) that are tracked by table `calsUsed`. Finally the recommended calibration for each measurement setup and calibration model can be found in table `calsRecommended`.

For accurate spectrum analysis data on background and blank measurements are important. Data of the recommended background for each measurement setup and background model are available in table `backgroundsRecommended`. Corresponding data of the blank can be found in `blanksRecommended`.

### 5.1 Calibrations

Table 5.1: Calibrations

calibrations			
Field	Type	Length	Flags
<code>idCal</code>	<code>int</code>	4	PAN
<code>measSetupId</code>	<code>varchar</code>	80	FN
<code>calModelId</code>	<code>varchar</code>	20	R
<code>calId</code>	<code>varchar</code>	80	UNC
<code>blankIdAnalysis</code>	<code>int</code>	4	F
<code>backgroundIdAnalysis</code>	<code>int</code>	4	F
<code>comments</code>	<code>text</code>	0	
( <code>measSetupId</code> ) → <code>measurementSetups(measSetupId)</code>			
( <code>blankIdAnalysis</code> ) → <code>analyses(idAnalysis)</code>			
( <code>backgroundIdAnalysis</code> ) → <code>analyses(idAnalysis)</code>			

*end of table.*

<code>idCal</code>	Identification number of the full calibration. The full calibration consists of multiple calibrations further defined in table 5.2 <code>cal-Types</code> .
<code>measSetupId</code>	Unique name of the measurement setup, to which this calibration can be applied. See table 6.5 <code>measurementSetups</code> .
<code>calModelId</code>	Identifier of the calibration model. This is often specific to the software used for analysis and calibration.
<code>calId</code>	Unique name describing the calibration. For the <i>Linssi</i> naming convention see scripts and interfaces manual [3].
<code>blankIdAnalysis</code>	Identification number of the analysis of the blank used in producing this calibration. See table 8.1 <code>analyses</code> .
<code>backgroundIdAnalysis</code>	Identification number of the analysis of the background used in producing this calibration. See table 8.1 <code>analyses</code> .
<code>comments</code>	Comments in English [en].

## 5.2 Calibration Types

Table 5.2: Calibration types

calTypes			
Field	Type	Length	Flags
idCal	int	4	PFN
calTypeId	varchar	20	PINR
inputIdCal	int	4	FS
changed	Boolean	1	
class	varchar	20	R
description	varchar	40	
creationTime	datetime	8	
calInfo	text	0	
calCertificate	text	0	
facilityId	varchar	80	F
contactPerson	varchar	40	
startOfRange	double	8	
endOfRange	double	8	
idFunction	int	4	F
parameters	blob	0	
comments	text	0	
(idCal ) → calibrations(idCal) (inputIdCal ) → calibrations(idCal) (facilityId ) → facilities(facilityId) (idFunction ) → functions(idFunction)			

*end of table.*

Each complete calibration, `idCal`, in *Linssi* consists of calibrations of more than one parameter of the measurement setup. Calibrations of these individual parameters are identified by the calibration type identifiers, `calTypeId`, in tables `calPoints` and `calTypes`. The exact meaning and support of each calibration type depends on the analysis software. The currently reserved calibration types are described below.

<code>idCal</code>	Identification number of the full calibration. See table 5.1 <code>calibrations</code> .
<code>calTypeId</code>	Name of the type of calibration. See below for discussion of the reserved <code>calTypeId</code> 's.
<code>inputIdCal</code>	Pointer to the calibration that is the basis for this calibration. Using this pointer the calibration chain can be tracked to the original calibration. This key is a self referencing foreign key, i.e., it points to another record in this table.
<code>changed</code>	This field is true if the calibration ( <code>idCal,calTypeId</code> ) has been changed when compared with the calibration ( <code>inputIdCal,calTypeId</code> ).
<code>class</code>	Additional information on the calibration chain. The following values have been defined:

PHD: External calibration information received together with the spectrum.

EXTERNAL: External calibration information of unspecified origin.

SOH: A State Of Health calibration update generated by the spectrum analysis software.

FIT: A (new) fit to the calibration points by the spectrum analysis software.

SOURCE: A corrected calibration due to source geometry. That is, the spectrum analysis software has made a correction to the calibration based on the differences in the measuring geometry between the source defined in the measurement setup and that actually measured. The correction may include source self-attenuation correction, source volume correction, source distance correction etc. See field `sourceId` in table 6.6 `measurements`. For the corrections actually performed the software documentation must be consulted.

<code>description</code>	An identifier given by the analyst to the calibration (possibly unique, not required).
<code>creationTime</code>	Date and time of creation of the calibration. <i>Note:</i> A valid date and time must be given. If old calibrations are searched for, this field is useful as an identifier.
<code>calInfo</code>	Free form description of the calibration by the analyst.
<code>calCertificate</code>	Calibration certificate block as described in Ref. [9]. This is an alternative certificate of the calibration. The preferred method is described in Ch. 3.1. If both methods are used for the same calibration, they must be consistent.
<code>facilityId</code>	Name of the facility where the calibration has been performed. See table 2.1 <code>facilities</code> .
<code>contactPerson</code>	Name of the person responsible for the calibration.
<code>startOfRange</code>	Start of the validity range of the calibration function in channels [ch], or in kiloelectronvolts [keV] for efficiency calibrations.
<code>endOfRange</code>	End of the validity range of the calibration function in channels [ch], or in kiloelectronvolts [keV] for efficiency calibrations.
<code>idFunction</code>	Function number of the calibration function. See table 9.1 <code>functions</code> .
<code>parameters</code>	Values of the parameters of <code>idFunction</code> . The values are given as pairs of parameter value and its one sigma absolute uncertainty. Each pair is separated with the newline character. If the parameter uncertainty is not known it must not be given. It is recommended that known zero uncertainties, which is the case for integers, for example, should be explicitly stated. The pairs are associated with the function by the order given in the function definition.
<code>comments</code>	Comments in English [en].

## 5.2.1 Peak Shape Calibrations

The current version of *Linssi* reserves the following `calTypeId`'s for Gaussian peak shape with extended exponential tails and a baseline step: `width`, `lowTail`, `highTail`, `lowTailExp`, `highTailExp` and `step`. These parameters are given as a function of channel in *Linssi*. For the exact meaning of the parameters see UNISAMPO, SHAMAN and *Aatami* manuals [4, 5, 6].

## 5.2.2 Energy Calibration

For energy calibration of photopeak centroids `calTypeId energy` is reserved. Here energy is given in kiloelectronvolts [keV] as a function of channel.

## 5.2.3 Efficiency Calibration

For detector efficiency calibration two `calTypeId`'s are reserved: `efficiency` for the photopeak efficiency and `totalEfficiency` for the total efficiency of the detector. Both efficiencies are defined as a function of energy in kiloelectronvolts [keV]. The efficiencies are absolute and describe the full measurement setup.

## 5.2.4 Calibration Functions

In *Linssi* the measurement setup calibration functions are used in tables 5.2 `calTypes`. The functions are fitted to points given in table 5.3 `calPoints`. In the functions the dependent variable  $y$  is defined as a function of one independent variable  $x$  with varying number of fitted parameters  $a_i$ .  $x$  is given either in kiloelectronvolts [keV] for efficiency calibration or in channels for all other calibration functions. The fitted parameters are stored in `parameters` together with their uncertainties. The available functions are defined in table 9.1 `functions`. In principle any function can be applied for any calibration type. In practice, however, function 1 may be used for any calibration, functions 2..4 are used for energy and shape calibrations, and functions 5..9 are used only for efficiency calibration. The user is advised to consult his analysis software manuals for physical interpretation and support for these functions.

## 5.3 Calibration Points

Table 5.3: Calibration points

calPoints			
Field	Type	Length	Flags
idCal	int	4	PF1N
calTypeId	varchar	20	PIF1Nr
idCalPoint	int	4	PN
idAnalysis	int	4	I1F2
idPeak	int	4	I1F2
xValue	double	8	
uncXValue	double	8	
yValue	double	8	
uncYValue	double	8	
comments	text	0	

(idCal ,calTypeId ) → calTypes(idCal,calTypeId)  
(idAnalysis ,idPeak ) → peaks(idAnalysis,idPeak)

*end of table.*

This table contains calibration point values and their uncertainties given as quadruplets (xValue, uncXvalue, yValue, uncYValue).

It is not mandatory to have all the calibration points associated with spectrum analysis results. For those calibrations idAnalysis = NULL. This might be the situation in the case of off-line calibrations. It is, however, recommended that the user assigns an entry in the analyses table even for these cases. The entry and the corresponding peaks table provide ample space also for justifying the off-line calibration.

idCal	Identification number of the full calibration. The full calibration consists of multiple calibrations identified by idCal.calTypeId pairs. See table 5.2 calTypes.
calTypeId	Name of the type of calibration. See Ch. 5.2 for discussion of the reserved calTypeId's.
idCalPoint	Index of the calibration point. For each idcal.calTypeId the numbering starts from 1.
idAnalysis	Identification number of the analysis results used to obtain this calibration point. See table 8.1 analyses.
idPeak	Identification number of spectrum peak for this calibration point. See table 8.3 peaks.
xValue	Channel, or energy [keV] for efficiency calibration, of the calibration point.
uncXValue	The one sigma absolute uncertainty of the xValue.
yValue	Value of the calibrated parameter at xValue.
uncYValue	The one sigma absolute uncertainty of the yValue.
comments	Comments in English [en].



## 5.4 Recommended Calibrations

Table 5.4: Recommended calibrations

calsRecommended			
Field	Type	Length	Flags
measSetupId	varchar	80	PFN
calModelId	varchar	20	PNr
idCal	int	4	FN
comments	text	0	
(measSetupId ) → measurementSetups(measSetupId)			
(idCal ) → calibrations(idCal)			

*end of table.*

It is a good practice to have a recommended calibration for each measurement setup and calibration model. `idCal` of that most up to date calibration is given in this table.

<code>measSetupId</code>	Name of the measurement setup. See table 6.5 <code>measurementSetups</code> .
<code>calModelId</code>	Identifier of the calibration model. This is often specific to the software used for analysis and calibration.
<code>idCal</code>	Identification number of the full calibration recommended to be used together this measurement setup and calibration model. See table 5.1 <code>calibrations</code> .
<code>comments</code>	Comments in English [en].

## 5.5 Used Calibrations

Table 5.5: Used calibrations

calsUsed			
Field	Type	Length	Flags
<code>idAnalysis</code>	int	4	PFN
<code>idCal</code>	int	4	PIFN
<code>usedInAnalysis</code>	Boolean	1	N
<code>comments</code>	text	0	
(idAnalysis ) → analyses(idAnalysis)			
(idCal ) → calibrations(idCal)			

*end of table.*

This table supports many-to-many relationships between analysis results and calibrations. A single full calibration, `idCal`, may, of course, be used in many analyses, `idAnalysis`. It is also possible that a single analysis takes advantage of many calibrations. For illustration, the analysis software may receive a calibration with the spectrum, `calTypes.class = PHD`, but the analyst selects to use the recommended calibration for the measurement setup in question: the fields `idcal`, `measSetupId` and `calModelId` all match in tables `calibrations` and `calsRecommended`. Finally, the software fine tunes the calibration for source geometry differences and uses the adjusted calibration to obtain the final results and stores it as `calTypes.class = SOURCE` to database. Note that only the last calibration is used to calculate the final results, i.e., `usedInAnalysis` will be TRUE for that calibration only. For the other calibrations of our example `usedInAnalysis` will be FALSE. However, after the analysis, at the latest, all the three above mentioned calibrations are stored in *Linssi*.

For information about the `class` field see table 5.2 `calTypes`.

<code>idAnalysis</code>	Identification number of the analysis, which has used, or has considered to use this calibration. See table 8.1 <code>analyses</code> .
<code>idCal</code>	Identification number of the full calibration. See table 5.1 <code>calibrations</code> .
<code>usedInAnalysis</code>	TRUE if this is the calibration on which the analysis results in <code>idAnalysis</code> are based on.
<code>comments</code>	Comments in English [en].

## 5.6 Recommended Backgrounds

Table 5.6: Recommended backgrounds

backgroundsRecommended			
Field	Type	Length	Flags
measSetupId	varchar	80	PF
backgroundModelId	varchar	20	PR
idAnalysis	int	4	IF
idMeas	int	4	IF
comments	text	0	
(measSetupId ) → measurementSetups(measSetupId)			
(idAnalysis ) → analyses(idAnalysis)			
(idMeas ) → measurements(idMeas)			

*end of table.*

In order to be able to do background subtraction, a background measurement and its analysis results must be available for each measurement setup. Since different analysis methods use different background models, compatibility of the results is indicated by the appropriate model identifier.

measSetupId	A unique identifier of the measurement setup. See table <a href="#">6.5 measurementSetups</a> .
backgroundModelId	A unique identifier of the background model used in background analysis
idAnalysis	Identification number of the analysis of the background measurement. See table <a href="#">8.1 analyses</a>
idMeas	Identification number of the background measurement. See table <a href="#">6.6 measurements</a> .
comments	Comments in English [en].

## 5.7 Recommended Blanks

Table 5.7: Recommended blanks

blanksRecommended			
Field	Type	Length	Flags
measSetupId	varchar	80	PF
blankModelId	varchar	20	PR
idAnalysis	int	4	IF
idMeas	int	4	IF
idSample	int	4	IF
comments	text	0	
(measSetupId ) → measurementSetups(measSetupId)			
(idAnalysis ) → analyses(idAnalysis)			
(idMeas ) → measurements(idMeas)			
(idSample ) → samples(idSample)			

*end of table.*

In order to be able to do blank subtraction a blank measurement and its analysis results must be available for each measurement setup. Since different analysis methods use different blank models compatibility of the results is indicated by the appropriate model identifier.

measSetupId	A unique identifier of the measurement setup. See table 6.5 measurementSetups.
blankModelId	A unique identifiers of the blank model used in the analysis of the blank.
idAnalysis	Identification number of the analysis of the blank measurement. See table 8.1 analyses
idMeas	Identification number of the blank measurement. See table 6.6 measurements.
idSample	Identification number of the blank sample measured. See table 4.1 samples.
comments	Comments in English [en].

# Chapter 6

## Measurements

Measurements of sample radioactivity are done in a measuring setup (`measurementSetups`), which combines the properties and geometry of the radioactive source (`sources`), detector (`detectors`), attenuator (`attenuators`), and shielding (`shields`). The source contains the radioactivity of the sample (`idSample`) to be measured, even though its chemical and physical form can be different.

Measurement parameters are stored in table `measurements` and the resulting spectrum, or spectra, are stored in table `spectra`. Additional data required for alpha measurements are stored in table `alphaMeasurements`.

Creating the primary key `idMeas` in table `measurements` is the **entry point 2** to *Linssi*. Measurement data cannot be input to *Linssi* before a corresponding entry of `idSample` in table 4.1 `samples` is available.

### 6.1 Sources

Table 6.1: Source geometry and material description

sources			
Field	Type	Length	Flags
<code>sourceId</code>	<code>varchar</code>	80	PN
<code>idSample</code>	<code>int</code>	4	IF
<code>sourceGeometry</code>	<code>varchar</code>	20	
<code>sourceThickness</code>	<code>double</code>	8	
<code>sourceHeightMar</code>	<code>double</code>	8	
<code>sourceWidth</code>	<code>double</code>	8	
<code>sourceLength</code>	<code>double</code>	8	
<code>sourceDiam1</code>	<code>double</code>	8	
<code>sourceDiam2</code>	<code>double</code>	8	
<code>sourceLayers</code>	<code>smallint</code>	2	
<code>sourceDensity</code>	<code>double</code>	8	
<code>sourceMass</code>	<code>double</code>	8	
<code>sourceMaterial</code>	<code>varchar</code>	20	R
<code>contDens</code>	<code>double</code>	8	

*continued on next page ...*

... continued from previous page

Field	Type	Length	Flags
contThick	double	8	
contMaterial	varchar	40	R
preparationMethod	text	0	R
sourceRFID	text	0	
sourceBarcode	text	0	
comments	text	0	
(idSample ) → samples(idSample)			

end of table.

This table describes the source geometry when the source to be measured is in the measuring position. Typically it describes the standard reference source geometry used in this measurement setup, see *Case 1* below. However, the source geometry can also be defined to apply sample specific corrections to this standard geometry, see *Case 2* below.

This table aims to describe the geometry and material of the source, which includes the container, to the extent sufficient for Monte Carlo simulation of the measuring process when used together with the rest of the measurement setup.

sourceId	A unique name of the source.
idSample	Identifies the sample. See table 4.1 <b>samples</b> . We can identify two cases. <i>Case 1:</i> If the source is used to describe a standard measurement setup, i.e., its <b>sourceId</b> appears in table 6.5 <b>measurementSetups</b> , then <b>idSample</b> here should be set to NULL. <i>Case 2:</i> If the source is not describing a standard geometry, i.e., not having proper calibrations directly available, <b>idSample</b> must be given. In this case the analysis software is using the geometry information of the source to calculate a new, or to adjust an existing, calibration. See field <b>sourceId</b> in table 6.6 <b>measurements</b> , and field <b>class</b> in table 5.2 <b>calTypes</b> .
sourceGeometry	Name of the source geometry. This name identifies the meaning of the source measures below.
sourceThickness	Thickness of the source in millimeters [mm].
sourceHeightMar	Height of the Marinelli beaker source in millimeters [mm].
sourceWidth	Width of the source in millimeters [mm].
sourceLength	Length of the source in millimeters [mm].
sourceDiam1	Outer diameter of the source in millimeters [mm].
sourceDiam2	Inner diameter of the source in millimeters [mm].
sourceLayers	Number of layers in the source.
sourceDensity	Density of the source in kilograms per cubic meter [kg/m <sup>3</sup> ].
sourceMass	Mass of the source in kilograms [kg].
sourceMaterial	Name of the source material.
contDens	Density of the source container in kilograms per cubic meter [kg/m <sup>3</sup> ].
contThick	Thickness of the source container in millimeters [mm].
contMaterial	Name of the source container material.
preparationMethod	Describes how the source has been made from the sample.

<code>sourceBarcode</code>	Bar code on the source.
<code>sourceRFID</code>	Radio Frequency IDentification a.k.a. RFID-tag on the source.
<code>comments</code>	Comments in English [en].

## 6.2 Detectors

Table 6.2: Detectors

detectors			
Field	Type	Length	Flags
detectorId	varchar	80	PN
facilityId	varchar	80	F
contactPerson	varchar	40	
detectorModel	varchar	40	
detectorType	varchar	20	R
absEfficiency	double	8	
volume	double	8	
diameter	double	8	
thickness	double	8	
coreDiameter	double	8	
coreLength	double	8	
endcapToCrystal	double	8	
windowMaterial	varchar	20	R
windowThickness	double	8	
deadLayerThickness	double	8	
biasVoltage	double	8	
polarity	varchar	10	
comments	text	0	
(facilityId ) → facilities(facilityId)			

*end of table.*

This table describes detectors used. This table aims to describe the detector to the extent sufficient for Monte Carlo simulation of the measuring process when used together with the rest of the measurement setup.

detectorId	A unique name of the detector.
facilityId	See table 2.1 facilities.
contactPerson	Name of the contact person.
detectorModel	Detector manufacturer's model name and number for the detector.
detectorType	Type of the detector, e.g., p-type.
absEfficiency	Absolute efficiency of the detector.
volume	Volume of the detector in cubic millimeters [mm <sup>3</sup> ].
diameter	Detector diameter in millimeters [mm].
thickness	Detector thickness in millimeters [mm].
coreDiameter	Diameter of the detector core in millimeters [mm].
coreLength	Length of the detector core in millimeters [mm].
endcapToCrystal	Distance from the endcap to the crystal surface in millimeters [mm].
windowMaterial	Name of the detector window material.
windowThickness	Detector window thickness in millimeters [mm].
deadLayerThickness	Detector dead layer thickness in millimeters [mm].
biasVoltage	Bias voltage in volts [V].
polarity	Polarity of the voltage, positive or negative.
comments	Comments in English [en].



## 6.3 Shields

Table 6.3: Shields

shields			
Field	Type	Length	Flags
<code>shieldId</code>	<code>varchar</code>	80	PN
<code>comments</code>	<code>text</code>	0	

*end of table.*

This table describes shielding used in measurements. (To be defined.)

This table aims to describe the shielding to the extent sufficient for Monte Carlo simulation of the measuring process when used together with the rest of the measurement setup.

`shieldId` A unique name of the shield.  
`comments` Comments in English [en].

## 6.4 Attenuators

Table 6.4: Attenuators

attenuators			
Field	Type	Length	Flags
<code>attenuatorId</code>	<code>varchar</code>	80	PN
<code>comments</code>	<code>text</code>	0	

*end of table.*

This table describes attenuators used in measurements. (To be defined.)

This table aims to describe the attenuator to the extent sufficient for Monte Carlo simulation of the measuring process when used together with the rest of the measurement setup.

`attenuatorId` A unique name of the attenuator.  
`comments` Comments in English [en].

## 6.5 Measurement Setups

Table 6.5: Measurement setups identify the components used and their positions

measurementSetups			
Field	Type	Length	Flags
measSetupId	varchar	80	PN
detectorId	varchar	80	F
sourceId	varchar	80	F
attenuatorId	varchar	80	F
shieldId	varchar	80	F
sampleDistX	double	8	
sampleDistY	double	8	
sampleDistZ	double	8	
attenDistX	double	8	
attenDistY	double	8	
attenDistZ	double	8	
shieldDistX	double	8	
shieldDistY	double	8	
shieldDistZ	double	8	
comments	text	0	
(detectorId ) → detectors(detectorId) (sourceId ) → sources(sourceId) (attenuatorId ) → attenuators(attenuatorId) (shieldId ) → shields(shieldId)			

*end of table.*

The Cartesian right hand coordinate system used here has its origin at the center of the detector's active front face window on its outer surface. The positive z-axis points outwards of the detector. If the z-axis is horizontal the y-axis is vertical and points upwards. Otherwise the y-axis is parallel to the plane of maximum possible symmetry and points outwards of the center of the detector system.

Sample, attenuator and shield coordinate axes are parallel to the coordinate axes defined here, but their origins are offset as described below.

The description of the measurement setup is meant to be accurate enough for Monte Carlo simulation of the measurements.

measSetupId	Unique name of the measurement setup.
detectorId	See table 6.2 detectors.
sourceId	See table 6.1 sources.
attenuatorId	See table 6.4 attenuators.
shieldId	See table 6.3 shields.
sampleDistX	x-component of the offset of the sample origin in millimeters [mm].
sampleDistY	y-component of the offset of the sample origin in millimeters [mm].
sampleDistZ	z-component of the offset of the sample origin in millimeters [mm].

<code>attenDistX</code>	x-component of the offset of the attenuator origin in millimeters [mm].
<code>attenDistY</code>	y-component of the offset of the attenuator origin in millimeters [mm].
<code>attenDistZ</code>	z-component of the offset of the attenuator origin in millimeters [mm].
<code>shieldDistX</code>	x-component of the offset of the shield origin in millimeters [mm].
<code>shieldDistY</code>	y-component of the offset of the shield origin in millimeters [mm].
<code>shieldDistZ</code>	z-component of the offset of the shield origin in millimeters [mm].
<code>comments</code>	Comments in English [en].

## 6.6 Measurements

Table 6.6: Measurement parameters

measurements			
Field	Type	Length	Flags
idMeas	int	4	PA
idSample	int	4	IFN
sourceId	varchar	80	F
measSetupId	varchar	80	F
blankIdMeas	int	4	FS
backgroundIdMeas	int	4	FS
idCal	int	4	F
measId	varchar	80	UNC
phdMeasName	varchar	80	C
extMeasName	varchar	80	
sourcePreparedBy	varchar	40	
sourceReady	datetime	8	
facilityId	varchar	80	F
contactPerson	varchar	40	
detectorTemperature	double	8	
uncDetectorTemperature	double	8	
ambientTemperature	double	8	
ambientHumidity	double	8	
waitTime	double	8	
acqStart	datetime	8	I
acqStartNanoSeconds	int	4	
acqEnd	datetime	8	
acqRealTime	double	8	
acqLiveTime	double	8	
measurementType	varchar	20	R
measuredDoseRate	double	8	
uncMeasuredDoseRate	double	8	
spectrumSent	datetime	8	
extFileName	text	0	
comments	text	0	
(idSample ) → samples(idSample) (sourceId ) → sources(sourceId) (measSetupId ) → measurementSetups(measSetupId) (blankIdMeas ) → measurements(idMeas) (backgroundIdMeas ) → measurements(idMeas) (idCal ) → calibrations(idCal) (facilityId ) → facilities(facilityId)			

*end of table.*

idMeas  
idSample

A unique measurement identifier.  
See table [4.1](#) samples.

<code>sourceId</code>	The identifier of the measured source. See table 6.1 <code>sources</code> . We can identify two cases. <i>Case 1:</i> If we are using a standard measurement setup without changes in the source geometry, then <code>sourceId</code> here should be set to <code>NULL</code> . In this case the relevant geometry information is found using <code>measSetupId</code> below. <i>Case 2:</i> If the source is not describing a standard geometry the geometry is defined by <code>sourceId</code> . The analysis software is able to calculate a new, or adjust an existing, calibration by comparing the geometry given here with the geometry of the standard setup defined by the source associated with the standard geometry <code>measurementSetup.sourceId</code> .
<code>measSetupId</code>	See table 6.5 <code>measurementSetups</code> .
<code>blankIdMeas</code>	A name of the blank measurement relevant for this sample measurement. This is a unique identifier of the type <code>idMeas</code> pointing to a record in this <code>measurements</code> table.
<code>backgroundIdMeas</code>	A name of the background measurement relevant for this sample measurement. This is a unique identifier of the type <code>idMeas</code> pointing to a record in this <code>measurements</code> table.
<code>idCal</code>	Identification number of the calibration, which was received with the measured spectrum. See table 5.1 <code>calibrations</code> . Also for an outside measurement there may be a corresponding measurement setup already defined in this <i>Linssi</i> database. In that case it is left to the analyst to decide whether to use the calibration provided there or the one received with the spectrum.
<code>measId</code>	A unique measurement name, i.e., this has one-to-one correspondence with the primary key <code>idMeas</code> above. This name is normally a result of the naming rules applied in the measurement facility. For the <i>Linssi</i> naming convention see scripts and interfaces manual [3].
<code>phdMeasName</code>	A name following the naming rules of the measurement identification (MID) of the CTBTO [9]. Even though not required by the database this name is globally unique across different FULL spectra. However, the MID for eventual PREL spectra is identical to that of the FULL spectrum. The naming convention is also described in <i>Linssi</i> scripts and interfaces manual [3].
<code>extMeasName</code>	A name of the measurement given or required by the customer. This name is normally a result of the naming rules used by the customer. This name is most probably unique but it is not required by the database.
<code>sourcePreparedBy</code>	Name of the person who prepared the source.
<code>sourceReady</code>	Date and time the source was ready for first measurement.
<code>facilityId</code>	Identifier of the facility that performed the measurement. See table 2.1 <code>facilities</code> .
<code>contactPerson</code>	Contact person for the measurement. See table 2.1 <code>facilities</code> .
<code>detectorTemperature</code>	Detector temperature in degrees centigrade [°C].
<code>uncDetectorTemperature</code>	One sigma absolute uncertainty of the detector temperature in degrees centigrade [°C].
<code>ambientTemperature</code>	Ambient temperature in degrees centigrade [°C].
<code>ambientHumidity</code>	Ambient relative humidity in percent [%].

<code>waitTime</code>	Time between end of sampling (or end of irradiation, or equivalent) and start of acquisition in seconds [s].
<code>acqStart</code>	Date and time of the start of acquisition.
<code>acqStartNanoSeconds</code>	Number of nanoseconds [ns] past last full second of acquisition start. If not zero <code>acqStart</code> must be truncated (not rounded) to the nearest second.
<code>acqEnd</code>	Date and time of the end of acquisition.
<code>acqRealTime</code>	Real time of acquisition in seconds [s].
<code>acqLiveTime</code>	Measurement system effective live time of acquisition in seconds [s].
<code>measurementType</code>	A reserved keyword describing the measurement type, e.g. <code>ZeroDeadTime</code> .
<code>measuredDoseRate</code>	Measured dose rate in Sieverts per second [Sv/s].
<code>uncMeasuredDoseRate</code>	Uncertainty of the measured dose rate in Sieverts per second [Sv/s].
<code>spectrumSent</code>	Date and time when the spectrum, or spectra, were sent to analysis. If the analysis is carried out in the measuring facility, this is most probably the time the spectrum, or spectra, were stored to the database.
<code>extFileName</code>	Pointer to external file for extra measurement results.
<code>comments</code>	Comments in English [en].

## 6.7 Spectra

Table 6.7: Measured spectra

spectra			
Field	Type	Length	Flags
idMeas	int	4	PF
idSpectrum	int	4	P
idSample	int	4	IFN
spectrumType	varchar	20	R
spectrumFormat	varchar	80	R
firstChannel	int	4	
lastChannel	int	4	
firstValidChannel	int	4	
lastValidChannel	int	4	
spectrum	longblob	0	
comments	text	0	
(idMeas ) → measurements(idMeas)			
(idSample ) → samples(idSample)			

*end of table.*

The measured spectra are stored in this table. In a normal gamma spectrum measurement the result is a single spectrum. However, there are cases when the result is two or more spectra, e.g., variance spectrum of the Ortec ZDT method, coincidence spectra, etc.

idMeas	Identification number of the measurement. See table 6.6 <b>measurements</b> .
idSpectrum	Unique spectrum number.
idSample	Identification number of the sample measured. See table 4.1 <b>samples</b> .
spectrumType	Type of the spectrum. Reserved types: FULL, a normal full time spectrum PREL, denoting a time slice of the full spectrum [9]. VARIANCE, variance spectrum of the ZDT method by Ortec.
spectrumFormat	Format of the spectrum. Reserved formats NULL or NORMAL, Only the counts (y-values) are stored, i.e., no channel numbers (x-values). The format is one channel per line. NORMAL-ZIPPED, like NORMAL but in the compressed in zip format
firstChannel	The channel number of the first number of counts in the <b>spectrum</b> .
lastChannel	The channel number of the last number of counts in the <b>spectrum</b> .
firstValidChannel	The channel number of the first valid number of counts in the <b>spectrum</b> . This could be the first channel above the lower level discriminator, for example.
lastValidChannel	The channel number of the last valid number of counts in the <b>spectrum</b> . This could be the last channel below the upper level discriminator, for example.



<code>spectrum</code>	The measured spectrum in a format given in <code>spectrumFormat</code> above.
<code>comments</code>	Comments in English [en].

## 6.8 Alpha Measurements

Table 6.8: Alpha measurement parameters and results

alphaMeasurements			
Field	Type	Length	Flags
idMeas	int	4	PF
idSample	int	4	IFN
reverseBias	boolean	1	
airPressureStart	double	8	
airPressureEnd	double	8	
detectorLeakCurrent	double	8	
lowDiscriminator	double	8	
highDiscriminator	double	8	
sourceBackingDiameter	double	8	
sourceBackingThickness	double	8	
sourceBackingMaterial	varchar	80	R
tracerNuclide1	varchar	10	c
actTracerNuclide1	double	8	
uncActTracerNuclide1	double	8	
refTimeTracerNuclide1	datetime	8	
tracerNuclide2	varchar	10	c
actTracerNuclide2	double	8	
uncActTracerNuclide2	double	8	
refTimeTracerNuclide2	datetime	8	
comments	text	0	
(idMeas ) → measurements(idMeas)			
(idSample ) → samples(idSample)			

*end of table.*

Even though *Linssi* is the database for gamma-ray spectrometry, its basic structure can be useful also for other spectrometers. Table `alphaMeasurements` is used to complement the `measurements` table so that also alpha spectrometry measurements can be stored in *Linssi*.

Below the `sourceBacking`-group of fields is designed for describing the backing element of a source used in the measurement. It is used typically with sources with active substance spread over a metal disc.

The `tracer`-group of fields is designed to be used with sources that have a known radioactive component added during the source preparation process.

<code>idMeas</code>	Measurement identification number. See table 6.6 <code>measurements</code> .
<code>idSample</code>	Identification number of the sample measured. See table 4.1 <code>samples</code> .
<code>reverseBias</code>	Denotes if battery plate was mounted in the vacuum chamber during measurement.
<code>airPressureStart</code>	Air pressure in pascals [Pa] in the vacuum chamber at the beginning of the measurement.

<code>airPressureEnd</code>	Air pressure in pascals [Pa] in the vacuum chamber at the end of the measurement.
<code>detectorLeakCurrent</code>	Detector leak current during measurement in amperes [A].
<code>lowDiscriminator</code>	Low level discriminator used in kiloelectronvolts [keV].
<code>highDiscriminator</code>	High level discriminator used in kiloelectronvolts [keV].
<code>sourceBackingDiameter</code>	The diameter of the metal disc on which the source is spread in millimeters [mm].
<code>sourceBackingThickness</code>	The source backing material thickness in millimeters [mm].
<code>sourceBackingMaterial</code>	Describes the material of the backing disc.
<code>tracerNuclide1</code>	<code>nuclideId</code> for the added component. For syntax see table <a href="#">3.3</a>
<code>actTracerNuclide1</code>	The activity of the added nuclide in becquerels [Bq].
<code>uncActTracerNuclide1</code>	Uncertainty of the tracer nuclide activity in becquerels [Bq].
<code>refTimeTracerNuclide1</code>	Reference time of the activity.
<code>tracerNuclide2</code>	<code>nuclideId</code> for the second added component. For syntax see table <a href="#">3.3</a>
<code>actTracerNuclide2</code>	The activity of the second added nuclide in becquerels [Bq].
<code>uncActTracerNuclide2</code>	Uncertainty of the second tracer nuclide activity in becquerels [Bq].
<code>refTimeTracerNuclide2</code>	Reference time of the activity of the second tracer.
<code>comments</code>	Comments in English [en].



# Chapter 7

## Regions of Interest and Spectrum Components

In addition to peak based spectrum analysis, as mainly discussed in Ch. 8.1, other spectral components can be used. Two different types of components are described here. First, one or multidimensional Regions of Interest, ROIs, can be used either to store measurements or already analyzed spectrum regions. Multidimensionality provides mechanisms for coincidence analysis, for example. Second, a spectrum can be considered to be a weighted sum of components obtained on the basis of previously analyzed ‘calibration’ spectra.

### 7.1 Regions of Interest (ROIs)

Table 7.1: Regions of Interest (ROIs)

Field	rois Type	Length	Flags
idRoi	int	4	PA
idAnalysis	int	4	F
idMeas	int	4	F
idSample	int	4	F
roiName	varchar	10	R
nuclideName	varchar	10	c
roiEfficiency	double	8	
uncRoiEfficiency	double	8	
roiArea	double	8	
roiNetArea	double	8	
roiStrippedArea	double	8	
roiCps	double	8	
uncRoiCps	double	8	
roiAnalysis	double	8	
uncRoiAnalysis	double	8	
comments	text	0	

(idAnalysis ) → analyses(idAnalysis)

(idMeas ) → measurements(idMeas)

*continued on next page ...*

... continued from previous page

Field	Type	Length	Flags
(idSample )	→ samples(idSample)		

end of table.

This table contains analysis results of regions of interest (ROI). One record is used for each ROI analyzed. Depending of the methodology used the ROI analysis and peak analysis results may both be available for the measurement. By providing foreign keys `idMeas`, `idSample` and `idAnalysis` ROIs can be used in applications where ROIs are either measurement or analysis results.

<code>idRoi</code>	Identification number of the ROI.
<code>idAnalysis</code>	Identification number of the analysis. See table 8.1 analyses.
<code>idMeas</code>	Identification number of the measurement. See table 6.6 measurements.
<code>idSample</code>	Identification number of the sample measured. See table 4.1 samples.
<code>roiName</code>	Name of the ROI.
<code>nuclideName</code>	Name of the nuclide associated with the ROI. For syntax see table 3.3
<code>roiEfficiency</code>	Counts in ROI/emission.
<code>uncRoiEfficiency</code>	One sigma absolute uncertainty of <code>roiEfficiency</code> .
<code>roiArea</code>	Total ROI area in counts. Note: No baseline area has been subtracted.
<code>roiNetArea</code>	Net ROI area in counts. Note: baseline area has been subtracted but not the contributions from stripping or from the blank and background.
<code>roiStrippedArea</code>	Stripped ROI area in counts. Note: baseline area and ROI stripping has been subtracted but the contributions from the blank and background not.
<code>roiCps</code>	Counts per seconds in stripped ROI. Note: baseline area and ROI stripping has been subtracted but the contributions from the blank and background not.
<code>uncRoiCps</code>	One sigma absolute uncertainty of <code>roiCPS</code> .
<code>roiAnalysis</code>	Numerical, single value giving the result of a specialized ROI analysis, e.g. ratio of measured to predicted counts in a ROI. The exact definition (and unit) of this field is dependent on the analysis software.
<code>uncRoiAnalysis</code>	Uncertainty of the ROI analysis result.
<code>comments</code>	Comments in English [en].

## 7.2 ROI Limits

Table 7.2: ROI limits

roiLimits			
Field	Type	Length	Flags
idRoi	int	4	PF
idLim	int	4	P
roiStartChannel	int	4	
roiEndChannel	int	4	
roiStartEnergy	double	8	
roiEndEnergy	double	8	
comments	text	0	
(idRoi ) → rois(idRoi)			

*end of table.*

The ROI limits in energy and channels are given in this table. The possibility to have multiple limits, `idLim` for each ROI, facilitates multidimensional ROIs. In 3D beta-gamma coincidence spectrometry, for example, two dimensional ROIs are used.

<code>idRoi</code>	Identification number of the ROI. See table 7.1 <code>rois</code> .
<code>idLim</code>	Identification number of the ROI limit. For beta-gamma spectroscopy systems <code>idLim = 1</code> and <code>2</code> are reserved beta and gamma channel/energy limits, respectively.
<code>roiStartChannel</code>	Start channel of the ROI. Depending on the application it is recommended to use either channels or energy (i.e. not both) to define the ROIs due to possibly changing energy calibrations.
<code>roiEndChannel</code>	End channel of the ROI.
<code>roiStartEnergy</code>	Start energy of the ROI in keV.
<code>roiEndEnergy</code>	End energy of the ROI in keV.
<code>comments</code>	Comments in English [en].

## 7.3 ROI Components

Table 7.3: ROI components

roiComponents			
Field	Type	Length	Flags
<code>idRoi</code>	<code>int</code>	4	PF
<code>componentTypeId</code>	<code>varchar</code>	10	PR
<code>roiComponent</code>	<code>longblob</code>	0	
<code>comments</code>	<code>text</code>	0	
(idRoi ) → rois(idRoi)			

*end of table.*

<code>idRoi</code>	Identification number of the ROI, to which the components are related. See table 7.1 <code>rois</code> .
<code>componentTypeId</code>	Type of the component, e.g., <code>beta</code> for the beta component of the ROI in beta-gamma analyses.
<code>roiComponent</code>	The ROI component, e.g., channel by channel gamma counts between <code>roiStartEnergy</code> and <code>roiEndEnergy</code> given in table 7.2 <code>roiLimits</code> . The exact type of the component depends on the component type and analysis software.
<code>comments</code>	Comments in English [en].



## 7.4 ROI Ratios

Table 7.4: ROI ratios

roiRatios			
Field	Type	Length	Flags
<code>firstIdRoi</code>	int	4	PF
<code>secondIdRoi</code>	int	4	PF
<code>roiRatio</code>	varchar	20	
<code>countRatio</code>	double	8	
<code>uncCountRatio</code>	double	8	
<code>comments</code>	text	0	
(firstIdRoi ) → rois(idRoi)			
(secondIdRoi ) → rois(idRoi)			

*end of table.*

<code>firstIdRoi</code>	idRoi of the higher energy ROI. See table 7.1 rois.
<code>secondIdRoi</code>	idRoi of the lower energy ROI. See table 7.1 rois.
<code>roiRatio</code>	Name of the ratio, eg."PB214_352:242" (from b-g phd format).
<code>countRatio</code>	Ratio of counts in <code>firstIdRoi</code> to be stripped from the counts in <code>secondIdRoi</code> .
<code>uncCountRatio</code>	One sigma absolute uncertainty of <code>countRatio</code> .
<code>comments</code>	Comments in English [en].

## 7.5 Spectrum Components

Table 7.5: Spectrum components

spectrumComponents			
Field	Type	Length	Flags
<code>idComponent</code>	int	4	PA
<code>idAnalysis</code>	int	4	F
<code>idMeas</code>	int	4	F
<code>idSample</code>	int	4	F
<code>componentType</code>	varchar	10	R
<code>spectrumComponent</code>	longblob	0	
<code>comments</code>	text	0	
(idAnalysis ) → analyses(idAnalysis) (idMeas ) → measurements(idMeas) (idSample ) → samples(idSample)			

*end of table.*

<code>idComponent</code>	Auto-incrementing identification number of the spectrum component.
<code>idAnalysis</code>	Identification number of the analysis where the component was calculated. See table <a href="#">8.1 analyses</a> .
<code>idMeas</code>	Identification number of the measurement, to which the components are related. See table <a href="#">6.6 measurements</a> .
<code>idSample</code>	Identification number of the sample measured. See table <a href="#">6.6 samples</a> .
<code>componentType</code>	Type of the spectrum component, e.g. FSC for spectrum component used in spectrum fitting with spectral components.
<code>spectrumComponent</code>	The spectrum component, e.g. channel by channel residual spectrum after removing an average spectrum from the measured spectrum. The exact type of the component depends on the spectrum component type and analysis software.
<code>comments</code>	Comments in English [en].

## 7.6 Spectrum Components Used

Table 7.6: Components used

componentsUsed			
Field	Type	Length	Flags
<code>idAnalysis</code>	int	4	PF
<code>idComponent</code>	int	4	PF
<code>idMeas</code>	int	4	F
<code>idSample</code>	int	4	F
<code>componentWeight</code>	double	8	
<code>uncComponentWeight</code>	double	8	
<code>comments</code>	text	0	

(`idAnalysis` ) → `analyses(idAnalysis)`  
 (`idComponent` ) → `spectrumComponents(idComponent)`  
 (`idMeas` ) → `measurements(idMeas)`  
 (`idSample` ) → `samples(idSample)`

*end of table.*

<code>idAnalysis</code>	Identification number of the analysis where the spectrum component number <code>idComponent</code> was used. See table 8.1 <code>analyses</code> .
<code>idComponent</code>	Identification number of the component. See table 7.5 <code>spectrumComponents</code> .
<code>idMeas</code>	Identification number of the measurement. See table 6.6 <code>measurements</code> . NULL, if site is a sample production site.
<code>idSample</code>	Identification number of the sample. See table 4.1 <code>samples</code> . NULL, if site is a measurement site.
<code>componentWeight</code>	Dimensionless weighting factor of the used spectrum component.
<code>uncComponentWeight</code>	One sigma absolute uncertainty of the component weight.
<code>comments</code>	Comments in English [en].



# Chapter 8

## Analysis

The analysis group of tables is in the core of the *Linssi* database. *Linssi* is designed to be useful in a facility doing gamma-ray spectrum analysis. From the version 2.00 on *Linssi* has also provided tables for alpha and beta-gamma spectrometry. The sample collection and measurement can very well be performed elsewhere and only the analysis results produced in-house. However, *Linssi* design requires that at least dummy entries of `idSample` and `idMeas` must exist in tables `samples` and `measurements`, respectively. Of course, only limited analysis is possible without knowledge of the measurement parameters, for example.

Starting an analysis, i.e., creating the primary key `idAnalysis` in table `analyses` is the **entry point 3** to the database.

All the tables in this group are filled with analysis and identification software. The amount of information may vary depending on the software used. The number of fields in these tables is large and many programs are not able to provide enough information to fill them all. *Linssi* developers have been using UNISAMPO – SHAMAN and *Aatami* – SHAMAN chains of software to provide all the information in gamma-ray spectrometry analysis tables.

Our philosophy has been to be able to store multiple analysis results, identified by `idAnalysis`, for any measurement without overwriting the earlier ones. This facilitates either a fully automatic pipeline in a once-through fashion, or an iterative approach where the earlier results are used as a starting point for further interactive or automatic processing. The best results are then identified in table 8.9 `finalResults`.

### 8.1 Analyses

Table 8.1: General data on analysis

analyses			
Field	Type	Length	Flags
<code>idAnalysis</code>	int	4	PA
<code>analysisId</code>	varchar	80	UNC
<code>idMeas</code>	int	4	IFN
<code>idSample</code>	int	4	IFN
<code>blankIdAnalysis</code>	int	4	FS
<code>backgroundIdAnalysis</code>	int	4	FS

*continued on next page ...*

... continued from previous page

Field	Type	Length	Flags
inputIdAnalysis	int	4	FS
spectrumArrival	datetime	8	
analysisBegin	datetime	8	
analysisEnd	datetime	8	
inputParam	text	0	R
interactiveLog	text	0	
type	varchar	20	R
software	varchar	20	R
swVersion	varchar	80	
facilityId	varchar	80	F
contactPerson	varchar	40	I
refTime1	datetime	8	
decayTime1	double	8	
refTime2	datetime	8	
decayTime2	double	8	
refConstants	text	0	R
baselineMethod	text	0	R
peaksMethod	text	0	R
roiMethod	text	0	R
nuclideMethod	text	0	R
uncCalcMethod	text	0	R
lcMethod	text	0	R
alpha	double	8	
beta	double	8	
searchStartChannel	mediumint	3	
searchEndChannel	mediumint	3	
searchThreshold	double	8	
numberOfPeaks	mediumint	3	
numberOfIterations	mediumint	3	
totalCounts	int	4	
comments	text	0	
(idMeas ) → measurements(idMeas) (idSample ) → samples(idSample) (blankIdAnalysis ) → analyses(idAnalysis) (backgroundIdAnalysis ) → analyses(idAnalysis) (inputIdAnalysis ) → analyses(idAnalysis) (facilityId ) → facilities(facilityId)			

end of table.

This table contains general data about the analysis, its input and how it has been performed. Some general analysis results are also provided.

**idAnalysis**                      A unique auto incrementing identification number of the analysis.

<code>analysisId</code>	A unique analysis identifier. This character string follows the naming convention applied by the facility running the database. For the <i>Linssi</i> naming convention see scripts and interfaces manual [3].
<code>idMeas</code>	Identification number of the measurement, i.e., spectrum analyzed. See table 6.6 measurements.
<code>idSample</code>	Identification number of the sample measured. See table 4.1 samples.
<code>blankIdAnalysis</code>	Identification number of the analysis of the blank. The analysis results of the blank are used in this analysis for blank subtraction.
<code>backgroundIdAnalysis</code>	Identification number of the analysis of the background. The analysis results of the background are used in this analysis for background subtraction.
<code>inputIdAnalysis</code>	Identification number of the analysis used as input for this analysis. This analysis may be a continuation of the <code>inputIdAnalysis</code> or <code>inputIdAnalysis</code> is an analysis of a similar sample and thus a good starting point for this analysis
<code>spectrumArrival</code>	Date and time when the spectrum arrived for analysis.
<code>analysisBegin</code>	Date and time when the spectrum analysis started.
<code>analysisEnd</code>	Date and time when the spectrum analysis ended.
<code>inputParam</code>	The most essential input parameters used for the analysis.
<code>interactiveLog</code>	Log storing the interactive commands for the analysis given by the analyst.
<code>type</code>	Type of the analysis. This could be interactive, batch, pipeline, for example.
<code>software</code>	Name of the software used in analysis.
<code>swVersion</code>	Version of the software used in analysis.
<code>facilityId</code>	Identifier of the facility that performed the analysis. See table 2.1 facilities.
<code>contactPerson</code>	Contact person for the analysis. See table 2.1 facilities.
<code>refTime1</code>	First reference date and time which the activities may be corrected to.
<code>decayTime1</code>	Decay time in seconds [s] from start of spectrum acquisition to <code>refTime1</code> .
<code>refTime2</code>	Second reference date and time which the activities may be corrected to.
<code>decayTime2</code>	Decay time in seconds [s] from end of activity collection to <code>refTime2</code> .
<code>refConstants</code>	References to the physical constants used. (e.g., ENSDF version, year, etc.)
<code>baselineMethod</code>	Spectrum baseline calculation method.
<code>peaksMethod</code>	Spectrum peak analysis method.
<code>roiMethod</code>	Region Of Interest (ROI) analysis method.
<code>nuclideMethod</code>	Nuclide identification and activity calculation method.
<code>uncCalcMethod</code>	Method of calculating uncertainties.
<code>lcMethod</code>	Method of calculating the decision level $L_c$ .
<code>alpha</code>	<code>alpha</code> defines the confidence, $1 - \alpha$ , on the <i>a posteriori</i> decision for accepting a gamma peak. Value of $\alpha$ is used to reject against <i>error of the first kind</i> .

<code>beta</code>	<code>beta</code> defines the confidence, $1 - \beta$ , on the <i>a priori</i> detection of a gamma line. Value of $\beta$ is used to reject against <i>error of the second kind</i> .
<code>searchStartChannel</code>	Peak search starts from this channel of the spectrum.
<code>searchEndChannel</code>	Peak search ends to this channel of the spectrum.
<code>searchThreshold</code>	Value of the search threshold used given in the units of $L_s$ . See <code>peakSearchSignificance</code> above. The value is dependent on the search method used.
<code>numberOfPeaks</code>	Number of peaks in the spectrum.
<code>numberOfIterations</code>	Number of iterations required for the analysis results. Exact meaning depends on the analysis method used.
<code>totalCounts</code>	Total number of counts in the spectrum.
<code>comments</code>	Comments in English [en].



## 8.2 Analysis Blocks

Table 8.2: Large data blocks resulting from peak analysis

analysisBlocks			
Field	Type	Length	Flags
<code>idAnalysis</code>	int	4	PF
<code>dataType</code>	varchar	20	PR
<code>idMeas</code>	int	4	F
<code>idSample</code>	int	4	F
<code>dataBlock</code>	longblob	0	
<code>dataBlockFormat</code>	varchar	80	R
<code>comments</code>	text	0	
( <code>idAnalysis</code> ) → <code>analyses(idAnalysis)</code> ( <code>idMeas</code> ) → <code>measurements(idMeas)</code> ( <code>idSample</code> ) → <code>samples(idSample)</code>			

*end of table.*

This table contains the large data blocks resulting from peak analysis, i.e., so called Binary Large Objects (BLOB). They are separated from the table `analyses` to facilitate faster queries on it, and to make it easy to delete these BLOBs, if found necessary due to storage space or speed reasons.

<code>idAnalysis</code>	Identification number of the analysis. See table 8.1 <code>analyses</code> .
<code>dataType</code>	Type of the blob. <code>dataType= BASELINE</code> . Channel by channel spectrum baseline used or generated in the analysis. <code>dataType= STRIPPED_SPECTRUM</code> . Channel by channel stripped spectrum used or generated in the analysis, i.e., the spectrum minus the peak functions. <code>dataType= SEARCH_SIGNIFICANCE</code> . Channel by channel peak search significance, i.e., significance values given by the peak search algorithm, very often a digital filter. The peak search significance, $L_s$ , usually aims to be directly proportional to the decision limit $L_c$ , i.e., $L_s = C \times L_c$ but the constant $C$ is dependent on the method, and not always constant, either. For the exact definition the software manual should be consulted. <i>Note</i> : usually the maxima of $L_s$ do not fall on integer channels.
<code>idMeas</code>	Identification number of the measurement, i.e., spectrum analyzed. See table 6.6 <code>measurements</code> .
<code>idSample</code>	Identification number of the sample measured. See table 4.1 <code>samples</code> .
<code>dataBlock</code>	The large data block.
<code>dataBlockFormat</code>	Format used in the <code>dataBlock</code> .

`dataBlockFormat=` NULL. The format is a channel by channel value where the channel range is identical to `spectra.spectrum` but the values are floating point values. See table [6.7 spectra](#). This is the format used for `dataTypes` BASELINE, STRIPPED\_SPECTRUM, and SEARCH\_SIGNIFICANCE, for instance.

`comments`

Comments in English [en].

## 8.3 Peaks

Table 8.3: Peak analysis results

Field	peaks Type	Length	Flags
idAnalysis	int	4	PFN
idPeak	int	4	PN
idMeas	int	4	IFN
idSample	int	4	IFN
centroidChannel	double	8	
uncCentroidChannel	double	8	
energy	double	8	
uncEnergy	double	8	
area	double	8	
uncArea	double	8	
netCountRate	double	8	
uncNetCountRate	double	8	
efficiency	double	8	
uncEfficiency	double	8	
fwhm	double	8	
fwtm	double	8	
significance	double	8	
significanceFlag	char	1	
decisionLimit	double	8	
detectionLimit	double	8	
searchSignificance	double	8	
peakOrigin	varchar	20	
outOfRange	Boolean	1	
baselineIndex	mediumint	3	
peakIdFunction	int	4	F
peakParameters	blob	0	
baselineArea	double	8	
baselinePerChannel	double	8	
uncBaselinePerChannel	double	8	
baselineStart	mediumint	3	
baselineEnd	mediumint	3	
baselineIdFunction	int	4	F
baselineParameters	blob	0	
backgroundCps	double	8	
uncBackgroundCps	double	8	
backgroundType	varchar	20	R
blankCps	double	8	
uncBlankCps	double	8	
blankType	varchar	20	R
comments	text	0	

(idAnalysis ) → analyses(idAnalysis)

*continued on next page ...*

... continued from previous page

Field	Type	Length	Flags
(idMeas )	→ measurements(idMeas)		
(idSample )	→ samples(idSample)		
(peakIdFunction )	→ functions(idFunction)		
(baselineIdFunction )	→ functions(idFunction)		

end of table.

This table contains analysis results of spectral peaks. One record is used for each peak analyzed. In addition to the normal software dependence of all the results, it should be noted that many peak parameters may either be taken directly from the calibrations or calculated from spectral data.

idAnalysis	Identification number of the analysis. See table 8.1 analyses
idPeak	Peak index, i.e., the number of the peak in the analysis idAnalysis. The list is sorted in an ascending order according to peak channel and starting from 1.
idMeas	Identification number of the measurement. See table 6.6 measurements.
idSample	Identification number of the sample measured. See table 4.1 samples.
centroidChannel	Channel of the peak centroid [ch].
uncCentroidChannel	One sigma absolute uncertainty of the centroid channel in channels [ch].
energy	Peak energy in kiloelectronvolts [keV].
uncEnergy	One sigma absolute uncertainty of peak energy in kiloelectronvolts [keV].
area	Peak area in counts. <i>Note:</i> baseline area has been subtracted but the contributions from the blank and background not.
uncArea	One sigma absolute uncertainty of area in counts.
netCountRate	Net count rate at the peak, i.e., peak-area-to-live-time-ratio in counts per second [1/s]. The contribution from blank and background have been subtracted.
uncNetCountRate	One sigma absolute uncertainty of the net count rate in counts per second [1/s].
efficiency	Absolute efficiency of the measuring system at the peak energy.
uncEfficiency	One sigma absolute uncertainty of the absolute efficiency.
fwhm	Full width at half maximum of the peak in channels [ch].
fwtm	Full width at tenth maximum of the peak in channels [ch].
significance	Peak significance in multiples of the decision limit [ $L_c$ ].
significanceFlag	Peak significance flags. They may correspond, for example, to high, medium, low, insignificant etc. The levels are decided by the software.
decisionLimit	Decision limit $L_c$ in counts for this peak (in this spectrum).
detectionLimit	Detection limit $L_d$ in counts for peak at this position (assuming the baseline of this spectrum).
peakOrigin	A set of peak flags giving the origin of the peak. For the reserved values see Tab. 8.4.

<code>outOfRange</code>	TRUE if the centroid of this peak is outside of the valid channel range.
<code>baselineIndex</code>	Index of the channel interval used to calculate the baseline. The same interval is usually used to calculate the peaks, i.e., often synonymous to <code>index</code> of the fitting interval.
<code>peakIdFunction</code>	Function identification number of the peak shape function. See Ch. 9.3.
<code>peakParameters</code>	Values of the parameters of <code>peakIdFunction</code> . The values are given as pairs of parameter value and its one sigma absolute uncertainty. Each pair is separated with the newline character. If the parameter uncertainty is not known it must not be given. It is recommended that known zero uncertainties, which is the case for integers, for example, should be explicitly stated. The pairs are associated with the function by the order given in the function definition. Note that there may be many peaks under one <code>baselineIndex</code> and all their parameters will be included in <code>peakParameters</code> . See Ch. 9.3.
<code>baselineArea</code>	Area of the baseline under the peak in counts, i.e., integral of the baseline function over the interval around peak deemed appropriate by the software.
<code>baselinePerChannel</code>	Baseline area in counts divided by the number of channels included in its calculation.
<code>uncBaselinePerChannel</code>	One sigma absolute uncertainty of the <code>baselinePerChannel</code> in counts.
<code>baselineStart</code>	Validity of the baseline and peak functions start at this channel.
<code>baselineEnd</code>	Validity of the baseline and peak functions end at this channel.
<code>baselineIdFunction</code>	Function identification number of the baseline function. See Ch. 9.3.
<code>baselineParameters</code>	Values of the parameters of <code>baselineIdFunction</code> . The values are given as pairs of parameter value and its one sigma absolute uncertainty. Each pair is separated with the newline character. If the parameter uncertainty is not known it must not be given. It is recommended that known zero uncertainties, which is the case for integers, for example, should be explicitly stated. The pairs are associated with the function by the order given in the function definition. See Ch. 9.3.
<code>backgroundCps</code>	Contribution of the background to peak count rate in counts per second [1/s] as derived from <code>backgroundIdAnalysis</code> . See table 8.1 analyses. <i>Note:</i> these are peak counts not baseline counts.
<code>uncBackgroundCps</code>	One sigma absolute uncertainty of background count rate in counts per second [1/s].
<code>backgroundType</code>	Type of the background: <code>detector</code> , <code>detector+blank</code> , etc., where <code>detector</code> means background only and <code>detector+blank</code> gives the contribution from both the background and blank. If <code>detector+blank</code> is given, <code>blankCps</code> below must be zero.
<code>blankCps</code>	Contribution of the blank to peak count rate in counts per second [1/s] as derived from <code>blankIdAnalysis</code> . See table 8.1 analyses. <i>Note:</i> these are peak counts not baseline counts
<code>uncBlankCps</code>	One sigma absolute uncertainty of blank count rate in counts per second [1/s].

<code>blankType</code>	Type of the blank: <code>blank</code> , <code>blank+detector</code> , etc., where <code>blank</code> means blank only, i.e., the background contribution to blank has been subtracted or negligible, and <code>blank+detector</code> gives the contribution from both the background and blank. If <code>blank+detector</code> is given, <code>backgroundCps</code> above must be zero.
<code>comments</code>	Comments in English [en].

### Note on peak origin flags

The syntax of Aatami [6] has been adopted for the peak origin flags. The available flags are shown in Tab. 8.4. The syntax requires that two flags from each of the four groups are given and the order shown in the table is followed. However, the second flag of the first group is always an underscore. Thus a peak found using Mariscotti peak search; fitted using Full fit keeping the centroid fixed; area also calculated with Full fit but, of course, keeping its value free; and taking the FWHM from peak shape calibration and thus keeping its value fixed in last fitting, would read: M\_F0F1C0

<b>1. Peak finding source flags (one flag and an underscore)</b>	
Flag	Explanation
A	Artificial peak
B	Background peak
E	External
M	Mariscotti peak search
R	Residual peak search (stripped spectrum)
I	Manually inserted (in certain channel)
l	Manually inserted (in library position)
L	Library inserted (multiplet fitting)
N	Inserted by natural radionuclide model
S	Inserted by summation peak model
<b>2. Centroid source flags (2 flags)</b>	
Flag	Explanation
Q	Quick fit
F	Full fit
U	User defined
E	External software
R	Multiplet fitting
M	Mariscotti center of gravity
L	Converted library energy
T	Tight fitting
0	Fixed in last fitting
1	Free in last fitting
<b>3. Net area source flags (2 flags)</b>	
Flag	Explanation
Q	Quick fit
F	Full fit
U	User defined
E	External software
R	Multiplet fitting
C	Calculated from reference line
S	Summation
s	Quick summation
0	Fixed in last fitting
1	Free in last fitting
<b>4. Full width at half maximum source flags (2 flags)</b>	
Flag	Explanation
Q	Quick fit
F	Full fit
U	User defined
E	External software
C	Calibration
0	Fixed in last fitting
1	Free in last fitting

Table 8.4: Peak origin flags. See note on p. 86

## 8.4 Line Associations

Table 8.5: Line associations

lineAssociations			
Field	Type	Length	Flags
idAnalysis	int	4	PI1F1N
nuclideId	varchar	10	PNc
idLine	smallint	2	PN
idPeak	int	4	I1F1
idMeas	int	4	IFN
idSample	int	4	IFN
lineEnergy	double	8	
uncLineEnergy	double	8	
emissionProb	double	8	
uncEmissionProb	double	8	
CCfactor	double	8	
uncCCfactor	double	8	
lineSignificance	double	8	
explLevel	double	8	
lorentzGamma	double	8	
xray	Boolean	1	
background	Boolean	1	
annihilation	Boolean	1	
singleEscape	Boolean	1	
doubleEscape	Boolean	1	
xrayEscape	Boolean	1	
backscatter	Boolean	1	
coincSum	Boolean	1	
randomSum	Boolean	1	
neutronScatter	Boolean	1	
neutronCapture	Boolean	1	
userGiven	Boolean	1	
found	Boolean	1	
foundClose	Boolean	1	
thresholdLine	Boolean	1	
primaryLine	Boolean	1	
actMan	Boolean	1	
comments	text	0	

(idMeas ) → measurements(idMeas)  
(idSample ) → samples(idSample)  
(idAnalysis ,idPeak ) → peaks(idAnalysis,idPeak)

*end of table.*

Line association results are stored in this table. For each gamma line of each identified nuclide there may be at maximum one spectrum peak associated with it. At maximum, since small library lines may not be visible in the spectrum. Put it the other way round: for



each spectrum peak there may be any number of library lines associated with it.

It is not necessary for each gamma line to be associated with a spectrum peak. These lines have been used by the analysis software to support in nuclide identification.

For unidentified spectrum peaks `idLine` is 0 and `nuclideId` is `NO_ID1`, `NO_ID2`, `NO_ID3`,...

<code>idAnalysis</code>	Identification number of the analysis. See table 8.1 analyses.
<code>nuclideId</code>	Name of a nuclide. If identified the LSQ activity of this nuclide is based on the line(s) in this table for which the <code>idPeak</code> is not NULL. See table 8.6 activities. For syntax see table 3.3 calibrationLibraries. If not identified, see <code>idLine</code> below.
<code>idLine</code>	Index of the nuclide line starting from 1. This is a unique internal number for nuclide line given by the analysis software. It is recommended that this is the number of the line in the nuclide reference library. However, for SHAMAN this is an internal index, since the properties of coincidence and escape lines are internally calculated in SHAMAN. If this index is 0 <code>idPeak</code> has not been identified, i.e., there are no lines associated with it. In that case the <code>nuclideId</code> is <code>NO_ID1</code> , <code>NO_ID2</code> , <code>NO_ID3</code> ,...
<code>idPeak</code>	Identification number of the peak if there is a peak associated with this gamma line. Otherwise <code>idPeak</code> is NULL. See table 8.3 peaks. This peak has been used in calculation of <code>actRawLSQ</code> . Depending on values of <code>actMan</code> and <code>primaryLine</code> this peak has been used also in calculation of <code>actRawMan</code> and <code>actRawPriLine</code> . See below.
<code>idMeas</code>	Identification number of the measurement. See table 6.6 measurements.
<code>idSample</code>	Identification number of the sample measured. See table 4.1 samples.
<code>lineEnergy</code>	Energy of the gamma line in kiloelectronvolts [keV]. <i>Note:</i> this is not the energy of the gamma peak in the spectrum.
<code>uncLineEnergy</code>	Absolute one sigma uncertainty of <code>lineEnergy</code> .
<code>emissionProb</code>	Emission probability of the gamma line (fraction, NOT in %).
<code>uncEmissionProb</code>	One sigma absolute uncertainty of the emission probability.
<code>CCfactor</code>	True coincidence correction factor of the gamma line.
<code>uncCCfactor</code>	One sigma absolute uncertainty of the true coincidence correction factor of the gamma line.
<code>lineSignificance</code>	The significance of the gamma line in units of the detection level $L_d$ . This is the calculated line significance based on the activity of the identified nuclide. It gives a measure whether this gamma line should be visible in the spectrum as a gamma peak or not.
<code>explLevel</code>	The fraction of peak area explained by this gamma line.
<code>lorentzGamma</code>	Value of the Lorentz gamma width of this X-ray line in kiloelectronvolts [keV].
<code>xray</code>	TRUE if this line is an X-ray line.
<code>background</code>	TRUE if this line is a background line.
<code>annihilation</code>	TRUE if this line is an annihilation line.
<code>singleEscape</code>	TRUE if this line is a single escape line.
<code>doubleEscape</code>	TRUE if this line is a double escape line.
<code>xrayEscape</code>	TRUE if this line is an X-ray escape line.
<code>backscatter</code>	TRUE if this line is a backscatter line.

<code>coincSum</code>	TRUE if this line is a true coincidence sum line.
<code>randomSum</code>	TRUE if this line is a random coincidence sum line.
<code>neutronScatter</code>	TRUE if this line is due to neutron scatter in the detector.
<code>neutronCapture</code>	TRUE if this line is due to neutron capture in the detector.
<code>userGiven</code>	TRUE if this line is a user defined line.
<code>found</code>	TRUE if this line is found, i.e., associated with a peak in the spectrum. The LSQ calculation of nuclide activity is based on all nuclide lines for which <code>found</code> is TRUE. See table 8.6 activities.
<code>foundClose</code>	TRUE if this line is found close to a spectrum peak but not associated with it.
<code>thresholdLine</code>	TRUE if this line is the threshold line. The threshold line is the most significant line of a nuclide that has NOT been associated with a peak in the spectrum.
<code>primaryLine</code>	TRUE if this line is the primary line, i.e., the most significant line of the nuclide given the spectral baseline. The primary line activity <code>actRawPriLine</code> is based on the peak associated with this gamma line. See table 8.6 activities.
<code>actMan</code>	TRUE if this line has been used in the off-line, possibly manual, calculation of the nuclide activity <code>actRawMan</code> . In that case the line must have been associated with a peak, i.e., <code>idPeak</code> must not be NULL. See table 8.6 activities.
<code>comments</code>	Comments in English [en].

## 8.5 Nuclides and Their Activities

Table 8.6: Identified nuclides and their activities

Field	Type	Length	Flags
idAnalysis	int	4	PFN
nuclideId	varchar	10	PINc
idMeas	int	4	IFN
idSample	int	4	IFN
confidence	double	8	
effPrecursor	varchar	10	c
effHalflife	double	8	
uncEffHalflife	double	8	
isBackground	Boolean	1	
actRawLSQ	double	8	
uncActRawLSQ	double	8	
interfCorrLSQ	Boolean	1	
actRawPriLine	double	8	
uncActRawPriLine	double	8	
interfCorrPriLine	Boolean	1	
actRawMan	double	8	
uncActRawMan	double	8	
actRawManMethod	text	0	R
actSelect	varchar	10	R
acqCorr	double	8	
uncAcqCorr	double	8	
decayCorr1	double	8	
uncDecayCorr1	double	8	
waitCorr	double	8	
uncWaitCorr	double	8	
irrCorr	double	8	
uncIrrCorr	double	8	
collCorr	double	8	
uncCollCorr	double	8	
decayCorr2	double	8	
uncDecayCorr2	double	8	
comments	text	0	
(idAnalysis ) → analyses(idAnalysis)			
(idMeas ) → measurements(idMeas)			
(idSample ) → samples(idSample)			

*end of table.*

Identified nuclides and their raw activities are stored in this table. The table contains activities calculated using only the primary lines and the activities where all found peaks of the nuclides are used in the least squares (LSQ) sense. The interfering nuclides may have been resolved during the process. There is also a field for off-line calculated raw activity. If the contributions from blank and background are known (see table 8.3 peaks) their contribution

has been subtracted, i.e., peaked background subtraction (PBS) has been applied to all activities.

The decay correction factors, for evaluation of the nuclide activity at specific dates and times from the raw activities, can also be stored. If the decay chain is in equilibrium these factors have been calculated using the effective half-lives of the nuclides. In the case of a decay chain not in equilibrium, the effective half-lives cannot be used. Even in this case the given correction factors can be used to obtain the decay corrected activities. The user should, however, consult the analysis software manual to find out whether the assumption of equilibrium has been applied or not.

Since the correction factors are cumulative it would be tempting to calculate the total uncertainty from the uncertainties of the individual factors using the normal error propagation law of Gauss. However, since the decay corrections are strongly correlated due to identical half-lives, it must also be taken into account. The corrections and the associated times are illustrated in Fig. 8.1 below. Decay correction factors are multiplicative.

<code>idAnalysis</code>	Identification number of the analysis. See table 8.1 analyses.
<code>nuclideId</code>	Name of the identified nuclide. For syntax see table 3.3 calibrationLibraries.
<code>idMeas</code>	Identification number of the measurement. See table 6.6 measurements.
<code>idSample</code>	Identification number of the sample measured. See table 4.1 samples.
<code>confidence</code>	A figure of credit, $p$ , describing the confidence on the presence of this nuclide in the sample, i.e., there is a probability of $1 - p$ that this nuclide is not present. <i>Note:</i> identification is a binary decision where valid calculations of the statistics are extremely difficult. See the manual of the software producing this number.
<code>effPrecursor</code>	Name of the effective precursor of the nuclide. For syntax see table 3.3 calibrationLibraries.
<code>effHalflife</code>	Effective half-life of the nuclide, i.e., half-life of <code>effPrecursor</code> in seconds [s].
<code>uncEffHalflife</code>	Absolute one sigma uncertainty of <code>effHalflife</code> in seconds [s].
<code>isBackground</code>	True if background efficiency is used to calculate the activity. This is the activity due to surrounding walls etc., i.e., not the activity in the sample. Normally its absolute value is not known and only the shape of the background efficiency curve has been used to facilitate more reliable nuclide identification. This background should not be confused with the peaked background subtraction (PBS) due to background measurement. As a matter of fact, they are incompatible and should not be used together.
<code>actRawLSQ</code>	Raw activity in becquerels [Bq] based on all the peaks of this nuclide. Weighted least squares fitting (LSQ) has been used to obtain the effective peak area. Raw activity is the net peak area multiplied by <code>CCfactor</code> (see table 8.5 lineAssociations), and divided by efficiency, live measuring time and emission probability of the gamma line. The contribution from blank and background has been subtracted. See also note on specific activity on p. 98.
<code>uncActRawLSQ</code>	One sigma absolute uncertainty in becquerels [Bq] of <code>actRawLSQ</code> .

<code>interfCorrLSQ</code>	TRUE if the interference due to other identified nuclides has been accounted for in <code>actRawLSQ</code> .
<code>actRawPriLine</code>	Raw activity in becquerels [Bq] based on the primary line of this nuclide. Raw activity is the net peak area multiplied by <code>CCfactor</code> (see table 8.5 <code>lineAssociations</code> ), and divided by efficiency, live measuring time and emission probability of the gamma line. The contribution from blank and background has been subtracted. See also note on specific activity on p. 98.
<code>uncActRawPriLine</code>	One sigma absolute uncertainty in becquerels [Bq] of <code>actRawPriLine</code> .
<code>interfCorrPriLine</code>	TRUE if the interference due to other identified nuclides has been accounted for in <code>actRawPriLine</code> .
<code>actRawMan</code>	Off-line, possibly manually, calculated raw activity in becquerels [Bq]. See also note on specific activity on p. 98.
<code>uncActRawMan</code>	One sigma absolute uncertainty in becquerels [Bq] of <code>actRawMan</code> .
<code>actRawManMethod</code>	Method used to calculate <code>actRawMan</code> .
<code>actSelect</code>	Identifies the activity used to calculate the final results. The possible values are LSQ: <code>actRawLSQ</code> has been used in table 8.9 <code>finalResults</code> . PRI: <code>actRawPriLine</code> has been used in table 8.9 <code>finalResults</code> . MAN: <code>actRawMan</code> has been used in table 8.9 <code>finalResults</code> . The decision which activity is to be used must be made by the analyst. That is, the analysis software must set this field to NULL. Which gamma lines have been used in the activity calculation can be found in table 8.5 <code>lineAssociations</code> .
<code>acqCorr</code>	Acquisition correction multiplier corrects for decay during spectrum measurement. See <code>acqRealTime</code> in table 8.1 <code>analyses</code> .
<code>uncAcqCorr</code>	One sigma absolute uncertainty of <code>acqCorr</code> .
<code>decayCorr1</code>	Decay correction multiplier #1. See <code>decayTime1</code> in table 8.1 <code>analyses</code> .
<code>uncDecayCorr1</code>	One sigma absolute uncertainty of <code>decayCorr1</code> .
<code>waitCorr</code>	Decay correction multiplier for wait time. See <code>waitTime</code> in table 6.6 <code>measurements</code> .
<code>uncWaitCorr</code>	One sigma absolute uncertainty of <code>waitCorr</code> .
<code>irrCorr</code>	This correction takes into account the activity production rate and decay during irradiation/collection. When applied to the activity at the end of collection it gives the saturation activity of the sample, i.e., the activity assuming infinite irradiation/collection time with the average activity production rate. <code>irrCorr</code> depends on the time profile of the collection rate. If it is not known, a common assumption is a constant rate. See your software documentation for the actual method used.
<code>uncIrrCorr</code>	One sigma absolute uncertainty of <code>irrCorr</code> .
<code>collCorr</code>	This correction takes into account the activity production/collection rate and decay during collection. When applied to the activity at the end of collection it gives the total activity collected. <code>collCorr</code> depends on the time profile of the collection rate. If it is not known, a common assumption is a constant rate. See your software documentation for the actual method used.

<code>uncCollCorr</code>	One sigma absolute uncertainty of <code>collCorr</code> .
<code>decayCorr2</code>	Decay correction multiplier #2. See <code>decayTime2</code> in table <a href="#">8.1 analyses</a> .
<code>uncDecayCorr2</code>	One sigma absolute uncertainty of <code>decayCorr2</code> .
<code>comments</code>	Comments in English [en].

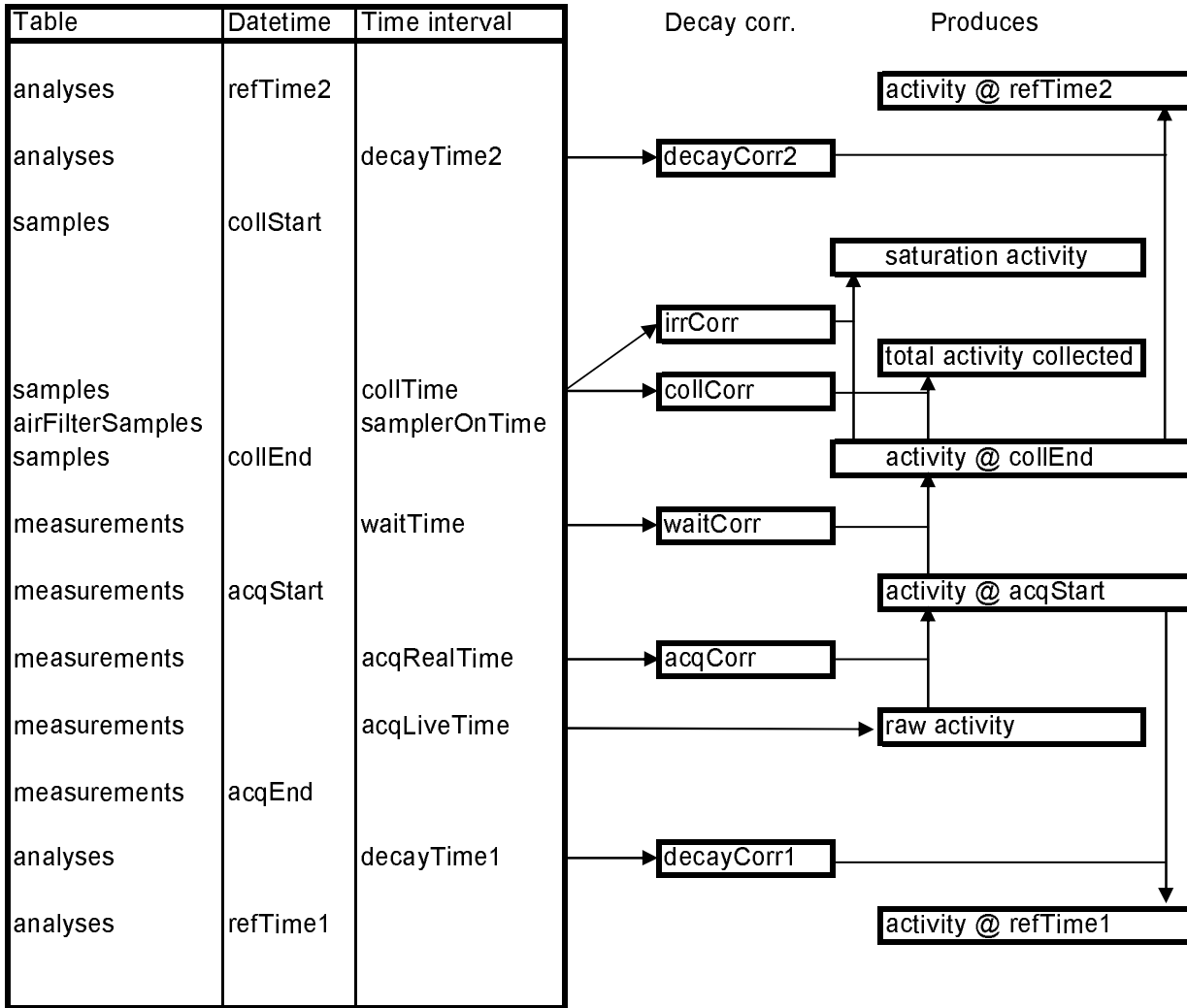


Figure 8.1: Decay corrections in *Linssi*. The time intervals used to calculate the various corrections are shown in the left hand part of the figure together with the times and dates from which they are calculated. It also includes the names of the tables where the corresponding fields can be found. The right hand part of the figure shows the decay correction factors in table `activities` and how they are used to obtain the decay corrected activities at given dates and times. The activity at the point of an arrow is obtained by **multiplying** the activity at the tail of the arrow with the correction factor shown.

*Note 1:* Raw activity, saturation activity and total activity collected are not activities at any specific date and time.

*Note 2:* `collCorr` and `irrCorr` depend on the shape of the collection rate profile and generally cannot be calculated from `collTime` and `samplerOnTime` alone.

*Note 3:* `decayTime1` and `decayTime2` may be positive or negative. The other times, by definition, are positive.

*Note 4:* If the decay chains are in equilibrium, the decay corrections can be easily obtained from the effective half-lives of the identified nuclides. In non-equilibrium chains the effective half-lives do not exist. Depending on the analysis software the correction factors may still be correct. Consult your software manual.

## 8.6 Activity Limits

Table 8.7: Activity limits and minimum detectable activities

activityLimits			
Field	Type	Length	Flags
idAnalysis	int	4	PFN
nuclideId	varchar	10	PINc
idMeas	int	4	IFN
idSample	int	4	IFN
energyPriLine	double	8	
area	double	8	
uncArea	double	8	
baselineArea	double	8	
uncBaselineArea	double	8	
decisionLimit	double	8	
detectionLimit	double	8	
significance	double	8	
mdaRaw	double	8	
effPrecursor	varchar	10	c
effHalflife	double	8	
uncEffHalflife	double	8	
acqCorr	double	8	
uncAcqCorr	double	8	
decayCorr1	double	8	
uncDecayCorr1	double	8	
waitCorr	double	8	
uncWaitCorr	double	8	
irrCorr	double	8	
uncIrrCorr	double	8	
collCorr	double	8	
uncCollCorr	double	8	
decayCorr2	double	8	
uncDecayCorr2	double	8	
comments	text	0	
(idAnalysis ) → analyses(idAnalysis)			
(idMeas ) → measurements(idMeas)			
(idSample ) → samples(idSample)			

*end of table.*

There are many methods to calculate minimum detectable activities (MDA) and the manual of the software performing the analysis should always be consulted for exact meaning of the fields below. It should be noted that the peak area may be based on the peak actually found in peak analysis or its calculation may have been forced by the MDA-algorithm itself. It is also possible that background peaks or interfering nuclides have been taken into account in the process of defining the peak and baseline areas.



<code>idAnalysis</code>	Identification number of the analysis. See table 8.1 analyses.
<code>nuclideId</code>	Name of the nuclide for which this MDA is calculated. <i>Note:</i> this nuclide does not need to be a nuclide identified in the spectrum. For identified nuclides more data can be found in table 8.6 activities. For syntax see table 3.3 calibrationLibraries.
<code>idMeas</code>	Identification number of the measurement. See table 6.6 measurements.
<code>idSample</code>	Identification number of the sample measured. See table 4.1 samples.
<code>energyPriLine</code>	Energy of the primary line of this nuclide in kiloelectronvolts [keV]. <i>Note:</i> the primary line is not necessary the one with the highest emission probability. The primary line is defined by the analysis software on the basis of maximum ‘visibility’ in this analysis. The MDA is calculated on the basis of this line.
<code>area</code>	Area of the peak at energy <code>energyPriLine</code> in counts.
<code>uncArea</code>	Absolute one sigma uncertainty of <code>area</code> in counts.
<code>baselineArea</code>	Area of the baseline at energy <code>energyPriLine</code> in counts used to calculate activity limits. <i>Note:</i> this is NOT the <code>baselineArea</code> of table 8.3 peaks.
<code>uncBaselineArea</code>	Absolute one sigma uncertainty of <code>baselineArea</code> in counts.
<code>decisionLimit</code>	Decision limit in counts. This corresponds the value of $\alpha$ in 8.1 analyses.
<code>detectionLimit</code>	Detection limit in counts. This corresponds the value of $\beta$ in 8.1 analyses.
<code>significance</code>	Peak area in the units of decision limit, i.e., <code>area</code> divided by <code>decisionLimit</code> .
<code>mdaRaw</code>	Raw minimum detectable activity in becquerels [Bq]. It is based on the minimum detectable peak area of the primary line of this nuclide. Raw activity is the net peak area multiplied by <code>CCfactor</code> (see table 8.5 lineAssociations), and divided by efficiency, live measuring time and emission probability of the gamma line. The contribution from blank and background has been subtracted. See also note on specific activity on p. 98.
<code>effPrecursor</code>	Name of the effective precursor of the nuclide. For syntax see table 3.3 calibrationLibraries.
<code>effHalflife</code>	Effective half-life of the nuclide, i.e., half-life of <code>effPrecursor</code> in seconds [s].
<code>uncEffHalflife</code>	Absolute one sigma uncertainty of <code>effHalflife</code> in seconds [s].
<code>acqCorr</code>	Acquisition correction multiplier corrects for decay during spectrum measurement. See <code>acqRealTime</code> in table 8.1 analyses.
<code>uncAcqCorr</code>	One sigma absolute uncertainty of <code>acqCorr</code> .
<code>decayCorr1</code>	Decay correction multiplier #1. See <code>decayTime1</code> in table 8.1 analyses.
<code>uncDecayCorr1</code>	One sigma absolute uncertainty of <code>decayCorr1</code> .
<code>waitCorr</code>	Decay correction multiplier for wait time. See <code>waitTime</code> in table 6.6 measurements.
<code>uncWaitCorr</code>	One sigma absolute uncertainty of <code>waitCorr</code> .

<code>irrCorr</code>	This correction takes into account the activity production rate and decay during irradiation/collection. When applied to the activity at the end of collection it gives the saturation activity of the sample, i.e., the activity assuming infinite irradiation time with the average activity production rate. <code>irrCorr</code> depends on the time profile of the collection rate. If it is not known, a common assumption is a constant rate. See your software documentation for the actual method used.
<code>uncIrrCorr</code>	One sigma absolute uncertainty of <code>irrCorr</code> .
<code>collCorr</code>	This correction takes into account the activity production/collection rate and decay during collection. When applied to the activity at the end of collection it gives the total activity collected. <code>collCorr</code> depends on the time profile of the collection rate. If it is not known, a common assumption is a constant rate. See your software documentation for the actual method used.
<code>uncCollCorr</code>	One sigma absolute uncertainty of <code>collCorr</code> .
<code>decayCorr2</code>	Decay correction multiplier #2. See <code>decayTime2</code> in table 8.1 analyses.
<code>uncDecayCorr2</code>	One sigma absolute uncertainty of <code>decayCorr2</code> .
<code>comments</code>	Comments in English [en].

### Note on specific activity

The activities in table 8.6 **activities** are raw sample activities in Bq. There are, however, cases where sample activities cannot be defined. For example, an *in situ* measurement of the surface activity of soil from the distance of 1 m above the ground does not give the total surface activity of the globe in Bq. In cases like this, efficiency calibration must be performed to give directly the specific activity, in this case activity per area at the measuring position, the unit being Bq/m<sup>2</sup>. Accordingly, the MDA-values in the above table must refer to the corresponding specific activity.

What is the denominator in the unit of the specific activity depends on the specific sample production model and is defined in the sample production group of tables, (Ch. 3). In the case where the MDA is the real activity in Bq, i.e., not the specific activity, the quantity and its unit needed to divide the raw MDA to get the raw minimum detectable specific activity are given in table 4.1 **samples**.

## 8.7 Nuclide Ratios

Table 8.8: Activity ratios of relevant nuclide pairs

nuclideRatios			
Field	Type	Length	Flags
idAnalysis	int	4	PF12N
firstNuclideId	varchar	10	PF1Nc
secondNuclideId	varchar	10	PF2Nc
idMeas	int	4	IFN
idSample	int	4	IFN
firstIsDaughter	Boolean	1	
secondIsDaughter	Boolean	1	
firstHalflife	double	8	
uncFirstHalflife	double	8	
secondHalflife	double	8	
uncSecondHalflife	double	8	
netBranching	double	8	
uncNetBranching	double	8	
refRatio	double	8	
uncRefRatio	double	8	
zeroRatio	double	8	
uncZeroRatio	double	8	
refTime	datetime	8	
zeroTime	datetime	8	
uncZeroTimeLow	double	8	
uncZeroTimeHigh	double	8	
comments	text	0	

(idMeas ) → measurements(idMeas)  
 (idSample ) → samples(idSample)  
 (idAnalysis ,firstNuclideId ) → activities(idAnalysis,nuclideId)  
 (idAnalysis ,secondNuclideId ) → activities(idAnalysis,nuclideId)

*end of table.*

This table contains the information of activity ratios of nuclide pairs that facilitates calculation of the birth time of activity, **zeroTime**. The nuclides may belong to the same decay chain or decay independently. If they decay independently, it is, of course, necessary to have *á priori* information on their relative yields, **zeroRatio**, at time zero.

<b>idAnalysis</b>	Identification number of the analysis. See table 8.1 analyses.
<b>firstNuclideId</b>	Name of the first nuclide. See table 8.6 activities. For syntax see table 3.3
<b>secondNuclideId</b>	Name of the second nuclide. See table 8.6 activities. For syntax see table 3.3
<b>idMeas</b>	Identification number of the measurement. See table 6.6 measurements.
<b>idSample</b>	Identification number of the sample measured. See table 4.1 samples.

<code>firstIsDaughter</code>	The first nuclide follows the second in the decay chain.
<code>secondIsDaughter</code>	The second nuclide follows the first in the decay chain. <i>Note 1:</i> Both <code>firstIsDaughter</code> and <code>secondIsDaughter</code> may be false, but both cannot be true. <i>Note 2:</i> If one is true the other one is a parent, not necessarily a direct parent.
<code>firstHalflife</code>	Half-life of the first nuclide in seconds [s].
<code>uncFirstHalflife</code>	Absolute one sigma uncertainty of <code>firstHalflife</code> in seconds [s].
<code>secondHalflife</code>	Half-life of the second nuclide in seconds [s].
<code>uncSecondHalflife</code>	Absolute one sigma uncertainty of <code>secondHalflife</code> in seconds [s].
<code>netBranching</code>	Net decay branching from parent to daughter through the decay chain, i.e., the product of branching fractions leading from parent to daughter.
<code>uncNetBranching</code>	One sigma absolute uncertainty of <code>netBranching</code> .
<code>refRatio</code>	Activity of the first nuclide divided by the activity of the second nuclide at <code>refTime</code> .
<code>uncRefRatio</code>	One sigma absolute uncertainty of <code>refRatio</code> .
<code>zeroRatio</code>	<i>A priori</i> assumed initial activity ratio between the first and the second nuclide at <code>zeroTime</code> . If not given, the nuclides must belong to the same decay chain, i.e., either <code>firstIsDaughter</code> or <code>secondIsDaughter</code> must be true. In that case it is assumed that the activity of the daughter is zero at <code>zeroTime</code> .
<code>uncZeroRatio</code>	One sigma absolute uncertainty of <code>zeroRatio</code> .
<code>refTime</code>	Reference date and time at which the <code>refRatio</code> is calculated. <i>Note:</i> If the half-lives are short when compared to the acquisition time, the decay corrections of activities to <code>refTime</code> should take the non-equilibrium of the decay chain into account. See Fig. 8.1.
<code>zeroTime</code>	Date and time when the initial activity was created.
<code>uncZeroTimeLow</code>	Uncertainty towards past of the <code>zeroTime</code> in seconds [s].
<code>uncZeroTimeHigh</code>	Uncertainty towards future of the <code>zeroTime</code> in seconds [s]. <i>Note:</i> Since the uncertainty of <code>zeroTime</code> is not necessarily symmetric, there are two uncertainties defined. The time together with the corresponding uncertainties is given as <code>zeroTime(-uncZeroTimeLow,+uncZeroTimeHigh)</code> , e.g., 2004-09-08 22:34:28 (-32000.0s,+71000.0s). The uncertainties correspond to one sigma absolute Gaussian uncertainties, i.e., 34% of the probability mass is between <code>zeroTime-uncZeroTimeLow</code> and <code>zeroTime</code> as well as between <code>zeroTime</code> and <code>zeroTime+uncZeroTimeHigh</code> .
<code>comments</code>	Comments in English [en].

## 8.8 Final Analysis Results

Table 8.9: Final analysis results

finalResults			
Field	Type	Length	Flags
idAnalysis	int	4	PFN
idMeas	int	4	IFN
idSample	int	4	IFN
completionTime	timestamp	4	
category	smallint	2	R
categoryReason	text	0	R
facilityId	varchar	80	F
contactPerson	varchar	40	I
analysisStatus	varchar	20	R
purpose	text	0	R
testType	text	0	R
comparison	text	0	R
conclusions	text	0	R
projectFile	longblob	0	
lowQualitySpectrum	Boolean	1	
doseRate	double	8	
uncDoseRate	double	8	
refResults	text	0	
comments	text	0	
(idAnalysis ) → analyses(idAnalysis)			
(idMeas ) → measurements(idMeas)			
(idSample ) → samples(idSample)			
(facilityId ) → facilities(facilityId)			

*end of table.*

idAnalysis	Identification number of the analysis. See table 8.1 analyses.
idMeas	Identification number of the measurement. See table 6.6 measurements.
idSample	Identification number of the sample measured. See table 4.1 samples.
completionTime	Date and time when these results were stored in the database.
category	Facility specific.
categoryReason	Motivations for category.
facilityId	Identifier of the facility that performed the final analysis. See table 2.1 facilities.
contactPerson	Contact person for the final results. See table 2.1 facilities.
analysisStatus	Status of the analysis. Valid values are: Preliminary or Final.
purpose	Facility specific.
testType	Facility specific.
comparison	Facility specific.
conclusions	Facility specific.

<code>projectFile</code>	File containing the full project information. The contents are software specific. Typically this would be a script able to reconstruct all the final analysis results for the given sample and measurement. Alternatively this may be the name of the project file, or URI, where project information is to be found.
<code>lowQualitySpectrum</code>	Set TRUE if the analyst has deemed the spectrum as being of low quality.
<code>doseRate</code>	Dose rate in Sieverts per second [Sv/s].
<code>uncDoseRate</code>	Uncertainty of the dose rate in Sieverts per second [Sv/s].
<code>refResults</code>	Description and availability of reference results are described here.
<code>comments</code>	Comments in English [en].

# Chapter 9

## Functions

### 9.1 Functions

Table 9.1: Functions

functions			
Field	Type	Length	Flags
<code>idFunction</code>	int	4	P
<code>functionNameId</code>	varchar	40	UR
<code>comments</code>	text	0	

*end of table.*

This table is a link between the function definitions, table `functionDefs`, and function reference, `idFunction`, in other tables.

<code>idFunction</code>	Unique identification number of the predefined function. See list of Ch. <a href="#">9.3</a> .
<code>functionNameId</code>	Unique standardized name of the function. See list of Ch. <a href="#">9.3</a> .
<code>comments</code>	Comments in English [en].

## 9.2 Function Definitions

Table 9.2: Function definitions

functionDefs			
Field	Type	Length	Flags
<code>idFunction</code>	int	4	PFR
<code>idFunctionDef</code>	int	4	PR
<code>functionDef</code>	text	0	
<code>comments</code>	text	0	

(`idFunction` )  $\rightarrow$  functions(`idFunction`)

*end of table.*

This table facilitates multiple presentations of any specific function `idFunction`.

<code>idFunction</code>	Unique function number. See table 9.1 functions.
<code>idFunctionDef</code>	Defines the type of definition given in <code>functionDef</code> . The current types are: = 0 or NULL No function definition given. = 1, Function definition is given in free form text in English [en]. = 2, Function definition is given in MathML, content markup. = 3, Function definition is given in MathML, presentation markup. = 4, Function definition is given in L <sup>A</sup> T <sub>E</sub> X. = 5, Function definition is given in T <sub>E</sub> X.
<code>functionDef</code>	Function definition in the form defined by <code>idFunctionDef</code> .
<code>comments</code>	Comments in English [en].



## 9.3 Function List

The reserved functions are selected by the field `idFunction` in the table referencing the function. Currently (*Linssi* 2.3) these functions are used for peak and baseline shape definitions in table 8.3 `peaks` and for calibration curves in table 5.2 `calTypes`. The parameter values needed for function evaluation at each point,  $x$ , are given in the field `parameters` of the table referencing the function. Their values are given as pairs of value and its one sigma absolute uncertainty. Each pair is separated with the newline character. If the parameter uncertainty is not known it must not be given. It is recommended that known zero uncertainties, which is the case for integers, for example, should be explicitly stated. The pairs are associated with the function in the order given in the function definitions below. Note that the parameter uncertainties are not used in the function definitions. However, their values, together with the definitions, may be used by the analysis software to calculate the uncertainties of the evaluated function values.

The `idFunction` values from 0 to 1000 are reserved for registered use of *Linssi* community. This is to avoid clashes between function names. For your private use you should use values greater than 1000. It is, however, recommended that you send your function definitions to us and we add them to the list of registered functions. The current registered functions are:

`idFunction: 1`

`functionNameId: Data points`

`parameter order: No parameters`

Only data points are available, i.e., no functional form of data is given. Interpolation between the data points should be used. Depending on the software and the type of data, different types of interpolation may be used, i.e., linear, quadratic, logarithmic etc.

`idFunction: 2`

`functionNameId: Polynomial`

`parameter order:  $a_0, a_1, a_2, \dots, a_n$`

$$y(x) = \sum_{i=0}^n a_i x^i \quad (9.1)$$

`idFunction: 3`

`functionNameId: Square root polynomial`

`parameter order:  $a_0, a_1, a_2, \dots, a_n$`

$$y(x) = \sum_{i=0}^n a_i x^{i/2} \quad (9.2)$$

`idFunction: 4`

`functionNameId: Square root of polynomial`

`parameter order:  $a_0, a_1, a_2, \dots, a_n$`

$$y(x) = \sqrt{\sum_{i=0}^n a_i x^i} \quad (9.3)$$

idFunction: 5

functionNameId: **Exponential rollover**

parameter order:  $a_0, a_1, a_2, a_3, a_4$

$$y(x) = a_0 \exp\left(\frac{-a_1}{x}\right)^{a_3} \left[1 - \exp\left(\frac{-a_2}{x}\right)^{a_4}\right] \quad (9.4)$$

idFunction: 6

functionNameId: **Polynomial in  $\ln y$  against  $\ln x$**

parameter order:  $a_0, a_1, a_2, \dots, a_n$

$$\ln y(x) = \sum_{i=0}^n a_i [\ln x]^i \quad (9.5)$$

idFunction: 7

functionNameId: **Polynomial in  $\ln y$  against  $x$**

parameter order:  $a_0, a_1, a_2, \dots, a_n$

$$\ln y(x) = \sum_{i=0}^n a_i x^{1-i} \quad (9.6)$$

idFunction: 8

functionNameId: **Polynomial in  $\ln y$  against  $1/x$**

parameter order:  $a_0, a_1, a_2, \dots, a_n$

$$\ln y(x) = \sum_{i=0}^{n-1} a_i \left[\ln \frac{a_n}{x}\right]^i \quad (9.7)$$

idFunction: 9

functionNameId: **Inverse exponential**

parameter order:  $a_0, a_1, a_2, a_3$

$$y(x) = \frac{1}{a_0 x^{-a_2} + a_1 x^{-a_3}} \quad (9.8)$$

idFunction: 10

functionNameId: **Gaussian**

parameter order:  $c, \sigma$

$$y(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-c)^2}{2\sigma^2}\right) \quad (9.9)$$

idFunction: 11

functionNameId: **Sampo peak function**

parameter order:  $c, p, \sigma, l, h, s$

$$y(x) = \begin{cases} p \exp \frac{-(x-c)^2}{2\sigma^2} + s \operatorname{erfc} \frac{(x-c)}{\sqrt{2}\sigma} & \text{if } -l \leq x - c \leq h \\ p \exp \frac{l(2(x-c)+l)}{2\sigma^2} + s \operatorname{erfc} \frac{(x-c)}{\sqrt{2}\sigma} & \text{if } x - c < -l \\ p \exp \frac{h(-2(x-c)+h)}{2\sigma^2} + s \operatorname{erfc} \frac{(x-c)}{\sqrt{2}\sigma} & \text{if } x - c > h \end{cases} \quad (9.10)$$

idFunction: 12

functionNameId: **General polynomial**

parameter order:  $n, c, a_0, a_1, \dots, a_n$

$$y(x) = \sum_{i=0}^n a_i (x - c)^i \quad (9.11)$$

idFunction: 13

functionNameId: **General square root polynomial**

parameter order:  $n, c, a_0, a_1, \dots, a_n$

$$y(x) = \sum_{i=0}^n a_i (x - c)^{i/2} \quad (9.12)$$

idFunction: 14

functionNameId: **General square root of polynomial**

parameter order:  $n, c, a_0, a_1, \dots, a_n$

$$y(x) = \sqrt{\sum_{i=0}^n a_i (x - c)^i} \quad (9.13)$$

idFunction: 15

functionNameId: **General exponential rollover**

parameter order:  $c, a_0, a_1, a_2, a_3, a_4$

$$y(x) = a_0 \exp \left( \frac{-a_1}{x - c} \right)^{a_3} \left[ 1 - \exp \left( \frac{-a_2}{x - c} \right)^{a_4} \right] \quad (9.14)$$

idFunction: 16

functionNameId: **General polynomial in  $\ln y$  against  $\ln x$**

parameter order:  $n, c, a_0, a_1, \dots, a_n$

$$\ln y(x) = \sum_{i=0}^n a_i [\ln(x - c)]^i \quad (9.15)$$

idFunction: 17

functionNameId: **General polynomial in  $\ln y$  against  $x$**

parameter order:  $n, c, a_0, a_1, \dots, a_n$

$$\ln y(x) = \sum_{i=0}^n a_i (x - c)^{1-i} \quad (9.16)$$

idFunction: 18

functionNameId: **General polynomial in  $\ln y$  against  $1/x$**

parameter order:  $n, c, a_0, a_1, \dots, a_n$

$$\ln y(x) = \sum_{i=0}^{n-1} a_i \left[ \ln \frac{a_n}{x - c} \right]^i \quad (9.17)$$

idFunction: 19

functionNameId: **General inverse exponential**

parameter order:  $c, a_0, a_1, a_2, a_3, a_4$

$$y(x) = \frac{1}{a_0(x - c)^{-a_2} + a_1(x - c)^{-a_3}} \quad (9.18)$$

idFunction: 93

functionNameId: **HAE efficiency 1-3**

parameter order:  $S, E_1, k, E_3, n$

$$y(x) = S f_1(x) f_3(x), \quad (9.19)$$

where  $f_1$  and  $f_3$  are the same as in function 95.

idFunction: 94

functionNameId: **HAE efficiency 1-2**

parameter order:  $S, E_1, k, E_2, b, m$

$$y(x) = S f_1(x) f_2(x), \quad (9.20)$$

where  $f_1$  and  $f_2$  are the same as in function 95.

idFunction: 95

functionNameId: **HAE efficiency 1-2-3**

parameter order:  $S, E_1, k, E_2, b, m, E_3, n$

$$y(x) = S f_1(x) f_2(x) f_3(x), \quad (9.21)$$

where

$$f_1(x) = e^{-\left(\frac{E_1}{x}\right)^k} \quad (9.22)$$

$$f_2(x) = \begin{cases} 1 - e^{-b\left(\frac{-E_2}{x}\right)^m} & \text{if } x > E_2 \\ 1 & \text{if } x \leq E_2 \end{cases} \quad (9.23)$$

$$f_3(x) = \begin{cases} 1 - e^{-\left(\frac{2E_3}{x-E_3}\right)^n} & \text{if } x > E_3 \\ 1 & \text{if } x \leq E_3 \end{cases} \quad (9.24)$$

idFunction: 96

functionNameId: **Inverse power**

parameter order:  $a_0, a_1, a_2$

$$y(x) = a_0 + \left(\frac{x}{a_1}\right)^{-a_2} \quad (9.25)$$

idFunction: 97

functionNameId: **Exponential sum**

parameter order:  $n, a_0, a_1, \lambda_1, a_2, \lambda_2, a_3, \lambda_3 \dots, a_n, \lambda_n$

$$y(x) = a_0 + \sum_{i=1}^n a_i e^{\lambda_i x} \quad (9.26)$$

idFunction: 99

functionNameId: **Linear with cut-off**

parameter order:  $a_0, a_1, a_2$

$$y(x) = \begin{cases} a_0 + a_1 x & \text{if } x \geq a_1 \\ 0 & \text{if } x < a_1 \end{cases} \quad (9.27)$$

### 9.3.1 MathML presentation of the functions

Mathematical Markup Language, MathML [10], is a markup language defined using Extensible Markup Language, XML [11]. If `idFunctionDef=2` a MathML presentation of the function definition is stored into `functionDef`. The parameters given in the table referencing the function are used to evaluate the MathML function value. The MathML function itself shall be encapsulated as

```
<mathml xmlns:z="http://www.w3.org/1998/Math/MathML" >
...
</mathml>
```

where the `http`-reference is to the applicable schema of MathML itself and `z` is the namespace prefix adopted in *Linssi*. These are the only requirements set by *Linssi* database specifications.

### 9.3.2 MathML support in Shaman

Even though *Linssi* sets minimal requirements for MathML functions, it should be pointed out that analysis software may set more stringent limits on the MathML features it is able to take advantage of. As far as we know, currently the only analysis program able to use MathML input is SHAMAN [5]. It uses a MathML subset briefly discussed here.

In SHAMAN lambda calculus is used to evaluate the function values. The parameters given in the table referencing the function are associated with **bvar**'s of the lambda construct. The association is by order, i.e., the first parameter, refers to the first **bvar**. The last **bvar** is the independent variable. A function definition starts with a list of **bvar**'s followed by a container tag **apply**, **ci**, **cn**, **or**, **piecewise**, which must evaluate to a single value.

The following MathML tags are supported in SHAMAN:

**Qualifier:** **bvar**

**Arithmetic and functions:** **plus**, **minus**, **times**, **divide**, **power**, **root** (degree tag not supported, hence only square root available), **exp**, **ln**, **sin**, **cos**, **tan**

**Binary forms of logical operators:** **eq**, **gt**, **lt**, **geq**, **leq**

**Logical operators:** **neq**, **not**, **and**, **or**

**Containers:** **lambda**, **ci**, **cn**, **apply**, **piecewise**, **piece**, **otherwise** (Type attributes of **ci** and **cn** are not supported, hence always treated as real.)

The application producing the calibration function in MathML lambda notation is responsible for the correctness and numerical robustness of the function. While SHAMAN does perform some rudimentary error checking on the function before use, the conversion of the MathML function may involve some reordering of the function terms, which while mathematically equivalent, may cause numerically unstable functions to break.

An example of a linear function,  $a+bx$ , is presented in the lambda calculus of MathML below. Assuming, for example, that the two parameters have values of 2.1 and 1.2, respectively, the function below evaluates to  $2.1 + 1.2x$ .

```
<mathml xmlns:z="http://www.w3.org/1998/Math/MathML" >
  <z:lambda>
    <z:bvar>
      <z:ci>a</z:ci>
    </z:bvar>
    <z:bvar>
      <z:ci>b</z:ci>
    </z:bvar>
    <z:bvar>
      <z:ci>x</z:ci>
    </z:bvar>
    <z:apply>
      <z:plus/>
      <z:ci>a</z:ci>
      <z:apply>
        <z:times/>
        <z:ci>b</z:ci>
        <z:ci>x</z:ci>
      </z:apply>
    </z:apply>
  </z:lambda>
```

</mathml>

See SHAMAN input parser manual for more details on how to write these lambda constructs [\[12\]](#).





# Chapter 10

## Connections

### 10.1 Database Information

Table 10.1: Database information

linssiInfo			
Field	Type	Length	Flags
<code>infoId</code>	<code>varchar</code>	20	PN
<code>value</code>	<code>text</code>	0	
<code>comments</code>	<code>text</code>	0	

*end of table.*

The purpose of this table is to provide meta information about the *Linssi* database itself.

- `infoId` A unique identifier of the `value` field. Reserved `infoId`:s are:  
DATABASEID, `value` contains the name of the database.  
VERSION, `value` contains *Linssi* version number of the database.  
BACKUP, `value` contains the full backup contents of the database.  
BACKUPDATE, `value` contains the date of the backup.
- `value` The data defined by `infoId`.
- `comments` Comments in English [en].

## 10.2 Messages

Table 10.2: Messages

messages			
Field	Type	Length	Flags
idMessage	int	4	PA
messageId	varchar	20	NR
senderFacilityId	varchar	80	F
senderContactPerson	varchar	40	
receiverFacilityId	varchar	80	F
receiverContactPerson	varchar	40	
idSample	int	4	F
idMeas	int	4	F
idAnalysis	int	4	F
reportNumber	int	4	
messageType	varchar	40	R
messageDataType	varchar	40	R
reportStatus	varchar	8	R
refMessageId	varchar	80	
refMessageFacilityId	varchar	80	F
transmission	datetime	8	
receipt	datetime	8	
messageBody	longblob	0	
dataVersion	varchar	20	R
dataVersionText	text	0	
receivedFrom	varchar	80	
sentTo	text	0	
ccTo	text	0	
authentication	Boolean	1	
authenticationPass	Boolean	1	
comments	text	0	
(senderFacilityId ) → facilities(facilityId) (receiverFacilityId ) → facilities(facilityId) (idSample ) → samples(idSample) (idMeas ) → measurements(idMeas) (idAnalysis ) → analyses(idAnalysis) (refMessageFacilityId ) → facilities(facilityId)			

*end of table.*

The purpose of this table is to keep track of the messages exchanged between facilities. Our original use of this table has been to track messages between certified laboratories and the CTBTO. However, the description given here should be fully adequate for facilities wishing to set up their own message passing protocols on the basis of this table.

If the user needs to fulfill the requirements of the messaging protocol of the the CTBTO, the user is asked to consult the report IDC-3.4.1Rev6 [9] to which the descriptions below are referencing.

The reference RLR/#xxx below refers to the data block #xxx of the radionuclide laboratory report (RLR). RLR and the related data blocks are described in Chapter ‘Radionuclide Laboratory Reports’ of the report IDC3.4.1Rev6 [9].

idMessage	Linssi identification number of the message.
messageId	Identification code of the message as represented in the message itself. For CTBT messages see id string of MSG ID in Chapter Message Structure of IDC3.4.1Rev6.
senderFacilityId	Site code of facility, where the message is originated. For CTBT messages see source in MSG ID in Chapter Message Structure of IDC3.4.1Rev6. See table 2.1 facilities.
senderContactPerson	contact person of the message sender.
receiverFacilityId	facilityId of the receiver of the message. See table 2.1 facilities.
receiverContactPerson	contact person of the message receiver.
idSample	Identification number of the sample measured. See table 4.1 samples.
idMeas	Identification number of the measurement, i.e., spectrum analyzed. See table 6.6 measurements.
idAnalysis	Identification number of the analysis. See table 8.1 analyses.
reportNumber	Number of the report. for CTBT messages see RLR/#Header.
messageType	Message type, i.e. for CTBT: request, subscription, data, lab-data, command_request, or command_response. See MSG TYPE in Chapter Message Structure of IDC3.4.1Rev6 [6].
messageDataType	Message data type. for CTBT see DATA TYPE in Chapter Message Structure and the specific radionuclide data types in Chapter Radionuclide Messages in IDC3.4.1Rev6 [6].
reportStatus	Status of the report i.e. PREL, FIN.
refMessageId	A reference to the message for which this message is a response to, i.e., for CTBT messages the ref str of the REF ID. See MSG TYPE and REF ID in Chapter Message Structure of IDC3.4.1Rev6 [6].
refMessageFacilityId	A reference to the message for which this message is a response to, i.e., for CTBT messages the ref src in REF ID in Chapter Message Structure of IDC3.4.1Rev6 [6]. See table 2.1 facilities.
transmission	Date and time of the transmission of the message (UTC).
receipt	Date and time of the receipt of the message (UTC).
messageBody	The full text of the message.
dataVersion	Version of message format used. For CTBT see RLR/#LabDataVersion.
dataVersionText	Notes regarding the message format. For CTBT see RLR/#LabDataVersion.
receivedFrom	Email address of the sender (From:).
sentTo	Email address of the recipient (To:).
ccTo	Email address of additional recipients (Cc:).
authentication	True if the message was authenticated.
authenticationPass	True if the authentication passed the checking.
comments	Comments in English [en].

## 10.3 External Keys

Table 10.3: External keys

externalKeys			
Field	Type	Length	Flags
keyId	varchar	80	PNc
databaseId	varchar	80	PN
keyType	varchar	20	PNR
idKey	int	4	N
comments	text	0	

*end of table.*

When data from an external *Linssi* database are imported to the local database, associations between the two instances of the same data must be retained. This is normally accomplished by using identical identifiers (**keyId**'s) for the same data in both databases. This requires that the **keyId**'s are unique across the databases. For this purpose it is sufficient to follow the agreed naming conventions for **sampleId**, **measId**, **calId** and **analysisId**, i.e., the table **externalKeys** is not needed. See *Linssi* scripts and interfaces manual [3].

However, if there were lots of common instances between the databases and also a need for oral discussion about them, the discussion would be easier when using the numeric **idKey**'s instead. For this purpose the table **externalKeys** is useful. It associates the local (and external) **keyId**'s with the external **idKey**'s facilitating the use of numbers in the communication. If a corresponding association is also needed by the users of the external database, **idKey**'s of the local database should be sent there to be used in their **externalKeys** table.

<b>keyId</b>	Identifier of the key in the local and external <i>Linssi</i> databases.
<b>databaseId</b>	A unique identifier of the external <i>Linssi</i> database.
<b>keyType</b>	Type of of the key. Reserved types: SAMPLE, <b>idKey</b> equals <b>idSample</b> of the external <i>Linssi</i> database. MEAS, <b>idKey</b> equals <b>idMeas</b> of the external <i>Linssi</i> database. CAL, <b>idKey</b> equals <b>idCal</b> of the external <i>Linssi</i> database. ANALYSIS, <b>idKey</b> equals <b>idAnalysis</b> of the external <i>Linssi</i> database.
<b>idKey</b>	Value of the key in the external <i>Linssi</i> database.
<b>comments</b>	Comments in English [en].

# Bibliography

- [1] *Linssi* – SQL Database for Gamma-Ray Spectrometry, Part I: DATABASE, Version 1.1, Report TKK-F-A841, Helsinki University of Technology, Espoo, Finland, 2006.
- [2] *Linssi* – SQL Database for Gamma-Ray Spectrometry, Part II: SCRIPTS AND INTERFACES, Version 1.1, Report TKK-F-A842, Helsinki University of Technology, Espoo, Finland, 2006.
- [3] *Linssi* – SQL Database for Gamma-Ray Spectrometry, Part II: SCRIPTS AND INTERFACES, Version 2.3, Report TKK-F-A862, Aalto University, Espoo, Finland, 2011.
- [4] UniSAMPO - Advanced Gamma Spectrum Analysis Software, Version 2.4, User's Guide. 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 2007.
- [6] User Manual of Radionuclide Analysis and Evaluation Software Aatami, version 4.10, Comprehensive Nuclear-Test-Ban Treaty Organization, Office of the Executive Secretary, Evaluation Section, Vienna, 2007.
- [7] <http://www.mysql.com>
- [8] Le Système international d'unités (SI), 7e édition 1998, Organisation intergouvernementale de la Convention du Mètre, Stedi Paris, ISBN 92-822-2154-7.
- [9] Formats and Protocols for Messages, IDC-3.4.1 Revision 6, IDC Documentation.
- [10] Mathematical Markup Language (MathML) Version 2.0 (Second Edition), W3C Recommendation 21 October 2003, <http://www.w3.org/TR/2003/REC-MathML2-20031021/>
- [11] Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000, <http://www.w3.org/TR/2000/REC-xml-20001006>
- [12] XML Input for Expert System Shaman, User's Guide version 1.0, Baryon Oy, Ltd. Espoo, Finland. 2003.





ISBN 978-952-60-3262-7 (printed)  
ISBN 978-952-60-3581-9 (pdf)  
ISSN 1456-3320