

---

# RCM-DX Specification

Version 2.0

Swiss federal railways SBB, Schweizerische  
Bundesbahnen SBB, Chemins de fer fédéraux suisses  
CFF, Ferrovie federali svizzere FFS



## Contents

<b>1</b>	<b>Specification of the RCM-DX Format version 2.0</b>	<b>4</b>
1.1	History . . . . .	4
1.2	Approval . . . . .	5
1.3	Introduction . . . . .	6
1.3.1	Motivation . . . . .	6
1.3.2	Hints . . . . .	6
1.4	Definitions . . . . .	7
1.4.1	File names . . . . .	7
1.4.2	Primitive and extended data types . . . . .	8
1.4.3	(HDF5) Group . . . . .	8
1.4.4	(HDF5) Attribut . . . . .	9
1.4.5	(HDF5) Datasets . . . . .	9
1.5	Data structures . . . . .	11
1.5.1	Array . . . . .	11
1.5.2	Preview data . . . . .	11
1.5.3	Limits . . . . .	12
1.5.4	Coordinates . . . . .	14
1.5.5	Pictures . . . . .	15
1.5.6	Videos . . . . .	17
1.6	Time-based data structures . . . . .	19
1.6.1	Timestamp . . . . .	19
1.6.2	Time Indices . . . . .	19
1.6.3	Durations . . . . .	24
1.7	RCM-DX file format . . . . .	25
1.8	RCM-DX Data hierarchy . . . . .	25
1.8.1	Root Group . . . . .	26
1.8.2	Session Group . . . . .	26
1.8.3	Section Group . . . . .	28
1.8.4	Position Group . . . . .	30
1.8.5	Platform Group . . . . .	31
1.8.6	Environment Group . . . . .	33
1.8.7	Measuring System Group . . . . .	36
1.8.8	Datasource Group . . . . .	38
1.8.9	Channel Group . . . . .	39
1.8.10	Logging Group . . . . .	43
1.8.11	Topology Group . . . . .	45

1.8.12	Track Group . . . . .	46
1.8.13	Line Group . . . . .	48
1.8.14	Switch Track Group . . . . .	49
1.8.15	Track Object Group . . . . .	50
1.8.16	Track Point Group . . . . .	50
1.8.17	Property Group . . . . .	51
1.8.18	Event Group . . . . .	52
1.8.19	Record Group . . . . .	57
1.8.20	Configuration Group . . . . .	61
1.8.21	Data Processing Group . . . . .	62
1.8.22	Clearance Information Group . . . . .	62
1.9	XML Schema Definitions . . . . .	63
1.9.1	Events Comment . . . . .	63
1.9.2	Events Defect . . . . .	64
1.9.3	Events Generic . . . . .	64
1.9.4	RCM-DX Data Types . . . . .	67

## List of Figures

1	RCM-DX Diagram Overview . . . . .	7
2	Structure for the recording of limit exceedances . . . . .	13
3	Image of a rail cross section measurement . . . . .	15
4	Binary tree, structure of node number . . . . .	22
5	Table overview 1: <i>BlockNumber</i> for each time stamp . . . . .	22
6	Table overview 2: <i>BlockNumber</i> for each timestamp . . . . .	22
7	Timestamp search in table . . . . .	24
8	RCM-DX Structure Overview . . . . .	25
9	Overview of the platform structure . . . . .	32
10	Overview of the Environment Structure . . . . .	34
11	Overview of the measuring system structure . . . . .	36
12	Overview of the logging structure . . . . .	43
13	Overview topology structure . . . . .	45

# 1 Specification of the RCM-DX Format version 2.0

## 1.1 History

Document Version	RCM-DX version	Datum	Autor	Beschreibung
V0.1	-	06.03.2015	Martin Frey (SCS)	Initial version
V0.2	-	20.03.2015	Martin Frey (SCS)	Extensions
V0.3	-	15.04.2015	Patrik Wernli (SCS)	Review
V0.4	-	20.04.2015	Martin Frey (SCS)	Extensions and revised
V0.5	-	03.05.2015	Patrik Wernli (SCS)	Formal Adaptions
V0.6	-	12.05.2015	Martin Frey (SCS)	PDR Feedback: Storing of Booleans, comments allowed on all levels, format independent of video codec, flags (including simulation) on session level. Schemas for exceedances, comments and drawings added
V0.7	-	13.07.2015	Patrik Wernli (SCS)	Finalized for CDR
V0.8	-	02.10.2015	Martin Frey (SCS)	Event model added, reference to specification event schema added.
V0.9	-	30.11.2015	Patrik Wernli (SCS)	Adaptions for Infotrans position model. Version concept removed. Event model updated.
V0.10	-	21.12.2015	Martin Frey (SCS)	Review
V0.11	-	21.12.2015	Patrik Wernli (SCS)	Revised after review
V0.12	-	16.02.2016	Pascal Brem (SCS)	Topology model in configuration.
V0.13	-	17.02.2016	Martin Frey (SCS)	Review topology model
V0.14	-	19.02.2016	Pascal Brem (SCS)	Sections added to file format.
V0.15	-	23.02.2016	Martin Frey (SCS)	Global configuration and settings updated

Document Version	RCM-DX version	Datum	Autor	Beschreibung
V0.16	-	11.07.2016	Pascal Brem (SCS)	New Hash code attribute for the topology.
V0.17	-	15.07.2016	Pascal Brem (SCS)	New units and data types for positions
V0.18	-	03.01.2018	Pascal Brem (SCS)	New GTG Track Id in the Topology.
V0.19	-	03.01.2018	Pascal Brem (SCS)	Events are stored on session level.
V0.20	-	09.01.2018	Pascal Brem (SCS)	Events and Sections in a group.
V0.21	-	09.01.2018	Patrik Wernli (SCS)	Added chapter “HDF5 File Format Versions”
V0.22	-	11.04.2018	Patrik Wernli (SCS)	Changed document template to official publishing template
V0.23	-	16.08.2018	Pascal Brem (SCS)	Changes in the channel basis definition.
V0.24	-	16.08.2018	Pascal Brem (SCS)	New attributes on the picture block channel.
V0.25	-	04.09.2018	Pascal Brem (SCS)	New minor version.
V0.26	-	28.11.2018	Pascal Brem (SCS)	New minor version for the topology attributes.
V0.27	-	08.01.2019	Pascal Brem (SCS)	New availability group.
V0.28	-	05.06.2019	Pascal Brem (SCS)	New switchtracks in the DfA
V0.29	2.0	29.10.2019	Michael Ammann (SBB)	Major changes to the format as well as the documentation. New major version for RCM-DX format.

## 1.2 Approval

Date	Name	Function
-	-	-

## 1.3 Introduction

### 1.3.1 Motivation

Railroad companies continuously gather data of their rail, overhead line, and telecommunications networks by means of mobile and stationary measuring systems. Data flows from these systems through processing units – which enrich, evaluate and validate the data –, to systems that display the data to subject matter experts and also to systems that automatically analyse it.

This specification defines the rail condition monitoring data exchange format (RCM-DX format) which is a data format optimised for data in the railroad context, i.e. for data points localised within a railroad network. The RCM-DX format is a file format based on the HDF5 specification and defines a structure of HDF5 groups, datasets, and attributes. The document at hands also describes the content of the elements defined. Although the format is open and can in principle be implemented right away by any railroad company, this specification contains a few non-generic elements and naming conventions that are specific to SBB. The reason for this is that any file that adheres to this specification can be used with the *RCM Viewer*, an application available soon to the public via a website.

The RCM-DX format is developed and maintained by the SBB company. An extension of the specification is permitted, yet, it must be taken into account that the resulting data file may no longer be read or processed by other systems supporting the RCM-DX format.

The RCM-DX format is a file format detailing the HDF5 format version 2.0. HDF5 was chosen for several reasons, including that it is an open format. HDF5 is a hierarchical data storage where the data is arranged in a tree structure. The HDF5 format is described on the webpage of the [HDF5-group](#), in particular on the site [HDF5 file format specification](#). The HDF5 group offers tools and libraries for various programming languages and operating systems that allow to read and write HDF5 files.

### 1.3.2 Hints

#### 1.3.2.1 RCM-DX structure

RCM-DX defines a structure of HDF5 groups, datasets, and attributes that software solutions that use this format must adhere to.

The extension of the specification is permitted. However, it must be taken into account that such data may no longer be read or processed by existing systems.

#### 1.3.2.2 Versioning

The RCM-DX data format is subject to changes, these are indicated by the version number in the document, see chapter [1.8.1 Root Group](#). The version number consists of three numbers separated by

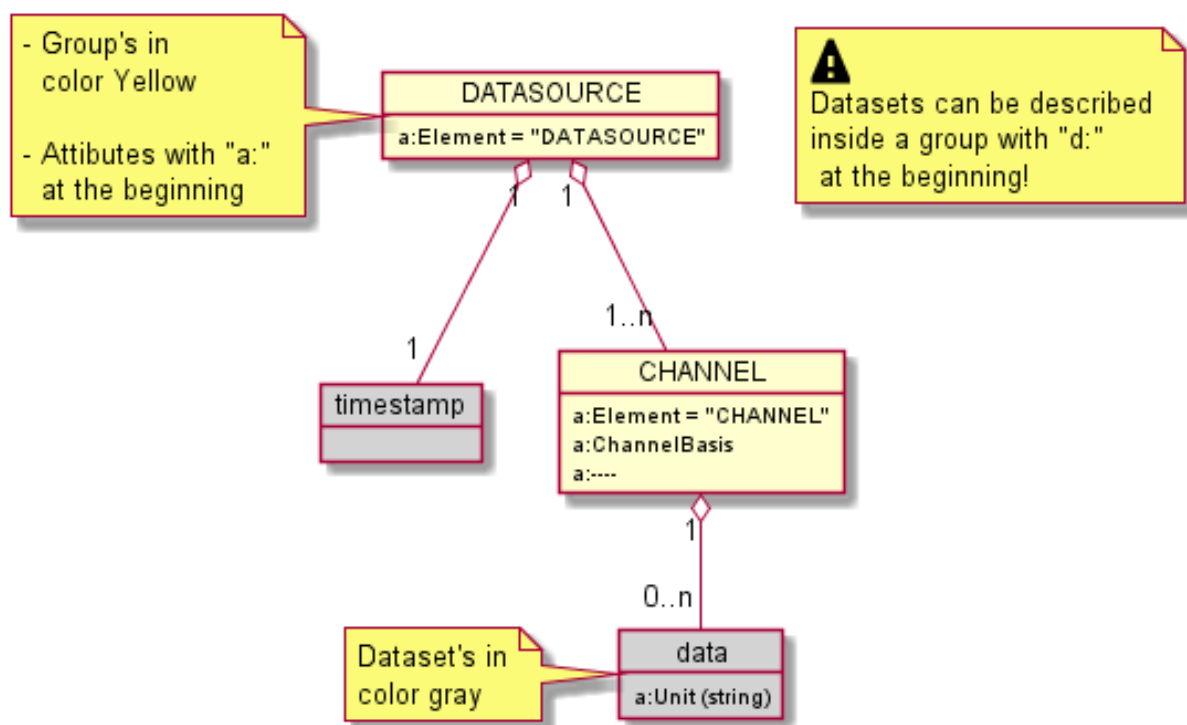
dots and is composed as follows: **[Major].[Minor]**. Example: **1.0**

**Major:** Indicates large change in data format that are not backward compatible. Examples are the modification of structures or the naming of existing groups.

**Minor:** Minor changes that represent an extension and are still backward compatible. Examples include defining new groups for datasets or new datatypes, etc.

### 1.3.2.3 Diagrams

The following figure shows the color coding used in diagrams:



**Figure 1:** RCM-DX Diagram Overview

## 1.4 Definitions

This document defines technical restrictions as well as content descriptions. This chapter provides an overview of the data types used, types of names, and other important points.

### 1.4.1 File names

Files following the RCM-DX specification get their own defined file extension, which is `rcmdx`.

Example of a file name: 20201228\_081522\_TGMS.rcmdx.

### 1.4.2 Primitive and extended data types

Possible data types for channels should be taken from the HDF5 specification.

Data types that are used in this specification and require a more detailed description are described below.

Name	Description	Example
UUID	Universally Unique Identifier as Unique ID as defined in <a href="https://wikipedia.org/wiki/Universally_Unique_Identifier">wikipedia.org/wiki/Universally_Unique_Identifier</a>	550e8400-e29b-11d4-a716-4466554400
Timestamp	Unique and worldwide defined format of a time, since January 1, 1970 00:00 UTC without leap seconds, defined under <a href="https://wikipedia.org/wiki/Timestamp">wikipedia.org/wiki/Timestamp</a>	1553237099000000000

### 1.4.3 (HDF5) Group

If we are talking about a group in this document, we mean the groups in HDF5 format (of the type [HDF5 Group](#)). These contain additional groups or datasets.

If a name of a group in this document is written in capital letters (for example [TOPOLOGY](#)), it is exactly the same as in the RCM-DX file. If the naming of a group is not fixed, the corresponding chapter describes in more detail how the name is composed.

Groups are described in this specification as follows:

Name	Parent object	Mandatory
<a href="#">SESSION</a>	<a href="#">RCMDX</a>	yes

**Name:** The name of the group.

**Parent object:** A group can be a subgroup of a group, here the name of this group is mentioned. If the name is written in quotation marks, it can be freely chosen by the creator of the file. Without quotation marks, the name of the group is meant.

**Mandatory:** If the group is absolutely necessary and must exist, [yes](#) is written here, otherwise [no](#).



Group names whose ending is "\_NAME" are wildcard names and are replaced as described in the corresponding paragraph. Example: *SESSION\_NAME*

#### 1.4.4 (HDF5) Attribut

In the RCM-DX, attributes, groups and datasets can be assigned. The names of the attributes are written in the UpperCamelCase-Notation<sup>1</sup>. Attributes are always of type **HDF5 attribute** unless otherwise specified.

Attributes are described in this specification as follows:

Name	Data Type	Parent object	Mandatory	Description
StartTime	64 bit integer	<i>SESSION_NAME</i>	yes	Start time in milliseconds, for example: 1553237099000000000

**Name:** The name of the attribute

**Data Type:** Primitive data type of the attribute, this describes the type of the content in the attribute itself.

**Parent object:** An attribute is always assigned to a group or a dataset, here the name of this group or dataset is mentioned.

**Mandatory:** If the attribute is absolutely necessary and must exist as well as contain a value, **yes** is written here, otherwise **no**.

**Description:** Description and or examples of the attribute

<sup>1</sup>Upper-Camel-Case-Notation: The Upper Camel Case Notation defines the way a composite name is written. Further information can be found at the following link: [Upper Camel Case](#)

#### 1.4.5 (HDF5) Datasets

A channel and its dataset can hold different types and types of data. The HDF5 group defines the way of storage, but not the names.

A dataset is always of the HDF5 type **HDF5 Dataset**.

Below is a list of ossible ways in which data can be stored in the RCM-DX:

Type of storage	Description
Array	Data array of arbitrary length

Type of storage	Description
Indexed single values	Simple array of dimension 1D. Beside a dataset <code>timestamp</code> a dataset <code>timeindex</code> is created, which contains an indexing of the data and simplifies the reading of the data. The dataset <code>timeindex</code> is described in more detail in the chapter <a href="#">1.6.2 Time Indices</a>
Image	An image taken at a defined time
Video	A video that has been streamed into several individual blocks of defined size, split and saved

The datasets are described in the lowerCamelCase-Notation<sup>2</sup>. Datasets are described in this specification as follows:

Name	Data type	Parent object	Mandatory	Storage type
timestamp	64 bit integer	<i>DATASOURCE_NAME</i>	yes	<code>array[n]</code>

**Name:** The name of the dataset.

**Data Type:** Primitive data type of the content in the dataset, thus the data type of the contained data.

**Parent object:** A dataset is always assigned to a group, here the name of this group is mentioned.

**Mandatory:** If the dataset is absolutely necessary and must be present, `yes` is written here, otherwise `no`.

**Storage type:** One of the storage types described in this chapter

Descriptions of the dataset are added outside the table.

<sup>2</sup>**lower-Camel-Case-Notation:** The lower camel case notation defines the way a composite name is written. Further information can be found under the following link: [www.wikipedia.org/wiki/Camel\\_case](http://www.wikipedia.org/wiki/Camel_case)

#### 1.4.5.1 HDF5 Compression

To save space, compression filters can be applied to datasets. Below you can see for all datasets whether this is recommended or not. The compression and which functions the HDF5 group offers is described in more detail on its website under the link: [www.support.hdfgroup.org/HDF5/faq/compression.html](http://www.support.hdfgroup.org/HDF5/faq/compression.html). For a dataset, for example, it says that “HDF5 compression” is allowed or not.

#### 1.4.5.2 HDF5 Chunking

Besides the “HDF5 Compression” there is the “HDF5 Chunking” for data within a dataset. This means that the data is divided into blocks, which in turn can be processed independently. This also allows faster access to parts of the data. Whether a splitting is allowed and recommended can be seen with each dataset, for example: “HDF5 Chunking” is allowed and recommended. The HDF5 chunking is described in more detail on the website of the HDF5 group: [www.support.hdfgroup.org/HDF5/doc/H5.user/Chunking.html](http://www.support.hdfgroup.org/HDF5/doc/H5.user/Chunking.html)

## 1.5 Data structures

### 1.5.1 Array

Channels which record individual measured values contain a dataset with the name `data`, this dataset is mandatory. Single values are stored in this dataset as a 1D array, the length of this array (or list) is not limited.

The possible data types are defined by the HDF5 Group, and can be read on the website of the HDF5 Group.

Boolean values (`true/false`) are represented as 8 bit integer, little endian, zero means `false`, all values greater than zero mean `true`.

If measurement data are recorded in 3D space, three different channels must be created.

The following attributes are assigned to this type of dataset:

Name	Data Type	Parent object	Mandatory	Description
<code>Unit</code>	string	a dataset	yes	A physical unit or empty if the data does not correspond to a physical unit

### 1.5.2 Preview data

In addition to integer and floating-point datasets, special datasets with preview data can be created. For a preview value, a defined set of entries (data block) from a dataset is calculated together to give, for example, an average value. This type of dataset provides a quick overview of the recorded data. The unit within this preview dataset is the same as in the channel dataset.

Three datasets with unique names are defined as follows to include preview values:

Name	Contents
data.PRE.AVG.X	Calculated average of data block
data.PRE.MIN.X	Smallest value in data block
data.PRE.MAX.X	Largest value in data block

The **X** is replaced by the block size. For example, if the block size of ten entries is defined and the average of the values is calculated, the dataset would be named as follows: `data.PRE.AVG.10`.

By specifying the block size, it is not necessary to create a dataset that contains the start time stamp of the data block.

If a dataset contains 600 entries, the dataset `timestamp` contains the same number of entries but the dataset `data.PRE.AVG.10` contains only 60 entries.

Further dind the two datasets `timestamp` and `duration` within the group contain.

Name	Data type	Parent object	Mandatory	Storage type
timestamp	64 bit integer	<i>LIMIT_NAME</i>	yes	<code>array[n]</code>
duration	64 bit integer	<i>LIMIT_NAME</i>	no	<code>array[n]</code>

### 1.5.3 Limits

A channel group can contain one or more limit groups. Each limit group contains its own `timestamp` dataset and can also contain a `duration` dataset. If a defined limit value of a measured value of the channel is exceeded, an entry in the `timestamp` dataset follows. Using the optional dataset `duration`, the duration of a limit value exceedance can be specified per entry in the dataset `timestamp`. If both datasets exist, they contain the same number of entries!

The group of limit values is defined as follows:

Name	Parent object	Mandatory
<b>LIMIT</b>	<i>CHANNEL_NAME</i>	no

The group **LIMIT** now contains further groups, each with the name of the limit exceeding, e.g. **TEMP**:

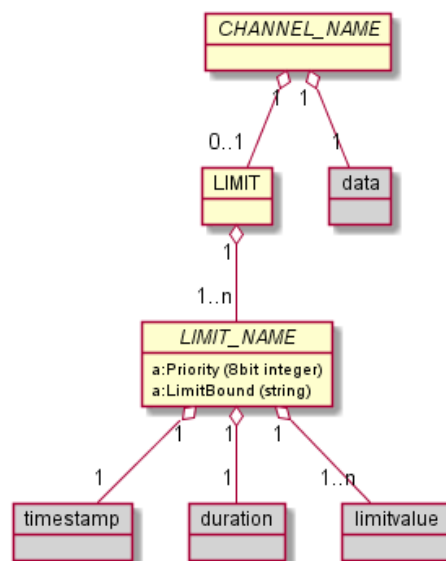
Name	Parent object	Mandatory
<i>LIMIT_NAME</i>	<b>LIMIT</b>	no

The following attributes are assigned to this group:

Name	Data Type	Parent object	Mandatory	Description
Priority	8 bit integer	<i>LIMIT_NAME</i>	yes	Priority of defined limit, lower values are priorities
LimitBound	string	<i>LIMIT_NAME</i>	yes	Defines the type of limit, whether it is a maximum or minimum limit.

It contains the following datasets:

Name	Data type	Parent object	Mandatory	Storage type
limitvalue	integer or float	<i>LIMIT_NAME</i>	yes	<code>array[n]</code>
timestamp	64 bit integer	<i>LIMIT_NAME</i>	yes	<code>array[n]</code>
duration	64 bit integer	<i>LIMIT_NAME</i>	no	<code>array[n]</code>



**Figure 2:** Structure for the recording of limit exceedances

#### 1.5.4 Coordinates

Measurement data that can be assigned to a coordinate system are given a defined name according to the following pattern: `coord.CN`.

This type of data storage allows several entries to be recorded per measurement time. Thus there are more entries in these datasets than in the dataset `timestamp`. How many entries per timestamp belong to each other (as a group) is stored in another dataset with the name `sampleindex`. The data set `sampleindex` is describes in chapter [1.5.4.1 Sample index](#).

Element	Description
<code>coord</code>	Simple character string for identifying data of type Coordinates
<code>.</code>	Separators
<code>C</code>	Additional character for identifying data of type Coordinates
<code>N</code>	Index Number beginning with “0”, increasing for each additional coordinate Datasets

The dataset is defined as follows:

Name	Data type	Parent object	Mandatory	Storage type
<code>coord.CN</code>	8, 32 or 64 bit signed integer or 64 or 32 bit floating point; (little endian in each case)	<code>CHANNEL_NA<sub>i</sub></code>	yes	Single

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

The following attributes are assigned to this type of dataset:

Name	Data Type	Parent object	Mandatory	Description
<code>Unit</code>	string	Dataset <code>coord.CN</code>	yes	One physical unit or empty if the data does not correspond to any physical unit

##### 1.5.4.1 Sample Index

If datasets are created for coordinates, a dataset on the same level and with the name `sampleindex`

must be available. The index number of an entry in `coord.CN`, `timestamp`, is entered as the start of the next group. If the dataset `sampleIndex` has a value of 21 at index zero, the first 20 entries from the dataset `coord.CN` belong together.

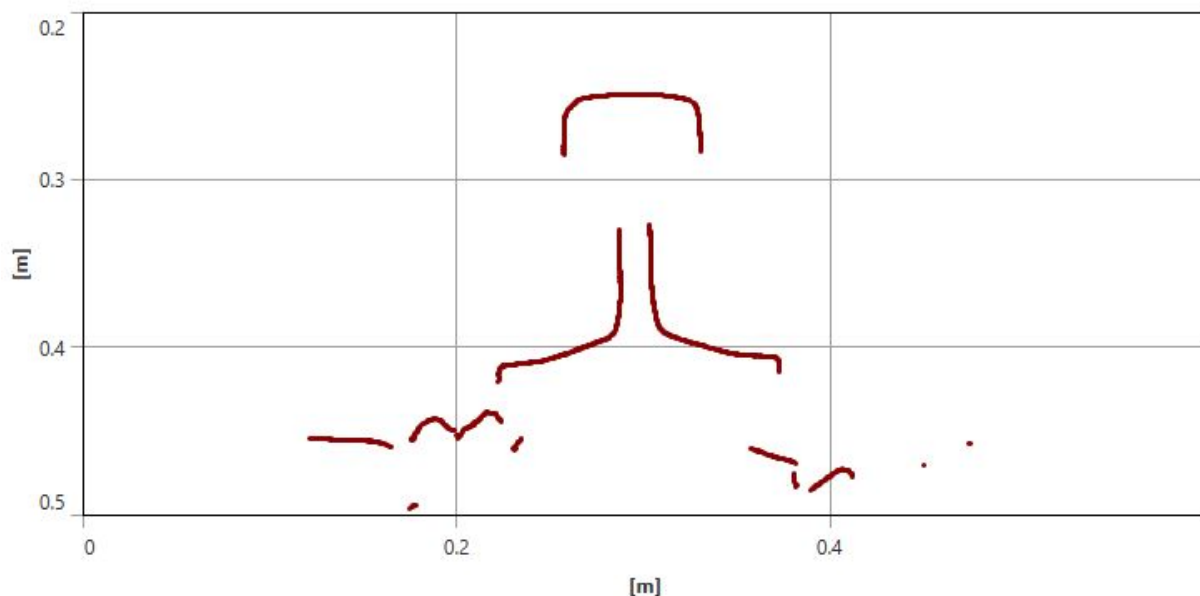
The resulting group size in the dataset `sampleIndex` may differ.

#### Example

The rail cross profile serves as an example here. At one point, several points of a rail profile are measured and stored. A channel with the name `coord.C0` for the X-axis and `coord.C1` is created for the Y-axis.

The dataset `sampleIndex` now contains the number of entries that belong together.

Below is a picture of a rail cross section measurement with about 2000 measuring points:



**Figure 3:** Image of a rail cross section measurement

### 1.5.5 Pictures

Images can be saved in compressed or uncompressed form. The format of the images is stored in an attribute so that the image can be read correctly.

Images are stored as binary data blocks, so an image results in a dataset. All images are stored in a group called `IMG`. All images in this group have the same properties that are stored in the attributes.

Name	Parent object	Mandatory
<code>IMG</code>	<code>DATASOURCE_NAME</code>	yes

The group **IMG** gets the following attributes for the more detailed description of the images contained therein:

Name	Data Type	Parent object	Mandatory	Description
ContentType	string	<b>IMG</b>	yes	Data type of images specified as MIME <sup>3</sup> type, for example <b>Content-Type</b> : <code>&lt;image/jpeg&gt;</code>
DataType	string	<b>IMG</b>	no	Description of data type, if no standard image, see <b>1.5.5.1 ContentType without Image MIME type</b>
ResolutionType	string	<b>IMG</b>	yes	Information about the point density of the images, defined in specification of the measuring system.
ResolutionX	32 bit integer	<b>IMG</b>	yes	Resolution in X direction
ResolutionY	32 bit integer	<b>IMG</b>	yes	Resolution in Y-direction

Images can have different resolutions in X and Y direction, this must be considered for a correct representation and evaluation of the images. HDF5 chunking is recommended and HDF5 compression is allowed.

**<sup>3</sup>MIME:** A list of possible MIME types can be found at the link [www.iana.org/assignments/media-types/media-types.xhtml](http://www.iana.org/assignments/media-types/media-types.xhtml). This is maintained by the [www.iana.org/](http://www.iana.org/).

#### 1.5.5.1 ContentType without Image MIME Type

If the created and saved image requires a special software to display it, the following MIME type should be added to the attribute **ContentType**: **Content-Type**: `<application/octet-stream>`.

To store more information, for example which system created the data, a new attribute can be added to the group **IMG**. The attribute gets the name **DataType**.

#### 1.5.5.2 Naming the dataset for an image



Name	Data type	Parent object	Mandatory	Storage type
img.NNNNNNNNN	integer, bit depth depending on color depth	IMG	yes	image

The images are named according to the following pattern: `img.NNNNNNNNN`, hereinafter a description of the individual elements.

Element	Description
img	String for the name of an image
.	Separators.
NNNNNNNNN	Picture number, beginning with 000000000 (nine characters)

### 1.5.6 Videos

Name	Data type	Parent object	Mandatory	Storage type
vid.NNNNNNNNN	integer, bit depth depending on color depth	VID	yes	image

As with the images, videos can be saved in compressed or uncompressed form. The format is stored in an attribute to make it easier to read the images.

Videos are stored as streams in individual blocks. The blocks are single datasets with a given name.

Name	Parent object	Mandatory
VID	<i>DATASOURCE_NAME</i>	yes

Below is a list of the attributes assigned to the data group `VID`:

Name	Data Type	Parent object	Mandatory	Description
ContentType	string	VID	yes	Data type of the video stream specified as MIME <sup>4</sup> type, for example <code>Content-Type: &lt;video/h264&gt;</code>
DataType	string	VID	no	Description of data type if no standard video format, see 1.5.6.1 <b>ContentType without video MIME type</b>
ResolutionX	32 bit integer	VID	yes	Resolution in X direction in pixels
ResolutionY	32 bit integer	VID	yes	Resolution in Y direction in pixels
FramesPerSecond	16 bit integer	VID	yes	Number of frames per second (fps) in which the video was recorded

<sup>4</sup>**MIME:** A list of possible MIME types can be found at the link [www.iana.org/assignments/media-types/media-types.xhtml](http://www.iana.org/assignments/media-types/media-types.xhtml). This is maintained by the [www.iana.org](http://www.iana.org).

#### 1.5.6.1 ContentType without Video MIME Type

If the created and saved video stream needs its own special software to display it, the following MIME type should be added to the `ContentType` attribute: `Content-Type: <application/octet-stream>`.

To store more information, for example which system created the data, a new attribute can be added to the group `VID`. The attribute gets the name `DataTyp`.

#### 1.5.6.2 Name of the dataset for a video

A video data block is named according to the following pattern: `vid.NNNNNNNNN`, hereinafter a description of the individual elements.

Element	Description
vid	String for the name of a video

Element	Description
.	Separator
NNNNNNNNN	Video number, starting with 000000000 (nine characters), ascending +1

HDF5 Chunking and HDF5 Compression is not recommended.

## 1.6 Time-based data structures

### 1.6.1 Timestamp

Each entry in a dataset of a channel has a reference to an entry in a dataset with the name `timestamp`, which lies within the data source group. In this `timestamp` dataset, there are as many entries as there are entries in a dataset of a channel. A timestamp is entered in nanoseconds since 01.01.1970 at 00:00 UTC.

The time stamps are always stored in ascending order and must not contain any jumps.

Name	Data type	Parent object	Mandatory	Storage type
timestamp	64 bit integer	<code>DATASOURCE_NAME</code>	yes	<code>array[n]</code>

HDF5 chunking is allowed and HDF5 compression is recommended.

These time stamps are recorded either by a defined distance travelled or by a frequency, this is described in more detail in the chapter [1.8.9.2 Common Trigger Distance or Frequency](#). In addition to the time stamps, the measuring devices follow this specification and also record measurement data at the same time. A central system serves as a clock generator for the data acquisition of all systems (measurement data and time stamps).

### 1.6.2 Time Indices

For a quick finding of timestamps this dataset is created in addition to the dataset `timestamp`. The time index dataset stores an offset value of a position of timestamp groups and is located in the `Datasource Group`, at the same level as the `timestamp` dataset. A detailed description of the contents can be found in the following chapter [1.6.2.1 Dataset content Time Indices](#). This dataset does not contain as many entries as the dataset `timestamp`.

Name	Datatype	Parent object	Mandatory	Storage type
timeindex	64 bit integer	<i>DATASOURCE_NAME</i>	yes	<code>array[n]</code>

HDF5 chunking is allowed and HDF5 compression is recommended.

The following attributes are assigned to the `timeindex` dataset:

Name	Data Type	Parent object	Mandatory	Description
BlockSize	long	Dataset <code>timeindex</code>	yes	Duration in nanoseconds of a time block. Time stamps within the same time block are indexed with the same value
LogTimeBlocks	integer	Dataset <code>timeindex</code>	yes	$2^{LogTimeBlocks}$ is equal to the number of blocks used to generate the binary tree. There may be time shifts greater than $2^{LogTimeBlocks} * BlockSize$ in the timestamp dataset, but these will not be indexed.
Depth	integer	Dataset <code>timeindex</code>	yes	Height of the binary tree

### 1.6.2.1 Dataset content Time Indices

In order to understand the content in the dataset `timeindex`, it must first be explained how it was created. The following example describes the process that leads to the result and back again.

In the example we want to save and index timestamps between  $10s$  and  $155s$ . These timestamps are contained in the `timestamp` dataset. The distances between the individual timestamps do not follow a uniform pattern.

First we define a *Offset*, which results from the first entry in the dataset `timestamp`. In our example the first entry is  $10s$ , so  $Offset = 10s$ . With the *Offset* of  $10 * 10^9ns$  we have a value range from  $0ns$  to  $145 * 10^9ns$  which we index.

If we now use a block size of  $BlockSize = 2 * 10^9ns$ , we get 72 blocks, which we index, because

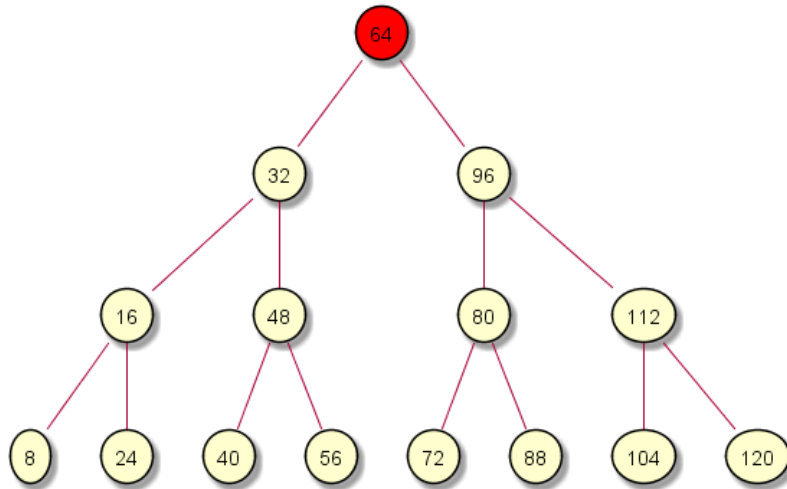
$10s + 72 * 2x10^9ns$  covers the values up to  $156s$  and is therefore sufficient for our range of values. To index all 72 blocks, we now need a sufficiently large value for *LogTimeBlocks*, we take a value of  $LogTimeBlocks = 7$ , because  $2^7 = 128$  is larger than 72.

Note: Possible would be a *LogTimeBlocks* value of  $LogTimeBlocks = 6 \Rightarrow 2^6 = 64$ . All values between 64 and 72 would then not be covered in the indexing!

If we now define a depth of  $Depth = 4$ , we get the following node numbers:

node number	height
64	1
32	2
96	2
16	3
48	3
80	3
112	3
8	4
24	4
40	4
56	4
72	4
88	4
104	4
120	4

These numbers and heights result from the binary tree, which we generate from the defined depth. To create the table, the tree is run through in “level-order”. The following picture shows this tree:



**Figure 4:** Binary tree, structure of node number

The first entry in the table has the value  $2^{\text{LogTimeBlocks}-1}$ , in our case  $2^{7-1} = 64$ , this entry has a height of one. Next, for each timestamp, we calculate the matching number of the corresponding block:

$$\text{BlockNumber} = \frac{\text{timestamp} - \text{Offset}}{\text{BlockSize}}$$

Each *BlockNumber* gets an index number in ascending order, starting from zero to 40. This number is used later to determine the offset position written to the `timeindex` dataset. Below the calculated *BlockNumber* and the corresponding timestamp, as an overview in a table:

Timestamp	10	14	16	19	27	28	33	35	38	45	46	50	52	56	60	62	65	68	74	75	80
BlockNumber	0	2	3	4	8	9	11	12	14	17	18	20	21	23	25	26	27	29	32	32	35
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

**Figure 5:** Table overview 1: *BlockNumber* for each time stamp

Timestamp	81	86	91	92	96	100	103	108	113	116	118	123	125	128	136	140	141	146	154	155
BlockNumber	35	38	40	41	43	45	46	49	52	53	54	56	57	59	63	65	65	68	72	72
Index	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40

**Figure 6:** Table overview 2: *BlockNumber* for each timestamp

In the next step we use the previously created table of the binary tree and write for each entry, the corresponding *BlockNumber* in the dataset `timeindex`. The first number of the binary tree is 64. So we look in the created table for the largest possible *BlockNumber*, which is smaller or equal to the value 64. Thus we find the number 63 in the table with the index 35. So we write the number 35 into the dataset `timeindex`. The second of the table with the binary tree has the value 32. In the table with

the *BlockNumber* we find the number 32, so the number 32 is written into the dataset `timeindex`. Now follows the number 96, for this there is no entry in the table with the *BlockNumber*, so we write the one -1 into the dataset `timeindex`. If we continue this way, we get the following table, which represents the dataset `timeindex`:

NodeNumber	Contents <code>timeindex</code>
64	35
32	18
96	-1
16	8
48	27
80	-1
112	-1
8	4
24	13
40	23
56	32
72	40
88	-1
104	-1
120	-1

### 1.6.2.2 Localization of the time stamp from dataset Time Indices

This chapter describes how to calculate the position of a timestamp using the dataset `timeindex`. Relevant are the defined values from the attributes *BlockSize*, *LogTimeBlocks* and *Depth*. With these values we can rebuild the binary tree.

Assuming we search for the position in the `timestamp` dataset for the timestamp 86s, we first calculate the node number with the following formula:

$$NodeNumber = \left( \frac{timestamp - Offset}{BlockSize} \right) - \left( \frac{timestamp - Offset}{BlockSize} \bmod 8 \right)$$

The calculation for a timestamp with the value 86s would look like this:

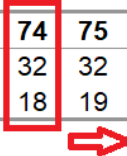
$$NodeNumber = \left( \frac{86s - 10s}{2s} \right) - \left( \frac{86s - 10s}{2s} \bmod 8 \right) = 38 - 6 = 32$$

With the calculated *NodeNumber* we can now find out the position using the `timeindex` dataset. Additionally we need the table of the binary tree. There we look for the index of the calculated *NodeNumber*, this would be the index 1. Now we find in the dataset `timeindex` at the same index, the position of the block by searching our timestamp, so we search from position 18.

<i>NodeNumber</i>	Dataset <i>timeindex</i>
64	35
<b>32</b>	<b>18</b>
96	-1
...	...

Now the search for the timestamp starts at the position 18 and searches via the values 74, 75, 80, 81 to 86 at position 22.

<b>Timestamp</b>	65	68	74	75	80	81	86	91	92	96
<b>BlockNumber</b>	27	29	32	32	35	35	38	40	41	43
<b>Index</b>	16	17	18	19	20	21	22	23	24	25



**Figure 7:** Timestamp search in table

The calculation above is always rounded down and not the next larger value (here 40) is used. This is because the position at the value 40 could be higher (end of the block) than the position in the block itself. So the timestamp would not be found!

### 1.6.3 Durations

If data is recorded that is valid for a certain period of time, the dataset with the name *duration* is added to the dataset *timestamp*. The timestamp recorded in the *timestamp* dataset specifies the time at which the value was recorded and the *duration* dataset specifies how long this value is valid in nanoseconds. The dataset *duration* is *timestamp* within a data source group next to the dataset.

Name	Data type	Parent object	Mandatory	Storage type
duration	32 bit integer	<i>DATASOURCE_NAME</i>	yes	<i>array[n]</i>

HDF5 chunking is allowed and HDF5 compression is recommended.



## 1.7 RCM-DX file format

The RCM-DX consists of a file format of the HDF5-group [www.hdfgroup.org/HDF5](http://www.hdfgroup.org/HDF5) in version 2.0. This allows to save the data in a tree structure. This structure, or rather the naming of the groups and datasets, is not specified by the HDF5 group, but by the RCM-DX specified here. The datasets can hold different data, what exactly is contained is specified as metadata.

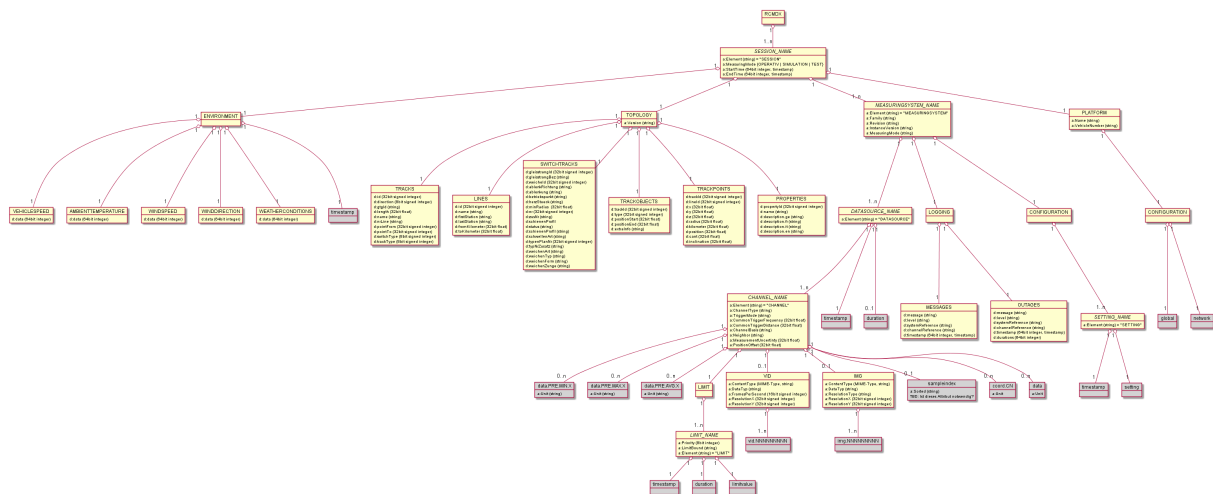
A change to the structure means a new version and thus a new release of this specification.

To read and write the HDF5 file format, the HDF5 group offers libraries for different languages. These can read and write the structure specified here without problems.

Further information about the structure of the HDF5 file format can be found under the following link: [www.portal.hdfgroup.org/display/HDF5/Introduction+to+HDF5](http://www.portal.hdfgroup.org/display/HDF5/Introduction+to+HDF5)

## 1.8 RCM-DX Data hierarchy

In the RCM-DX, the individual groups and datasets as well as their names are defined. Below is an overview of the structure specified in this document:



**Figure 8:** RCM-DX Structure Overview

Separate and more detailed specifications have been written for individual structure groups. Several measuring instruments can be installed on one measuring and inspection vehicle. Each of these measuring devices generates new channels of data, which flow into the RCM-DX. Since these channels can be different for each measuring device, the specifications were separated. Another reason for this is the fact that other railway operators use different measuring and inspection equipment.

The individual groups are specified in more detail below in the subcategories.

### 1.8.1 Root Group

The root group contains all other subgroups. This group defines the RCM-DX and bears its name and thus refers to this specification.

Name	Parent object	Mandatory
RCMDX	this is the root node	yes

#### 1.8.1.1 Attributes

The following attributes are assigned to the group [RCMDX](#):

Name	Data Type	Parent object	Mandatory	Description
clearance	8 bit integer (as boolean)	<a href="#">RCMDX</a>	yes	<b>Null (0)</b> for <b>false</b> , <b>one (1)</b> for <b>true</b> . With <b>true</b> the data in the whole file were released, otherwise the data are to be regarded as test data or are of lower quality.
Major	16-bit integer	<a href="#">RCMDX</a>	yes	Major Version of the RCM-DX specification that corresponds to the structure of the created file
Minor	16-bit integer	<a href="#">RCMDX</a>	yes	Minor Version of the RCM-DX specification that corresponds to the structure of the created file

### 1.8.2 Session Group

The session group contains data that was collected during the same period. A session group contains data from different sources. A RCM-DX file can contain several of these session groups and these in turn can overlap in time. Data of one source can only be in one session group at a time, so they cannot overlap, so at any time this source is only one session that records this data.

### 1.8.2.1 Naming

Since several session groups can be contained in one RCM-DX file, they must be given a unique name. To achieve this goal, the names are assigned according to the following pattern:

Name	Parent object	Mandatory
YYYYMMDD_hhmmss.SSS	RCMDX	yes

Example: 20190212\_231255.592

The individual elements and their meaning are described below:

Pattern	Content
YYYY	The year in four digit representation
MM	The month in the year (01 for January)
DD	The day in the month
hh	The hour in the day (0-23)
mm	The minute in the hour
ss	The seconds in the minute
SSS	The milliseconds in the seconds
"_" or "."	Characters as separator

A session group contains the data of the measuring equipment. Only one session can exist for a given millisecond, this must be implemented and ensured by the creator.

### 1.8.2.2 Attributes

Name	Data Type	Parent object	Mandatory	Description
StartTime	long	SESSION_NAME	yes	Time stamp in nanoseconds since January 1, 1970 UTC as start time of the session

Name	Data Type	Parent object	Mandatory	Description
EndTime	long	SESSION_NAME	no	Time stamp in nanoseconds since 1.1.1970 UTC as end time of the session. If the session has not yet been closed, this attribute is missing
Element	string	SESSION_NAME	yes	Contains the type of the group, this is fix "SESSION"

### 1.8.3 Section Group

The group [SECTION](#), contains information about a session.

Name	Parent object	Mandatory
<a href="#">SECTION</a>	SESSION_NAME	yes

#### 1.8.3.1 Section info

This group contains the information regarding the section itself.

Name	Parent object	Mandatory
<a href="#">SECTIONINFO</a>	<a href="#">SECTION</a>	yes

##### 1.8.3.1.1 Data fields

The following data fields are contained in the group "SECTIONINFO":

Name	Data type	Parent object	Mandatory	Storage type
coachOrientation	8 bit integer	<a href="#">SECTIONINFO</a>	yes	<a href="#">array[n]</a>
firstTrackOffset	64 bit float	<a href="#">SECTIONINFO</a>	yes	<a href="#">array[n]</a>
lastTrackOffset	64 bit float	<a href="#">SECTIONINFO</a>	yes	<a href="#">array[n]</a>
startTimestamp	64 bit integer	<a href="#">SECTIONINFO</a>	yes	<a href="#">array[n]</a>

Name	Data type	Parent object	Mandatory	Storage type
trackListOffset	64 bit float	SECTIONINFO	yes	array[n]

HDF5 chunking is allowed and recommended for all.

HDF5 compression is allowed.

coachOrientation

Defines the orientation of travel of the measuring vehicle per section. This array contains only as many entries as there are sections.

Number	Orientation of travel
0	forward
1	Reverse

firstTrackOffset

Indicates the distance in meters between the start of the track and the position at the beginning of the measurement. This array contains only as many entries as there are sections.

lastTrackOffset

Indicates the distance in meters between the position at the end of the measurement and the end of the rail. This array contains only as many entries as there are sections.

startTimestamp

Start time of the section as time stamp since 1.1.1970 at 00:00 UTC.

trackInfoOffset

This dataset defines how many entries in the datasets of the “Track list group” belong to a section. One entry is created per section in a session and the number of entries is defined. A group size can be determined by calculating the specified offset value at the  $x$  position minus the offset value at the  $x - 1$  position.

### 1.8.3.2 Track list

This group contains the information regarding the section itself.

Name	Parent object	Mandatory
<a href="#">TRACKLIST</a>	<a href="#">SECTION</a>	yes

#### 1.8.3.2.1 Data fields

Name	Data type	Parent object	Mandatory	Storage type
id	32 bit signed integer	<a href="#">TRACKLIST</a>	yes	<a href="#">array[n]</a>
startTimestamp	64 bit signed integer	<a href="#">TRACKLIST</a>	yes	<a href="#">array[n]</a>
endTimestamp	64 bit signed integer	<a href="#">TRACKLIST</a>	yes	<a href="#">array[n]</a>
orientation	8 bit signed integer	<a href="#">TRACKLIST</a>	yes	<a href="#">array[n]</a>
startCoveredDistance	64 bit float	<a href="#">TRACKLIST</a>	yes	<a href="#">array[n]</a>
endCoveredDistance	64 bit float	<a href="#">TRACKLIST</a>	yes	<a href="#">array[n]</a>

#### 1.8.4 Position Group

This group contains general information on the position.

Name	Parent object	Mandatory
<a href="#">POSITION</a>	<a href="#">SECTION</a>	yes

#### 1.8.4.1 Data fields

Name	Data type	Parent object	Mandatory	Storage type
coveredDistance	64 bit float	<a href="#">POSITION</a>	yes	<a href="#">array[n]</a>
coachOrientation	8 bit integer	<a href="#">POSITION</a>	yes	<a href="#">array[n]</a>
vehicleSpeed	64 bit float	<a href="#">POSITION</a>	yes	<a href="#">array[n]</a>
trackOrientation	8 bit integer	<a href="#">POSITION</a>	yes	<a href="#">array[n]</a>
trackId	32 bit integer	<a href="#">POSITION</a>	yes	<a href="#">array[n]</a>
trackOffset	64 bit float	<a href="#">POSITION</a>	yes	<a href="#">array[n]</a>
lineKilometer	64 bit float	<a href="#">POSITION</a>	yes	<a href="#">array[n]</a>
positionAccuracy	8 bit integer	<a href="#">POSITION</a>	yes	<a href="#">array[n]</a>
positionQuality	8 bit integer	<a href="#">POSITION</a>	yes	<a href="#">array[n]</a>

**coveredDistance:** Total length of a session **vehicleSpeed:** Speed of the vehicle at the time  
**trackId:** Defined track ID on which the vehicle is located at the time of recording  
**trackOffset:** Distance between starting point of track and current position  
**lineKilometer:** Absolute position on the line travelled at the time of recording  
**positionQuality:** Quality of the position measurement between zero (0) very good to 15 very bad  
**positionAccuracy:** The position accuracy

Unit's are defined in the attribute [Unit](#) of each data field.

#### 1.8.4.2 Coach Orientation

The dataset [coachOrientation](#) contains the coach direction of the vehicle. This information influences the position of the measuring systems.

This dataset can contain the following values:

Value	Description
0	Vehicle moving forward
1	Vehicle reversing

#### 1.8.4.3 Track Orientation

The dataset [trackOrientation](#) contains the alignment of the track. This information serves the correct evaluation of the kilometre data of the line, see dataset [trackOffset](#).

This dataset can contain the following values:

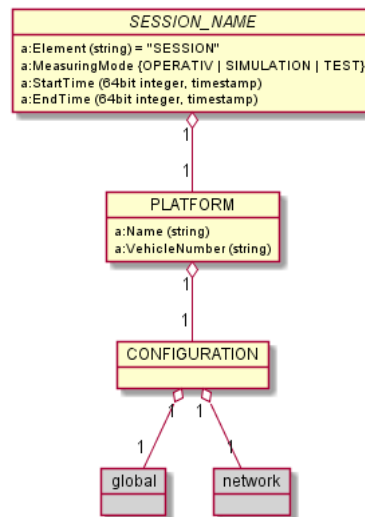
Value	Description
0	The kilometre indications of the line are <b>ascending</b> , based on the indicated vehicle orientation
1	The kilometre indications of the line are <b>decreasing</b> , based on the indicated vehicle orientation

#### 1.8.5 Platform Group

A platform group contains information about a measuring vehicle that collects the data.

The naming of the group is defined according to which platform produced the data. An overview of all names and the corresponding platform is specified in the chapter [1.8.5.2 Platforms at SBB](#).

Name	Parent object	Mandatory
Platform	SESSION_NAME	yes



**Figure 9:** Overview of the platform structure

### 1.8.5.1 Attributes

The platform group contains the following attributes:

Name	Data Type	Mandatory	Description
Name	string	yes	Unique name of the vehicle
VehicleNumber	string	yes	Unique number of the vehicle

### 1.8.5.2 Platforms at the SBB

Below is a list of the defined unique names of the platforms and their names.

Platform Name	Abbreviation	Vehicle Number
DFZ00	DFZ	-
DFZ01	gDFZ	-
DFZ02	SPZ	-



### 1.8.5.3 Configuration

Configurations of various systems can be stored in the datasets of this group. The datasets are designed so that global and network specific configurations can be stored.

Name	Parent object	Mandatory
CONFIGURATION	PLATFORM	no

Subsequent datasets are subordinate to this group:

Name	Data type	Parent object	Mandatory	Storage type
global	string	CONFIGURATION	yes	Single values
network	string	CONFIGURATION	yes	Single values

HDF5 chunking is allowed and recommended for all.

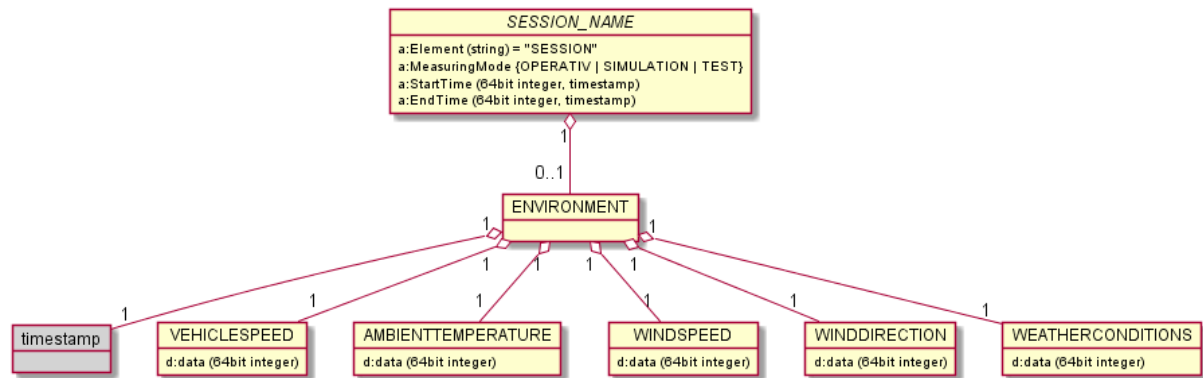
HDF5 compression is allowed.

### 1.8.6 Environment Group

Name	Parent object	Mandatory
ENVIRONMENT	SESSION_NAME	no

Information about the environment can be stored in the subgroups and their datasets. Since such information applies to all measurement systems, this is the right place for it.

As always with a data source, the dataset `timestamp` must not be forgotten.



**Figure 10:** Overview of the Environment Structure

### 1.8.6.1 Vehicle Speed

Name	Parent object	Mandatory
VEHICLESPEED	ENVIRONMENT	yes

The dataset contains a measured vehicle speed for each time stamp.

Name	Data type	Parent object	Mandatory	Storage type
data	64 bit integer	VEHICLESPEED	yes	array[n]

This data source also has a common dataset `timestamp`.

### 1.8.6.2 Ambient Temperature Group

This group contains a dataset containing the ambient temperatures. One temperature measurement is performed per time stamp.

Name	Parent object	Mandatory
AMBIENTTEMPERATURE	ENVIRONMENT	yes

For each time stamp, the ambient temperature is entered in the dataset.

Name	Data type	Parent object	Mandatory	Storage type
data	64 bit integer	<a href="#">AMBIENTTEMPERATURE</a>	yes	<a href="#">array[n]</a>

### 1.8.6.3 Wind Speed Group

The wind speed can be stored in the dataset of the group [WINDSPEED](#).

Name	Parent object	Mandatory
<a href="#">WINDSPEED</a>	<a href="#">ENVIRONMENT</a>	yes

For each time stamp, the wind speed is entered in the dataset.

Name	Data type	Parent object	Mandatory	Storage type
data	64 bit integer	<a href="#">WINDSPEED</a>	yes	<a href="#">array[n]</a>

### 1.8.6.4 Wind Direction Group

In addition to the wind speed, the wind direction is also saved, this is done in this group.

Name	Parent object	Mandatory
<a href="#">WINDDIRECTION</a>	<a href="#">ENVIRONMENT</a>	yes

For each time stamp, the wind direction is entered in the dataset.

Name	Data type	Parent object	Mandatory	Storage type
data	64 bit integer	<a href="#">WINDDIRECTION</a>	yes	<a href="#">array[n]</a>

### 1.8.6.5 Weather Conditions Group

The weather has an influence on the measurements. How the weather was at the time of the measurements is recorded in this group.

Name	Parent object	Mandatory
<a href="#">WEATHERCONDITIONS</a>	<a href="#">ENVIRONMENT</a>	yes

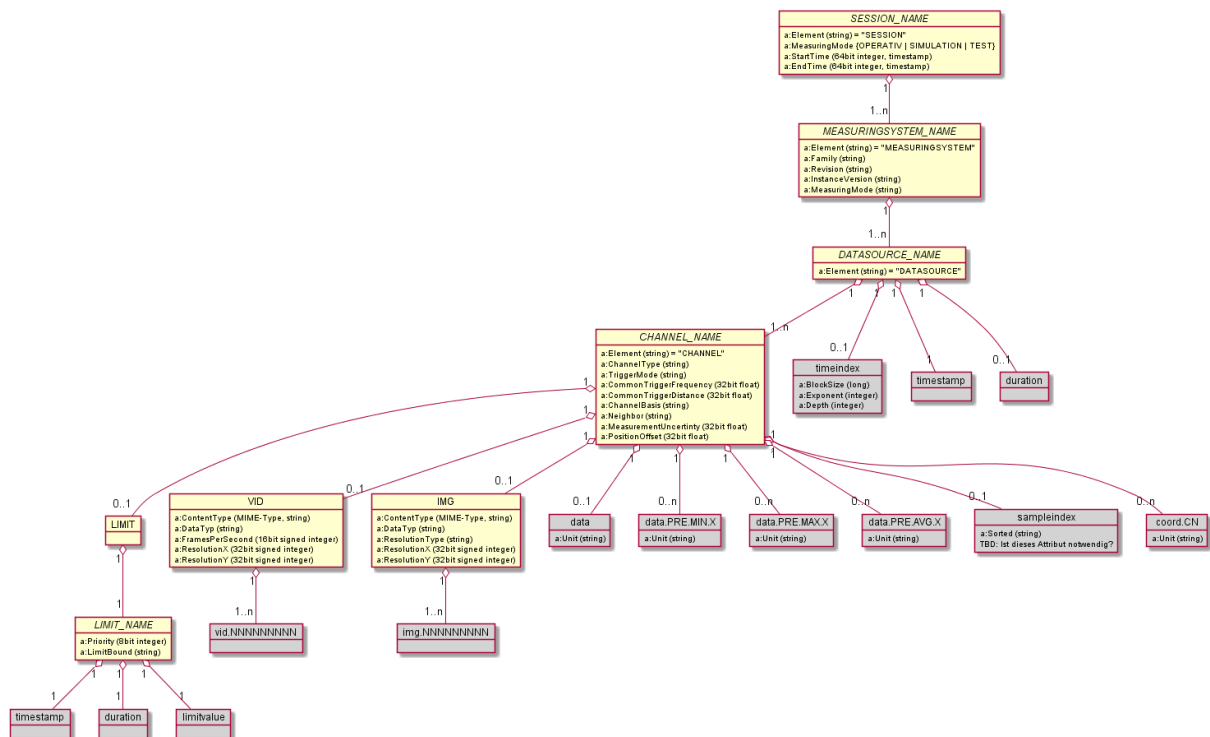
For each time stamp, the weather conditions are entered in the dataset. This could be for example “rain, fog, snowfall”.

Name	Data type	Parent object	Mandatory	Storage type
data	64 bit integer	WEATHERCONDITIONS	yes	array[n]

### 1.8.7 Measuring System Group

Each measuring system has its own data sources, which have their own names, as well as their own channels, which in turn have their own names. Common features are described in this specification, everything else is defined in a separate specification. Since this part differs greatly from railway companies and measuring equipment, a rigid specification has been dispensed with, but a certain framework is still given.

A group is created for each system that collects data. The name of the group is unique for each system. The composition of this name is not predefined. Each system contains further subgroups, each of which contains a data source at the end.



**Figure 11:** Overview of the measuring system structure

### 1.8.7.1 Example

As an example, a measuring system that records environmental data is described here. This measuring system can acquire several types of data, but belongs to the measuring system group “ADDITIONAL\_DATA”. The data are stored in individual data source groups, this is described in the chapter [1.8.8 Datasource Group](#).

### 1.8.7.2 Attributes

The following attributes are contained in the group of the measuring system:

Name	Data Type	Parent object	Mandatory	Description
Family	string	<i>MEASURINGSYSTEM_NAME</i>	yes	General name of the measuring system
Revision	string	<i>MEASURINGSYSTEM_NAME</i>	yes	Version of the software on the measuring system, issued by the owner of the platform
InstanceVersion	string	<i>MEASURINGSYSTEM_NAME</i>	yes	Version of the data format created by the measuring instrument. This version can be different within different gauges of the same family
Element	string	<i>MEASURINGSYSTEM_NAME</i>	yes	Contains the type of the group, this is fixed <i>MEASURINGSYSTEM</i>
MeasuringMode	string	<i>MEASURINGSYSTEM_NAME</i>	yes	Indicates the measuring mode.

#### 1.8.7.2.1 Measurement mode

There are three different measurement modes, which are explained individually below.

Name	Description
OPERATIV	Productive data that will be further used.

Name	Description
TEST	Test data recorded during a diagnostic run with the aim of checking and testing the measuring equipment.
SIMULATION	Simulated values that the measuring systems produce themselves and are no longer used.

### 1.8.8 Datasource Group

A data source group can contain several channels and thus several data sources. This group combines these channels. The naming can be freely selected, but must be unique.

A time stamp is available for each individual measuring point within a data source group. There are two types of data acquisition for a data source group. One is always after a defined distance (e.g. every 250 millimeters) and the other is the recording of measurement data at a certain frequency (e.g. 4000 Hz). The way the measurement data was recorded is shown in two attributes for each channel group. For a description see [1.8.9.2 Common Trigger Distance or Frequency](#).

#### 1.8.8.1 Attributes

The following attribute is assigned to the group:

Name	Data Type	Parent object	Mandatory	Description
Element	string	<i>DATASOURCE_NAME</i>	yes	Contains the type of the group, this is fix <a href="#">DATASOURCE</a>

#### 1.8.8.2 Example

In our example the name of the data source group is assigned, which should contain our environmental measurement data, which we call [ENVIRONMENT](#).

This group now also contains the dataset [timestamp](#).

#### 1.8.8.3 Timestamp dataset

Each data source group contains a dataset called [timestamp](#). It contains all timestamps at which a measurement was recorded. The size of this list of timestamps is the same as the size of the datasets per channel.

A more detailed description can be found in the chapter [1.6.1 Timestamp Array](#)!

### 1.8.9 Channel Group

A channel group contains metadata for the actual measurement data and thus for the various channels. The naming can be freely selected, but must be unique within the data source group.

The following attributes are contained in this group:

Name	Data Type	Parent object	Mandatory	Description
Element	string	<i>CHANNEL_NAME</i>	yes	Contains the type of the group, this is fix <a href="#">CHANNEL</a>
TriggerMode	string	<i>CHANNEL_NAME</i>	yes	See chapter <a href="#">1.8.9.6 Trigger Mode</a>
ChannelBasis	string	<i>CHANNEL_NAME</i>	yes	See chapter <a href="#">1.8.9.1 Channel Base</a>
CommonTriggerDistance	32 bit float	<i>CHANNEL_NAME</i>	yes	See chapter <a href="#">“1.8.9.2 Common Trigger Distance or Frequency”</a>
CommonTriggerFrequency	32 bit float	<i>CHANNEL_NAME</i>	yes	See chapter <a href="#">“1.8.9.2 Common Trigger Distance or Frequency”</a>
ChannelType	string	<i>CHANNEL_NAME</i>	yes	See chapter <a href="#">1.8.9.3 Channel Type</a>
Neighbor	string	<i>CHANNEL_NAME</i>	yes	See chapter <a href="#">1.8.9.4 Neighbor</a>
MeasurementUncertainty	32 bit float	<i>CHANNEL_NAME</i>	yes	This attribute contains the measurement accuracy of the channel according to the specifications of the measurement system.
PositionOffset	32 bit float	<i>CHANNEL_NAME</i>	yes	See chapter <a href="#">1.8.9.5 Position Offset</a>

### 1.8.9.1 Channel base

Description of the channel, what was measured and in which direction. Since a measuring vehicle can move on a rail in two directions and the sensor could therefore be on the other side, it should be possible to indicate this. Here is the place for it.

Possible values are:

Value	Description
COACH_LEFT	Sensor is installed on the left hand side of the measuring platform. The data has not been corrected for direction of travel
COACH_RIGHT	Sensor is installed on the right hand side of the measuring platform. The data has not been corrected for direction of travel
RAIL_LEFT	Channel contains data of the left rail in terms of track direction. The data has been corrected for direction of travel
RAIL_RIGHT	Channel contains data of the right rail in terms of track direction. The data has been corrected for direction of travel
ABSOLUTE	Channel is not associated with a single rail

### 1.8.9.2 Common Trigger Distance or Frequency

The two attributes `CommonTriggerFreq` and `CommonTriggerDistance` define the type of measurement data recording and the distance at which these measurement data were recorded.

If the measurement data are recorded at a certain distance, for example every 20 millimeters, the attribute `CommonTriggerDistance` is specified and the value "0.0" is specified for `CommonTriggerFreq`. At the attribute `TriggerMode` the value `DISTANCE` is noted.

If the data is recorded at a certain frequency, the frequency is given in Herz for the attribute `CommonTriggerFreq`, for example 40000.0 for 40kHz. With `CommonTriggerDistance` the value remains "0.0". At the attribute `TriggerMode` the value `TIME` is noted.

In a data source group there is always only one common trigger frequency or one trigger distance. A mixture within the group is not permitted! Both attributes have the value "0.0" if `TriggerMode` contains the value `EVENT`.

### 1.8.9.3 Channel Type

Defines how a value was created. This can be measured, calculated or taken from a previously defined data source that was read from there and inserted into the file.



The following values are possible:

Value	Description
MEASURED	Measured values
REFERENCE	A setpoint of a third source

An example for reference values are defined target values which flow in from another source (file, database etc.) and are to be used for comparisons.

#### 1.8.9.4 Neighbor

Refers to the name of an adjacent channel. This can be the right rail, for example, when measuring the track temperature of the left rail. Thus the attribute `neighbor` of the channel “TEMP\_RAIL\_L” would contain the name “TEMP\_RAIL\_R” and vice versa.

#### 1.8.9.5 Position Offset

Describes the distance between a defined zero point (position) on the measuring vehicle and a measuring system. This specification is used to convert the exact time at which the measurement was taken to a defined zero point using the speed driven and the position taken at the time.

#### 1.8.9.6 Trigger Mode

This attribute defines how the data was recorded.

Possible values are:

Value	Description
TIME	Time-based measurement data recording
DISTANCE	Distance-based measurement data acquisition
EVENT	Event based recording

#### 1.8.9.7 Example

All types of data that are recorded at the same time follow as the channel here. The names of the channels are freely selectable, here as an example:

Name	Mandatory	Description
TEMP	no	Temperature
WIND_DIR	no	Wind direction
WIND_SPEED	no	Wind speed
HUM	no	moisture

Furthermore, all channels receive the following attributes and values:

Attribute Name	Value (as example)
ChannelBasis	ABSOLUTE
CommonTriggerDistance	0.0
CommonTriggerFreq	1.0
MeasurementType	MEASURED_VALUE
Neighbor	""
PositioOffset	0.0
TriggerMode	TIME

#### 1.8.9.7.1 Data object

Each channel group receives a dataset with the actual measurement data:

Name	HDF5 Type	Mandatory
data	HDF5 Dataset	yes

There are as many measurement data entries as there are timestamps in the dataset `timestamp` which is included in the channel group.

The dataset needs more information, this is given as attributes:

Name	Data type	Parent object	Mandatory	Storage type
Unit	string	<i>CHANNEL_NAME</i>	yes	<code>array[n]</code>

**Unit:** The physical unit of the measurement data, such as “millimeter”. If no physical unit can be assigned to the data, this attribute remains empty.

The dataset and the possible data that can be stored are described in more detail in the chapter [1.4.5 Dataset](#).

#### Example

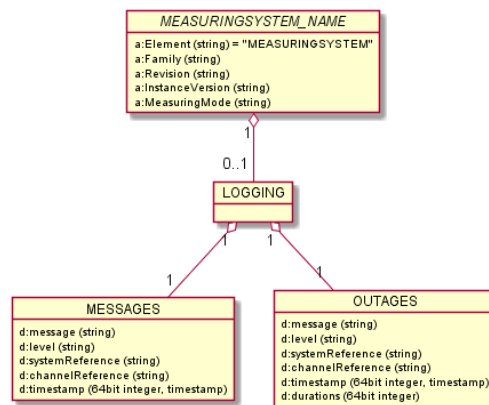
The units are now added to our defined channels. Each dataset now gets an attribute for this unit:

Channel Name	Attribute Name	Unit
TEMP	"Unit"	DegreeCelsius
WIND_DIR	"Unit"	Degree
WIND_SPEED	"Unit"	MeterPerSecond
HUM	"Unit"	RelativeHumidity

### 1.8.10 Logging Group

The logging group contains information about the status of the measuring systems. The data is divided into two subgroups, [OUTAGES](#) and [MESSAGES](#). These are described in separate chapters.

Name	Parent object	Mandatory
<a href="#">LOGGING</a>	<i>MEASURINGSYSTEM_NAME</i>	no



**Figure 12:** Overview of the logging structure

#### 1.8.10.1 Outage Group

In this group, failures and interruptions of measurement systems are recorded in a defined structure, each as its own dataset.

The following datasets are included in this group:

Name	Data type	Parent object	Mandatory	Storage type
message	string	LOGGING	yes	array[n]
systemReference	string	LOGGING	yes	array[n]
channelReference	string	LOGGING	yes	array[n]
level	string	LOGGING	yes	array[n]

HDF5 chunking is allowed and HDF5 compression is allowed.

This group receives a `timestamp` dataset as well as a `duration` dataset to indicate the time of the measurement failure.

**systemReference:** A reference to the measurement system.

**channelReference:** A reference to a channel.

**level:** Defines the severity of the failure or interruption of a measurement system.

**message:** This dataset contains one message per entry about a failure of a measuring instrument.

#### 1.8.10.2 Message Group

Status messages for a system or data source are stored in this group. If no messages exist, this group remains empty.

The following datasets are contained in this group:

Name	Data type	Parent object	Mandatory	Storage type
message	string	LOGGING	yes	array[n]
systemReference	string	LOGGING	yes	array[n]
channelReference	string	LOGGING	yes	array[n]
level	string	LOGGING	yes	array[n]
timestamp	64 bit integer	LOGGING	yes	array[n]

HDF5 chunking is recommended and HDF5 compression is allowed.

**systemReference:** Fully qualified name of the measuring system that sent the message.

**channelReference:** Fully qualified name of the channel that the message refers to. empty if the message is measuring system wide.

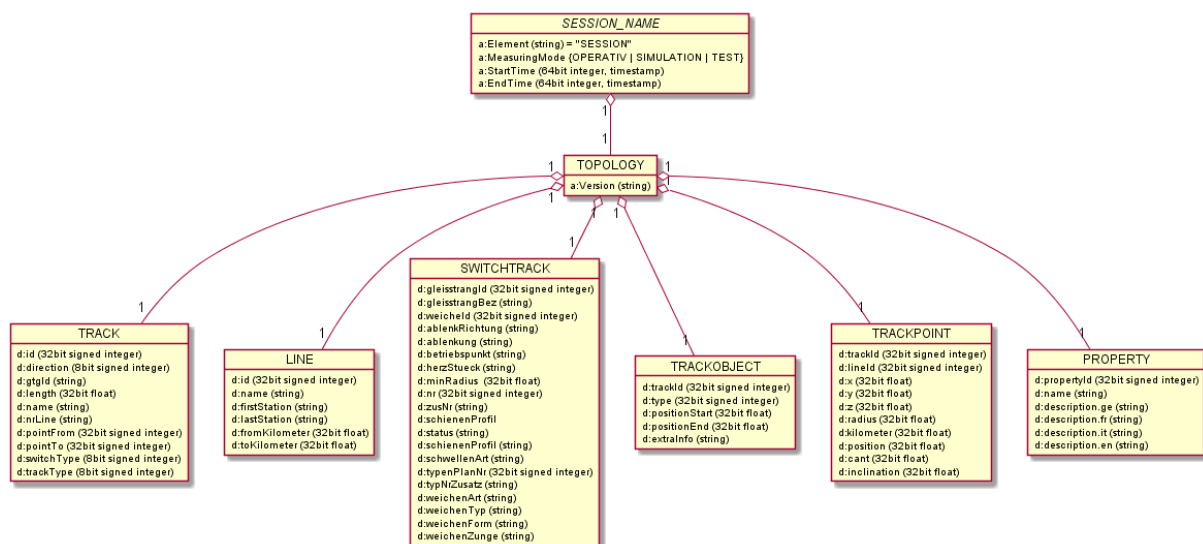
**level: TODO:** Not yet defined....

**message:** The actual message to be recorded for a measurement system.

### 1.8.11 Topology Group

A topology group contains all information on the route network of the respective railway company. This chapter has been optimised for SBB and may differ between railway companies. SBB's data processing chain provides for this structure, which is why it is described here.

Name	HDF5 Type	Parent object	Mandatory
<b>TOPOLOGY</b>	HDF5 Group	<i>SESSION_NAME</i>	yes



**Figure 13:** Overview topology structure

#### 1.8.11.1 Attributes

The group **TOPOLOGY** contains the following attributes:

Name	Data type	Parent object	Mandatory	Storage type
Version	string	<b>TOPOLOGY</b>	yes	<code>array[n]</code>

**Version:** Version number of topology, included to check validity.

The DfA (Database of fixed assets) is a SBB construct and reflects the SBB route network. The data comes from a database and is distributed as a file to the SBB measuring vehicles. They can read the information contained therein and also add it to the RCM-DX. This DfA is used for positioning and it is therefore possible to assign the measured data to an object from the route network.

### 1.8.12 Track Group

This group contains information on the tracks of the railway network. The information is stored in separate datasets.

Name	Parent object	Mandatory
TRACK	TOPOLOGY	yes

The following datasets are included in this group, some of which are described in more detail in the subchapters:

Name	Data type	Parent object	Mandatory	Storage type
direction	8 bit signed integer, little endian	TRACK	yes	<a href="#">array[n]</a>
id	32 bit integer, little endian	TRACK	yes	<a href="#">array[n]</a>
gtgId	string	TRACK	yes	<a href="#">array[n]</a>
length	string	TRACK	yes	<a href="#">array[n]</a>
name	string	TRACK	yes	<a href="#">array[n]</a>
nrLine	string	TRACK	yes	<a href="#">array[n]</a>
pointFrom	32 bit integer, little endian	TRACK	yes	<a href="#">array[n]</a>
pointTo	32 bit integer, little endian	TRACK	yes	<a href="#">array[n]</a>
switchType	8 bit signed integer, little endian	TRACK	yes	<a href="#">array[n]</a>
trackType	8 bit signed integer, little endian	TRACK	yes	<a href="#">array[n]</a>

**direction:** Will be decided in a separate chapter: [1.8.12.1 direction](#)

**id:** ID of the track section

**gtgId:** Unique GTG ID of a GTG string, this ID is stored as UUID

**length:** The length of the track section

**name:** name of the track section

**nrLine:** Number of the line for the track section

**pointFrom:** ID of the starting point of the track section

**pointTo:** ID of the end point of the track section

**switchType:** Will be reviewed in a separate chapter: [1.8.12.2.1 switchType](#)

**trackType:** Will be published in a separate chapter: [1.8.12.2 trackType](#)

### 1.8.12.1 direction

The direction of a switch is specified in this dataset.

If the track is of the type “switch”, a value greater than zero must be selected here. Which number means what is shown in the following table:

Value	Description
0	No crossover
1	Straight line switch track
2	Left-handed switch
3	switch running to the right

### 1.8.12.2 trackType

The number in the *trackType* dataset defines the type of track that belongs to it. Which number means what is shown in the following table:

Value	Description
0	Station track
1	Track
2	Switch

#### 1.8.12.2.1 switchType

If the track is of the type “Switch”, a value greater than zero must be selected here. Which number means what is shown in the following table:

Value	Description
0	Anything but a turnout
1	simple switch
2	Double switch
3	Simple crossovers
4	Double track connection
5	Double crossover

### 1.8.13 Line Group

This group contains information about a line in the route network. The information is stored in separate datasets.

Name	Parent object	Mandatory
<a href="#">LINE</a>	<a href="#">TOPOLOGY</a>	yes

The following datasets are included in this group:

Name	Data type	Parent object	Mandatory	Storage type
id	32 bit signed integer	<a href="#">LINE</a>	yes	<a href="#">array[n]</a>
name	string	<a href="#">LINE</a>	yes	<a href="#">array[n]</a>
firstStation	string	<a href="#">LINE</a>	yes	<a href="#">array[n]</a>
lastStation	string	<a href="#">LINE</a>	yes	<a href="#">array[n]</a>
fromKilometer	string	<a href="#">LINE</a>	yes	<a href="#">array[n]</a>
toKilometer	string	<a href="#">LINE</a>	yes	<a href="#">array[n]</a>

**id:** Defines the ID of the line, this is unique

**name:** The name of the line

**firstStation:** The name of the first station of this line

**LastStation:** The name of the last station of this line.

**km:** Start kilometre of the line, expressed in kilometres

**toKilometer:** Final kilometer of the line, in kilometers



### 1.8.14 Switch Track Group

This group contains information about switches in the route network. The information is stored in separate datasets.

Name	Parent object	Mandatory
<a href="#">SWITCHTRACK</a>	<a href="#">TOPOLOGY</a>	yes

The following datasets are included in this group:

Name	Data type	Parent object	Mandatory	Storage type
track strand	32 bit signed integer	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
gleisstrangBez	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
soft	32 bit signed integer	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
deflecting direction	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
distraction	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
operating point	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
herzStueck	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
minRadius	32 bit signed integer	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
nr	32 bit signed integer	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
zusNr	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
rail profile	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
status	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
rail profile	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
thresholdArt	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
typesPlanNr	32 bit signed integer	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
typeNraddition	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
softArt	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
softType	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
softForm	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>
soft tongue	string	<a href="#">SWITCHTRACK</a>	yes	<a href="#">array[n]</a>

**trackId:** a reference to the GTG-ID

**trackBez:** Contains a description of the track section

**softId:** Contains the ID's of the switches as a reference

### 1.8.15 Track Object Group

This group contains information about objects in the route network, for example a balise. The information is stored in separate datasets.

Name	Parent object	Mandatory
TRACKOBJECT	TOPOLOGY	yes

The following datasets are included in this group:

Name	Data type	Parent object	Mandatory	Storage type
trackId	32 bit signed integer	TRACKOBJECT	yes	array[n]
type	32 bit signed integer	TRACKOBJECT	yes	array[n]
positionStart	32 bit signed integer	TRACKOBJECT	yes	array[n]
positionEnd	32 bit signed integer	TRACKOBJECT	yes	array[n]
extraInfo	string	TRACKOBJECT	yes	array[n]

**trackId:** Contains the ID of the track to which the track is connected.

**type:** Type of the object

**positionStart:** Start position of the object in meters

**positionEnd:** End position of the object in meters

**ExtraInfo:** Additional information about the object, for example, the ID of a balise.

### 1.8.16 Track Point Group

This group contains information about defined points on the route network. The information is stored in separate datasets.

Name	Parent object	Mandatory
<a href="#">TRACKPOINT</a>	<a href="#">TOPOLOGY</a>	yes

The following datasets are included in this group:

Name	Data type	Parent object	Mandatory	Storage type
trackId	32 bit signed integer	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>
lineId	32 bit signed integer	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>
x	32 bit float	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>
y	32 bit float	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>
z	32 bit float	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>
radius	32 bit float	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>
kilometers	32 bit float	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>
position	32 bit float	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>
cant	32 bit float	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>
inclination	32 bit float	<a href="#">TRACKPOINT</a>	yes	<a href="#">array[n]</a>

**trackId:** Reference to the ID of the track section

**lineId:** Reference to the ID of the line

**x:** X-Koordinate of the point

**y:** Y-Koordinate of the point

**z:** Z-Koordinate of the point

**radius:** The radius of a point, given in meters.

**Kilometres:** Contains the line kilometre of the point in the route network, expressed in kilometres.

**Position:** Position of the point, in meters

**cant:** The inclination at this point, expressed in millimetres.

**Inclination:** Gradient at this point, expressed in parts per thousand

### 1.8.17 Property Group

This group contains information about properties of the topology itself. The information is stored in separate datasets.

Name	Parent object	Mandatory
<a href="#">PROPERTY</a>	<a href="#">TOPOLOGY</a>	yes

The following datasets are included in this group:

Name	Data type	Parent object	Mandatory	Description
propertyId	32 bit signed integer	<a href="#">PROPERTY</a>	yes	<a href="#">array[n]</a>
name	string	<a href="#">PROPERTY</a>	yes	<a href="#">array[n]</a>
description.ge	string	<a href="#">PROPERTY</a>	yes	<a href="#">array[n]</a>
description.fr	string	<a href="#">PROPERTY</a>	yes	<a href="#">array[n]</a>
description.it	string	<a href="#">PROPERTY</a>	yes	<a href="#">array[n]</a>
description.en	string	<a href="#">PROPERTY</a>	yes	<a href="#">array[n]</a>

**propertyId:** Unique ID of the characteristic

**name:** Name of the characteristic

**description.ge:** Description of the characteristic in the language German

**description.fr:** Description of the characteristic in French language

**description.it:** Description of the characteristic in Italian language

**description.en:** Description of the feature in English language

### 1.8.18 Event Group

The Event group is used to store events that occurred during the recording of data. Events are bound to a channel, system or session and have a link to it. In addition to events, log entries can also be created, these are described in more detail in the chapter [1.8.19 Record Group](#). Systems can, for example, trigger an event when a limit value is exceeded. Events are always time-bound which means an event contains the exact time of occurrence and the duration of the event. The duration can also be zero, so the event occurred exactly at the specified time.

Name	HDF5 Type	Parent object	Mandatory
<a href="#">EVENT</a>	HDF5 Group	<a href="#">SESSION_NAME</a>	yes

Within the group there are the following data fields:

Name	Data type	Parent object	Mandatory	Storage type
systemReference	string	<a href="#">EVENT</a>	yes	<a href="#">array[n]</a>
channelReference	string	<a href="#">EVENT</a>	no	<a href="#">array[n]</a>
data	string	<a href="#">EVENT</a>	yes	<a href="#">array[n]</a>
type	string	<a href="#">EVENT</a>	yes	<a href="#">array[n]</a>
duration	64 bit signed integer little endian	<a href="#">EVENT</a>	yes	<a href="#">array[n]</a>
timestamp	64 bit signed integer little endian	<a href="#">EVENT</a>	yes	<a href="#">array[n]</a>

For all datasets, HDF5 chunking is recommended and HDF5 compression is allowed.

Each of these datasets contains a list with information about an entry, at a certain time. Each dataset is described in more detail in the following subchapters.

#### 1.8.18.1 System reference “systemReference”

Contains a list of entries containing the name of the system that triggered the event.

Name	Data Type	Parent Object	Mandatory	Storage Type
systemReference	string	<a href="#">EVENT</a>	yes	<a href="#">array[n]</a>

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

#### 1.8.18.2 Channel reference “channelReference”

Contains a list of entries that refers to a channel to which the event applies.

Name	Data Type	Parent Object	Mandatory	Storage Type
channelReference	string	<a href="#">EVENT</a>	yes	<a href="#">array[n]</a>

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

#### 1.8.18.3 Event Data

This dataset contains the actual information about an event, this in the XML notation which is described in more detail in each chapter of the event types.

The events are stored in this dataset as a list. A type can be stored for each event. These are explained in more detail in the chapter [1.8.18.5 Event Types](#).

Name	Data type	Parent object	Mandatory	Storage type
data	string	EVENT	yes	array[n]

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

#### 1.8.18.4 Duration

Defines for each event the duration of the event itself. This value can also be zero.

Name	Data type	Parent object	Mandatory	Storage type
duration	64 bit integer	EVENT	yes	array[n]

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

#### 1.8.18.5 Event Types

Contains the type of an event.

Name	Data type	Parent object	Mandatory	Storage type
type	string	EVENT	yes	array[n]

HDF5 chunking is allowed and recommended for all.

HDF5 compression is allowed.

In the list “type” the type of the recorded event is shown. The different types contain different information which is shown in the following subchapters. There are corresponding XML schemas for all types that define the technical specifications.

##### 1.8.18.5.1 Defect

A defect can be, for example, an image of a rail showing a damage of the surface. This defect is recorded by a system. However, it may happen that this error is not one (incorrectly detected), this information

can be specified afterwards (attribute “PossibleValidationResults”). Defects are always channel bound and recorded or evaluated by a system. In the following, the elements and attributes that occur in a *Defect* as XML are described in more detail.

The XML Schema can be found in the chapter [1.9.2 EventsDefect](#).

#### XML elements

Not all of these elements must be present, details can be taken from the XML Schema.

Name	Description	Parent object
Defect	XML Root Element	none
PossibleDefectNames	Name of a possible error	Defect
PossibleClassifications	Classification of a possible defect	Defect
PossibleValidationResults	Possible confirmations of the defect	Defect

#### XML attributes

Below are the attributes of the root element “Defect”:

Name	Description	Parent object
Classification	Classification of the error	Defect
DefectName	Name of the error	Defect
Details	Further information or more detailed descriptions of the error	Defect
Parameter1Name	Name of the parameter 1	Defect
Parameter1Value	Value of parameter 1	Defect
Parameter2Name	Name of the parameter 2	Defect
Parameter2Value	Value of Parameter 2	Defect
Parameter3Name	Name of the parameter 3	Defect
Parameter3Value	Value of parameter 3	Defect
ID	Unique number for identification of the error	Defect

#### 1.8.18.5.2 Detected Object

These events indicate an object found during a diagoose ride. These can be, for example, detected balises or tunnels. What exactly counts as a found object is not defined in this specification, only the information for a recorded event.

The XML Schema can be found in the chapter [1.9.3 EventsGeneric](#).

## XML elements

Not all of these elements must be present, details can be taken from the XML schema.

Name	Description	Parent object
DetectedObject	Root Element	none
object	Element with information about the found object in the element itself or in the attributes	DetectedObject
Reference	Reference to a list of known and uniquely assignable objects of the railway company	DetectedObject
ObjectAttribute	Further information about the object, the information is contained in the attributes	DetectedObject

## XML attributes

Name	Description	Parent object
Unique ID of the event	DetectedObject	
Type	Type of object found	object
Description	further description or information about/from object	object
ObjectConsistency	Reference to the correctness of the specified data	object
ReferenceSystem	Reference to the name of the system from which the data originates	Reference
Key	Information about the data contained in the “ObjectAttribute” element	ObjectAttribute

**1.8.18.5.3 Limit Violation**

Limit value exceedances of measured values of a channel can also be recorded as events.

The XML schema can be found in chapter [1.9.3 EventsGeneric](#).

## XML elements

Name	Description	Parent object
LimitViolation	Root Element	none



## XML attributes

Name	Description	Parent object
TimestampMaxViolation	Time at which limit value was exceeded	LimitViolation
ViolatedLimit	Name of the defined limit	LimitViolation
ID	Unique ID of the event	LimitViolation

**1.8.18.5.4 Consistency**

The message about the consistency of the data is triggered by a system that checks all data according to certain criteria. For example, this could be a check for black images in a video. If all frames in the video are black, something is wrong and the video is unusable. Messages are only created if a finding is present.

The XML Schema can be found in the chapter [1.9.3 EventsGeneric](#).

## XML element

Name	Description	Parent object
Consistency	Root Element	none

## XML attributes

Name	Description	Parent object
Type	Type or type of consistency check in response to the question "What has been checked?"	Consistency
ProcessName	Name of the process that checked consistency	Consistency
Result	Result of the consistency check.	Consistency
ID	Unique ID of the event (UUID)	Consistency

**Result:** The actual result of the consistency check. Each system that performs a consistency check has different results, which in turn must be described in more detail in its specification.

**1.8.19 Record Group**

Unlike events, logs are only created by a user and not by a system. For all protocol types, there are corresponding XML schemas that define the technical specifications. Metadata is defined in the

respective channels. Protocol entries can have references to systems, sessions, and channels.

Name	HDF5 Type	Mandatory
<a href="#">RECORD</a>	HDF5 Group	yes

Within the group there are the following data fields:

Name	Data type	Parent object	Mandatory	Storage type
type	string	<a href="#">RECORD</a>	yes	<a href="#">array[n]</a>
systemReference	string	<a href="#">RECORD</a>	yes	<a href="#">array[n]</a>
channelReference	string	<a href="#">RECORD</a>	no	<a href="#">array[n]</a>
data	string	<a href="#">RECORD</a>	yes	<a href="#">array[n]</a>
duration	64 bit signed integer little endian	<a href="#">RECORD</a>	yes	<a href="#">array[n]</a>
timestamp	64 bit signed integer little endian	<a href="#">RECORD</a>	yes	<a href="#">array[n]</a>

Each of these datasets contains a list with information about an entry at a specific time. Each dataset is described in more detail in the following subchapters.

#### 1.8.19.1 Record Data

This dataset contains the actual information for a protocol entry, this in the XML notation which is described in more detail in each chapter of the protocol types.

The protocol entries are stored in this dataset as a list. A type can be stored for each entry. These are explained in more detail in the chapter [1.8.19.5 Record Types](#).

Name	Data type	Parent object	Mandatory	Storage type
data	string	<a href="#">RECORD</a>	yes	<a href="#">array[n]</a>

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

#### 1.8.19.2 Duration

Defines for each entry the duration of the log entry itself. This value can also be zero.

Name	Data type	Parent object	Mandatory	Storage type
duration	64 bit integer	RECORD	yes	array[n]

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

#### 1.8.19.3 System reference systemReference

Contains a list of entries containing the name of the system that triggered the record.

Name	Data Type	Parent Object	Mandatory	Storage Type
systemReference	RECORD	string	yes	array[n]

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

#### 1.8.19.4 Channel reference channelReference

Contains a list of entries that refers to a channel to which the record applies.

Name	Data Type	Parent Object	Mandatory	Storage Type
channelReference	RECORD	string	yes	array[n]

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

#### 1.8.19.5 Record Types

Contains the type of a log entry.

Name	Data type	Parent object	Mandatory	Storage type
recordtype	string	RECORD	yes	array[n]

HDF5 chunking is allowed and recommended, HDF5 compression is allowed.

In the list “recordtype” the type of the recorded protocol entry is shown. The different types contain different information, which is shown in the following subchapters. Corresponding XML schemas are

available for all types, which define the technical specifications.

#### 1.8.19.5.1 Comment

Comments recorded during a diagnostic drive by the user. The content is not specified, only the XML structure. The XML schema can be found in chapter [1.9.1 EventsComment](#).

##### XML elements

Name	Description	Parent object
Comment	Root element and message, recorded by the user	none

##### XML attributes

Name	Description	Parent object
Username	Name of the user who recorded the message	Comment
ID	Unique ID of this message	Comment

#### 1.8.19.5.2 Corrupt

Messages of the type “damaged” or “unusable” do not receive a content specification, only the XML structure is predefined and described here. The XML schema can be found in chapter [1.9.3 Events-Generic](#).

##### XML elements

Name	Description	Parent object
Corrupt	Root element and message, recorded by the user	none

##### XML attributes

Name	Description	Parent object
Username	Name of the user who recorded the message	Corrupt
ID	Unique ID of this message	Corrupt

### 1.8.20 Configuration Group

In the configuration group, data can be stored in any format that was used for the configuration of one or more measuring systems. Each subgroup defines a measurement system.

This group is below that of a measurement system and thus within the group [1.8.7 Measuring System Group](#).

Name	Parent object	Mandatory
<a href="#">CONFIGURATION</a>	<i>MEASURINGSYSTEM_NAME</i>	no

#### 1.8.20.1 Configuration Group datasets

Within this group there are further groups whose names correspond to those of a measuring system to which the configuration contained therein belongs.

In the following *SETTING\_NAME* is used as placeholder of the actual name of the measuring system.

Name	Parent object	Mandatory
<i>SETTING_NAME</i>	<a href="#">CONFIGURATION</a>	no

This group contains two datasets:

Name	Data type	Parent object	Mandatory	Storage type
setting	string	<i>SETTING_NAME</i>	yes	<a href="#">array[n]</a>
timestamp	64 bit integer	<i>SETTING_NAME</i>	yes	<a href="#">array[n]</a>

**setting:** Contains the actual configuration.

**timestamp:** Contains the time from when this configuration is valid and was used.

The following attributes are contained in this group:

Name	Data Type	Parent object	Mandatory	Description
DataType	string	<a href="#">setting</a>	yes	Defines the datatype of the configuration within the dataset <a href="#">setting</a> . Data type specified as MIME <sup>3</sup> type, for example <a href="#">Content-Type: &lt; text/strings&gt;</a>

### 1.8.21 Data Processing Group

The data source group [DATAPROCESSING](#) contains information on data processing. This information is written by systems that make changes to the data. These changes, for example, can be a conversion from millimeters to meters.

Name	Parent object	Mandatory
<a href="#">DATAPROCESSING</a>	<a href="#">RCMDX</a>	yes

#### 1.8.21.1 Data Processing Group datasets

The group [DATAPROCESSING](#) contains one datasets:

Name	Data type	Parent object	Mandatory	Storage type
keyValue	string	<a href="#">DATAPROCESSING</a>	yes	<a href="#">array[n]</a>
timestamp	64 bit integer	<a href="#">DATAPROCESSING</a>	yes	<a href="#">array[n]</a>

**keyValue:** This record contains a unique key as a reference to a data processing step, followed by a value about what was done in that step or what the result was. Key- and value are separated by a colon character: [key:value](#)

**timestamp:** Contains the time of the acquisition of the entry in the [keyValue](#) dataset.

### 1.8.22 Clearance Information Group

This group is used by SBB to record information about the data release of all parties who have processed this data. The information is stored in the form of key-value pairs in a dataset.

Name	Parent object	Mandatory
CLEARANCEINFORMATION	RCMDX	no

The group **CLEARANCEINFORMATION** contains one datasets:

Name	Data type	Parent object	Mandatory	Storage type
keyValue	string	CLEARANCEINFORMATION	yes	array[n]
timestamp	64 bit integer	CLEARANCEINFORMATION	yes	array[n]

**keyValue:** This record contains a unique key as a reference to a clearance step, followed by a value about what was done in that step or what the result was. Key- and value are separated by a colon character: `key:value`

**timestamp:** Contains the time of the acquisition of the entry in the `keyValue` dataset.

## 1.9 XML Schema Definitions

### 1.9.1 Events Comment

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3   xmlns:tns="http://www.sbb.ch/RCMDX/Events/Comment"
4   targetNamespace="http://www.sbb.ch/RCMDX/Events/Comment"
5   elementFormDefault="qualified">
6
7   <xs:include schemaLocation="../RcmDxDatatypes.xsd" />
8
9   <xs:element name="Comment">
10     <xs:complexType>
11       <xs:simpleContent>
12         <xs:extension base="xs:string">
13           <xs:attribute name="Username" type="xs:string" use="required"
14             />
15           <xs:attribute name="ID" type="tns:UUID" use="required" />
16         </xs:extension>
17       </xs:simpleContent>
18     </xs:complexType>
19   </xs:element>
```

20 `</xs:schema>`

### 1.9.2 Events Defect

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3    xmlns:tns="http://www.sbb.ch/RCMDX/Events/Defect"
4    targetNamespace="http://www.sbb.ch/RCMDX/Events/Defect"
5    elementFormDefault="qualified">
6
7    <xs:include schemaLocation="../RcmDxDatatypes.xsd" />
8
9    <xs:element name="Defect">
10      <xs:complexType>
11        <xs:sequence>
12          <xs:element name="PossibleDefectNames" type="xs:string"
13            minOccurs="0" maxOccurs="unbounded" />
14          <xs:element name="PossibleClassifications" type="xs:string"
15            minOccurs="0" maxOccurs="unbounded" />
16          <xs:element name="PossibleValidationResults" type="xs:string"
17            minOccurs="0" maxOccurs="unbounded" />
18        </xs:sequence>
19        <xs:attribute name="Classification" type="xs:string" use="
20          required" />
21        <xs:attribute name="DefectName" type="xs:string" use="required"
22          />
23        <xs:attribute name="Details" type="xs:string" use="required" />
24        <xs:attribute name="Parameter1Name" type="xs:string" />
25        <xs:attribute name="Parameter1Value" type="xs:string" />
26        <xs:attribute name="Parameter2Name" type="xs:string" />
27        <xs:attribute name="Parameter2Value" type="xs:string" />
28        <xs:attribute name="Parameter3Name" type="xs:string" />
29        <xs:attribute name="Parameter3Value" type="xs:string" />
30        <xs:attribute name="ID" type="tns:UUID" use="required" />
31      </xs:complexType>
32    </xs:element>
33  </xs:schema>

```

### 1.9.3 Events Generic



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3   xmlns:tns="http://www.sbb.ch/RCMDX/Events/Generic"
4   targetNamespace="http://www.sbb.ch/RCMDX/Events/Generic"
5   elementFormDefault="qualified">
6
7   <xs:include schemaLocation="../../../RcmDxDatatypes.xsd" />
8
9   <xs:element name="Corrupt">
10     <xs:complexType>
11       <xs:simpleContent>
12         <xs:extension base="xs:string">
13           <xs:attribute name="Username" type="xs:string" use="required"
14             />
15           <xs:attribute name="ID" type="tns:UUID" use="required" />
16         </xs:extension>
17       </xs:simpleContent>
18     </xs:complexType>
19   </xs:element>
20
21   <xs:simpleType name="ObjectConsistencyXml">
22     <xs:restriction base="xs:string">
23       <xs:enumeration value="Ok" />
24       <xs:enumeration value="OnlyInReal" />
25       <xs:enumeration value="OnlyInData" />
26       <xs:enumeration value="Measured" />
27     </xs:restriction>
28   </xs:simpleType>
29
30   <xs:element name="DetectedObject">
31     <xs:complexType>
32       <xs:sequence>
33         <xs:element name="object" minOccurs="1" maxOccurs="1">
34           <xs:complexType>
35             <xs:simpleContent>
36               <xs:extension base="xs:string">
37                 <xs:attribute name="Type" type="xs:string" use="
38                   required" />
39                 <xs:attribute name="Description" type="xs:string" use="
40                   required" />
41                 <xs:attribute name="ObjectConsistency" type="tns:
42                   ObjectConsistencyXml" use="required" />
43               </xs:extension>
44             </xs:simpleContent>
45           </xs:complexType>
46         </xs:element>
47       </xs:sequence>
48     </xs:complexType>
49   </xs:element>
50 </xs:schema>
```

```

40         </xs:simpleContent>
41     </xs:complexType>
42 </xs:element>
43 <xs:element name="Reference" minOccurs="0" maxOccurs="unbounded
44     ">
45     <xs:complexType>
46     <xs:simpleContent>
47     <xs:extension base="xs:string">
48     <xs:attribute name="ReferenceSystem" type="xs:string"
49     use="required" />
50     </xs:extension>
51     </xs:simpleContent>
52 </xs:complexType>
53 </xs:element>
54 <xs:element name="ObjectAttribute" minOccurs="0" maxOccurs="
55     unbounded">
56     <xs:complexType>
57     <xs:simpleContent>
58     <xs:extension base="xs:string">
59     <xs:attribute name="Key" type="xs:string" use="required
60     " />
61     </xs:extension>
62     </xs:simpleContent>
63 </xs:complexType>
64 </xs:element>
65
66 <xs:element name="LimitViolation">
67     <xs:complexType>
68     <xs:attribute name="TimestampMaxViolation" type="xs:long" use="
69     required" />
70     <xs:attribute name="ViolatedLimit" type="xs:string" use="required
71     " />
72     <xs:attribute name="ID" type="tns:UUID" use="required" />
73     </xs:complexType>
74 </xs:element>
75
76 <xs:element name="Consistency">
77     <xs:complexType>
78     <xs:attribute name="Type" type="xs:string" use="required" />

```

```

77     <xs:attribute name="ProcessName" type="xs:string" use="required"
       />
78     <xs:attribute name="Result" type="xs:string" use="required" />
79     <xs:attribute name="ID" type="tns:UUID" use="required" />
80   </xs:complexType>
81 </xs:element>
82
83 </xs:schema>

```

### 1.9.4 RCM-DX Data Types

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
3    <xs:simpleType name="restrictedString">
4      <xs:restriction base="xs:string">
5        <xs:minLength value="1" />
6        <xs:maxLength value="512" />
7        <xs:pattern value="[a-zA-Z0-9_\-\.]+" />
8      </xs:restriction>
9    </xs:simpleType>
10   <xs:simpleType name="restrictedStringWithColon">
11     <xs:restriction base="xs:string">
12       <xs:minLength value="1" />
13       <xs:maxLength value="512" />
14       <xs:pattern value="[a-zA-Z0-9_\-\.:]+" />
15     </xs:restriction>
16   </xs:simpleType>
17   <xs:simpleType name="restrictedID">
18     <xs:restriction base="xs:ID">
19       <xs:minLength value="1" />
20       <xs:maxLength value="512" />
21       <xs:pattern value="[a-zA-Z0-9_\-\.]+" />
22     </xs:restriction>
23   </xs:simpleType>
24   <xs:simpleType name="restrictedIDREF">
25     <xs:restriction base="xs:IDREF">
26       <xs:minLength value="1" />
27       <xs:maxLength value="512" />
28       <xs:pattern value="[a-zA-Z0-9_\-\.]+" />
29     </xs:restriction>

```

```

30 </xs:simpleType>
31 <xs:simpleType name="versionString">
32   <xs:restriction base="xs:string">
33     <xs:minLength value="1" />
34     <xs:maxLength value="32" />
35     <xs:pattern value="[a-zA-Z0-9_-\.\.]" />
36   </xs:restriction>
37 </xs:simpleType>
38 <xs:simpleType name="portNumber">
39   <xs:restriction base="xs:int">
40     <xs:minExclusive value="0" />
41     <xs:maxInclusive value="65535" />
42   </xs:restriction>
43 </xs:simpleType>
44 <xs:simpleType name="ipAddress">
45   <xs:restriction base="xs:string">
46     <xs:pattern value="([0-9]{1,3}\.){3}[0-9]{1,3}" />
47   </xs:restriction>
48 </xs:simpleType>
49 <xs:simpleType name="network">
50   <xs:restriction base="xs:string">
51     <xs:pattern value="([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{1,2}" />
52   </xs:restriction>
53 </xs:simpleType>
54 <xs:simpleType name="hostName">
55   <xs:restriction base="xs:string">
56     <xs:pattern value="([a-zA-Z0-9] | [a-zA-Z0-9][a-zA-Z0-9\-\-]*[a-zA-
57       Z0-9])\.)*([A-Za-z0-9] | [A-Za-z0-9][A-Za-z0-9\-\-]*[A-Za-z0-9])"
58     />
59   </xs:restriction>
60 </xs:simpleType>
61 <xs:simpleType name="ipAddressOrHostName">
62   <xs:union memberTypes="ipAddress hostName" />
63 </xs:simpleType>
64 <xs:simpleType name="nonNegativeInt">
65   <xs:restriction base="xs:int">
66     <xs:minInclusive value="0" />
67   </xs:restriction>
68 </xs:simpleType>
69 <xs:simpleType name="positiveInt">
70   <xs:restriction base="xs:int">
71     <xs:minInclusive value="1" />
72   </xs:restriction>

```

```

71 </xs:simpleType>
72 <xs:simpleType name="positiveFloat">
73   <xs:restriction base="xs:float">
74     <xs:minExclusive value="0" />
75   </xs:restriction>
76 </xs:simpleType>
77 <xs:simpleType name="positiveIntOrMinus1">
78   <xs:restriction base="xs:int">
79     <xs:minInclusive value="-1" />
80   </xs:restriction>
81 </xs:simpleType>
82 <xs:simpleType name="positiveLong">
83   <xs:restriction base="xs:long">
84     <xs:minExclusive value="0" />
85   </xs:restriction>
86 </xs:simpleType>
87 <xs:simpleType name="nonNegativeLong">
88   <xs:restriction base="xs:long">
89     <xs:minInclusive value="0" />
90   </xs:restriction>
91 </xs:simpleType>
92 <xs:simpleType name="compressionLevel">
93   <xs:restriction base="xs:integer">
94     <xs:minInclusive value="0" />
95     <xs:maxInclusive value="9" />
96   </xs:restriction>
97 </xs:simpleType>
98 <xs:simpleType name="mimeType">
99   <xs:restriction base="xs:string">
100    <xs:pattern value="[!#$%'+*\-\0-9A-Z\^_'a-z{|}~]+/[!#$%'+*\-\0-9A-Z
      \^_'a-z{|}~]+(; *[^;]+)*" />
101   </xs:restriction>
102 </xs:simpleType>
103 <xs:simpleType name="nonEmptyString">
104   <xs:restriction base="xs:string">
105     <xs:minLength value="1" />
106   </xs:restriction>
107 </xs:simpleType>
108 <xs:simpleType name="UUID">
109   <xs:restriction base="xs:string">
110     <xs:pattern
111       value="(urn:uuid:)?[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]
        {4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}|\{[0-9a-fA-F]{8}-[0-9a-fA-F]

```

```
    ]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}\}" />
112    </xs:restriction>
113  </xs:simpleType>
114  <xs:simpleType name="vehicleNumber">
115    <xs:restriction base="xs:string">
116      <xs:pattern value="[0-9]{2} [0-9]{2} [0-9]{4} [0-9]{3}-[0-9]" />
117    </xs:restriction>
118  </xs:simpleType>
119  <xs:simpleType name="httpUrl">
120    <xs:restriction base="xs:anyURI">
121      <xs:pattern value="https?:/.+" />
122    </xs:restriction>
123  </xs:simpleType>
124 </xs:schema>
```