RCM-DX specification

Document Version 0.31_ase

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

Contents

List of Figures

1 Specification of the RCM-DX Format

1.1 License

> TODO: Lizenzhinweise notieren für RCM-DX TODO: Copyright hier vermerken TODO: Lizenzhinweise oder Copyright notieren für HDF5

1.2 Contribution

The RCM-DX specification is open source and freely accessible and usable by all with respect to the license. Any person who wants to improve this specification can do so. How exactly this works can be read on the Github website. The repository can be found at RCM-DX repository and the website at RCM-DX specification website.

1.3 Change history

RCM-DX version	Document version	Date	Autor	Change
1.0.0	0.1	06.03.2015	Martin Frey (SCS)	Initial version
1.0.0	0.2	20.03.2015	Martin Frey (SCS)	Extensions
1.0.0	0.3	15.04.2015	Patrik Wernli (SCS)	Review
1.0.0	0.4	20.04.2015	Martin Frey (SCS)	Extensions and revised
1.0.0	0.5	03.05.2015	Patrik Wernli (SCS)	Formal Adaptions
1.0.0	0.6	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
1.0.0	0.7	13.07.2015	Patrik Wernli (SCS)	Finalized for CDR
1.0.0	0.8	02.10.2015	Martin Frey (SCS)	Event model added, reference to specification event schema added.

RCM-DX version	Document version	Date	Autor	Change
1.0.0	0.9	30.11.2015	Patrik Wernli (SCS)	Adaptions for Infotrans position model. Version concept removed. Event model updated.
1.0.0	0.10	21.12.2015	Martin Frey (SCS)	Review
1.0.0	0.11	21.12.2015	Patrik Wernli (SCS)	Revised after review
1.0.0	0.12	16.02.2016	Pascal Brem (SCS)	Topology model in configuration.
1.0.0	0.13	17.02.2016	Martin Frey (SCS)	Review topology model
1.0.0	0.14	19.02.2016	Pascal Brem (SCS)	Sections added to file format.
1.0.0	0.15	23.02.2016	Martin Frey (SCS)	Global configuration and settings updated
1.0.0	0.16	11.07.2016	Pascal Brem (SCS)	New Hash code attribute for the topology.
1.0.0	0.17	15.07.2016	Pascal Brem (SCS)	New units and data types for positions
1.0.0	0.18	03.01.2018	Pascal Brem (SCS)	New GTG Track Id in the Topology.
1.0.0	0.19	03.01.2018	Pascal Brem (SCS)	Events are stored on session level.
1.1.0	0.20	09.01.2018	Pascal Brem (SCS)	Events and Sections in a group.
1.1.0	0.21	09.01.2018	Patrik Wernli (SCS)	Added chapter “HDF5 File Format Versions”
1.1.0	0.22	11.04.2018	Patrik Wernli (SCS)	Changed document template to official publishing template
1.1.0	0.23	16.08.2018	Pascal Brem (SCS)	Changes in the channel basis definition.
1.1.0	0.24	16.08.2018	Pascal Brem (SCS)	New attributes on the picture block channel.
1.2.0	0.25	04.09.2018	Pascal Brem (SCS)	New minor version.
1.3.0	0.26	28.11.2018	Pascal Brem (SCS)	New minor version for the topology attributes.

RCM-DX version	Document version	Date	Autor	Change
1.4.0	0.27	08.01.2019	Pascal Brem (SCS)	New availability group.
1.4.1	0.28	05.06.2019	Pascal Brem (SCS)	New switchtracks in the DfA
2.0.0	0.29	21.04.2020	Michael Ammann (SBB), Jakob Grilj (SBB)	Adaptation of the structure to new requirements. New major release with version number 2.0, due to major changes in the structure and goal for publication of the specification (open source).
2.0.0	0.30	27.10.2020	Aron Serafini (SCS)	Adjust consistency, measurementMode and processing chapters.
2.0.0	0.31	11.03.2021	Aron Serafini (SCS)	Adjust specification to implementation

1.4 Introduction

1.4.1 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.

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 in 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.

1.4.2 Hints

1.4.2.1 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.

1.4.2.2 Versioning

The RCM-DX data format is subject to changes, these are indicated by the version number in the document, see chapter 1.9.1 Root Group. The version number consists of two numbers, separated by dots and is composed as follows: **[Major].[Minor].[Feature]**. Example: **1.0.0**

Major
Defines the main version and indicates when major changes have been made. These are, for example, that changing the basic structure or renaming groups, datasets or attributes (major, minor attribute as an example), which are mandatory.

Minor
Indicates minor changes, such as changing the name of an attribute that is not mandatory or defining new groups, attributes, or datasets. These changes do not affect anything that cannot be read with an existing RCM-DX read-write library.

Feature
Indicates feature changes.

1.4.2.3 Diagrams

The following figure shows the color coding used in diagrams:

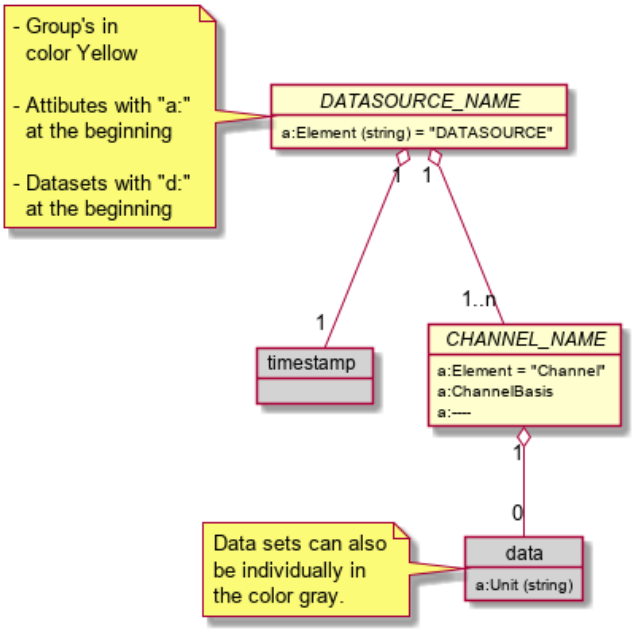

Figure 1: RCM-DX diagram overview

1.5 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.

1.5.1 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`.

1.5.2 Primitive and extended data types

1.5.2.1 Primitive data types

Below are the supported data types used in this specification including a short description. Not all data types possible with HDF5 are described. The read and write library specifies which data types can be used for data sets and attributes.

Data type	Description	Examples
boolean	Two values are possible, `true` or `false`, `0` or `1`	`true`
byte	One byte or an unlimited number of bytes are possible	-
signed integer	Positive and negative values are allowed. Possible bit depths are: 8, 16, 32 or 64	`1256442`; `-62334`
unsigned integer	Only positive values are allowed. Possible bit depths are: 8, 16, 32 or 64	`1256442`; `-62334`
float	Positive and negative values are allowed. Possible bit depths are: 32 or 64	`12.53`; `-3212.546`
string	Multiple characters of undefined length	“RCM-DX is great!”

1.5.2.2 Extended data types

Extended data types use the primitive data types, but have a more specialized format or meaning. The following is a list of the extended data types used.

Name	Description	Example
Timestamp	Unique and worldwide defined format of a time in nanoseconds since January 1, 1970 00:00 UTC without leap seconds, defined under wikipedia.org/wiki/Timestamp. Datatype is `64 bit unsigned integer`	`1553237099000000000`
Enum	Enumerations are predefined (and always capitalized) strings defined in this specification. Datatype is `string`	`MAX`, `MIN`, `RIGHT`, `LEFT`

1.5.3 (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	Optional
`SESSION`	`RCMDX`	no

Name
The name of the group.

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

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.

Optional
If a group is marked as optional (`optional = yes`), it can be omitted if it is not needed. If a group is marked as non-optional (`Optional = no`), then this group must exist if the parent group also exists, otherwise it does not.

1.5.4 (HDF5) Attribute

In the RCM-DX, attributes, groups and datasets can be assigned. The names of the attributes are written in the UpperCamelCase-Notation[1]. Attributes are always of type `HDF5 attribute` unless otherwise specified.

Attributes are described in this specification as follows:

Name	Data type	Parent object	Optional	Description
StartTime	64 bit integer	*SESSION_NAME*	no	Start time in miliseconds, 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 data set, here the name of this group or data set is mentioned.

Optional
If an attribute is marked as optional (`Optional = yes`), it can be omitted if it is not required. If an attribute is marked as non-optional (`Optional = no`), this attribute must be present if the group or dataset to which the attribute applies is also present, otherwise it is not.

Description
Description and or examples of the attribute.

[1]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

1.5.5 (HDF5) Datasets

A channel as a group always has one data set. A channel defines a type of sensor data that is stored in its data set. Several channels form a data source. Further information on the structure is described further down in this specification.

A data set is always of the HDF5 type `HDF5 Dataset`.
Below is a list of possible ways in which data can be stored in the RCM-DX:

Type of storage	Description
Array	Data array of arbitrary length
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[2]. Datasets are described in this specification as follows:

Name	Data type	Parent object	Optional	Storage type
timestamp	Timestamp	*DATASOURCE_NAME*	no	`Array`

Name
The name of the data set.

Data type
Primitive data type of the content in the data set, thus the data type of the contained data.

Parent object
A data set is always assigned to a group, here the name of this group is mentioned.

Optional
If the dataset is marked as optional (`Optional = yes`), it can be omitted if it is not needed. If the dataset is marked as non-optional (`Optional = no`), this dataset must be present if the parent group also has these datasets, otherwise not.

Storage type
One of the storage types described in this chapter.

Descriptions of the data set are added outside the table.

[2]**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

1.5.5.1 HDF5 Chunking

The “HDF5 Chunking” is for data within a data set. 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 data set, 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

1.6 Data structures

Within a channel group, one of the following structures can be contained: Array, Limits, Coordinates, Pictures or Videos. These structures are described in more detail in this chapter.

1.6.1 Array

Channels which record individual measured values contain a data set with the name `data`, this data set is never optional. Single values are stored in this data set as a 1D array, the length of this array (or list) is not limited.

The possible data types are defined in chapter 1.5.2 Primitiv and extended data types

Multidimensional measured values are given their own channel group per dimension and thus their own data set called `data`.

Name	Data type	Parent object	Optional	Storage type
data	A primitive or extended data type	*CHANNEL_NAME*	no	`Array`

The following attributes are assigned to this type of data set:

Name	Data type	Parent object	Optional	Description
`Unit`	string	data set `data`	no	A physical unit or empty if the data does not correspond to a physical unit

1.6.1.1 Limits

A channel group can contain zero or one limit groups. Each limit sub-group *`LIMIT_NAME`* contains its own `timestamp` data set and contains a `duration` data set. The data set `limitvalue` contains the limit values. When which limit value is valid is defined in the dataset `timestamp` and how long it is valid is defined in the dataset `duration`. Values at the same index position of all three data resources belong together.

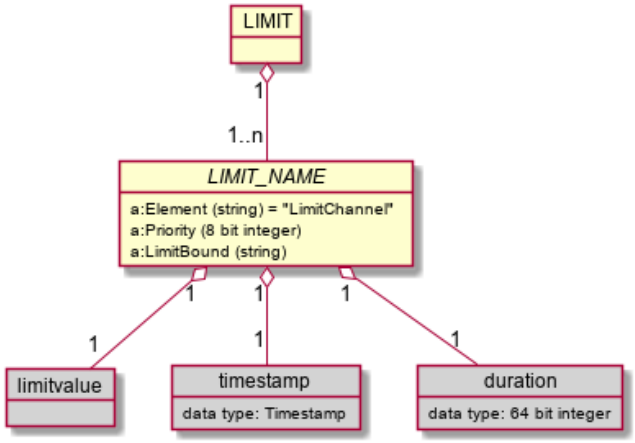

Figure 2: Limit group overview

The group of limit values is defined as follows:

Name	Parent object	Optional
`LIMIT`	*CHANNEL_NAME*	no

The group `LIMIT` now contains further groups, each with the name of the limit exceeding:

Name	Parent object	Optional
LIMIT_NAME	`LIMIT`	no

The following attributes are assigned to this group:

Name	Data type	Parent object	Optional	Description
Priority	8 bit integer	*LIMIT_NAME*	no	Priority of defined limit, lower values are priories.
LimitBound	Enum	*LIMIT_NAME*	no	Defines the type of limit, possible values are `MAX` or `MIN`.

It contains the following datasets:

Name	Data type	Parent object	Optional	Storage type
limitvalue	A primitive or extended data type	*LIMIT_NAME*	no	`Array`
timestamp	Timestamp	*LIMIT_NAME*	no	`Array`
duration	Timestamp	*LIMIT_NAME*	no	`Array`

1.6.2 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 timestamp. Thus there are more entries in these datasets than in the data set `timestamp`. How many entries per timestamp belong to each other (as a group) is stored in another data set with the name `sampleindex`. The data set `sampleindex` is describes in chapter 1.6.2.2 Sample index.

Element	Description
coord	Simple character string for identifying data of type Coordinates
.	Separators
C	Additional character for identifying data of type Coordinates
N	Index number beginning with “0”, increasing for each additional coordinate datasets

The data set is defined as follows:

Name	Data type	Parent object	Optional	Storage type
coord.CN	A primitive or extended data type	*CHANNEL_NA*	no	Single

The following attributes are assigned to this type of data set `coord.CN`:

Name	Data type	Parent object	Optional	Description
`Unit`	string	Dataset `coord.CN`	no	One physical unit or empty if the data does not have any physical unit

1.6.2.1 Coordinate related measured values

Further measured values can be recorded for each coordinate measuring point. These are stored in individual data sets. In the following the definition of these data sets:

Element	Description
value	Simple character string for identifying data of type Coordinates
.	Separators
V	Additional character for identifying data of type value
N	Index number beginning with “0”, increasing for each additional value set

The data set is defined as follows:

Name	Data type	Parent object	Optional	Storage type
value.VN	A primitive or extended data type	*CHANNEL_NAME*	yes	Single

The following attributes are assigned to this type of data set `value.VN`:

Name	Data type	Parent object	Optional	Description
`Type`	string	Dataset `value.VN`	no	Describes the content and type of the data it contains.

1.6.2.2 Sample Index

If datasets are created for coordinates, a data set on the same level and with the name `sampleindex` must be available. The index number of an entry in `coord.CN`, is entered as the start of the next

group. If the data set `sampleindex` has a value of 21 at index zero, the first 20 entries from the data set `coord.CN` belong together, the next data group start at with index number 21. The number of entries in `sampleindex` corresponds to the one in the data set `timestamp`.

The group sizes can vary among themselves, this can be seen from the index positions in `sampleIndex`.

Example
The rail cross profile serves as an example here. At one point on the railroad track, several data 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 data set `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:

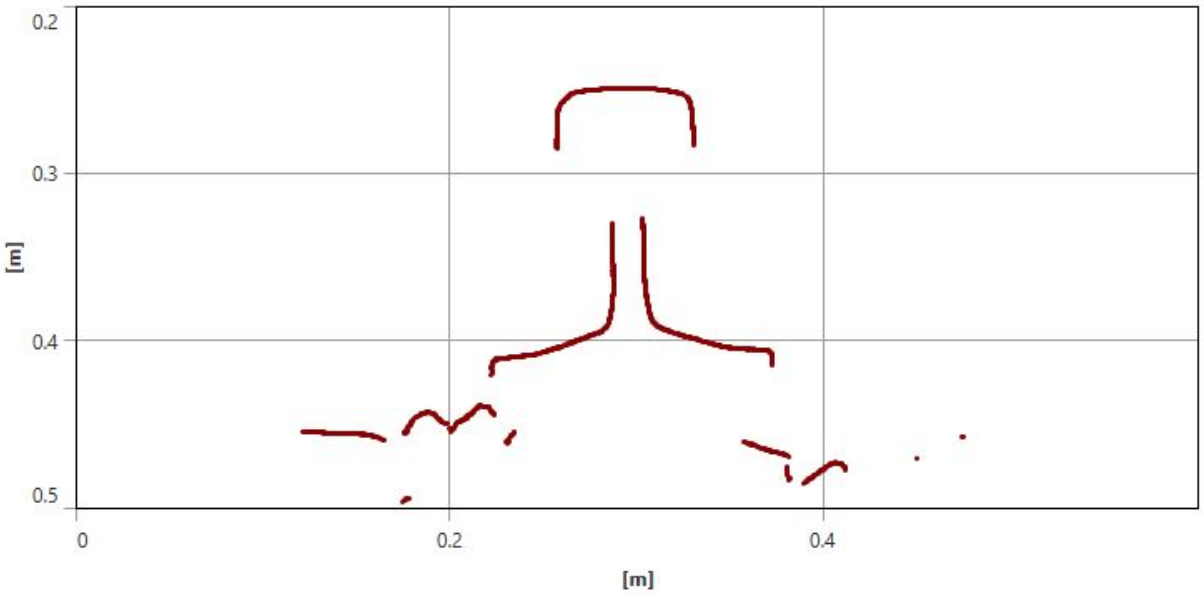

Figure 3: Image of a rail cross section measurement

1.6.3 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 data set. 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	Optional
IMG	*DATASOURCE_NAME*	no

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

Name	Data type	Parent object	Optional	Description
ImageContentTy	string	IMG	yes	Image content type, for example JPEG
ResolutionType	Enum	IMG	no	Description in chapter 1.6.3.1 Image resolution types
ResolutionInfoX	32 bit float	IMG	no	Resolution in X direction
ResolutionInfoY	32 bit float	IMG	no	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.

1.6.3.1 Image resolution types

The “ResolutionType” attribute contains information about the values of the “ResolutionInfoX” and “ResolutionInfoY” attributes.
ResolutionType’ can contain the following values: “none”, “mmPerPixel” or ’dimension

none
No information available.

mmPerPixel
Defines the pixel size in mm. The values in ResolutionInfoX and ResolutionInfoY give information about the real size of a pixel in millimeters.

dimension
Defines the resolution of the image. The values in ResolutionInfoX and ResolutionInfoY give information about the image size in pixels.

1.6.3.2 Naming the data set for an image

Name	Data type	Parent object	Optional	Storage type
img.NNNNNNNNN	integer, bit depth depending on color depth	`IMG`	no	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)

1.6.4 Videos

Name	Data type	Parent object	Optional	Storage type
vid.NNNNNNNNN	integer, bit depth depending on color depth	`VID`	no	image

As with the images, videos can be saved in compressed or uncompressed form. Videos are stored as streams in individual blocks. The blocks are single datasets with a given name.

Name	Parent object	Optional
VID	*DATASOURCE_NAME*	no

1.6.4.1 Name of the data set 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
.	Separator

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

1.7 Time-based data structures

1.7.1 Timestamp

Each entry in a data set of a channel has a reference to an entry in a data set with the name `timestamp`, which lies within the data source group. In this `timestamp` data set, there are as many entries as there are entries in a data set of a channel.

The time stamps are always stored in ascending order.

Name	Data type	Parent object	Optional	Storage type
timestamp	Timestamp	*DATASOURCE_NAME*	no	`Array`

These time stamps are recorded either by a defined distance travelled or by a frequency, this is described in more detail in the chapter 1.9.18.1 Trigger mode.

1.7.2 Durations

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

The differentiation between discrete (data for discrete `timestamp`) and continuous (data for `timestamp` with `duration`) data is done on data source level. The existence or absence of a `duration` array defines if the data source is **“Discrete”** (no `duration` array) or **“Continuous”** (with `duration` array).

Example: Assuming there is a data source with a temperature value every second and a calculated average temperature for every minute. Such an average temperature would be stored in a continuous data source, within an array, with a duration of 60 sec.

Name	Data type	Parent object	Optional	Storage type
duration	64 bit integer	*DATASOURCE_NAME*	**no** for continuous values, otherwise **yes**	`Array`

1.8 RCM-DX file format

The RCM-DX consists of a file format of the HDF5-group 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

1.9 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:

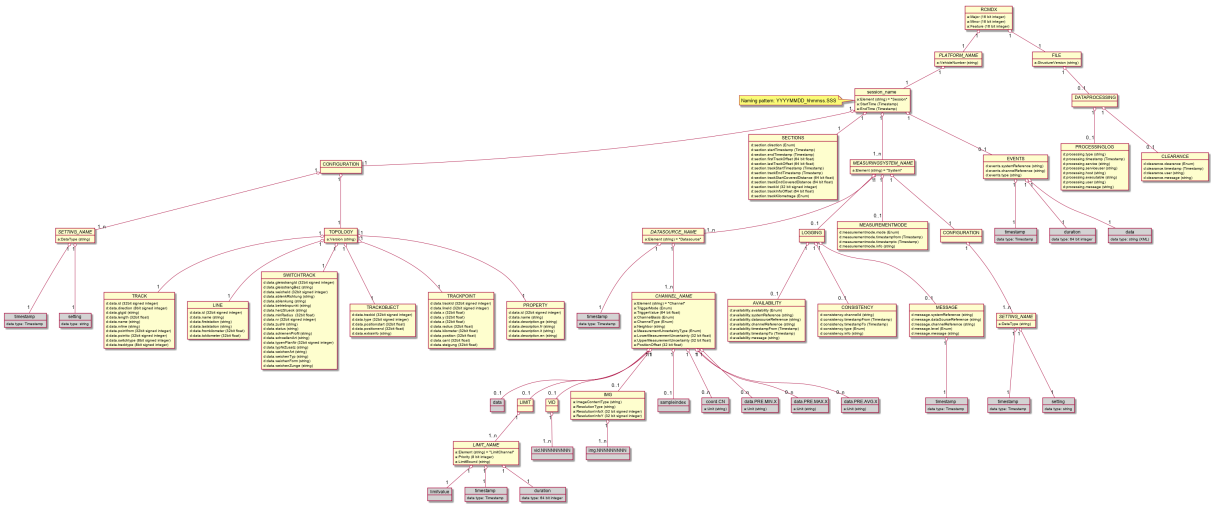

Figure 4: RCM-DX structure overview

Separate and more detailed specifications have been written for individual structure groups. Several measuring instruments can be installed on one measuring platform. 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.

1.9.1 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	Optional
RCMDX	this is the root node	no

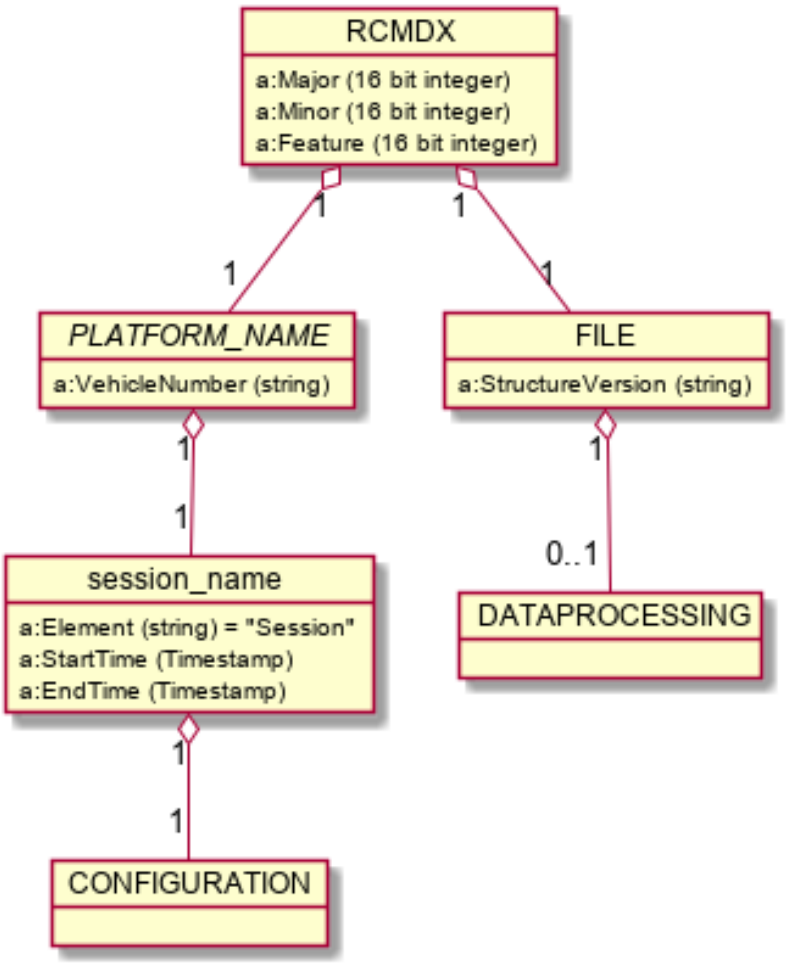

Figure 5: Root group overview

1.9.1.1 Attributes

The following attributes are assigned to the group RCMDX:

Name	Data type	Parent object	Optional	Description
Major	16 bit integer	RCMDX	no	Major Version of the RCM-DX specification that corresponds to the structure of the created file

Name	Data type	Parent object	Optional	Description
Minor	16 bit integer	RCMDX	no	Minor Version of the RCM-DX specification that corresponds to the structure of the created file
Feature	16 bit integer	RCMDX	no	Feature Version of the RCM-DX specification that corresponds to the structure of the created file

Major

Minor

Indicates minor changes, such as changing the name of an attribute that is not mandatory or defining new groups, attributes, or datasets. These changes do not affect anything that cannot be read with an existing RCM-DX read-write library. ### File Group

The file group contains file specific information.

Name	Parent object	Optional
FILE	RCMDX	no

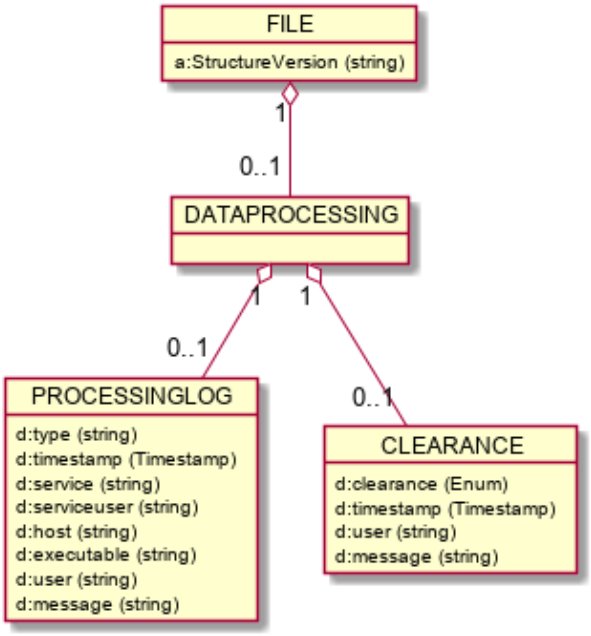

Figure 6: File group overview

1.9.1.2 Attributes

The file group contains the following attribute:

Name	Data type	Optional	Description
StructureVersion	string	no	Version identifier of the platform structure. All underlying systems, datasources and channels can be identified based on this version

1.9.2 Data Processing Group

The data source group `DATAPROCESSING` contains information about the data in this file, and the processing.

Name	Parent object	Optional
`DATAPROCESSING`	`FILE`	yes

1.9.3 Processing log Group

The data source group PROCESSINGLOG 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	Optional
PROCESSINGLOG	DATAPROCESSING	yes

1.9.3.1 Processing log Group datasets

The group PROCESSINGLOG contains one dataset:

Name	Data type	Parent object	Optional	Storage type
type	string	PROCESSINGLOG	no	Array
timestamp	64 bit integer	PROCESSINGLOG	no	Array
service	string	PROCESSINGLOG	no	Array
serviceUser	string	PROCESSINGLOG	no	Array
host	string	PROCESSINGLOG	no	Array
executable	string	PROCESSINGLOG	no	Array
user	string	PROCESSINGLOG	yes	Array
message	string	PROCESSINGLOG	yes	Array

All these datasets represent different columns in a table. Their sizes will therefore always be identical.

type
This record contains the type of the data processing step. The following are possible types: CREATION, CONVERSION, MERGE, CONSISTENCY_CHECK, UNIT_CONVERSION, COMMENT, ...

timestamp
Contains the time of the acquisition of the entry.

service
Name of the service that applied this processing step.

serviceUser
(Technical) User that runs the service.

host
Host that the service is run on.

executable
Executable that the service is run from.

user
Optional user-ID of the initiator of this processing step.

message
Optional message of the user.

1.9.4 Clearance 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 data set.

Name	Parent object	Optional
CLEARANCE	DATAPROCESSING	yes

The group CLEARANCE contains one dataset:

Name	Data type	Parent object	Optional	Storage type
clearance	Enum	CLEARANCE	no	Array
timestamp	64 bit integer	CLEARANCE	no	Array
user	string	CLEARANCE	no	Array
message	string	CLEARANCE	yes	Array

All these datasets represent different columns in a table. Their sizes will therefore always be identical.

clearance
This record contains the enum value of the clearance. The following values are possible:

Name	Description
RELEASED	This file has been released and is ready for analysis
PENDING	This file needs to be reviewed manually
UNRELEASED	This file has not yet been released

timestamp
Contains the time of the acquisition of the entry.

user
User-ID of the initiator of this clearance.

message
Optional message of the user.

1.9.5 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 1.9.5.2 Platforms at the SBB.

Name	Parent object	Optional
Platform	RCMDX	no

1.9.5.1 Attributes

The platform group contains the following attributes:

Name	Data type	Optional	Description
VehicleNumber	string	no	Unique number of the vehicle

1.9.5.2 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	-
DFZ04	OBM-N	-

1.9.6 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 contains exactly one session group.

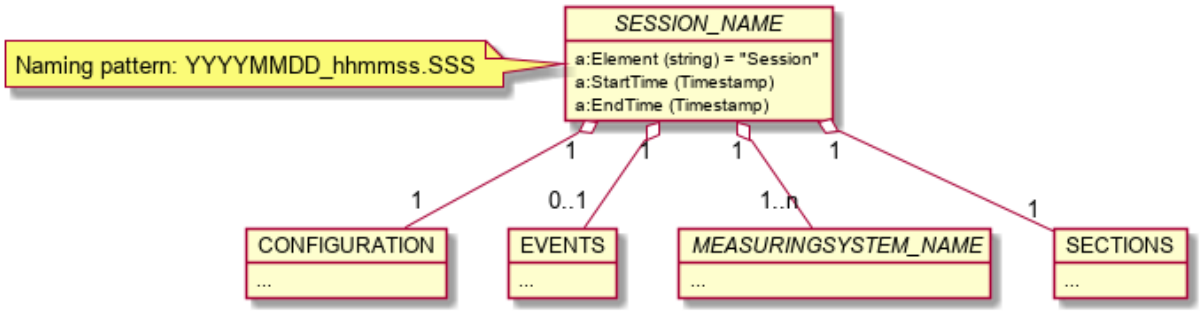

Figure 7: Session group overview

1.9.6.1 Naming

The name of a session group is assigned according to the following pattern:

Name	Parent object	Optional
YYYYMMDD_hhmmss.SSS	`PLATFORM`	no

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

For a certain period of time, only one session can exist in a file, this must be ensured by the creator of the file.

1.9.6.2 Attributes

Name	Data type	Parent object	Optional	Description
Element	string	*SESSION_NAME*	no	Names the type of the group, this is fix “Session”
StartTime	Timestamp	*SESSION_NAME*	no	Timestamp in nanoseconds as start time of the session
EndTime	Timestamp	*SESSION_NAME*	yes	Timestamp in nanoseconds as end time of the session. If the session has not yet been closed, this attribute is missing

1.9.7 Session configuration Group

Configurations can be stored in the datasets of this group. The datasets are designed so that global and network specific configurations can be stored. The configuration can change and have not to be the same in each session.

Name	Parent object	Optional
`CONFIGURATION`	*SESSION_NAME*	no

1.9.8 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	Optional
`TOPOLOGY`	HDF5 Group	*SESSION_NAME*	no

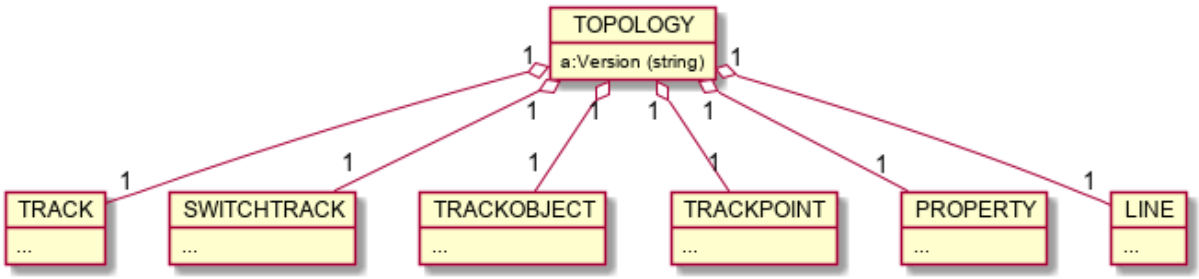

Figure 8: Topology group overview

1.9.8.1 Attributes

The group TOPOLOGY contains the following attributes:

Name	Data type	Parent object	Optional	Storage type
Version	string	TOPOLOGY	no	Array

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.

1.9.9 Track Group

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

Name	Parent object	Optional
TRACK	TOPOLOGY	no

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

Name	Data type	Parent object	Optional	Storage type
direction	8 bit signed integer	TRACK	no	Array
id	32 bit integer	TRACK	no	Array
gtgId	string	TRACK	no	Array
length	string	TRACK	no	Array
name	string	TRACK	no	Array
pointFrom	32 bit integer	TRACK	no	Array
pointTo	32 bit integer	TRACK	no	Array
switchType	8 bit signed integer	TRACK	no	Array
trackType	8 bit signed integer	TRACK	no	Array

direction
The direction of a switch is specified in this data set.
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

id
ID of the track.

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.

pointFrom
ID of the starting point of the track section.

pointTo
ID of the end point of the track section.

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

trackType
The number in the *trackType* data set 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

1.9.10 Line Group

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

Name	Parent object	Optional
`LINE`	`TOPOLOGY`	no

The following datasets are included in this group:

Name	Data type	Parent object	Optional	Storage type
id	32 bit signed integer	`LINE`	no	`Array`
name	string	`LINE`	no	`Array`

Name	Data type	Parent object	Optional	Storage type
firstStation	string	LINE	no	Array
lastStation	string	LINE	no	Array
fromKilometer	string	LINE	no	Array
toKilometer	string	LINE	no	Array

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.

1.9.11 Switch Track Group

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

Name	Parent object	Optional
SWITCHTRACK	TOPOLOGY	no

The following datasets are included in this group:

Name	Data type	Parent object	Optional	Storage type
gleisstrangId	32 bit signed integer	SWITCHTRACK	no	Array
gleisstrangBez	string	SWITCHTRACK	no	Array

Name	Data type	Parent object	Optional	Storage type
soft	32 bit signed integer	SWITCHTRACK	no	Array
deflecting direction	string	SWITCHTRACK	no	Array
distraction	string	SWITCHTRACK	no	Array
operating point	string	SWITCHTRACK	no	Array
herzStueck	string	SWITCHTRACK	no	Array
minRadius	32 bit signed integer	SWITCHTRACK	no	Array
nr	32 bit signed integer	SWITCHTRACK	no	Array
zusNr	string	SWITCHTRACK	no	Array
rail profile	string	SWITCHTRACK	no	Array
status	string	SWITCHTRACK	no	Array
rail profile	string	SWITCHTRACK	no	Array
thresholdArt	string	SWITCHTRACK	no	Array
typesPlanNr	32 bit signed integer	SWITCHTRACK	no	Array
typeNraddition	string	SWITCHTRACK	no	Array
softArt	string	SWITCHTRACK	no	Array
softType	string	SWITCHTRACK	no	Array
softForm	string	SWITCHTRACK	no	Array
soft tongue	string	SWITCHTRACK	no	Array

gleisstrangId
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.

1.9.12 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	Optional
TRACKOBJECT	TOPOLOGY	no

The following datasets are included in this group:

Name	Data type	Parent object	Optional	Storage type
trackId	32 bit signed integer	TRACKOBJECT	no	Array
type	32 bit signed integer	TRACKOBJECT	no	Array
positionStart	32 bit signed integer	TRACKOBJECT	no	Array
positionEnd	32 bit signed integer	TRACKOBJECT	no	Array
extraInfo	string	TRACKOBJECT	no	Array

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.

1.9.13 Track Point Group

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

Name	Parent object	Optional
TRACKPOINT	TOPOLOGY	no

The following datasets are included in this group:

Name	Data type	Parent object	Optional	Storage type
trackId	32 bit signed integer	TRACKPOINT	no	Array
lineId	32 bit signed integer	TRACKPOINT	no	Array
x	32 bit float	TRACKPOINT	no	Array
y	32 bit float	TRACKPOINT	no	Array
z	32 bit float	TRACKPOINT	no	Array
radius	32 bit float	TRACKPOINT	no	Array
kilometers	32 bit float	TRACKPOINT	no	Array
position	32 bit float	TRACKPOINT	no	Array
cant	32 bit float	TRACKPOINT	no	Array
inclination	32 bit float	TRACKPOINT	no	Array

trackId
Reference to the ID of the track section.

lineId
Reference to the ID of the line.

x
X-Koordiante of the point.

y
Y-Koordiante of the point.

z
Z-Koordiante of the point.

radius
The radius of a point, given in meters.

kilometers
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.

1.9.14 Property Group

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

Name	Parent object	Optional
PROPERTY	TOPOLOGY	no

The following datasets are included in this group:

Name	Data type	Parent object	Optional	Description
propertyId	32 bit signed integer	PROPERTY	no	Array
name	string	PROPERTY	no	Array
description.ge	string	PROPERTY	no	Array
description.fr	string	PROPERTY	no	Array
description.it	string	PROPERTY	no	Array
description.en	string	PROPERTY	yes	Array

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.

1.9.14.1 Setting Group

Within this group there are further groups whose names identify the type of setting.
In the following *SETTING_NAME* is used as placeholder of the actual name of a setting.

Name	Parent object	Optional
SETTING_NAME	`CONFIGURATION`	no

This group contains two datasets:

Name	Data type	Parent object	Optional	Storage type
setting	string	*SETTING_NAME*	no	`Array`
timestamp	64 bit integer	*SETTING_NAME*	no	`Array`

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	Optional	Description
DataType	string	`setting`	no	Defines the datatype of the configuration within the data set `setting`. Data type specified as MIME[3] type, for example `Content-Type: <text/strings>`

1.9.15 Section Group

The group `SECTION`, contains information about a session.

Name	Parent object	Optional
`SECTION`	*SESSION_NAME*	no

1.9.15.1 Data fields

The following data fields are contained in the group “SECTION”:

Name	Data type	Parent object	Optional	Storage type
direction	Enum	`SECTION`	no	`Array`
startTimestamp	Timestamp	`SECTION`	no	`Array`
endTimestamp	Timestamp	`SECTION`	no	`Array`
firstTrackOffset	64 bit float	`SECTION`	no	`Array`
lastTrackOffset	64 bit float	`SECTION`	no	`Array`
trackInfoOffset	64 bit float	`SECTION`	no	`Array`
trackId	32 bit signed integer	`SECTION`	no	`Array`
trackStartTimestamp	Timestamp	`SECTION`	no	`Array`
trackEndTimestamp	Timestamp	`SECTION`	no	`Array`
trackKilometrage	Enum	`SECTION`	no	`Array`
trackStartCoveredD	64 bit float	`SECTION`	no	`Array`
trackEndCoveredDistance	64 bit float	`SECTION`	no	`Array`

direction
Indicates the orientation of travel of the measuring vehicle in the given section. This array contains only as many entries as there are sections.

Possible values of this enumeration are:

Value	Orientation of travel
FORWARD	forward
BACKWARD	backward

startTimestamp

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

endTimestamp
End time of the section as time stamp since 1.1.1970 at 00:00 UTC. `endTimestamp` must be greater than `startTimestamp`.

firstTrackOffset
Indicates the distance in meters between the start of the track and the position at the beginning of the section. 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.

trackInfoOffset
This data set lists how many entries in the datasets of the “Track list group” belong to section. One entry is created per section in a session and the number of entries is defined. The number of tracks belonging to each section can be determined by calculating the specified offset value at the x position minus the offset value at the $x - 1$ position.

trackId
ID of the tracks that are part of the section.

trackStartTimestamp
Time since beginning of the session at which the measurement on a track started.

trackEndTimestamp
Time since the beginning of the session at which the measurement on a track ended. `endTimestamp` must be greater than `startTimestamp`.

trackKilometrage
Orientation of the track with respect to the section’s driving direction.

The following values are allowed:

Value	Meaning
ASCENDING	The track kilometrage increases with the driving direction
DESCENDING	The track kilometrage decreases with the driving direction

trackStartCoveredDistance
Start covered distance of the track in the section.

trackEndCoveredDistance
End covered distance of the track in the section.

1.9.16 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 among railway companies and with 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.

Name	Parent object	Optional
MEASURINGSYSTEM_NAME	`PLATFORM`	no

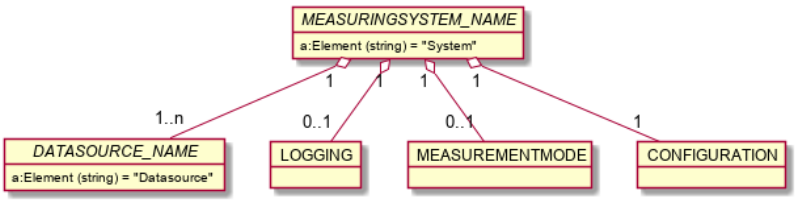

Figure 9: Measuring system overview

1.9.16.1 Attributes

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

Name	Data type	Parent object	Optional	Description
Element	string	*MEASURINGSYSTEM_NAM*	no	Indicates the type of the group, this is fixed `System`

1.9.17 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.

Name	Parent object	Optional
DATASOURCE_NAME	*MEASURINGSYSTEM_NAME*	no

A timestamp 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 1.9.18.1 Trigger mode.

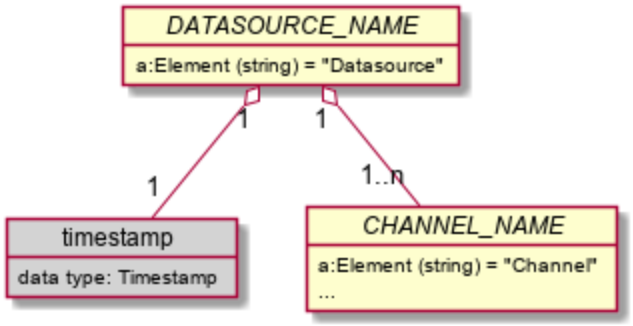

Figure 10: Datasource group overview

1.9.17.1 Attributes

The following attribute is assigned to the group:

Name	Data type	Parent object	Optional	Description
Element	string	*DATASOURCE_NAME*	no	Contains the type of the group, this is fix `Datasource`

1.9.17.2 Example

In our example the name of the data source group is assigned, which should contain our environmental measurement data, which we call `ENVIRONMENT`.

1.9.17.3 Timestamp data set

Each data source group contains a data set 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 1.7.1 Timestamp Array!

1.9.18 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.

Name	Parent object	Optional
CHANNEL_NAME	*DATASOURCE_NAME*	no

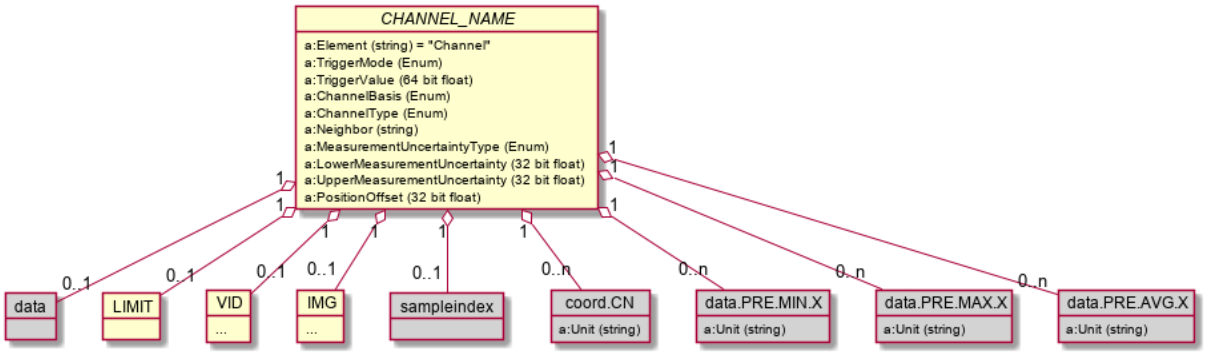

Figure 11: Channel group overview

The following attributes are contained in this group:

Name	Data type	Parent object	Optional	Description
TriggerMode	Enum	*CHANNEL_NAME*	no	See chapter 1.9.18.1 Trigger Mode
TriggerValue	64 bit float	*CHANNEL_NAME*	no	See below
ChannelBasis	Enum	*CHANNEL_NAME*	no	See below
ChannelType	Enum	*CHANNEL_NAME*	no	See below
Neighbor	string	*CHANNEL_NAME*	no	See below
MeasurementUncertaintyType	Enum	*CHANNEL_NAME*	no	See below
LowerMeasurementUncerta	32 bit float	*CHANNEL_NAME*	no	See below
UpperMeasurementUncertainty	32 bit float	*CHANNEL_NAME*	no	See below

Name	Data type	Parent object	Optional	Description
PositionOffset	32 bit signed float	*CHANNEL_NAME*	no	See below
Element	string	*CHANNEL_NAME*	no	Contains the type of the group, this is fix `Channel`

TriggerValue
The trigger value defines when a value is measured based on the trigger mode.

> In a data source group there is always only one common trigger mode! A mixture within the group is not permitted!

> The attribute has the value “0.0” if `TriggerMode` contains the value `EVENTS`.

ChannelBasis
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
TOTAL	The channel has no dependency on direction of motion or vehicle orientation
SENSOR_LEFT	Signal reflects the left hand side of the vehicle irrespective of it’s orientation or motion
SENSOR_RIGHT	Signal reflects the right hand side of the vehicle irrespective of it’s orientation or motion
RAIL_LEFT	Signal reflects the left side with respect to direction of motion
RAIL_RIGHT	Signal reflects the right side with respect to direction of motion
SENSOR_VERTICAL_LEFT	TODO
SENSOR_VERTICAL_RIGHT	TODO
MOVE_DIRECTION_VERTICAL_LEFT	TODO
MOVE_DIRECTION_VERTICAL_RIGHT	TODO

Value	Description
SENSOR_VERTICAL_TOTAL	TODO
MOVE_DIRECTION_VERTICAL_TOTAL	TODO

ChannelType
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.

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.

MeasurementUncertaintyType
Defines the type in which the measurement uncertainty is set. The following values are possible:

Value	Description
ABSOLUTE	Absolute values
RELATIVE	Relative values

LowerMeasurementUncertainty
This attribute contains the measurement accuracy of the channel according to the specifications of the measurement system. This is the lower bound for the measurement uncertainty range.

UpperMeasurementUncertainty
This attribute contains the measurement accuracy of the channel according to the specifications of the measurement system. This is the upper bound for the measurement uncertainty range.

PositionOffset

Describes the distance between a defined zero point (position) on the measuring vehicle and a the measuring sensor. This specification is used to convert the exact time at which the measurement was taken to a defined zero point. The value can be positive or negative and has the unit millimeter.

1.9.18.1 TriggerMode

This attribute defines how the data was recorded.

Possible values are:

Value	Description	Unit
TIME	Time-based measurement data recording	Nanoseconds
DISTANCE	Distance-based measurement data acquisition	Milimeter
EVENT	EVENT based recording	none

1.9.18.2 Data object

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

Name	HDF5 Type	Optional
data	HDF5 Dataset	no

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

The data set needs more information, this is given as attributes:

Name	Data type	Parent object	Optional	Storage type
Unit	string	*CHANNEL_NAME*	no	`Array`

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 data set and the possible data that can be stored are described in more detail in the chapter 1.5.5 Dataset.

1.9.19 Measurement mode group

The following group contains important information about the measurement mode of the system.

Name	Parent object	Optional
MEASUREMENTMODE	*MEASURINGSYSTEM_NAME*	yes

The group MEASUREMENTMODE contains one dataset:

Name	Data type	Parent object	Optional	Storage type
measurementM	Enum	MEASUREMENTMODE	no	Array
timestampFrom	64 bit integer	MEASUREMENTMODE	no	Array
timestampTo	64 bit integer	MEASUREMENTMODE	no	Array
info	string	MEASUREMENTMODE	yes	Array

All these datasets represent different columns in a table. Their sizes will therefore always be identical.

1.9.19.1 Measurement mode (enum)

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

Name	Description
PRODUCTIVE	Productive data that will be further used.
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.

1.9.20 Logging Group

The logging group contains information about the status of the measuring systems. The data is divided into two subgroups, AVAILABILITY and MESSAGES. These are described in separate chapters.

Name	Parent object	Optional
LOGGING	*MEASURINGSYSTEM_NAME*	yes

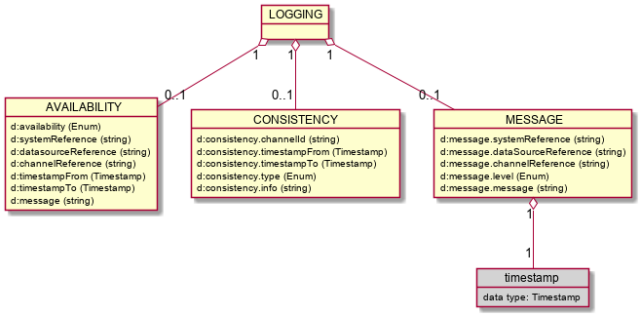

Figure 12: Logging group overview

1.9.20.1 Availability Group

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

Name	Parent object	Optional
AVAILABILITY	LOGGING	yes

The following datasets are included in this group:

Name	Data type	Parent object	Optional	Storage type
availability	Enum	AVAILABILITY	no	Array
systemReference	string	AVAILABILITY	no	Array
datasourceReference	string	AVAILABILITY	yes	Array
channelReference	string	AVAILABILITY	yes	Array
timestampFrom	Timestamp	AVAILABILITY	no	Array
timestampTo	Timestamp	AVAILABILITY	no	Array
message	string	AVAILABILITY	no	Array

availability

See chapter 1.9.20.1.1 Availability

systemReference

A reference to the measurement system.

datasourceReference
A reference to a data source.

channelReference
A reference to a channel.

message
This data set contains one message per entry about a failure of a measuring instrument.

timestampFrom
Start timestamp of the availability entry.

timestampTo
End timestamp of the availability entry.

1.9.20.1.1 Availability

Defines the severity of the failure or interruption of a measurement system. Following values are possible:

level	description
MONITORED	The measurement system or sensor was monitored during this period. All other
FAILURE	The measuring system or sensor has failed and has not recorded any measured values during the session.
WARNING	The measuring system or the sensor may have had a failure and the measured values may not be correct.

1.9.20.2 Consistency group

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.

Name	Parent object	Optional
CONSISTENCYDATA	LOGGING	yes

The group `CONSISTENCYDATA` contains one dataset:

Name	Data type	Parent object	Optional	Storage type
channelId	string	`CONSISTENCYDATA`	no	`Array`
timestampFrom	64 bit integer	`CONSISTENCYDATA`	no	`Array`
timestampTo	64 bit integer	`CONSISTENCYDATA`	no	`Array`
consistencyType	Enum	`CONSISTENCYDATA`	no	`Array`
consistencyInfo	string	`CONSISTENCYDATA`	yes	`Array`

All these datasets represent different columns in a table. Their sizes will therefore always be identical.

channelId
Channel reference in which the finding was detected.

timestampFrom
Start timestamp of the consistency.

timestampTo
End timestamp of the consistency. Note that consistencies can never overlap and will always be continuous (without holes) within sections.

consistencyType
The consistency type can have the following values:

Name	Description
CONSISTENT	The referenced data was checked and is consistent
INCONSISTENT	The referenced data was checked and is inconsistent
NO_DATA	Consistency has been checked, but no data was found

consistencyInfo
Contains additional information about this consistency (E.g. which rule decided the consistency type).

1.9.20.3 Message Group

This group contains messages, generated from a measurement system or a person, structured in data set.

Name	Parent object	Optional
MESSAGES	LOGGING	yes

The following datasets are included in this group:

Name	Data type	Parent object	Optional	Storage type
message	string	MESSAGES	no	Array
level	Enum	MESSAGES	no	Array
systemReference	string	MESSAGES	no	Array
datasourceReference	string	MESSAGES	no	Array
channelReference	string	MESSAGES	no	Array
timestamp	Timestamp	MESSAGES	no	Array

message
This data set contains one message per entry about a failure of a measuring instrument.

systemReference
A reference to the measurement system.

datasourceReference
A reference to a data source.

channelReference
A reference to a channel.

level
See chapter 1.9.20.3.1 Message level

1.9.20.3.1 Message level

Defines the importance of a message. Following values are possible:

level	description
INFO	The message is only informative.
WARNING	The message indicates a warning.
ERROR	The message indicates an error.

1.9.21 EVENTS Group

The EVENTS 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. 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	Optional
EVENTS	HDF5 Group	*SESSION_NAME*	yes

Within the group there are the following data fields:

Name	Data type	Parent object	Optional	Storage type
systemRef	string	EVENTS	no	Array
channelRef	string	EVENTS	yes	Array
eventtype	string	EVENTS	no	Array
data	string	EVENTS	no	Array
duration	64 bit signed integer	EVENTS	no	Array
timestamp	Timestamp	EVENTS	no	Array

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

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

Name	Data type	Parent Object	Optional	Storage Type
systemRef	string	EVENTS	no	Array

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

Name	Data type	Parent Object	Optional	Storage Type
channelRef	string	EVENTS	yes	Array

data This data set 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 (Defect EVENT type, Detected object EVENT type, Limit EVENT type).
A type can be stored for each EVENT. These are explained in more detail below.

Name	Data type	Parent object	Optional	Storage type
data	string	EVENTS	no	Array

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

Name	Data type	Parent object	Optional	Storage type
duration	64 bit integer	EVENTS	no	Array

type Contains the type of an EVENT.

Name	Data type	Parent object	Optional	Storage type
type	string	EVENTS	no	Array

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.

1.9.21.1 Defect EVENT type

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 1.10.2 EventsDefect.

1.9.21.1.1 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

1.9.21.1.2 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

1.9.21.2 Detected object EVENT type

These events indicate an object found during a diagrose 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 1.10.3 EventsGeneric.

1.9.21.2.1 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

Name	Description	Parent object
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

1.9.21.2.2 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

1.9.21.3 Limit violation EVENT type

Limit value exceedances of measured values of a channel can also be recorded as events.
The XML schema can be found in chapter 1.10.3 EventsGeneric.

1.9.21.3.1 XML elements

Name	Description	Parent object
LimitViolation	Root Element	none

1.9.21.3.2 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 EVENTS	LimitViolation

1.9.21.4 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 1.10.1 EventsComment.

1.9.21.4.1 XML elements

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

1.9.21.4.2 XML attributes

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

1.9.21.5 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 1.10.3 Events-Generic.

1.9.21.5.1 XML elements

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

1.9.21.5.2 XML attributes

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

1.9.22 Measurement system configuration Group

The structure of this group is the **same** as for the configuration group below the group *SESSION_NAME*: 1.9.7 Session configuration group, but **the parent group** is *MEASURINGSYSTEM_NAME*: 1.9.16 Measuring system group. The measuring system configuration group also does not contain the Topology.

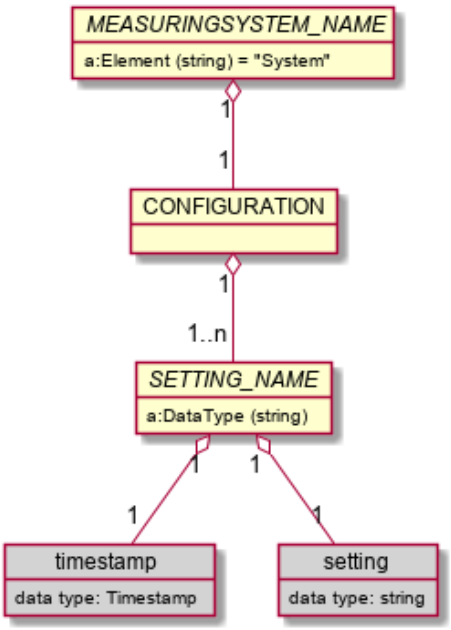

Figure 13: Measurement configuration group overview

1.10 XML Schema Definitions

1.10.1 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"
```

```
  elementFormDefault="qualified">

  <xs:include schemaLocation="../RcmDxDataTypes.xsd" />

  <xs:element name="Comment">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="Username" type="xs:string" use="required"
              />
          <xs:attribute name="ID" type="tns:UUID" use="required" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

1.10.2 Events Defect

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://www.sbb.ch/RCMDX/Events/Defect"
  targetNamespace="http://www.sbb.ch/RCMDX/Events/Defect"
  elementFormDefault="qualified">

  <xs:include schemaLocation="../RcmDxDataTypes.xsd" />

  <xs:element name="Defect">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PossibleDefectNames" type="xs:string"
   minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="PossibleClassifications" type="xs:string"
           minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="PossibleValidationResults" type="xs:string"
           minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="Classification" type="xs:string" use="
         required" />
```

```
      <xs:attribute name="DefectName" type="xs:string" use="required" />
      <xs:attribute name="Details" type="xs:string" use="required" />
      <xs:attribute name="Parameter1Name" type="xs:string" />
      <xs:attribute name="Parameter1Value" type="xs:string" />
      <xs:attribute name="Parameter2Name" type="xs:string" />
      <xs:attribute name="Parameter2Value" type="xs:string" />
      <xs:attribute name="Parameter3Name" type="xs:string" />
      <xs:attribute name="Parameter3Value" type="xs:string" />
      <xs:attribute name="ID" type="tns:UUID" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

1.10.3 Events Generic

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://www.sbb.ch/RCMDX/Events/Generic"
  targetNamespace="http://www.sbb.ch/RCMDX/Events/Generic"
  elementFormDefault="qualified">

  <xs:include schemaLocation="../RcmDxDataTypes.xsd" />

  <xs:element name="Corrupt">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="Username" type="xs:string" use="required" />
          <xs:attribute name="ID" type="tns:UUID" use="required" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="ObjectConsistencyXml">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Ok" />
      <xs:enumeration value="OnlyInReal" />
      <xs:enumeration value="OnlyInData" />
```

```
25        <xs:enumeration value="Measured" />
26      </xs:restriction>
27    </xs:simpleType>
28
29    <xs:element name="DetectedObject">
30      <xs:complexType>
31        <xs:sequence>
32          <xs:element name="object" minOccurs="1" maxOccurs="1">
33            <xs:complexType>
34              <xs:simpleContent>
35                <xs:extension base="xs:string">
36                  <xs:attribute name="Type" type="xs:string" use="
                        required" />
37                  <xs:attribute name="Description" type="xs:string" use="
                        required" />
38                  <xs:attribute name="ObjectConsistency" type="tns:
                        ObjectConsistencyXml" use="required" />
39                </xs:extension>
40              </xs:simpleContent>
41            </xs:complexType>
42          </xs:element>
43          <xs:element name="Reference" minOccurs="0" maxOccurs="unbounded
              ">
44            <xs:complexType>
45              <xs:simpleContent>
46                <xs:extension base="xs:string">
47                  <xs:attribute name="ReferenceSystem" type="xs:string"
                        use="required" />
48                </xs:extension>
49              </xs:simpleContent>
50            </xs:complexType>
51          </xs:element>
52          <xs:element name="ObjectAttribute" minOccurs="0" maxOccurs="
              unbounded">
53            <xs:complexType>
54              <xs:simpleContent>
55                <xs:extension base="xs:string">
56                  <xs:attribute name="Key" type="xs:string" use="required
                        " />
57                </xs:extension>
58              </xs:simpleContent>
59            </xs:complexType>
60          </xs:element>
```

```
      </xs:sequence>
      <xs:attribute name="ID" type="tns:UUID" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="LimitViolation">
    <xs:complexType>
      <xs:attribute name="TimestampMaxViolation" type="xs:long" use="
    required" />
      <xs:attribute name="ViolatedLimit" type="xs:string" use="required
          " />
      <xs:attribute name="ID" type="tns:UUID" use="required" />
    </xs:complexType>
  </xs:element>

</xs:schema>
```

1.10.4 RCM-DX Data types

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
  <xs:simpleType name="restrictedString">
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="512" />
      <xs:pattern value="[a-zA-Z0-9_\-\.]+" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="restrictedStringWithColon">
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="512" />
      <xs:pattern value="[a-zA-Z0-9_\-\.:]+" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="restrictedID">
    <xs:restriction base="xs:ID">
      <xs:minLength value="1" />
      <xs:maxLength value="512" />
      <xs:pattern value="[a-zA-Z0-9_\-\.]+" />
```

```
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="restrictedIDREF">
    <xs:restriction base="xs:IDREF">
      <xs:minLength value="1" />
      <xs:maxLength value="512" />
      <xs:pattern value="[a-zA-Z0-9_\-\.]+" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="versionString">
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
      <xs:maxLength value="32" />
      <xs:pattern value="[a-zA-Z0-9_\-\.]+" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="portNumber">
    <xs:restriction base="xs:int">
      <xs:minExclusive value="0" />
      <xs:maxInclusive value="65535" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ipAddress">
    <xs:restriction base="xs:string">
      <xs:pattern value="(([0-9]{1,3}\.){3}[0-9]{1,3})" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="network">
    <xs:restriction base="xs:string">
      <xs:pattern value="(([0-9]{1,3}\.){3}[0-9]{1,3})/[0-9]{1,2}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="hostName">
    <xs:restriction base="xs:string">
      <xs:pattern value="(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-]*[a-zA-
          Z0-9])\.)*([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-]*[A-Za-z0-9])"
          />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ipAddressOrHostName">
    <xs:union memberTypes="ipAddress hostName" />
  </xs:simpleType>
  <xs:simpleType name="nonNegativeInt">
```

```
63      <xs:restriction base="xs:int">
64        <xs:minInclusive value="0" />
65      </xs:restriction>
66    </xs:simpleType>
67    <xs:simpleType name="positiveInt">
68      <xs:restriction base="xs:int">
69        <xs:minInclusive value="1" />
70      </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>
```