SiLA Data Capture Standard

Specification

Changes

Last storage: 1-Nov-13 16:08

Datum	Version	Reason	Description	Author
2011-06-19	0.01	First edit		TWe
2012-07-12	0.8	Unappropriate Structure	Reorder	HBä
2012-07-23	0.9	Reorganized chapter 4	Moved content of chapter 4 into chapter 2.	TWe
2012-08-29	0.91	A lot of changes	Adapted to changes in schema, new examples, etc.	ReHo
2013-03-01	0.10	A lot of changes	Edit text. Changed xml-listings from pictures to textboxes for better readability. Update Figures for V15. Add Chapter 5 with rules.	Bepa
2013-07-22	0.11		Correct wording and include feedback	Bepa
2013-08-20	1.0		Preparing document for ballot	Bepa
2013-11-01	1.0	Final Version	Include Feedback from balloting (wording)	Bepa

Contents

1. Scope of Document

In automated laboratory systems and also in standalone laboratory instruments data will be generated which are important for later analysis. The goal of the *SiLA Data Capture Standard*, described in this document, is to define rules for formatting such data in a device independent manner.

The main focus is on data produced by microtiterplate readers. But the standard is also suitable for

- liquid handling data (e.g. reagent transfer between microtiterplates)
- content data of storages (e.g. content of microtiterplates in an incubator)
- any other data

It is not the aim of this document to describe a standardized representation of image data. Following the herein described guidelines, it is however possible to generate a reference to stored image data by filename and file-storage-location.

In the following chapters most of the examples and illustrating descriptions are for measurements with plate readers.

For correct interpretation of the generated data and also for traceability there is additional information about the context of the data generation necessary. The standard defines, how this information should be included and related to the data.

The *SiLA Data Capture Standard* should be seen in relation to the SiLA standard *SiLA Device Control & Data Interface Specification* [DCDIS]. In the DCDIS is inter alia defined how data has to be transferred between the device and the controlling software. The standard in this document describes the formatting of such data.

1.1. Notational conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

References to documents, listed in section 6, are enclosed in square brackets like [XSD].

SiLA defined terms will be marked by *cursive letters.*

Figures are shown mostly as UML diagrams. Because they should be understandable without deep UML knowledge the diagrams may be incomplete in some details.

2. Architecture

All SiLA compliant instrument data shall be represented in Extended Markup Language XML. In the interface specification [DCDIS], a schema is defined for the XML data element of the Response Event and the Data Event http://sila-standard.org/schemata/ResponseType_1.2.xsd.

According to this schema, the data element can contain either a single Parameter, or several ParameterSets, or a structure that complies with the XML Schema-type anyType. If a command execution produces more complex data than can be expressed by a Parameter or by ParameterSets, the field of AnyData may be filled by an XML structure that complies with the schema of this document – **the Data Capture schema.**

Additionally to the actual result data, there is the response data definition in the interface specification. This is done by an annotation extension; see [DCDIS, Section 2.1.7]. The name for the TypeName shall be “containerData”.

The final XML must be validated according to the Data Capture schema, which is divided into two XSD-schema files:

1. *Data Capture Core*
2. *Data Capture Dictionary*

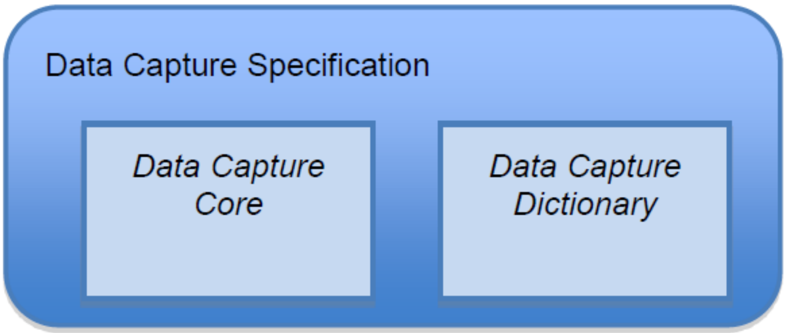

Figure 1 Data Capture Specification - Components

The separation of the Data capture schema into two files helps to simplify maintenance and version control of the specification, because we expect that the definition of the Core will be more stable than the Dictionary, which likely will be expanded regularly.

The *Data Capture Core* schema describes the complete structure of a SiLA compliant data file in XML. The Data Capture Core schema references the Data Capture Dictionary schema.

Because the knowledge of the Data Capture Dictionary is helpful to understand fundamental concepts of the Data Capture Core, the dictionary is described in the next chapter. Details of the *Data Capture Core* can be found in chapter 4.

3. Data Capture Dictionary

The *Data Capture Dictionary* schema defines properties, so called *commonProperties*, which can be used to describe the measurements itself and their results.

An additional part of the schema of the *Data Capture Dictionary* enables to create custom defined *specificProperties* that might be necessary to describe new or more specialized measurement techniques in greater detail.

It is not the goal of the standard to define a complete dictionary of all possible techniques for data generation. But with the *commonProperties* a large set of commonly used terms are contained in the schema. These terms MUST be used with highest priority. Only if there is no term within the *commonProperties* for certain information, a *specificProperty* shall be used.

Newly emerging and often used *specificProperties* will be added to the *Data Capture Dictionary* on regular basis so that the number of *commonProperties* will grow over time. Backwards compatibility shall be ensured.

The current version of the *Data Capture Dictionary* is available at:
http://sila-standard.org/schemata/DataCapture_Properties_1.0.xsd.

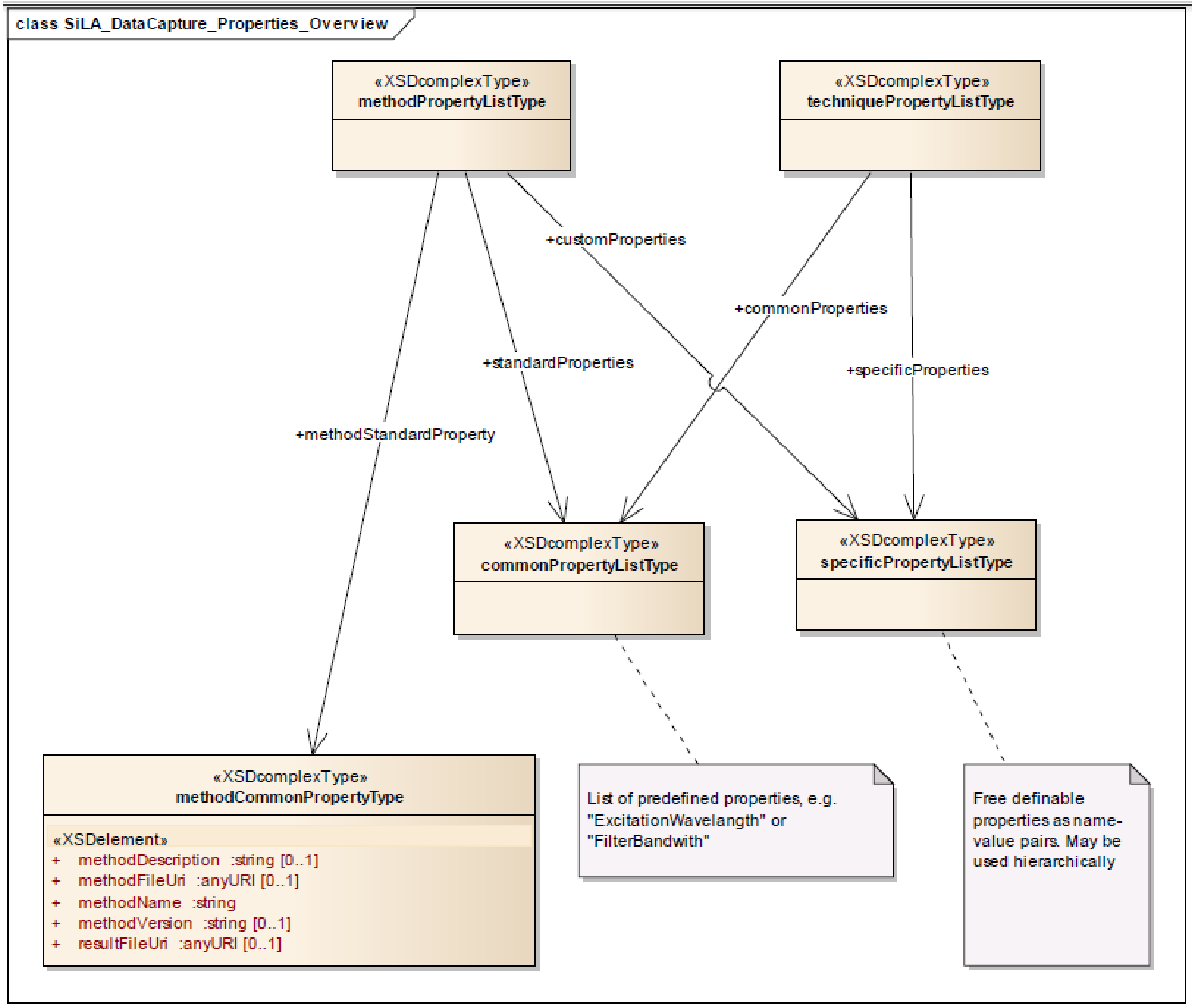

Figure 2 Data Capture Properties - Overview

3.1.1 Common Properties

The *commonProperties* comprise properties that are used very often in screening and drug discovery. They include the most common detection technologies of microtiterplate readers like fluorescence and absorption. But also for other data generation entities some useful properties are predefined.

For example fluorescence measurements would use the following common properties:

- ExcitationWavelength
- EmissionWavelength
- ExcitationBandwidth
- EmissionBandwidth
- …, etc.

to describe the measurements settings. Other properties like

- OperatorName
- Temperature
- LightSource
- MethodName
- …, etc.

might be used to describe who, when and how the experiment was performed.

In the *Data Capture Dictionary* schema properties are arranged according to their data type. There are properties of data type String, Integer, Double, Duration, DateTime, Boolean and Enumeration. For the latter also the definitions of these enumerations are included in the schema. Most of the properties also have units. Again a number of possible units are already defined in the dictionary. The Listing 1 shows the usage of some *commonProperties*.

```xml
<commonProperties>
   <AbsorptionWavelengthunit="nm">340</AbsorptionWavelength>
   <AbsorptionBandwidthunit="nm">25</AbsorptionBandwidth>
   <LightIntensityunit="%">1.0</LightIntensity>
   <NumberOfLightPulses>2</NumberOfLightPulses>
   <FocusPositionunit="mm">6.5</FocusPosition>
</commonProperties>
```

Listing 1 Example of Common Properties

The properties of type duration shall be used for longer times (minutes, hours, days). Shorter times and time spans can be represented with various “double” properties and the corresponding unit.

For a current list of all pre-defined properties see the Table 3 Common Properties in the appendix of this document.

3.1.2 Specific Properties

Since it is not possible to define a complete list of all properties and units, the *Data Capture Specification* enables to define customized *specificProperties* and units. The example below shows how such specific properties might be implemented within the *Data Capture Schema*.

SpecificProperties can also be used to include information which is not important for later analysis. So there can be information included about device details which are helpful for troubleshooting.

```xml
<specificProperties>
  <complexProperty name="Background information">
    <integerProperty name="Number of Flashes/Time">2</integerProperty>
    <integerPropertyname="Signal">448531</integerProperty>
    <durationPropertyname="Meastime">PT19.368S</durationProperty>
    <stringProperty name="MeasInfo">De=1st Ex=Btm Em=N/A Wdw=N/A</stringProperty>
  </complexProperty>
  <complexProperty name="Used filters">
    <complexProperty name="Filter 301">
      <stringProperty name="Filter name">Photometric 340</stringProperty>
      <stringPropertyname="Filter type">Excitation</stringProperty>
      <stringProperty name="Notes">P340 CWL=340nm BW=25nm Tmin=40%</stringProperty>
      <stringProperty name="Used with">
        Photometry,TrFluorometry,FP,LANCE</stringProperty>
      <stringProperty name="Last edited by">Installation</stringProperty>
      <dateTimePropertyname="Lastedited">2013-12-09T16:26:50</dateTimeProperty>
      <booleanPropertyname="Factorypreset">true</booleanProperty>
    </complexProperty>
  </complexProperty>
  <complexPropertyname="Other">
    <integerProperty name="Reference AD Gain">1</integerProperty>
    <integerPropertyname="Referencesignal">268725</integerProperty>
    <booleanPropertyname="Continuousmeasurement">false</booleanProperty>
    <integerProperty name="Number of flashes integrated">1</integerProperty>
    <stringPropertyname="Last edited by">Administrator</stringProperty>
    <dateTimePropertyname="Lastedited">2013-07-25T18:59:31</dateTimeProperty>
    <booleanPropertyname="Factorypreset">false</booleanProperty>
  </complexProperty>
  </specificProperties>
```


Listing 2 Example of Specific Properties

4. Data Capture Core

To understand the next chapters some terms with its meaning within this document will be introduced in the following section.

4.1 Terms

This standard is about data which are related to an **Experiment**. An experiment is an action which generates data. This may be for example a readout of a microtiterplate which delivers a value for every well of the plate. It can also be a reagent transfer which delivers information about the source, the destination and the volume of the transfer.

The experiment is related to a labware **Container** which may contain one or more targets for the experiment (**Experiment Targets)**. This may be for example a microtiterplate or a rack which contains vials.

The experiment may be divided in some ***ExperimentSteps*** where every step describes a part of the experiment. For example in an automated laboratory system the first experiment step can be an initial readout of a microtiterplate, the second step may be dispensing of a reagent into the plate and the last step the final readout. These steps can also be executed within one device (e.g. a reader with onboard-pipettor). In the first case the data will be delivered from several devices, in the second case only one device will deliver data of all steps.

4.2 Overview

The core schema of the Data Capture supports the representation of:

- generated data (e.g. measurement results)
- plate layouts, such as the count of wells, arrangement of wells
- meta information of the data generation, such as measurement wavelength or sample ID; general information about the measurement method, such as start time or operator

The current version of the *Data Capture Core Schema* is available at: http://sila-standard.org/schemata/DataCapture_1.0.xsd.

Figure 3 shows the main structure of the Data Capture Core. It begins with the root element "containerData" which is split into five sub-trees describing the experimental data, the labware, the device meta data, the user meta data and the method meta data.

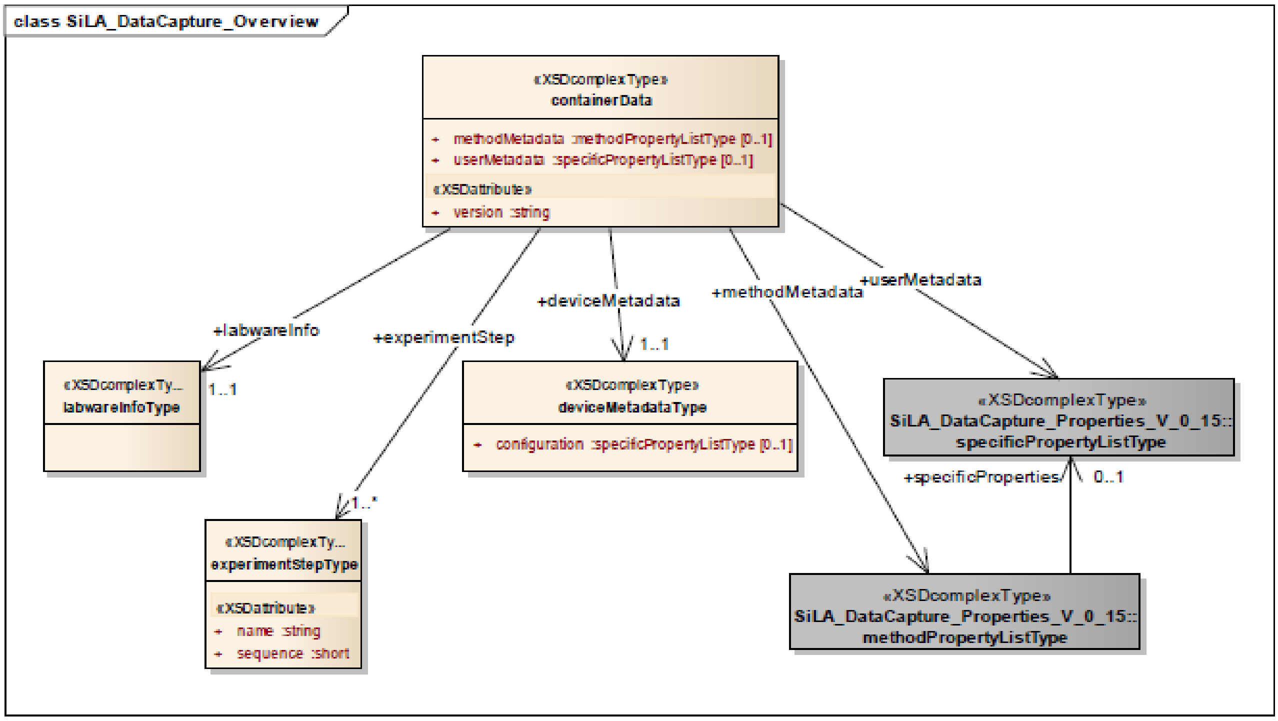

Figure 3 Data Capture - Overview

The following XML excerpt shows the mentioned sections on an example of a data capture element. Details are abbreviated by "…". The XML also starts with the root node "*containerData*", which has sub-elements for the results (one or more *experimentStep*), the labware, and meta data for device, user and method. User and method meta data are optional.

```xml
<?xml version="1.0" encoding="utf-8"?>
<containerData version="1.0"  xsi:noNamespaceSchemaLocation="SiLA_DataCapture_V_1_0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <experimentStep name="pre-incubation" sequence="1" type="Measurement">
      <dataSeries nameId="Abs 340 nm" repeat="1" description="De=1st Ex=Btm Em=N/A   Wdw=N/A"
          startDatetime="2012-10-21T16:28:28+01:00" endDatetime=
          "2022-10-21T16:35:13+01:00">
            ...
      </dataSeries>
      <dataSeries nameId="Abs @ 450 On-the_Fly(1)" repeat="1" description=""   startDatetime="2012-10-
          21T16:28:28+01:00"
          endDatetime="2002-10-21T16:35:13+01:00">
            ...
      </dataSeries>
   </experimentStep>
   <labwareInfo>
      <standardPlate labwareDefinitionSource="Other" name="AssayPlate"   labwareTypeName="XYZ 384"
        rows="16" cols="24" labwareTypeId="212"   identification="LX00345149"totalPositions="384">
            ...
      </standardPlate>
   </labwareInfo>
    <deviceMetadata>
      <software name="SiLA Reader Simulation Driver" manufacturer=   "EQUIcon
        Software GmbH Jena" version="10">
      </software>
      <identification manufacturer="PerkinElmer" serialNumber="1000087"   firmwareVersion="1.0"
        productName="Envision" productNumber="2100"/>
   </deviceMetadata>
   <userMetadata>
      <stringPropertyname="Assay-ID">457</stringProperty>
      <stringProperty name="Assay-Name">X10 inhibition</stringProperty>
   </userMetadata>
   <methodMetadata>
      <methodcommonProperty>
        <methodName>XtsTestGuy</methodName>
        <methodDescription>None.</methodDescription>
        <methodVersion>1.0</methodVersion>
      </methodcommonProperty>
      <commonProperties>
        <MethodStartTime>2012-10-21T16:28:48</MethodStartTime>
        <MethodDuration>PT5M25S</MethodDuration>
      </commonProperties>
      <specificProperties>
        <stringProperty name="Protocol ID">100050</stringProperty>
        <stringProperty name="Assay Error Information">No errors</stringProperty>
        <stringProperty name="Assay Notes"/>
      </specificProperties>
   </methodMetadata>
</containerData>
```


Listing 3 Example of containerData

The following sections will describe the sub-trees. It is important to understand the relation between data and the experiment targets, therefore a description of the so called labware will follow now.

4.3 Labware Info

The *labwareInfo* element is of complex type (*labwareInfoType*) and is meant to hold relevant information about the labware used in an experiment. In order to accommodate also other labware containers than microtiterplates (*standardLabwareType*) the *otherLabwareType* is introduced. The *labwareInfo* can comprise multiple entries of each of the two sub-types.

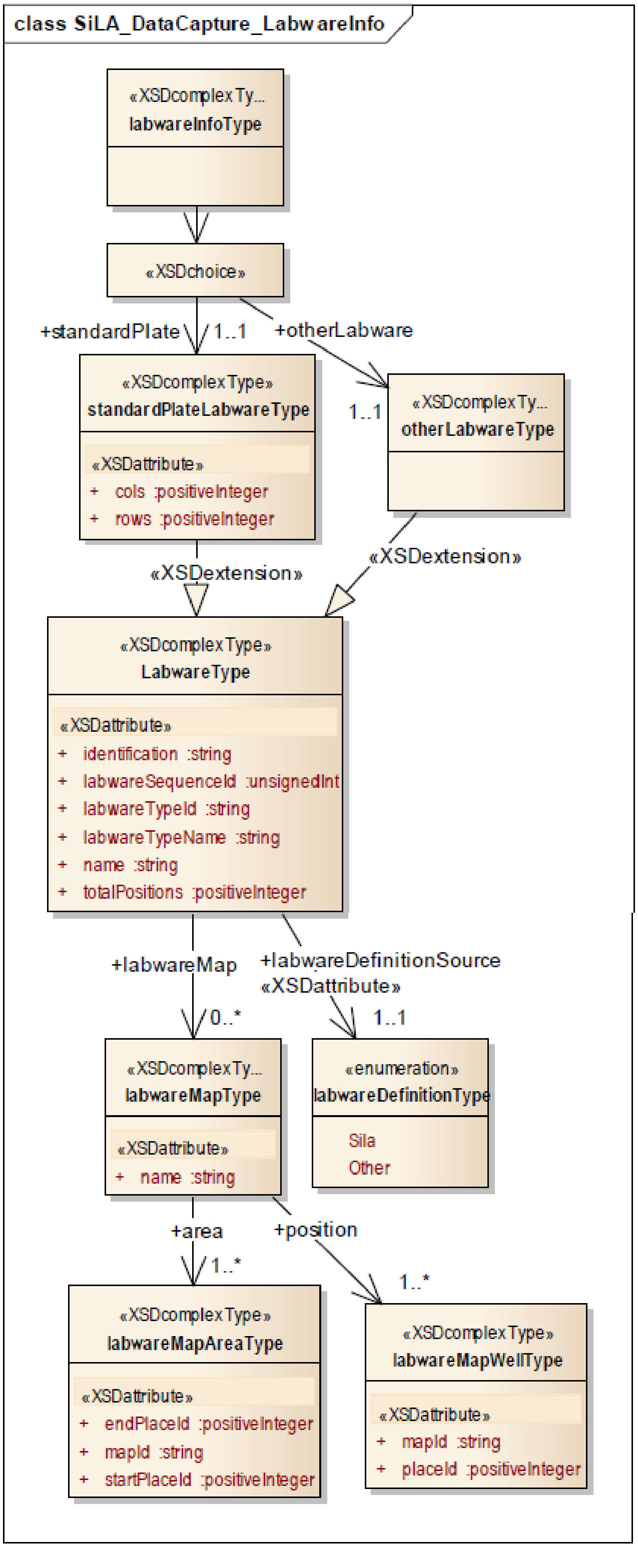

Figure 4 LabwareInfo

The Listing 4 contains an example for the labwareInfo element.

```xml
<labwareInfo>
  <standardPlate labwareDefinitionSource="Other" name="AssayPlate"  labwareTypeName="XYZ 384"
    rows="16" cols="24" labwareTypeId="212"  identification="LX00345149"totalPositions="384">
      <labwareMap name="384 assay plate well map">
      <position placeId="1" mapId="Assay_A1"/>
      <position placeId="2" mapId="Assay_A2"/>
      <position placeId="3" mapId="Assay_A3"/>
      <position placeId="4" mapId="Assay_A4"/>
         ...
      <position placeId="383" mapId="Assay_P23"/>
      <position placeId="384" mapId="Assay_P24"/>
    </labwareMap>
  </standardPlate>
</labwareInfo>
```

Listing 4 Example of LabwareInfo

For the relation between the data and the experiment target at least one plate or other labware used in the experiment has to be specified. For each plate or other labware described in the data file it is mandatory to define positions relevant for the data generation.

Standard microtiterplates use the *standardPlateLabwareType*, which is based on the more general *labwareType*, to describe the labware. The latter contains different information items like the name, the *labwareMap*, the *labwareSequenceId*, *labwareDefinitionSource* and the *identification*. The last mentioned attribute is used to report the barcode (or other identification information) of the plate.

The *labwareMap* section is used to map the sequential positions (the attribute *placeId*) to a more human readable format (the attribute *mapId*). This *mapId* is a string which must be unique in the xml document. It is realized by using the common “key/keyref” mechanism of the Xsd-Schema definition. Following the normal schema validation ensures the uniqueness.

In the following example the *mapId* “Assay_3” will be defined (line A, part of the element *labwareMap*) and in the line B (part of the data section below) the *mapId* will be used to assign a data value.

```
A: <position placeId="3" mapId="Assay_A3"/>

B: <doubleValue mapId="Assay_A3">5.652</doubleValue>
```

Listing 5 Usage of mapId

Each defined *mapId* can be used several times to assign values.

For the commonly used 96 and 384 well plates it is RECOMMENDED to compose the *mapId* from information about the purpose of the plate and the position of the well in row/column representation.

The use of the *mapId* also enables the reporting of result for multiple plates by identifying precisely the source positions on these plates.

Since there is no naming convention for microtiterplates in the 1536 format, SiLA defines that such wells MUST be identified by applying a sequential numbering schema (*placeId*)

Figure 1 shows how this numbering is organized. The first well will be identified based on the SBS standard starting from the top left [SBS]. The numbering continues from left to right and then from the top row down to the next row, and so on. Thus the definition of the *placeId* is valid for any of the commonly used rectangular grid plates, e.g. 24, 96, 384 or 1536 well plates (see also the picture below). Plates or labware arrangements in other formats can be addressed with a similar approach. There may be no position duplications however.

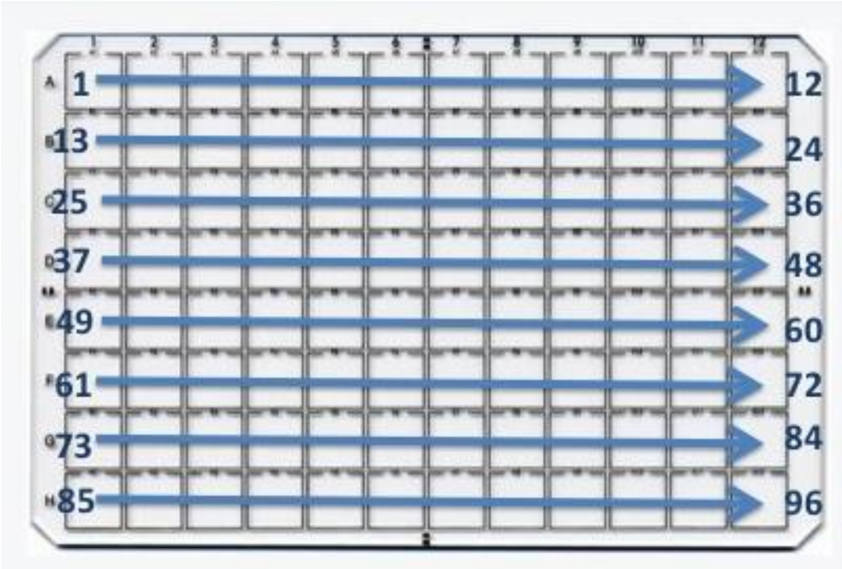

Figure 5 Schema of Numbering in Plate

4.4 Generated Data (Measurement Results)

The generated data are reported in the experiment step branches. This is a complex type (*experimentStepType*), which is meant to hold all the relevant information about the different experimental steps executed during the entire experiment.

Each *experimentStep* contains up to several *dataSeries* that contain the information about the results from a single measurement or different measurement techniques or from kinetic measurements. Such a *dataSeries* consists of values, which can be of one of the common scalar data types, including also dateTime and duration.

Figure 6 depictures the structure of the element *experimentStep*.

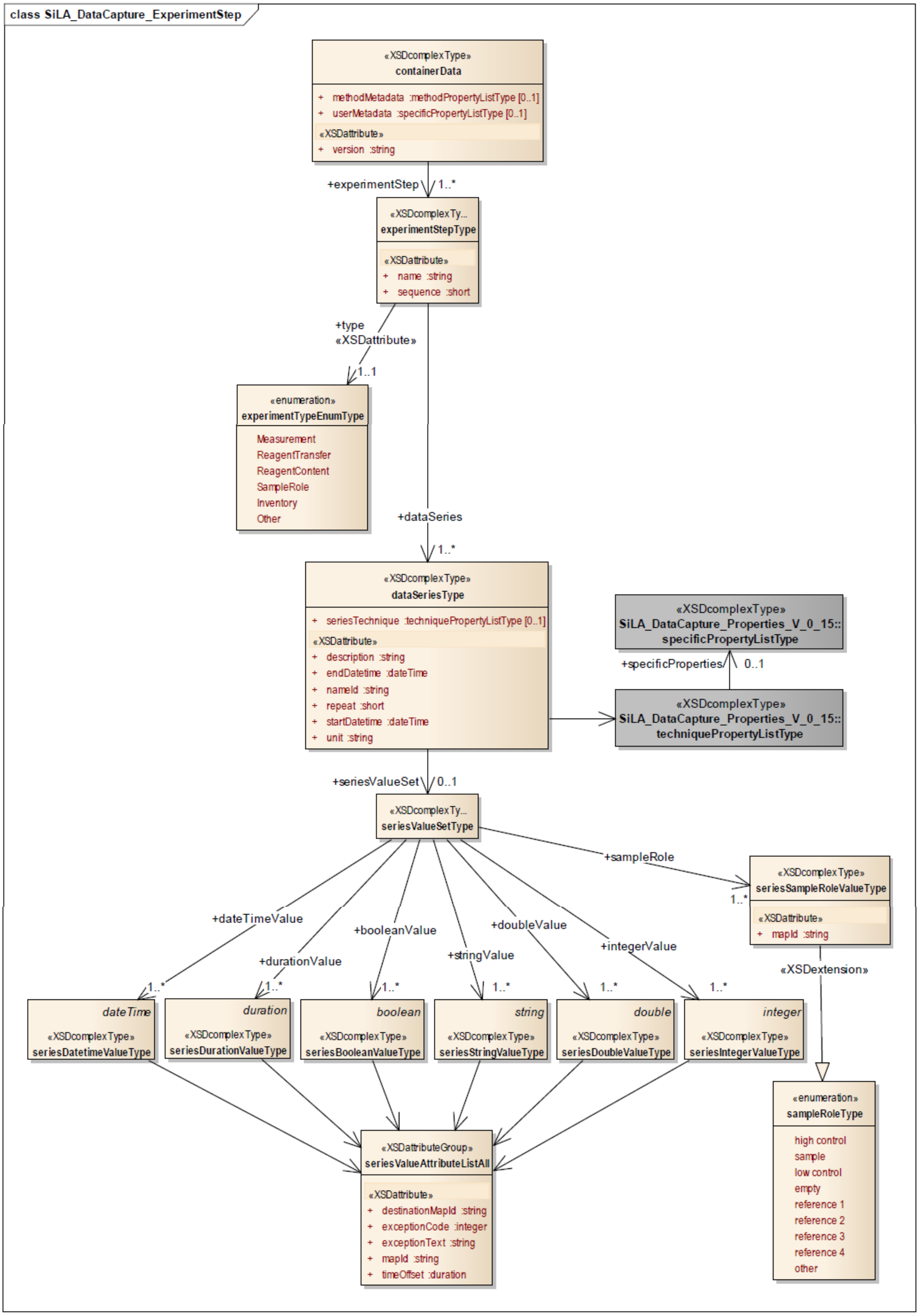

Figure 6 *experimentStep*

In the following subsections, the details for *experimentStep* and *dataSeries* are specified.

4.4.1 experimentStep

An experiment step is the structure to representing generated data (e.g. measured with a reader). To enable also representation of other processes than measurements or situations during the execution of automated laboratory procedures, the following types of *experimentSteps* are defined:

- **Measurement**: measurement data only
- **ReagentTransfer**: Information about the reagent as well as the source and destination well
- **ContainerContent**: Information about the current content of the labware
- **SampleRole**: information about the purpose of the content of the well, e.g. sample, control, reference, etc. (Allowed values are defined in the Data Capture schema)
- **InventoryContent**: Information about the current content of the labware and its location in an inventory system.
- **Other**: Other labware container related information. Specification of the meaning can be done by using the name attribute of the experimentStep or by using common and specific properties.

Based on these possibilities, it is clear that the present data capture standard is not limited to reader devices, but can also be applied to other data for example from storage systems or pipetting devices.

4.4.2 DataSeries

The data generated by the reader measurement is written into a *dataSeries*. The *dataSeries* consists of two elements, the *seriesTechnique* describes how the measurement was done and the *seriesValueSet* is listing the individual measurement values for each position in the labware container.

For the *seriesTechnique* both common and specific properties can be used.
In the *seriesValueSet* the measurement positions are identified via the *mapId* attribute (see below). The use of the *mapId* enables the representation of multiple measurements and also the listing of only selected wells.

Each value can be tagged with an exception code. This allows for well specific exception handling, such as “no liquid in well”.

The following passage shows an abbreviated *dataSeries* example. The *SeriesTechnique* contains common and specific properties. The data section in the *seriesValueSet* comprises the data values for a 384 well plate (most wells are represented by the …..).

```xml
<dataSeries nameId="Abs 340 nm" repeat="1" description="De=1st Ex=Btm Em=N/A   Wdw=N/A"
  startDatetime="2012-10-21T16:28:28+01:00"
  endDatetime="2012-10-21T16:35:13+01:00">
  <seriesTechnique>
    <commonProperties>
      <AbsorptionWavelengthunit="nm">340</AbsorptionWavelength>
      <AbsorptionBandwidthunit="nm">25</AbsorptionBandwidth>
      <LightIntensityunit="%">1.0</LightIntensity>
      <NumberOfLightPulses>2</NumberOfLightPulses>
      <FocusPositionunit="mm">6.5</FocusPosition>
    </commonProperties>
    <specificProperties>
      <complexProperty name="Background information">
        <integerProperty name="Number of Flashes/Time">2</integerProperty>
        <integerPropertyname="Signal">448531</integerProperty>
        <durationPropertyname="Meastime">PT19.368S</durationProperty>
        <stringProperty name="MeasInfo">De=1st Ex=Btm Em=N/A Wdw=N/A</stringProperty>
      </complexProperty>
      <complexProperty name="Used filters">
        <complexProperty name="Filter 301">
          <stringProperty name="Filter name">Photometric 340</stringProperty>
          <stringProperty name="Filter type">Excitation</stringProperty>
          <stringProperty name="Notes">P340 CWL=340nm BW=25nm Tmin=40%
              </stringProperty>
          <stringProperty name="Used with">Photometry,TrFluorometry,FP,LANCE
              </stringProperty>
          <stringProperty name="Last edited by">Installation</stringProperty>
          <dateTimeProperty name="Last edited">2012-12-09T16:26:50</dateTimeProperty>
          <booleanProperty name="Factory preset">true</booleanProperty>
        </complexProperty>
      </complexProperty>
      <complexProperty name="Other">
        <integerProperty name="Reference AD Gain">1</integerProperty>
        <integerProperty name="Reference signal">268725</integerProperty>
        <booleanProperty name="Continuous measurement">false</booleanProperty>
        <integerProperty name="Number of flashes integrated">1</integerProperty>
        <stringProperty name="Last edited by">Administrator</stringProperty>
        <dateTimeProperty name="Last edited">2012-07-25T18:59:31</dateTimeProperty>
        <booleanProperty name="Factory preset">false</booleanProperty>
      </complexProperty>
    </specificProperties>
  </seriesTechnique>
  <seriesValueSet>
    <doubleValue mapId="Assay_A1" timeOffset="PT01S">4.749</doubleValue>
    <doubleValue mapId="Assay_A2" exceptionCode="1"
       exceptionText="OVERFLOW">6.000</doubleValue>
    <doubleValue mapId="Assay_A3">           5.652  </doubleValue>
    <doubleValue mapId="Assay_A4">           5.652  </doubleValue>
      ...
    <doubleValue mapId="Assay_P23">          5.652  </doubleValue>
    <doubleValue mapId="Assay_P24">          5.652  </doubleValue>
  </seriesValueSet>
</dataSeries>
```

Listing 6 Example of dataSeries

4.5 Device Meta Data

To ensure the traceability of data some information about the device are contained in a separate type. Most of the defined items are “required” and therefore they must be specified.

The *deviceMetadata* is a complex type (*deviceMetadataType*), which is meant to hold relevant information about the device used in the experiment.

The inner structure of the device meta data consists of the software branch describing the software used to control the device. Further the configuration branch describes the actual configuration of the device and the identification branch to describe the instrument. The latter contains information that also can be obtained by the mandatory command *GetDeviceIdentification* (see DCDIS).

In all three branches there is the possibility to expand the contents with specific properties based on the *specificPropertyListType*.

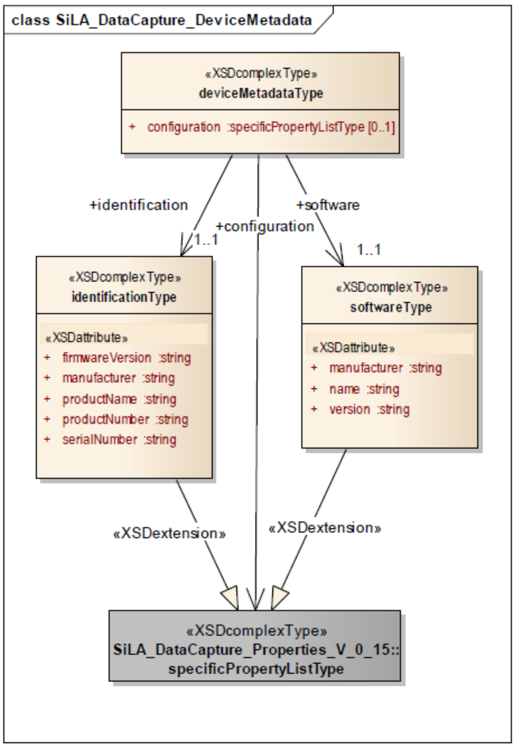

Figure 7 DeviceMetadata

The following cutout shows an example of *deviceMetadata* which contains information about the software, hardware and configuration of the measuring device used during the experiment.

```
<deviceMetadata>
  <software name="SiLA Reader Simulation Driver"  manufacturer="EQUIcon
    Software GmbH Jena" version="10">
  </software>
  <identification manufacturer="PerkinElmer" serialNumber="1000087"   firmwareVersion="1.0"
    productName="Envision" productNumber="2100"/>
</deviceMetadata>
```

Listing 7 Example of deviceMetadata

4.6 User Meta Data

The generated data can be associated to the environmental conditions of its origin by the “UserMetaData” subtree. This data can be set by the *SiLA Service Consumer* by using the common command. Possible information may be details about an assay run, an experiment Id or other information. The *SiLA Service Provider* stores this information and includes it unchanged in every container data. Therefore the data can be assigned to the generation of it. This may be important for later analysis, data mining etc. Because there are no restrictions on the *userMetadata* itself, every company can define its own data items.

The element *userMetadata* is of complex type (*specificPropertyListType*) and enables the reporting of user related data. For this kind of data no properties or attributes are mandatory.

```
<userMetadata>
  <stringPropertyname="Assay-ID">457</stringProperty>
  <stringProperty name="Assay-Name">X10 inhibition</stringProperty>
</userMetadata>
```

Listing 8 Example of userMetadata

4.7 Method Meta Data

In order to ensure the traceability of any generated data, there is information about the method, which was used to obtain the data at device level. This information can be included by the methodMetadata subtree.

The element *methodMetadata* again is of complex type (*methodPropertyListType*). When used, it contains some mandatory elements (*methodCommonProperty*) and can be extended with *commonPropertyListType* and *specificPropertyListType* elements.

The Listing 9 Example of methodMetadata shows an example of methodMetadata.

```
<methodMetadata>
  <methodcommonProperty>
    <methodName>XtsTestGuy</methodName>
    <methodDescription>None.</methodDescription>
    <methodVersion>1.0</methodVersion>
  </methodcommonProperty>
  <commonProperties>
    <MethodStartTime>2012-10-21T16:28:48</MethodStartTime>
    <MethodDuration>PT5M25S</MethodDuration>
  </commonProperties>
  <specificProperties>
    <stringProperty name="Protocol ID">100050</stringProperty>
    <stringProperty name="Assay Error Information">No errors</stringProperty>
    <stringProperty name="Assay Notes"/>
  </specificProperties>
</methodMetadata>
```

Listing 9 Example of methodMetadata

5 Rules for using Schema Elements

The hereby introduced standard has some freedom to use its elements. This section provides details about how to apply the above defined standard and to ensure each user will use this standard in a uniform manner.

Liberty consists in the following points.

- Some elements and attributes in the schema of the standard are required, however it is optional to leave some of these values empty.
- Common and specific properties may be used redundantly.
- The DCDIS enables to send the generated data for the whole container (by the *ResponseEvent*) but enables also to send the data while the data generation is running (by the *DataEvent*, e.g. for every well of a microtiterplate).
- If you have multiple phases of data generation of the same experiment targets, you can use multiple *dataSeries* within one *experimentStep* or one *dataSeries* and multiple *experimentStep*s or any combination of the above.
- If there are many different data features of an experiment target which have to be reported, it is possible to create a *dataSeries* for every position which contains the different features or creating *dataSeries* for every feature which contains the value for all positions.

The following rules MUST be followed by every application of the standard:

1. Every required item of the schema of the standard MUST NOT be empty. The correct values have to be assigned, if it is possible.

2. If there is an appropriate item in the common properties the user MUST use this property instead create a new specific property.

3. If the data generation takes only a few seconds the user MUST send the data with the *ResponseEvent*. This means that all data are included in one *containerData* structure. For example a barcode reader must deliver the code in the *ResponseEvent* and not in a *DataEvent*.

4. If data generation takes a longer time (e.g. due to complex data analysis of the content of a well of a microtiterplate) the user MAY use *DataEvent*s. This means that the data is distributed over multiple *containerData* structures.

5. If an *experimentStep* does not change the properties of the experiment target, all related data MUST be sent within this *experimentStep*. An *experimentStep* changes the properties of the experiment target, for example if there will be reagents added, it will not be changed by a readout.

6. If there are multiple data generation for the same feature of the experiment target but for increasing time (e.g. kinetic measurement) the user MUST use for every time point one *dataSeries* which contains the values for all positions.

7. If there are multiple different data features of an experiment target which have to be reported, for every feature a separate *dataSeries*, which contains the value of the feature of all positions, MUST be used.

The following examples illustrate some points.

If there is a measurement with two detection modes the user has to use two *dataSeries* within one *experimentStep*.

If the results of a kinetic measurement should be reported, where the labware container doesn't leave the instrument and the environment will be stable during the whole measurement, the user must use multiple *dataSeries* within one *experimentStep*.

If the results of a kinetic measurement should be reported, where the labware container doesn't leave the instrument and the environment will not be stable during the whole measurement (e.g. while controlled heating will be done), the user must use for every time point one *dataSeries* within multiple *experimentSteps*.

The next example illustrates a more complex scenario. Assume there is a reader with onboard-dispensing and there is one measurement, then a dispensing and then a second measurement. The *containerData* must contain three *experimentSteps*. The first contains one *dataSeries* of the first measurement. The second *experimentStep* contains one *dataSeries* with the information about the dispensing, the third *experimentStep* the *dataSeries* of the second measurement.

6 References

Reference	Name	Version	Date	Author	State
[XSD]	XML Schema Part 2: Data Types	Second Edition			
[SBS]	SBS Standard for micro titer plates				
[DCDIS]	SiLA Device Control & Data Interface Specification	1.2	2012	SiLA Consortium	released

Table 1: Document References

7 Attachments

Reference	Name	Version	Date	Author	State
[DC]	XSD Schema SiLA Data Capture	1.0	2013	SiLA Consortium	
[DCP]	XSD Schema SiLA Data Capture Properties	1.0	2013	SiLA Consortium	

Table 2: Document Attachments

8 Appendix

8.1 Data Capture Schema – Full Layout

Figure 8 Data Capture Schema - Full layoutdepicts the SiLA Data Capture Schema. Gray filled elements are references to types defined in the SiLA Data Capture Properties. The figure is a composition of figures shown earlier in this document.

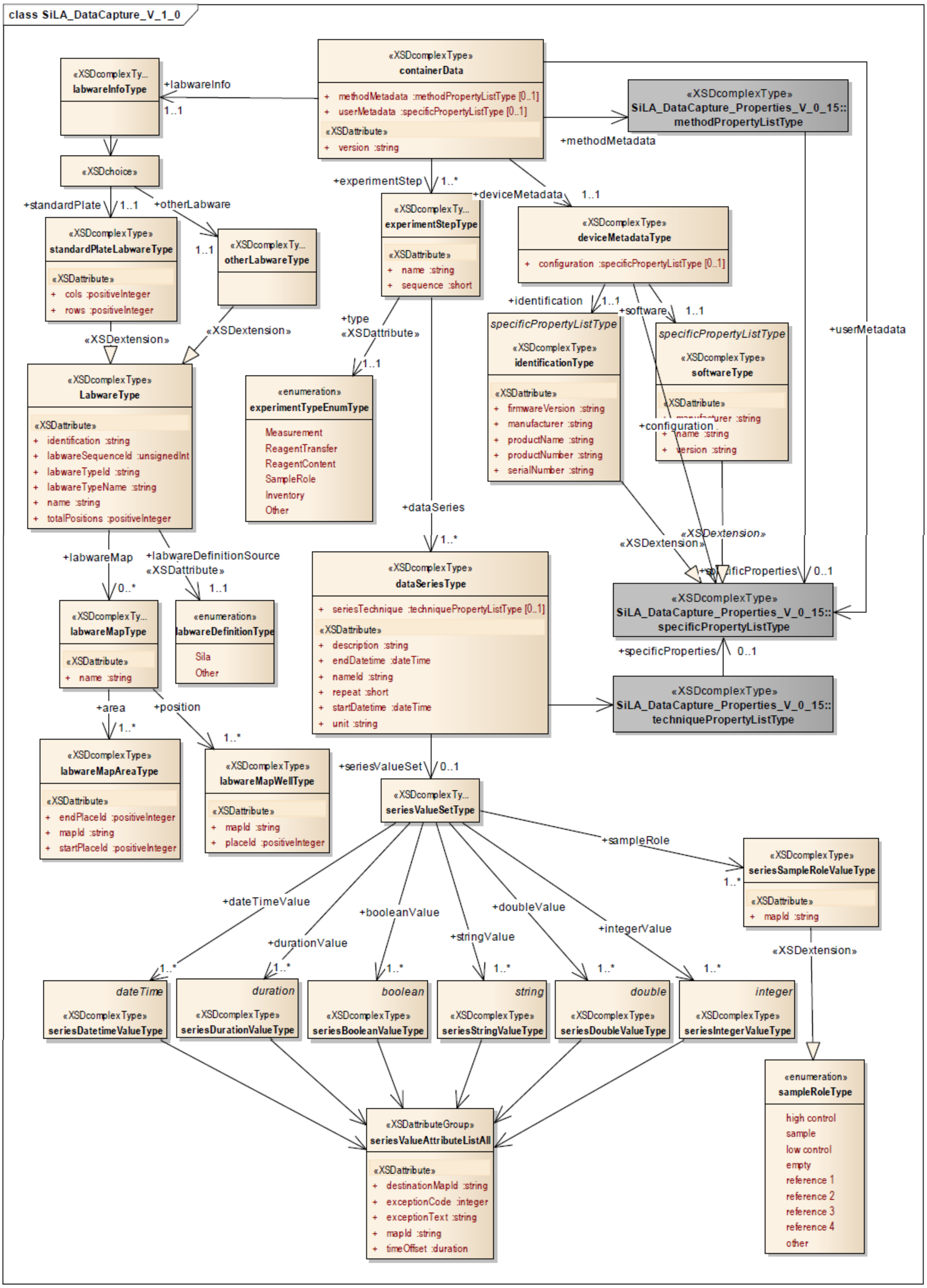

Figure 8 Data Capture Schema - Full layout

8.2 Currently defined Common Properties

The Table 3 shows the currently defined Common Properties. For more information see the SiLA Data Capture Schema definition.

String Properties	***Double Properties***	***Integer Properties***
Name	AbsorptionBandwidth	ChannelNumber
Description	AbsorptionWavelength	MaximumCounts
ID	AmbientPressure	MeasurementCycles
Type	AmbientTemperature	MinimumCounts
Mode	AngleMaximum	NumberOfLightPulses
OperatingSystem	AngleMinimum	NumberOfMeasurements
SerialNumber	AttenuationLevel	Partition
DeviceSerialNumber	Bandwidth	Repeats
Manufacturer	ChemicalShiftMaximum	WellNumber
DeviceManufacturer	ChemicalShiftMinimum	
DeviceName	CO2Concentration	***Duration Properties***
Version	Concentration	
FileUri	Delay	MethodDuration
CreatedBy	DetectorGain	MeasurementTime
LastEditedBy	DetectorSensitivity	
DataBaseID	DetectorVoltage	***DateTime Properties***
Protocol	Duration	
Parameters	EmissionBandwidth	CreationDate
InstrumentDescription	EmissionWavelength	LastEditionDate
ComputerName	ExcitationBandwidth	StartTime
OperatorName	ExcitationWavelength	MethodCreationDate
OperatorID	ExposureTime	MethodLastEditionDate
AssayName	FilterBandwidth	MethodStartTime
AssayDescription	FilterCenterWavelength	CalibrationTime
LabwareType	FocusPosition	
LabwareId	Gain	***Boolean Properties***
LabwareDescription	GateTime	
SiLALabwareDatabaseId	Humidity	Reference
SampleDescription	IntegrationTime	BackgroundCorrection
MethodName	Length	AutofluorescenceCorrection
MethodDescription	LightIntensity	DataNormalized
MethodVersion	MassMinimum	
MethodFileUri	MeasuredTemperature	
MethodID	MeasurementInterval	***Enumeration Properties***
MethodCreatedBy	PulsePower	
MethodLastEditedBy	PulseWidth	DetectionMode
ResultFileUri	Resolution	Polarization
FilterDescription	SampleConcentration	EmissionPolarization
BeamSplitterType	Sensitivity	ExcitationPolarization
BeamSplitterDescription	Temperature	AbsorptionPolarization
AttenuatorMode	TemperatureSetPoint	FilterType
MeasurementArea	TRFDelay	FilterFunction
WellName	TRFGateTime	LightSource
Notes	Voltage	DetectorType
AnalysisDescription	Volume	
AnalysisMode	Wavelength	
AnalysisProtocol	WavelengthMaximum	
AnalysisParameters	WavelengthMinimum	
CalibrationMethod	Wavenumber	
CalibrationFileUri	WavenumberMaximum	
SampleName	WavenumberMinimum	
SampleType	Weight	
Unit	XPosition	
	YPosition	
	ZPosition	

Table 3 Common Properties