canSAS1d Documentation

Release 1.1

canSAS

March 29, 2013

CONTENTS

The canSAS1d data format is used to communicate one-dimensional reduced small-angle scattering data. It is a text format and is based on XML.

Objective

One of the first aims of the **canSAS** (Collective Action for Nomadic Small-Angle Scatterers) forum of users, software developers, and facility staff was to discuss better sharing of SAS data analysis software. The canSAS (http://www.cansas.org/) identified that a significant need within the SAS community can be satisfied by a robust, self-describing, text-based, standard format to communicate reduced one-dimensional small-angle scattering data, $I(Q)$, between users of our facilities. Our goal has been to define such a format that leaves the data file instantly human-readable, editable in the simplest of editors, and importable by simple text import filters in programs that need not recognize advanced structure in the file nor require advanced programming interfaces. The file should contain both the primary data of $I(Q)$ and also any other descriptive information (metadata) about the sample, measurement, instrument, processing, or analysis steps.

The cansas1d:1.1 standard meets the objectives for a 1D standard, incorporating metadata about the measurement, parameters and results of processing or analysis steps. Even multiple measurements (related or unrelated) may be included within a single XML file.

Brief Contents

CHAPTER

ONE

TABLE OF CONTENTS

1.1 Preface

The name **canSAS** stands for *Collective Action for Nomadic Small-Angle Scatterers*. canSAS provides a forum for users, software developers and facility staff to meet and exchange ideas on all aspects of programming for X-ray and neutron small-angle scattering experiments. This includes experiment preparation and simulation through control to data-storage, reduction and analysis. The aim of the forum is to provide the best resources to the many nomadic experimenters who combine results measured at different facilities.

The aims of the canSAS meetings are to promote and simplify sharing SAS data analysis methods. Non-specialists will benefit from easily applicable methods for exchanging and merging of data from different synchrotron, laboratory and neutron facilities.

The canonical name for this format is *cansas1d:1.1*.

This work is the initiative of the canSAS 1D Data Formats Working Group, established at the canSAS-V workshop, NIST, Gaithersburg, Maryland, USA from October 29th to 31st 2007, which produced revision 1.0. A further workshop was held July 28th to 31st, 2012 at Uppsala University, Uppsala, Sweden which produced revision 1.1. The format derives many of its foundations from previous works such as the SASXML format, a joint collaboration between ISIS and ILL.

The home page of the canSAS 1D Data Formats Working Group describes the members, timelines, and current status. There is a discussion page for some matters that preceded this revision.

Home: http://www.cansas.org/wgwiki/index.php/1D_Data_Formats_Working_Group

Discussion: http://www.cansas.org/wgwiki/index.php/Talk:1D_Data_Formats_Working_Group

---

Disclaimer

This description is meant to inform the community how to arrange information within the structure of the XML files and to define the spelling of the terms to be used. However, should the information in this document and the cansas1d:1.1 SAS XML Schema differ, (http://www.cansas.org/trac/browser/1dwg/trunk/cansas1d.xsd) the XML Schema will be deemed to have the most correct description of the standard.

1.2 Specification

The canSAS 1-D standard for reduced 1-D SAS data is implemented using XML files. A single file can contain SAS data from a single experiment or multiple experiments. All types of relevant data ($I(Q)$, metadata) are described for each experiment. More details are provided below.

This is the definitive specification of *cansas1d:1.1*, the canSAS standard format for storing small-angle scattering data in XML files. The standard is defined using the rules of XML Schema (http://www.w3.org/XML/Schema).

Note that the cansas1d:1.1 XML data files must adhere to the XML rules which includes being well-formed (including the use of closing tags). [1] Files that can be validated against *XML Schema* (http://www.cansas.org/trac/browser/1dwg/trunk/cansas1d.xsd) are deemed to be valid cansas1d:1.1 data files.

In this document, curly braces, *{}*, are used to indicate text that is supplied by the user. Such as, an attribute may be written

name={text}

and this means that the user would replace *{text}* with text that gives, in this example, a name such as *final detector*. Thus resulting in:

```
name="final detector"
```

which is a well-formed XML attribute.

Another example is an instance of the *{any}* element. Suppose one had analysis data, then *{any}* would be replaced with *analysis* and the element might look like this:

```
<analysis>
   ... analysis content goes here ...
</analysis>
```

Contents

1.2.1 Overview

One of the first aims of the **canSAS** (Collective Action for Nomadic Small-Angle Scatterers) forum of users, software developers, and facility staff was to discuss better sharing of SAS data analysis software. The canSAS forum (http://www.cansas.org/canSAS) identified that a significant need within the SAS community can be satisfied by a robust, self-describing, text-based, standard format to communicate reduced one-dimensional small-angle scattering data, $I(Q)$, between users of our facilities. Our goal has been to define such a format that leaves the data file instantly human-readable, editable in the simplest of editors, and importable by simple text import filters in programs that need not recognise advanced structure in the file nor require advanced programming interfaces. The file should contain both the primary data of $I(Q)$ and also any other descriptive information (metadata) about the sample, measurement, instrument, processing, or analysis steps.

Objective

The cansas1d:1.1 standard meets the objectives for a 1D standard, incorporating metadata about the measurement, parameters and results of processing or analysis steps. Even multiple measurements (related or unrelated) may be included within a single XML file.

[1] For example, see http://www.w3schools.com/xmL/xml_syntax.asp for an explanation of the XML syntax.

General Layout of the XML Data

The canSAS 1-D standard for reduced 1-D SAS data is implemented using XML files. A single file can contain SAS data from a single experiment or multiple experiments. All types of relevant data ($I(Q)$, metadata) are described for each experiment. More details are provided below.

The basic elements of the cansas1d:1.1 standard are shown in the following table. After an XML header, the root element of the file is *SASroot* which contains one or more *SASentry* elements, each of which describes a single experiment (data set, time-slice, step in a series, new sample, etc.). Details of the *SASentry* element are also shown in the next figure. See the section *Elements of the canSAS XML standard* for examples of cansas1d:1.1 XML data files. Examples, Case Studies, and other background information are below. More discussion can be found on the canSAS 1D Data Formats Working Group page (http://www.cansas.org/wgwiki/index.php/1D_Data_Formats_Working_Group) and its discussion page. (http://www.cansas.org/wgwiki/index.php/Talk:1D_Data_Formats_Working_Group)

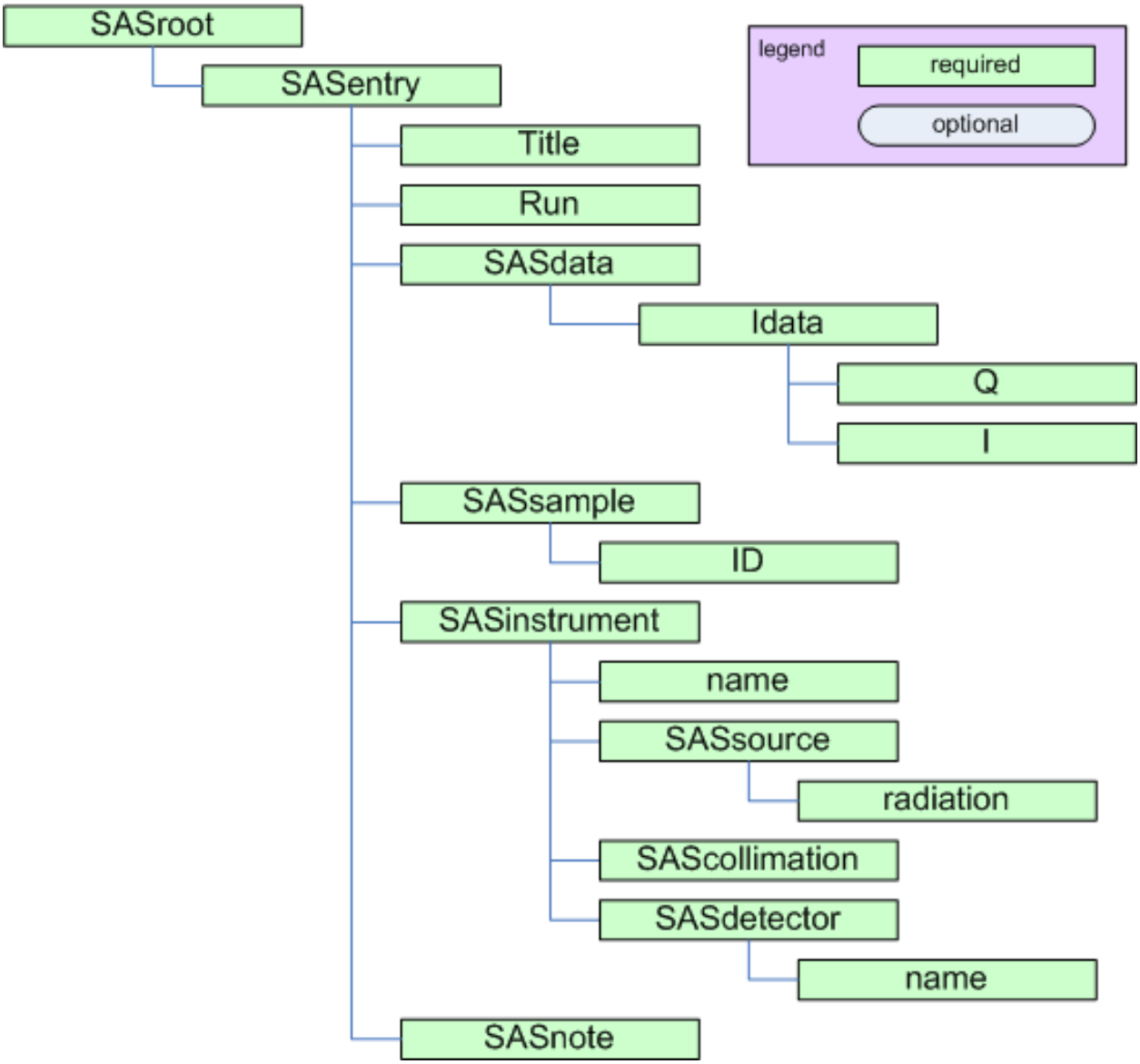


Figure 1.1: block diagram of minimum elements required for *cansas1d:1.1* standard

SASroot the root element of the file (after the XML header)

SASentry describes a single experiment (data set, time-slice, step in a series, new sample, etc.)

Required header for cansas1d:1.1 XML files

```
<?xml version="1.0"?>
<SASroot version="1.1"
        xmlns="cansas1d:1.1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="cansas1d:1.1
            http://www.cansas.org/svn/1dwg/trunk/cansas1d.xsd"
        >
```

Basic elements of the canSAS 1-D standard

Element	Description
XML Header	descriptive info required at the start of every XML file
SASroot	root element of XML file
SASentry	data set, time-slice, step in a series, new sample, etc.
Title	for this particular *SASentry*
Run	run number or ID number of experiment
{any}	any XML element can be used at this point
SASdata	this is where the reduced 1-D SAS data is stored
Idata	a single data point of $I(Q)$ (and related items) in the dataset
SAStransmission_spectrum	any transmission spectra may be stored here
Tdata element	a single data point in the transmission spectrum
{any}	any XML element can be used at this point
SASsample	description of the sample
SASinstrument	description of the instrument
SASsource	description of the source
SAScollimation	description of the collimation
SASdetector	description of the detector
SASprocess	description of each processing or analysis step
SASnote	anything at all

Rules

1. **A cansas1d:1.1 XML data files will adhere to the standard if it can** successfully *validate* against the established XML Schema.

   http://www.cansas.org/trac/browser/1dwg/trunk/cansas1d.xsd

2. $Q = (4\pi/\lambda)\sin(\theta)$ where λ is the wavelength of the radiation, and 2θ is the angle through which the detected radiation has been scattered.

3. **units to be given in standard SI abbreviations (eg, m, cm, mm, nm, K)** with the following exceptions:

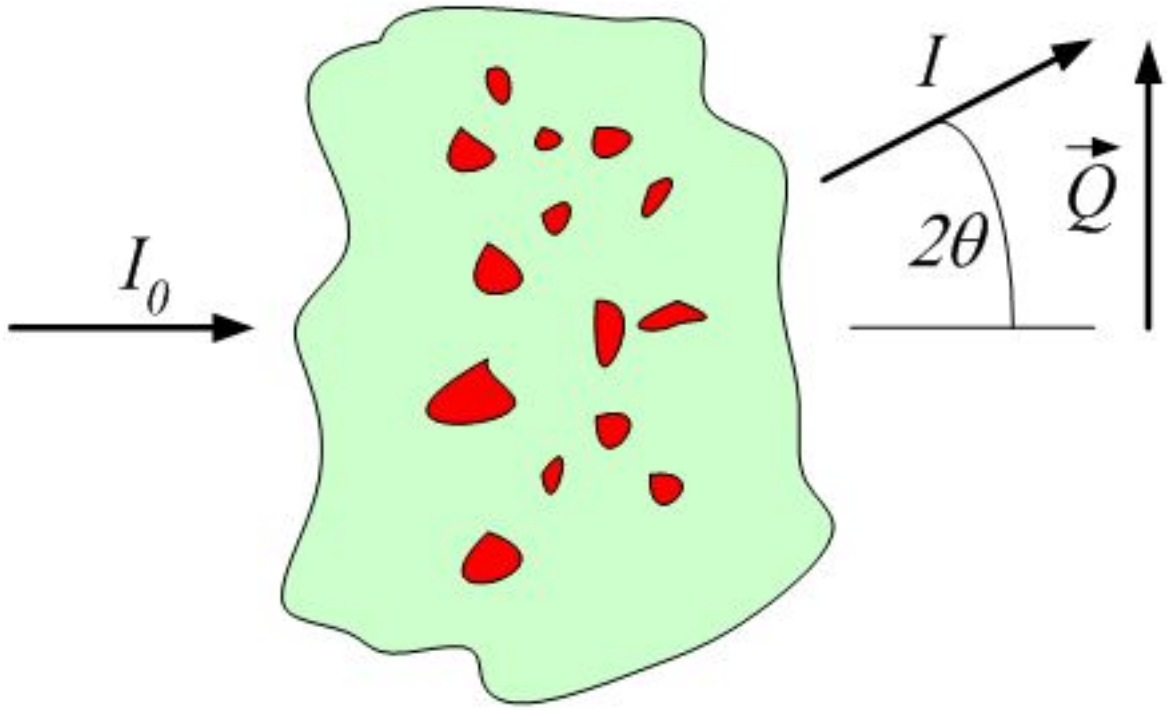


Figure 1.2: definition of Q geometry for small-angle scattering

use this	to mean this
um	micrometres
C	Celsius
A	Angstrom
percent	%
fraction	fraction
a.u.	arbitrary units
none	no units are relevant (such as dimensionless)

4. **where reciprocal units need to be quoted, the format shall be "1/abbreviation",** such as `1/A`
5. use `^` to indicate an exponent (rather than `**`), such as `m^2`
6. **when raised to a power, use similar to `A^3` or `1/m^4`** (and not `A3` or `A**3` or `m-4`)
7. **coordinate axes:** (See the sections titled *Definition of the coordinate axes* and *Compatibility of Geometry Definitions*.)
   (a) z is along the trajectory of the radiation (positive value in the direction towards the detector)
   (b) x is orthogonal to z in the horizontal plane (positive values increase to the right when viewed towards the incoming radiation)
   (c) y is orthogonal to z and x in the vertical plane (positive values increase upwards)
8. **orientation (angles) describes single-axis rotations (rotations about** multiple axes require more information):
   (a) roll is about z
   (b) pitch is about x
   (c) yaw is about y

9. Binary data is not supported

Definition of the coordinate axes

The definitions of the coordinate axes for translation and orientation geometry are described by the following two figures.

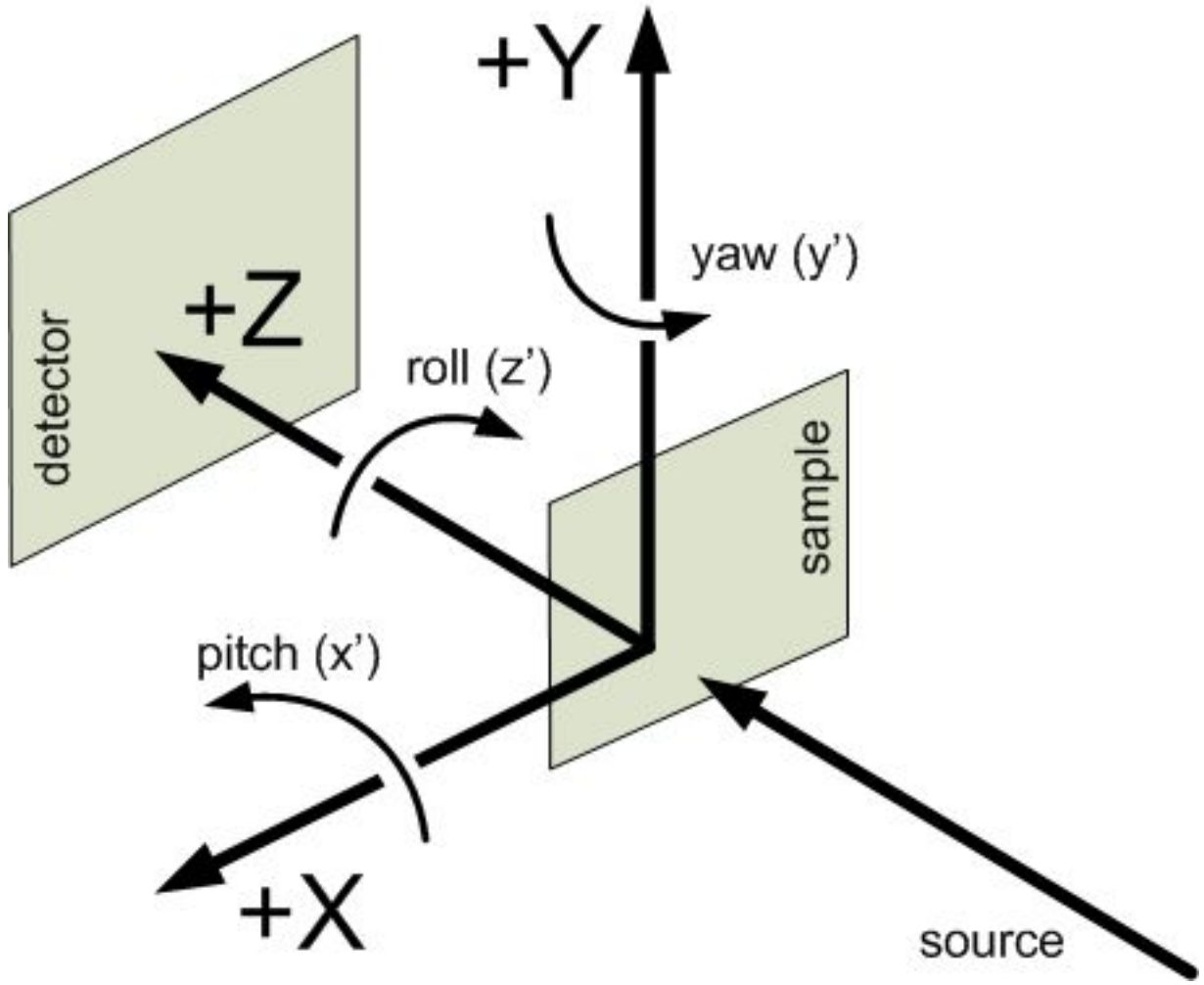


Figure 1.3: Coordinate axes as viewed from the source.

Converting data into the XML format: *XmlWriter*

The canSAS/xmlWriter (http://www.cansas.org/formats/tools/xmlWriter/) is a WWW form to translate three-column ASCII text data into the cansas1d:1.1 XML format. This form will help you in creating an XML file with all the required elements in the correct places. The form requests the SAS data of *Q*, *I*, and *Idev* (defined elsewhere on this page) and some basic metadata (title, run, sample info, ...).

Press the *Submit* button and you will receive a nicely formatted WWW page with the SAS data. If you then choose *View page source* (from one of your browser menus), you will see the raw XML of the cansas1d:1.1 XML format and you can copy/paste this into an XML file.

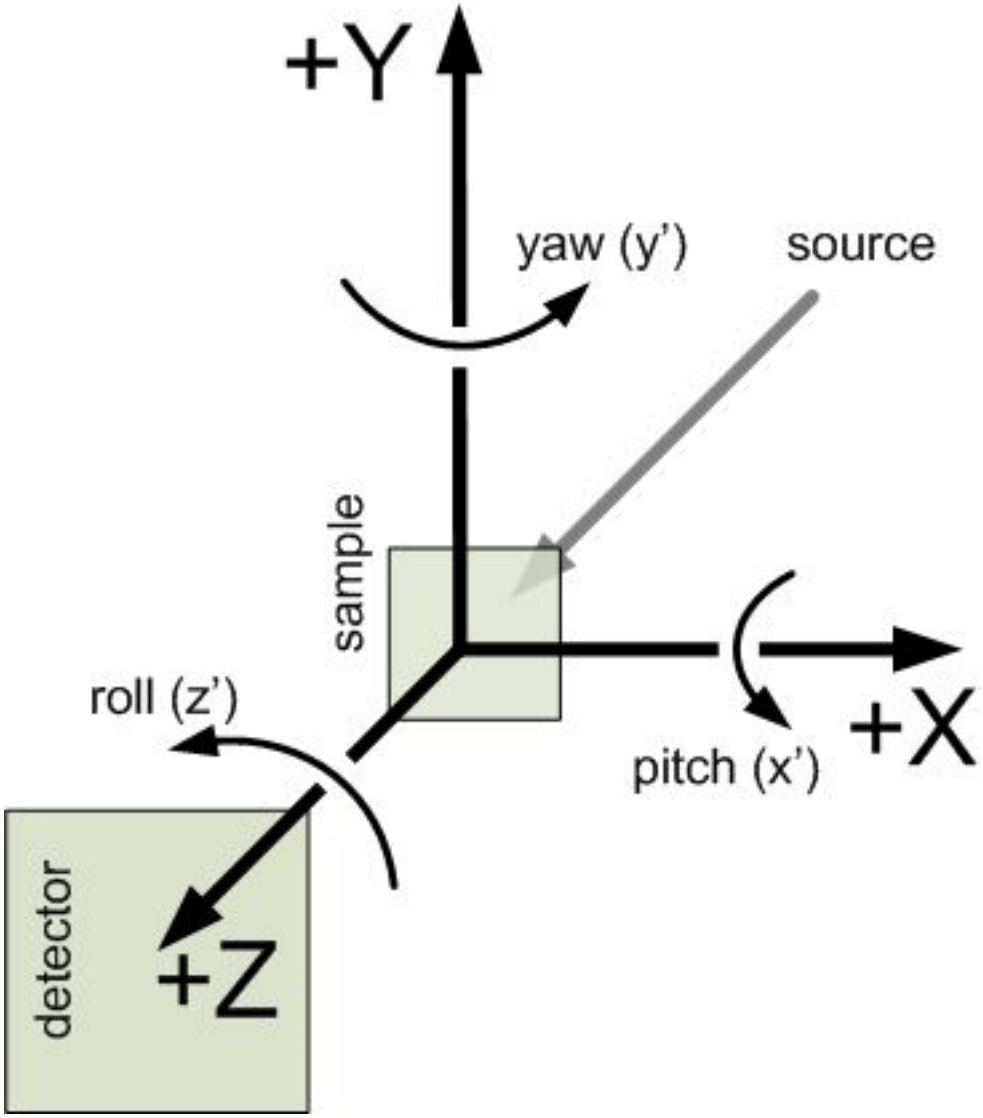


Figure 1.4: Coordinate axes as viewed from the detector.

The SAS data that you paste into the form box is likely to be copied directly from a 3-column ASCII file from a text editor. Line breaks are OK, they will be treated as white-space as will tabs and commas. Do not be concerned that the data looks awful in the form entry box, just check the result to see that it comes out OK.

Documentation and Definitions

XML Schema

The *cansas1d.xsd* XML Schema (http://www.cansas.org/trac/browser/1dwg/trunk/cansas1d.xsd) defines the rules for the XML file format and is used to validate any XML file for adherence to the format.

TRAC (view source code highlighted by bug tracking system)

http://www.cansas.org/trac/browser/1dwg/trunk/cansas1d.xsd

SVN (view raw source code from version control system)

http://www.cansas.org/svn/1dwg/trunk/cansas1d.xsd

XML stylesheets

An XML stylesheet, or *XSLT* (http://www.w3schools.com/xsl/), can be used to extract metadata or to convert into another file format. The default canSAS stylesheet *cansasxml-html.xsl* (http://www.cansas.org/svn/1dwg/trunk/cansasxml-html.xsl) should be copied into each folder with canSAS XML data file(s). It can be used to display the data in a supporting WWW browser (such as Firefox or Internet Explorer) or to import into Microsoft Excel (with the added XML support in Excel).

By default, MS Windows binds *.xml* files to start Internet Explorer. Double-clicking on a canSAS XML data file with the *cansasxml-html.xsl* (see above tip) stylesheet in the same directory will produce a WWW page with the SAS data and selected metadata.

Suggestions for support software that write cansas1d:1.1 XML data files

Some common best practices have been identified in the list below.

- be sure to update to the latest SVN repository revision:

  ```
  svn update
  ```

- check the output directory to see if it contains the default XSLT file
- **copy the latest XSLT file to the output directory if either:**
  - the output directory contains an older revision
  - the output directory does not have the default XSLT file
- **The most recent XSLT file can be identified by examining the file** for the *$ Revision: $* string, such as in the next example.

  ```
  # $Revision: 313 $
  ```

Examples and Case Studies

Basic example

http://www.cansas.org/trac/browser/1dwg/trunk/cansas1d.xml

Note that, for clarity, only one row of data is shown. This is probably a very good example to use as a starting point for creating XML files with a text editor.

Bimodal test data

http://www.cansas.org/trac/browser/1dwg/trunk/bimodal-test1.xml

Simulated SAS data (with added noise) calculated from model bimodal size distribution to test size distribution analysis routines.

Glassy Carbon Round Robin

http://www.cansas.org/wgwiki/index.php/Glassy_Carbon_Round_Robin

Samples of a commercial glassy carbon measured at several facilities worldwide.

dry chick collagen SAXS see *Case Study: Dry Chick Collagen* section

SAXS data from *dry chick collagen* illustrates the minimum information necessary to meet the requirements of the standard format.

AF1410 steel SANS see *Case Study: AF1410 Steel* section

SANS data from *AF1410 steel* using magnetic contrast variation (with multiple samples and multiple data sets for each sample), the files can be viewed from the TRAC site (no description yet).

http://www.cansas.org/trac/browser/1dwg/trunk/examples/af1410/

:Test all the cansas1d:1.1 rules:

http://www.cansas.org/trac/browser/1dwg/trunk/cansas1d-template.xml

The *cansas1d-template.xml* data file is used to test all the rules in the XML Schema. This is probably not a very good example to use as a starting point for creating XML files with a text editor since it tests many of the special-case rules.

XML layout for multiple experiments

Each experiment is described with a single *SASentry* element. The fragment below shows how multiple experiments with multiple data sets can be included in a single XML file. This illustrates using more than one *SASentry* and more than one *SASdata* element.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="cansasxml-html.xsl" ?>
<SASroot version="1.1"
        xmlns="urn:cansas1d:1.1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:cansas1d:1.1
            http://www.cansas.org/formats/1.1/cansas1d.xsd"
      >
     <!--
          Note:
          This file is not a valid cansas1d/1.1 data file.
          It is an example to show how to structure multiple data sets.
     -->
      <SASentry name="071121.dat#S22">
            <!-- contents of the first experiment in the file go here -->
      </SASentry>
      <SASentry name="example temperature series">
           <!-- example with two SAS data sets related to the same sample -->
           <Title>title of this series</Title>
           <Run name="run1">42-001</Run>
```

```
            <Run name="run2">42-002</Run>
            <SASdata name="run1">
                 <!-- data from 42-001 run comes here -->
            </SASdata>
            <SASdata name="run2">
                 <!-- data from 42-002 run comes here -->
            </SASdata>
            <!-- other elements come here for this entry -->
       </SASentry>
       <SASentry name="other sample">
            <!-- any number of additional experiments can be included, as desired -->
            <!-- SASentry elements in the same XML file do not have to be related -->
       </SASentry>
</SASroot>
```

Full examples of canSAS XML files with multiple experiments include:

multiple data sets ISIS LOQ SANS instrument:

http://www.cansas.org/trac/browser/1dwg/trunk/W1W2.XML

multiple samples, multiple data sets

AF1410 steel SANS contrast variation study from NIST:

http://www.cansas.org/trac/browser/1dwg/trunk/examples/af1410/cs_af1410.xml

SANS study using magnetic contrast variation (with multiple samples and multiple data sets for each sample), the files can be viewed from the TRAC site (no description yet).

Foreign Elements

To allow for inclusion of elements that are not defined by the *cansas1d.xsd* XML Schema, XML **foreign elements** are permitted at select locations in the cansas1d:1.1 format. Please refer to the *XML Help* section for more help with XML foreign elements.

Note: Need to make another example. This example was based on v1.0. With v1.1, there is no need for the foreign namespace in this example.

There is an example that demonstrates the use of a foreign namespace: http://www.cansas.org/trac/browser/1dwg/data/Glassy%20Carbon/ISIS/GLASSYC_C4G8G9_withTL.xml

This example uses a foreign namespace to record the transmission spectrum related to the acquisition of the SANS data at a time-of-flight facility. Look near line 153 for this element:

```
<transmission_spectrum xmlns="urn:transmission:spectrum">
```

The foreign namespace given (`urn:transmission:spectrum`) becomes the default namespace for just the *transmission_spectrum* element.*

Also refer to canSAS TRAC ticket #47 (http://www.cansas.org/trac/changeset/47) for an example of arranging the content in *SASprocessnote* to avoid the use of foreign namespace elements.

Support tools for Visualization & Analysis software

Support for importing cansas1d:1.1 files exists for these languages and environments:

FORTRAN See the *FORTRAN* section.

IgorPro See the *IgorPro* section.

Java JAXB See the *Java JAXB* section.

Microsoft Excel Support for Microsoft Excel is provided through the default canSAS stylesheet, *cansasxml-html.xsl* (http://www.cansas.org/svn/1dwg/trunk/cansasxml-html.xsl). The ISIS **LOQ** instrument (http://www.isis.stfc.ac.uk/instruments/loq/loq2470.html) has provided an excellent description of how to import data from the cansas1d:1.1 format into Excel. Also note that the old WWW site (http://www.isis.rl.ac.uk/archive/LargeScale/LOQ/loq.htm) may still be available.

PHP See the *PHP* section.

The *canSAS/xmlWriter* (http://www.cansas.org/xmlWriter/) is implemented in PHP (http://www.php.net) and writes a cansas1d:1.1 data file given three-column ASCII data as input. The code uses *DomDocument* (http://www.php.net/DomDocument) to build the XML file. Look for the line beginning with:

```
function prepare_cansasxml($post)
```

Another example of *DomDocument* is in the `function surveillance($post)` where logging information is inserted into an XML file.

PHP source: http://www.cansas.org/trac/browser/1dwg/trunk/php/xmlWriter/index.php

Python See the *Python* section.

XSLT *XSLT* (useful in a web browser) is described later in the *Example XML Stylesheets* section.

Software repositories (for cansas1d:1.1 standard)

TRAC (bug reporting) http://www.cansas.org/trac/browser/1dwg/tags/v1.1

SVN (*subversion* revision control system) http://www.cansas.org/svn/1dwg/tags/v1.1

1.2.2 Elements of the canSAS XML standard

There are various elements (tag names) in the cansas1d:1.1 standard. Each of these is described below.

Name XML tag to be used for this element of the standard.

Type A *Type* may be either of:

header Describes the required XML header lines. Without questions, use the header in the section titled *XML header*.

container A *container* element has subelements but no text of its own. These are similar to the NeXus NXDL `group` type.

floating-point number Elements of type *floating-point number* are obvious. In most cases, a `unit` attribute is required. This will be noted.

string Elements of type *string* are any valid string (non-whitespace) sequence.

Occurence The number of times a particular element may appear is described in the *occurence* column. A value of *[0..1]* indicates the element is optional but may appear one time. A value of *[0..inf]* indicates the element is optional but may appear an infinite number of times (also known as unbounded).

Attributes *Attributes* list the required or optional attributes of this element. Note that attributes must adhere to the well-formed [2] XML guidelines:

[2] well-formed XML: http://www.w3schools.com/xmL/xml_syntax.asp

```
attributename="value"
```

where either single or double quotes surround the value. All attributes must have a value. Attributes may be given in any order.

XML header

parent: None. This is the start of the XML file.

```
1 <?xml version="1.0"?>
2 <?xml-stylesheet type="text/xsl" href="cansasxml-html.xsl">
3 <SASroot version="1.1"
4                 xmlns="urn:cansas1d:1.1"
5                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6                 xsi:schemaLocation="urn:cansas1d:1.1
7                                     http://www.cansas.org/formats/1.1/cansas1d.xsd">
```

line	Description
1	declaration that this is an XML file, Some XML files have an additional attribute that is not part of the canSAS standard: `encoding='UTF-8'` that is not included since not all canSAS participants are comfortable supporting UTF-8 (http://en.wikipedia.org/wiki/UTF-8) yet.
2	Optional XML Stylesheet declaration to allow this XML file to display nicely in a (XSLT-compliant) WWW browser such as firefox. The XSLT file named as the argument of the *href* attribute (in this case: *cansasxml-html.xsl*) **must be in the same directory** as this XML file. (This requirement is part of the XSLT specification.) You can substitute a different XSLT file name to achieve a different formatted result. See http://www.w3schools.org/xsl for more help.
3	*SASroot* is the root element of the XML file. The *version* attribute is the version number of the canSAS standard to which this file has been written. The order in which the attributes of any XML element appear is not important. An attribute may not be given twice in an element if the file conforms to the XML standard. Indentation is optional and is ignored by the XML support library.
4	Attribute declaring the default XML namespace.
5	Attribute defining *xsi* as the prefix to identify content tags in the *XMLSchema-instance* namespace.
6 & 7	Location of the XML Schema that defines the allowed tags for this XML file. This attribute has two strings within the quotations, separated by white space (a newline is acceptable). The first string is the XML namespace, repeated from above. The second string is a suggested name of the XML Schema file. Some XML support libraries will follow the URL shown here to retrieve the XML Schema from the canSAS server. Since this behavior is not guaranteed by the XML standard, don't count on it.

SASroot

parent: *XML header*

Name	Type	Occurrence	Description	Attributes
SASentry	container	[1..inf]	A single SAS scan is reported in a *SASentry*. Include as many *SASentry* elements as desired. They may contain related or unrelated data. name is an optional attribute to provide a string for this *SASentry*. (Use of this string is not defined by this standard.)	`name="short-name"`

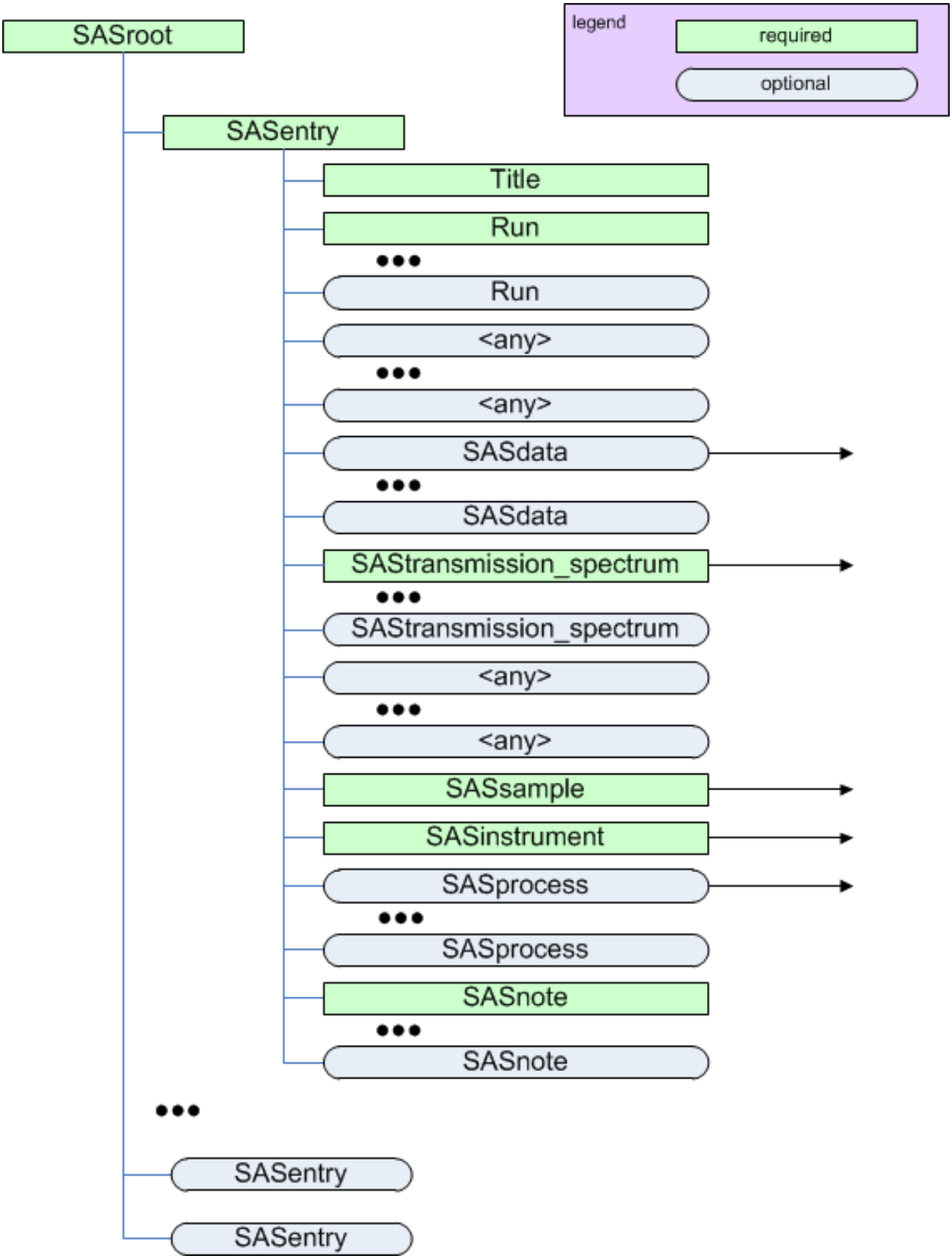


Figure 1.5: The *SASroot* element

SASentry

parent: *SASroot*

Refer to the figure in *SASroot*.

Name	Type	Occurrence	Description	Attributes
Title	string	[1..1]	Title of this *SASentry*.	*name={short-Run-identifier}*
Run	string	[1..inf]	Run identification for this *SASentry*. For many facilities, this is an integer. Use multiple instances of *Run* as needed. Note: How to correlate this with *SASdata* and *SASinstrument* configurations has not yet been defined. *name* is an optional string attribute to identify this particular *Run*. Could use this to associate (correlate) multiple *SASdata* elements with *Run* elements. (Give them the same *{short-Run-identifier}*.)	
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d:1.1 standard can be placed at this point.	*xmlns:{foreign-prefix}={foreign-namespace}*
SASdata	container	[1..inf]	Reduced 1-D SAS data for this *SASentry*. Use multiple *SASdata* elements to represent multiple frames. Use this to associate (correlate) multiple *SASdata* elements with *Run* elements. (Give them the same name.)	*name={short-Run-identifier}*
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d:1.1 standard can be placed at this point.	*xmlns:{foreign-prefix}={foreign-namespace}*
SASsample	container	[1..1]	Description of the sample.	*name={short-SASsample-identifier}*
SASinstrument	container	[1..1]	Description of the instrument.	
SASprocess	container	[0..inf]	Description of a processing or analysis step.	*name={short-SASprocess-identifier}*
SASnote	container	[1..inf]	Free form description of anything not covered by other elements.	*name={short-SASnote-identifier}*

SASdata

parent: *SASentry*

Also, see the *drawing of the Q geometry*.

Name	Type	Occurrence	Description	Attributes
Idata	container	[1..inf]	Each *Idata* describes a single SAS data point containing $I(Q)$ and other terms.	

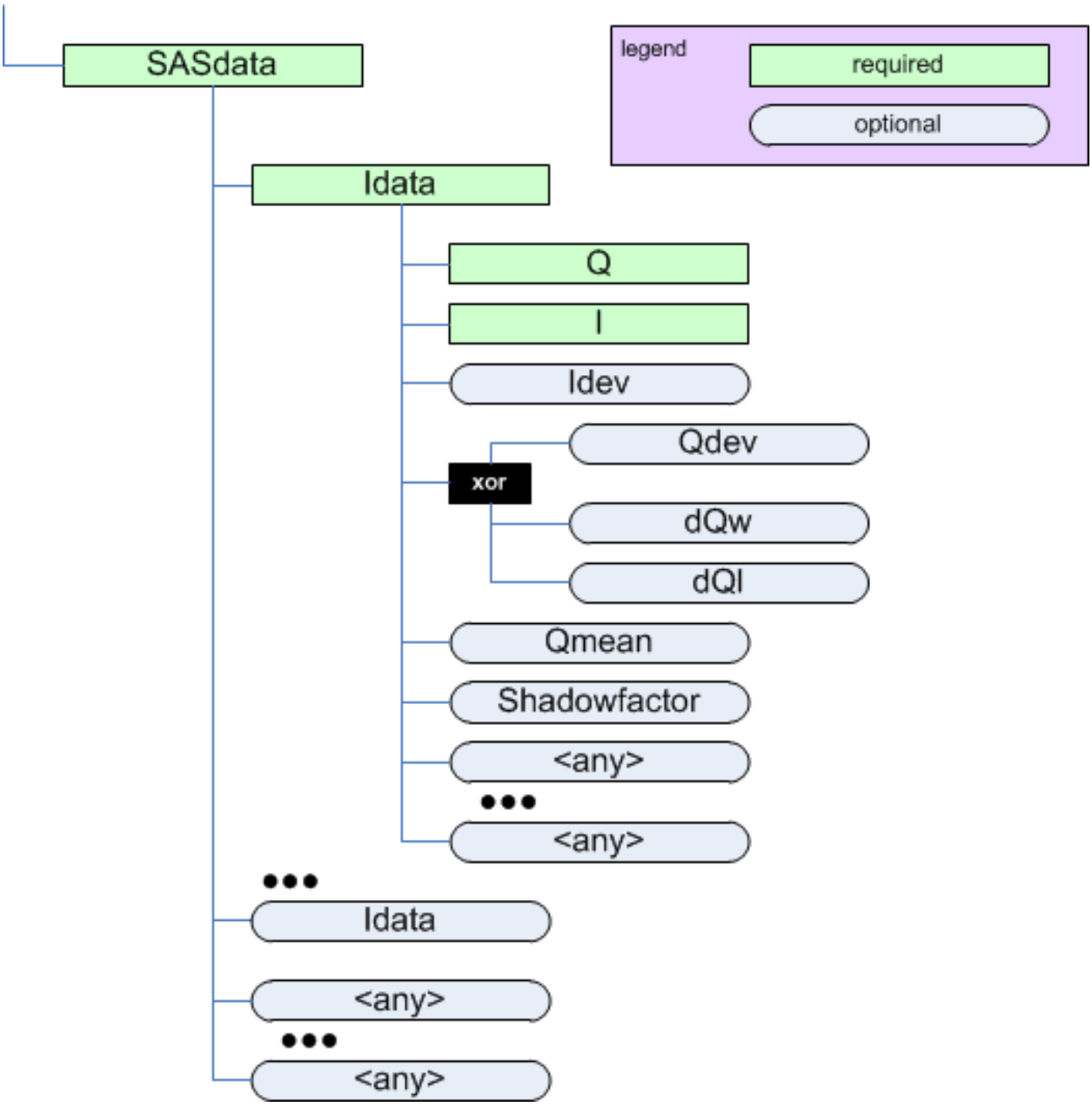


Figure 1.6: The *SASdata* element

Idata

parent: *SASdata*

Name	Type	Occurrence	Description	Attributes
Q	float	[1..1]	$Q = (4\pi/\lambda)\sin(\theta)$ as defined in the *Rules* section. Either *1/A* or *1/nm* are typical units.	*unit={unit}* [3]
I	float	[1..1]	Intensity of the detected radiation. as defined in the *Rules* section. See the section titled *The Intensity Problem* for ways to describe the SAS intensity. One possibility might be *1/cm* for absolute units when the intensity describes a **differential cross-section per unit volume per unit solid angle**.	*unit={unit}* [1] [4]
Idev	float	[0..1]	Estimated uncertainty (usually standard deviation) of *I*. [5] It is unexpected for *I* and *Idev* to have different units.	*unit={unit}* [1]
Qdev	float	[0..1]	Estimated uncertainty (usually standard deviation) of *Q*. [3] It is unexpected for *Q* and *Qdev* to have different units.	*unit={unit}* [1] [6]
dQw	float	[0..1]	*Q* resolution along the axis of scanning (the high-resolution *slit width* direction). Useful for defining resolution data from slit-smearing instruments such as Bonse-Hart geometry. [3] It is unexpected for *Q* and *dQw* to have different units.	*unit={unit}* [1] [4]
dQl	float	[0..1]	*Q* resolution perpendicular to the axis of scanning (the low-resolution *slit length* direction). Useful for defining resolution data from slit-smearing instruments such as Bonse-Hart geometry. [3] It is unexpected for *Q* and *dQl* to have different units.	*unit={unit}* [1] [4]
Qmean	float	[0..1]	Mean value of *Q* for this data point. [3] Useful when describing data that has been binned from higher-resolution or from area detectors. It is unexpected for *Q* and *Qmean* to have different units.	*unit={unit}* [1]
Shadowfactor	float	[0..1]	A numerical factor applied to pixels affected by the beam stop penumbra. [3] [7]	
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d:1.1 standard can be placed at this point.	*xmlns:{foreign-prefix}={foreign-namespace}*

[3] The *unit* attribute is required. See *Rules* for acceptable values.

[4] Because there are several different ways to describe the SAS intensity, One should be very careful to inspect the *unit* attribute to determine how to handle subsequent data processing, especially in the area of units conversion.

[5] When an optional element (such as *Idev*, *Qdev*, ...) is used, it **must** be used in every *Idata* within the enclosing *SASdata*.

[6] If either *dQw* or *dQl* are used, then *Qdev* is not permitted to be used.

[7] The *Shadowfactor* is used in data files from NIST/NCNR instruments. See: J.G. Barker & J.S. Pedersen (1995) J. Appl. Cryst. 28, 105-114.

Table Notes

SAStransmission_spectrum

parent: *SASentry*

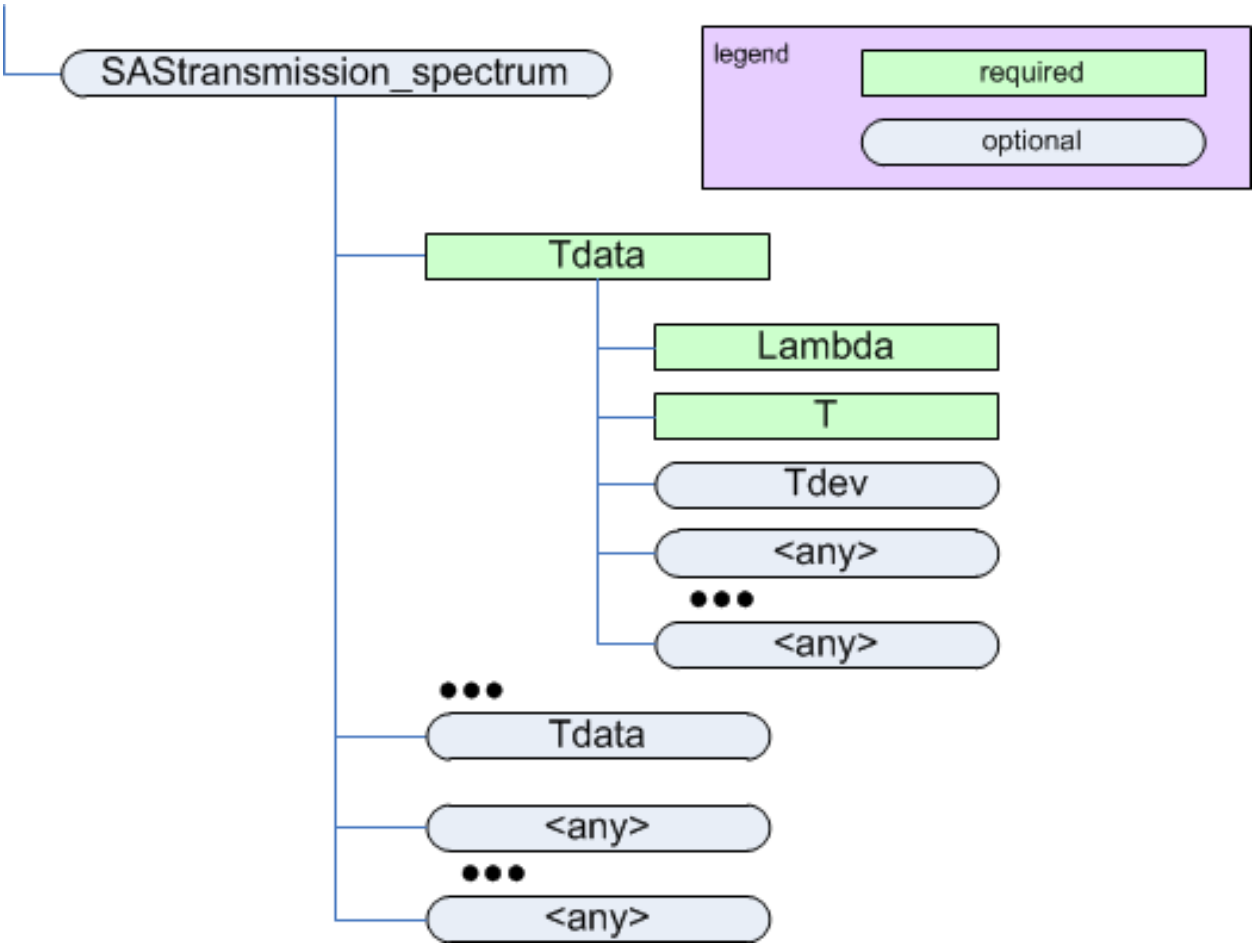


Figure 1.7: The *SAStransmission_spectrum* element

The *SAStransmission_spectrum* element has a *name* attribute to identify what type of spectrum is being described. It is expected that this value will take either of these two values:

value	meaning
sample	measurement with the sample and container
can	measurement with just the container

Name	Type	Occurrence	Description	Attributes
Tdata element	container	[1..inf]	Each *Tdata* describes a single data point of a transmission spectrum.	

Tdata element

parent: *SAStransmission_spectrum*

Name	Type	Occurrence	Description	Attributes
Lambda	float	[1..1]	Wavelength of the radiation bin. Either *A* or *nm* are typical units.	*unit={unit}* [8]
T	float	[1..1]	Transmission value (I/I_0)	*unit={unit}* [1]
Tdev	float	[0..1]	Estimated uncertainty (usually standard deviation) of *T*. [9] It is unexpected for *T* and *Tdev* to have different units.	*unit={unit}* [1]
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d:1.1 standard can be placed at this point.	*xmlns:{foreign-prefix}={foreign-namespace}*

Table Notes

SASsample

parent: *SASentry*

Name	Type	Occurrence	Description	Attributes
ID	string	[1..1]	Text string that identifies this sample.	
thickness	float	[0..1]	Thickness of this sample.	*unit={unit}* [10]
transmission	float	[0..1]	Transmission (I/I_0) of this sample. Note that there is no *unit* attribute as this number is dimensionless.	
temperature	float	[0..1]	Temperature of this sample.	*unit={unit}* [1]
position	container	[0..1]	Location of the sample in *X*, *Y*, and *Z*.	
orientation	container	[0..1]	Orientation (rotation) of the sample.	
details	string	[0..inf]	Any additional sample details.	
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d:1.1 standard can be placed at this point.	*xmlns:{foreign-prefix}={foreign-namespace}*

geometry

See the figures in *Definition of the coordinate axes*.

[8] The *unit* attribute is required. See *Rules* for acceptable values.
[9] When an optional element (such as *Tdev*, ...) is used, it must be used in every *Tdata* within the enclosing *SAStransmission_spectrum*.
[10] The *unit* attribute is required. See *Rules* for acceptable values.

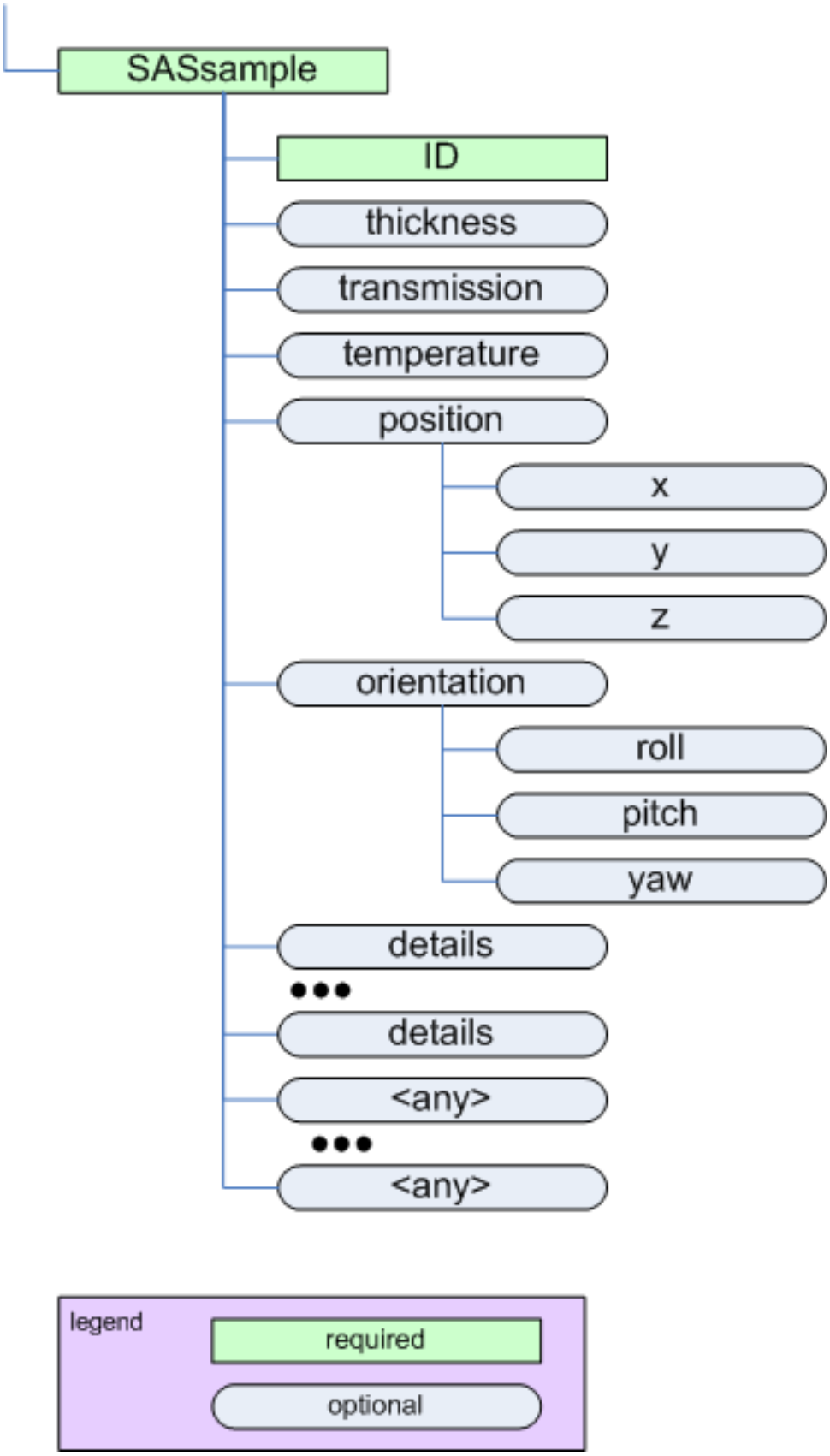


Figure 1.8: The *SASsample* element

position

Name	Type	Occurrence	Description	Attributes
x	float	[0..1]	Position of the sample in *X*.	*unit={unit}* [1]
y	float	[0..1]	Position of the sample in *Y*.	*unit={unit}* [1]
z	float	[0..1]	Position of the sample in *Z*.	*unit={unit}* [1] [11]

orientation

Note: The *orientation* element is intended to describe simple rotations about a single axis rather than a full set of rotations as in a crystallographic context.

Name	Type	Occurrence	Description	Attributes
roll	float	[0..1]	Rotation about the *Z* axis (roll).	*unit={unit}* [1]
pitch	float	[0..1]	Rotation about the *X* axis (pitch).	*unit={unit}* [1]
yaw	float	[0..1]	Rotation about the *Y* axis (yaw).	*unit={unit}* [1]

Table Notes

SASinstrument

parent: *SASentry*

Name	Type	Occurrence	Description	Attributes
name	string	[1..1]	Text string that identifies the name of this instrument	
SASsource	container	[1..1]	Description of the radiation source.	*name={name}*
SAScollimation	container	[1..inf]	Description of a collimation element.	*name={name}*
SASdetector	container	[1..inf]	Description of the detector.	*name={name}*

SASsource

parent: *SASinstrument*

[11] While z is allowed by the standard (provided by use of a standard *size* element in the XML Schema), it does not make sense to use it for small-angle scattering in some situations as noted. Use of z in such situations may be ignored by processing software.

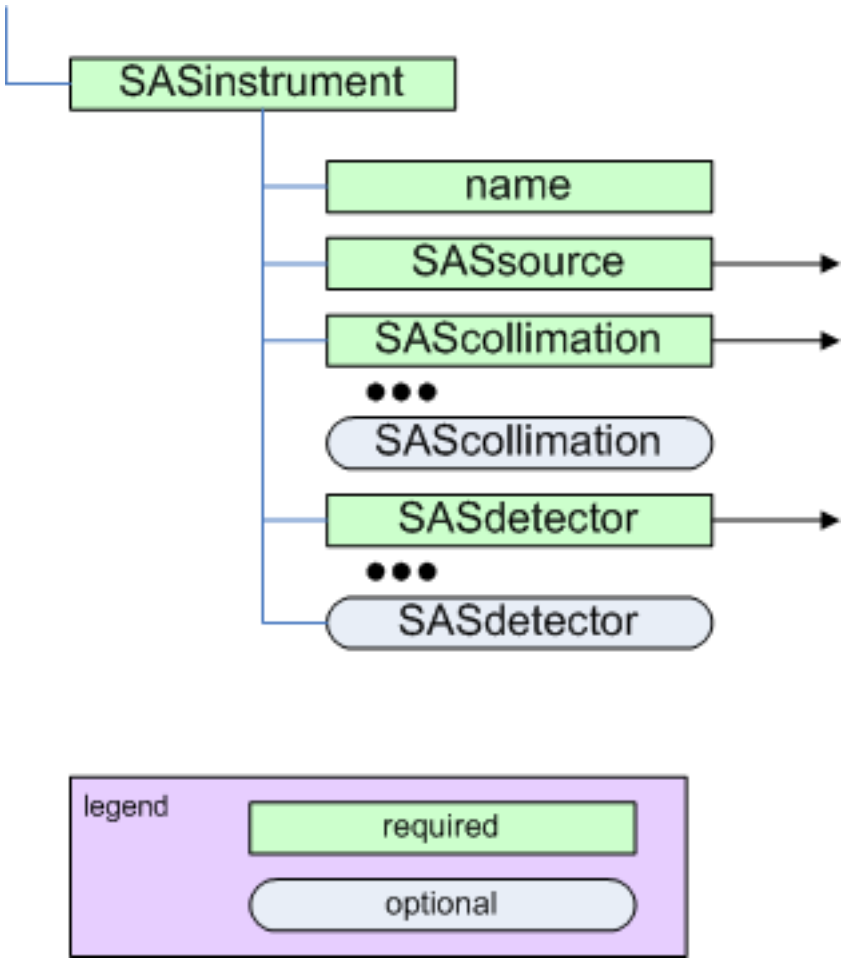


Figure 1.9: The *SASinstrument* element

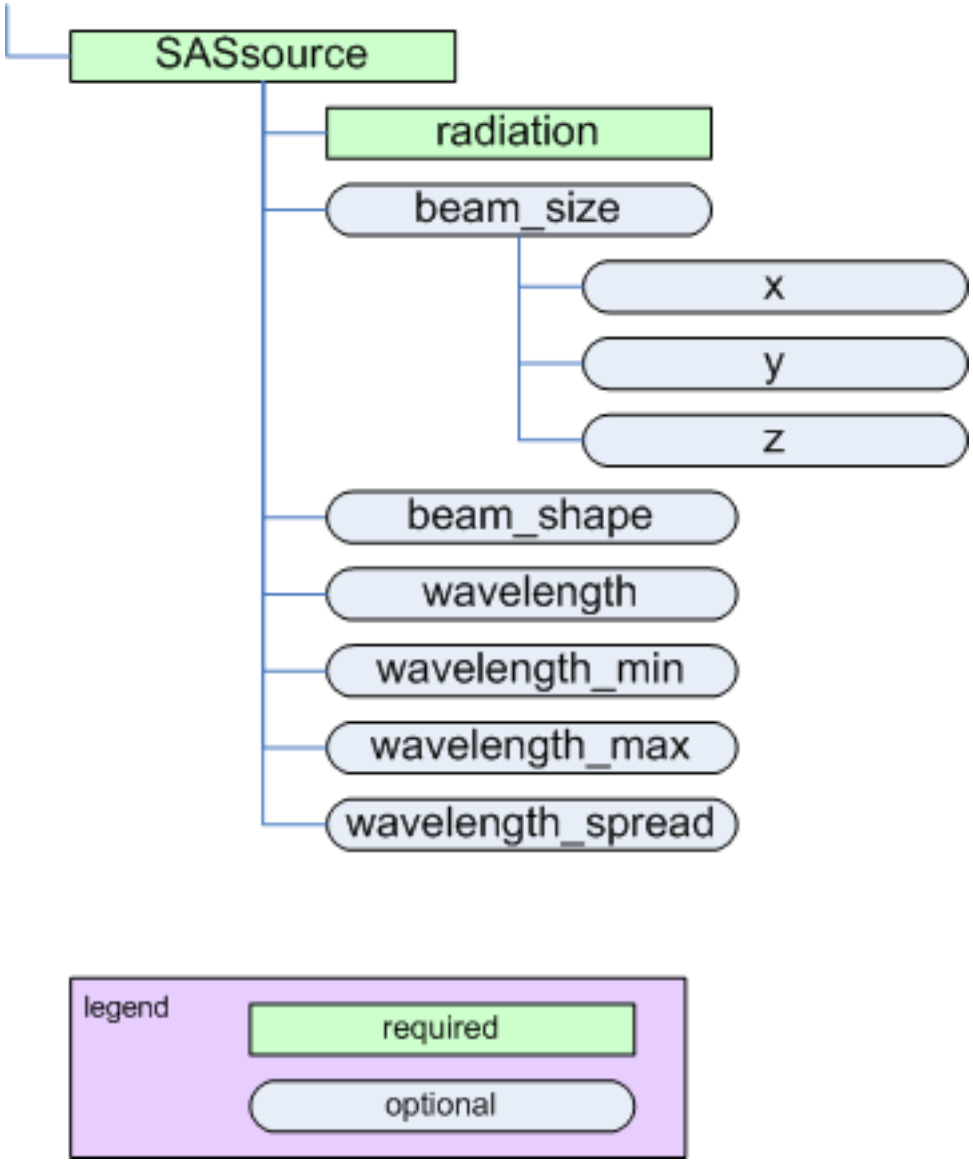


Figure 1.10: The *SASsource* element

Name	Type	Occurrence	Description	Attributes
radiation	string	[1..1]	Name of the radiation used. For maximum compatibility with NeXus, use one of the types defined by NeXus *NXsource*: [12] • Spallation Neutron Source • Pulsed Reactor Neutron Source • Reactor Neutron Source • Synchrotron X-ray Source • Pulsed Muon Source • Rotating Anode X-ray • Fixed Tube X-ray or the NeXus probe type: • neutron • x-ray • muon • electron	
beam_size	container	[0..1]	Physical dimension of the beam (incident on the sample). If beam is round, just use *X* dimension.	*name={name}* [13]
beam_shape	string	[0..1]	Text description of the shape of the beam (incident on the sample).	
wavelength	float	[0..1]	wavelength (λ) of radiation incident on the sample.	*unit={unit}* [14]
wavelength_min	float	[0..1]	Some facilities specify wavelength using a range. The minimum of such a range is given by* wavelength_min*.	*unit={unit}* [3]
wavelength_max	float	[0..1]	Some facilities specify wavelength using a range. The maximum of such a range is given by* wavelength_max*.	*unit={unit}* [3]
wavelength_spread	float	[0..1]	Some facilities specify the width of the wavelength spectrum. The width of such a range is given by *wavelength_spread*.	*unit={unit}* [3]

geometry

See the figures in *Definition of the coordinate axes*.

beam_size

Name	Type	Occurrence	Description	Attributes
x	float	[0..1]	Dimension of the beam in *X*.	*unit={unit}* [3]
y	float	[0..1]	Dimension of the beam in *Y*.	*unit={unit}* [3]
z	float	[0..1]	Dimension of the beam in *Z*.	*unit={unit}* [2 3]

Table Notes

SAScollimation

parent: *SASinstrument*

Name	Type	Occurrence	Description	Attributes
length	float	[0..1]	Amount/length of collimation inserted (on a SANS instrument)	*unit={unit}* [15]
aperture	container	[0..inf]	Description of a slit or aperture. *name*: Optional name attribute for this aperture. *type*: Optional text attribute to describe the type of aperture (pinhole, 4-blade slit, Soller slit, ...).	*name={name}* *type={type}*

aperture

Name	Type	Occurrence	Description	Attributes
size	container	[0..1]	Opening dimensions of this aperture.	*name={name}*
distance	float	[0..1]	Distance from this collimation element to the sample.	*unit={unit}* [1]

size

See the figures in *Definition of the coordinate axes*.

[15] The *unit* attribute is required. See *Rules* for acceptable values.

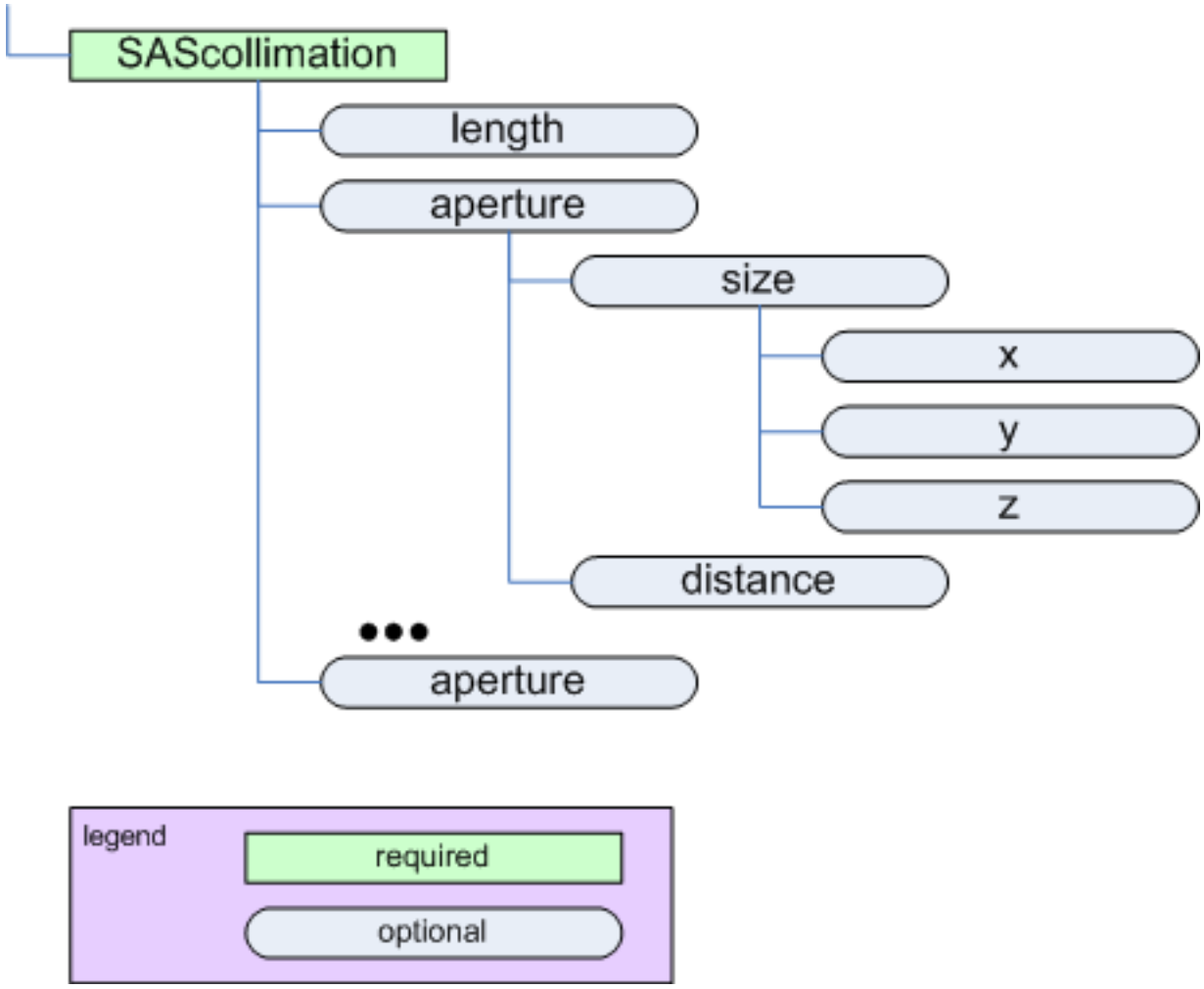


Figure 1.11: The *SAScollimation* element

Name	Type	Occurrence	Description	Attributes
x	float	[0..1]	Dimension of the collimation in X.	*unit={unit}* [1]
y	float	[0..1]	Dimension of the collimation in Y.	*unit={unit}* [1]
z	float	[0..1]	Dimension of the collimation in Z.	*unit={unit}* [1] [16]

Table Notes

SASdetector

parent: *SASinstrument*

Name	Type	Occurrence	Description	Attributes
name	string	[1..1]	Identifies the name of this detector.	
SDD	float	[0..1]	Distance between sample and detector.	*unit={unit}* [17]
offset	container	[0..1]	Offset of this detector position in *X*, *Y*, (and *Z* if necessary).	
orientation	container	[0..1]	Orientation (rotation) of this detector in roll, pitch, and yaw.	
beam_center	container	[0..1]	Center of the beam on the detector in *X*, *Y*, (and *Z* if necessary).	
pixel_size	container	[0..1]	Size of detector pixels in *X*, *Y*, (and *Z* if necessary).	
slit_length	float	[0..1]	Slit length of the instrument for this detector, expressed in the same units as Q. (See *Rules*)	*unit={unit}* [1]

geometry

See the figures in *Definition of the coordinate axes*.

[16] While z is allowed by the standard (provided by use of a standard *size* element in the XML Schema), it does not make sense to use it for small-angle scattering. Use of z here may be ignored by processing software.

[17] The *unit* attribute is required. See *Rules* for acceptable values.

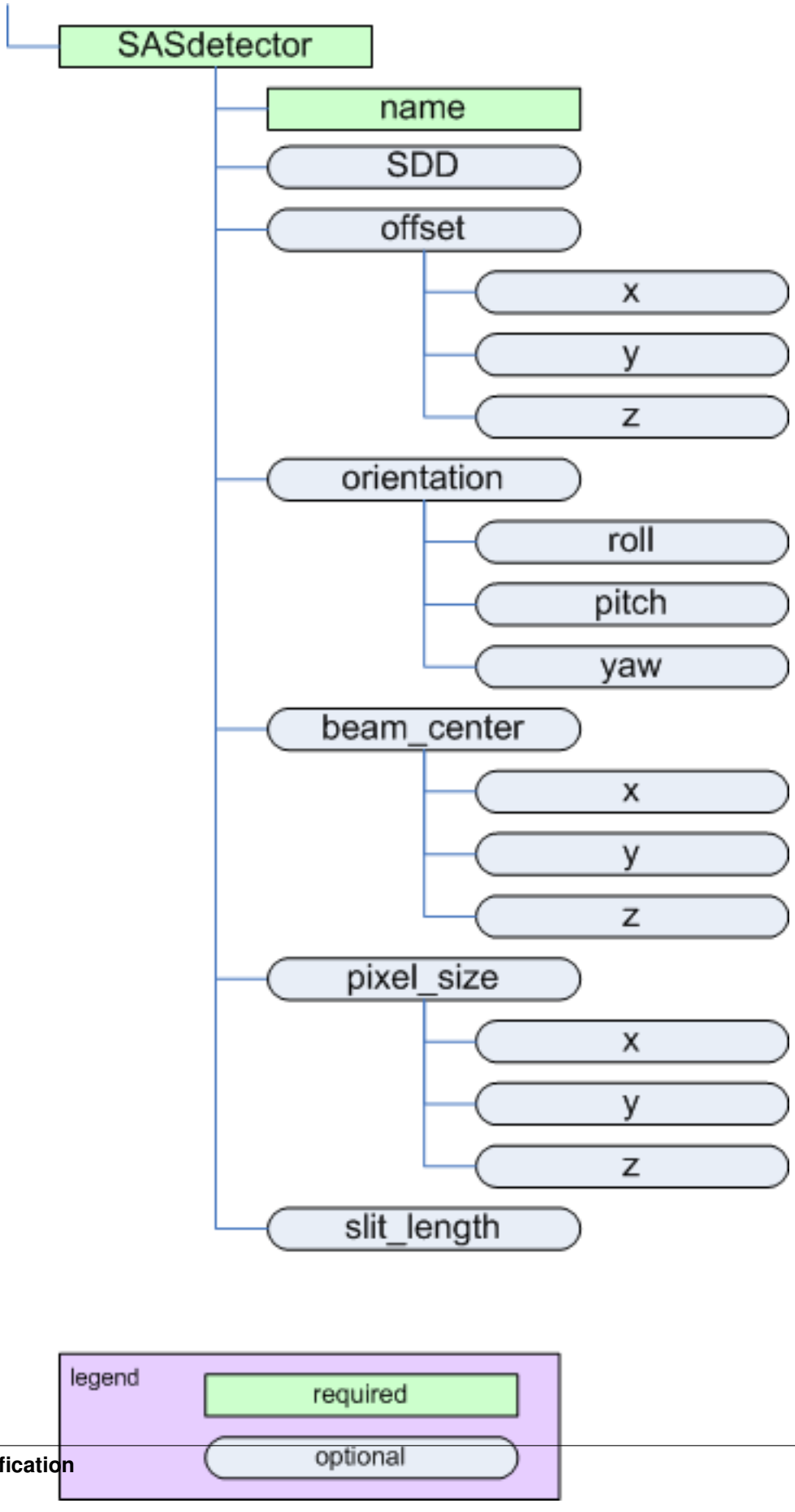


Figure 1.12: The *SASdetector* element

offset

Name	Type	Occurrence	Description	Attributes
x	float	[0..1]	Offset of the detector position in *X*.	*unit={unit}* [1]
y	float	[0..1]	Offset of the detector position in *Y*.	*unit={unit}* [1]
z	float	[0..1]	Offset of the detector position in *Z*.	*unit={unit}* [1] [18]

orientation

Note: The *orientation* element is intended to describe simple rotations about a single axis rather than a full set of rotations as in a crystallographic context.

Name	Type	Occurrence	Description	Attributes
roll	float	[0..1]	Rotation about the *Z* axis (roll).	*unit={unit}* [1]
pitch	float	[0..1]	Rotation about the *X* axis (pitch).	*unit={unit}* [1]
yaw	float	[0..1]	Rotation about the *Y* axis (yaw).	*unit={unit}* [1]

beam_center

Position of the beam center on the detector

Name	Type	Occurrence	Description	Attributes
x	float	[0..1]	Position of the beam center on the detector in *X*.	*unit={unit}* [1]
y	float	[0..1]	Position of the beam center on the detector in *Y*.	*unit={unit}* [1]
z	float	[0..1]	Position of the beam center on the detector in *Z*.	*unit={unit}* [1] [2]

pixel_size

Name	Type	Occurrence	Description	Attributes
x	float	[0..1]	Size of a detector pixel in *X*.	*unit={unit}* [1]
y	float	[0..1]	Size of a detector pixel in *Y*.	*unit={unit}* [1]
z	float	[0..1]	Size of a detector pixel in *Z*.	*unit={unit}* [1] [2]

---

[18] While z is allowed by the standard (provided by use of a standard *size* element in the XML Schema), it does not make sense to use it for small-angle scattering in some situations as noted. Use of z in such situations may be ignored by processing software.

Table Notes

SASprocess

parent: *SASentry*

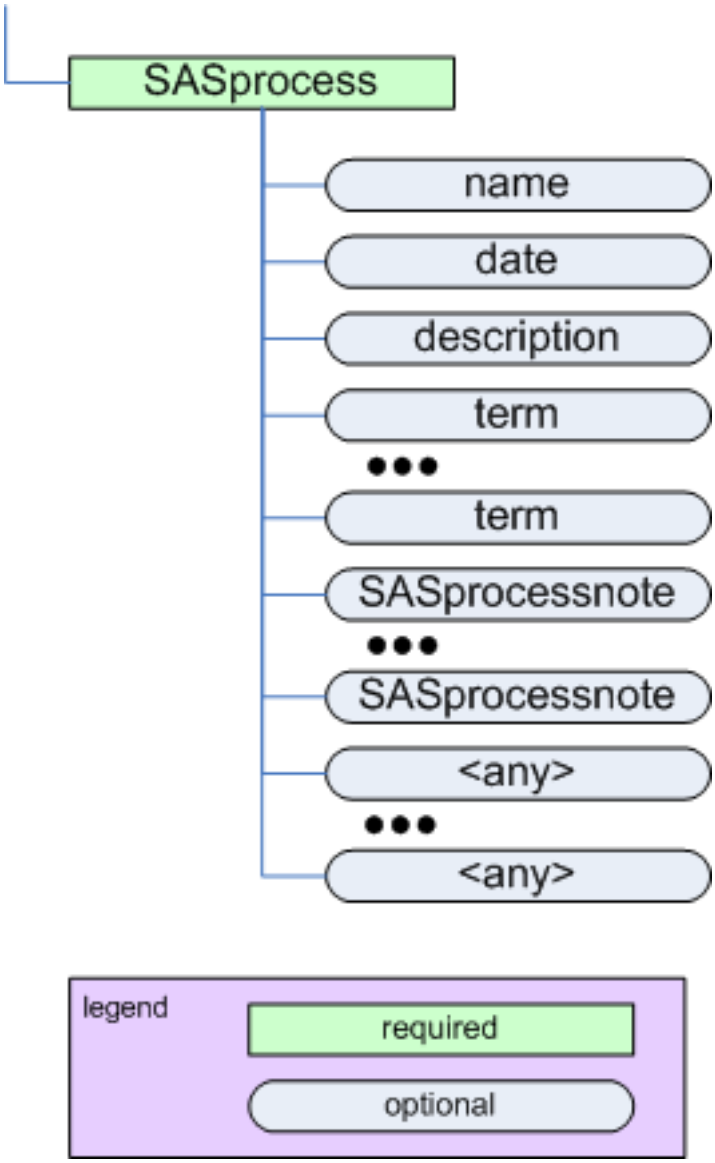


Figure 1.13: The *SASprocess* element

Name	Type	Occurrence	Description	Attributes
name	string	[0..1]	Optional name for this data processing or analysis step.	
date	string	[0..1]	Optional date for this data processing or analysis step. [19]	
description	string	[0..1]	Optional description for this data processing or analysis step.	
term	string	[0..1]	Specifies the value of a single variable, parameter, or term (while defined here as a string, it could be a number) related to the *SASprocess* step.	*unit={unit}* [20]
SASprocessnote	container	[1..inf]	Describes anything about *SASprocess* that is not already described.	
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d:1.1 standard can be placed at this point.	*xmlns:{foreign-prefix}={foreign-namespace}*

SASprocessnote

parent: *SASprocess*

Name	Type	Occurrence	Description	Attributes
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d:1.1 standard can be placed at this point.	*xmlns:{foreign-prefix}={foreign-namespace}*

Table Notes

SASnote

parent: *SASentry*

Name	Type	Occurrence	Description	Attributes
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d:1.1 standard can be placed at this point.	*xmlns:{foreign-prefix}={foreign-namespace}*

{any}

This element is used as a catch-all, to allow users to introduce other data (not defined by this canSAS standard) into the file.

[19] Use a format for the date which is easily machine-readable such as ISO-8601 (either `yyyy-mm-ddThh:mm:ss` or `yyyy-mm-dd hh:mm:ss`). See: http://www.w3.org/TR/NOTE-datetime or http://en.wikipedia.org/wiki/ISO_8601 for more details.

[20] The *unit* attribute is required for numerical data. See *Rules* for acceptable values.

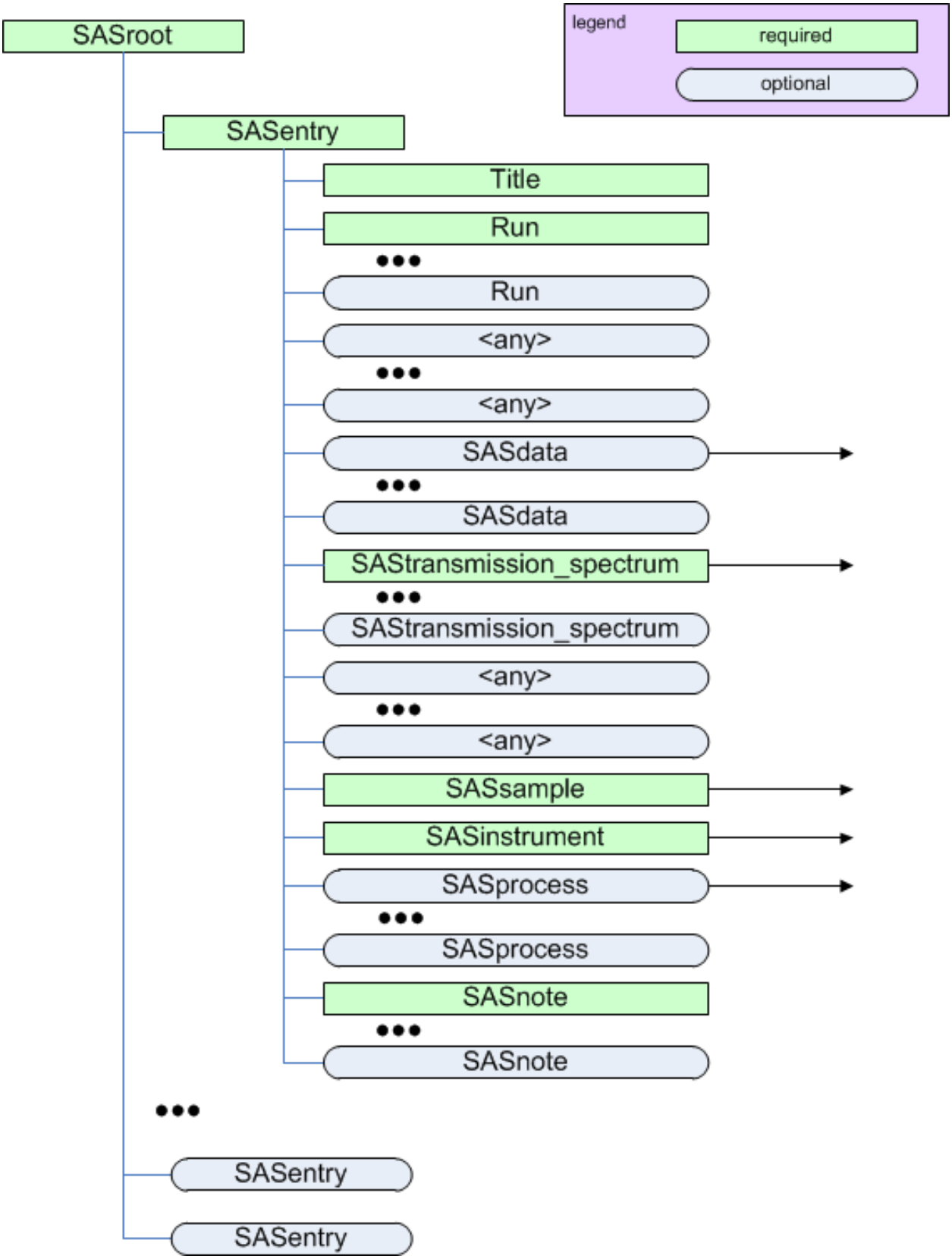


Figure 1.14: The *SASnote* element

Name	Type	Occurrence	Description	Attributes
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d:1.1 standard can be placed at this point. (These are called **foreign** elements. It is suggested to associate foreign elements with a foreign namespace to differentiate them from the canSAS elements in the XML file.)	*xmlns:{foreign-prefix}={foreign-namespace}*

1.2.3 Validation of XML against the Schema

1. open browser to: http://www.xmlvalidation.com/
2. **paste content of candidate XML file** (with reference in the header to the XML Schema as shown above) into the form
3. press `<validate>`
4. **paste content of `cansas1d.xsd` XSD file into form** and press `<continue validation>`

   (http://www.cansas.org/svn/1dwg/trunk/cansas1d.xsd)
5. check the results

1.2.4 Compatibility of Geometry Definitions

The translation and orientation geometry used by this canSAS standard are consistent with:

Cartesian http://en.wikipedia.org/wiki/Cartesian_coordinate_system

Right-hand rule http://en.wikipedia.org/wiki/Right-hand_rule

NeXus http://www.nexusformat.org/Coordinate_Systems

McStas http://mcstas.risoe.dk/documentation/tutorial/node6.html

---

SHADOW The translation and orientation geometry **definitions used here are different** than those used by *SHADOW* (http://www.nanotech.wisc.edu/shadow). In *SHADOW*, the y and z axes are swapped and the direction of x is changed.

1.3 Tutorial

This is a tutorial for *cansas1d:1.1*, the canSAS standard format for storing small-angle scattering data in XML files.

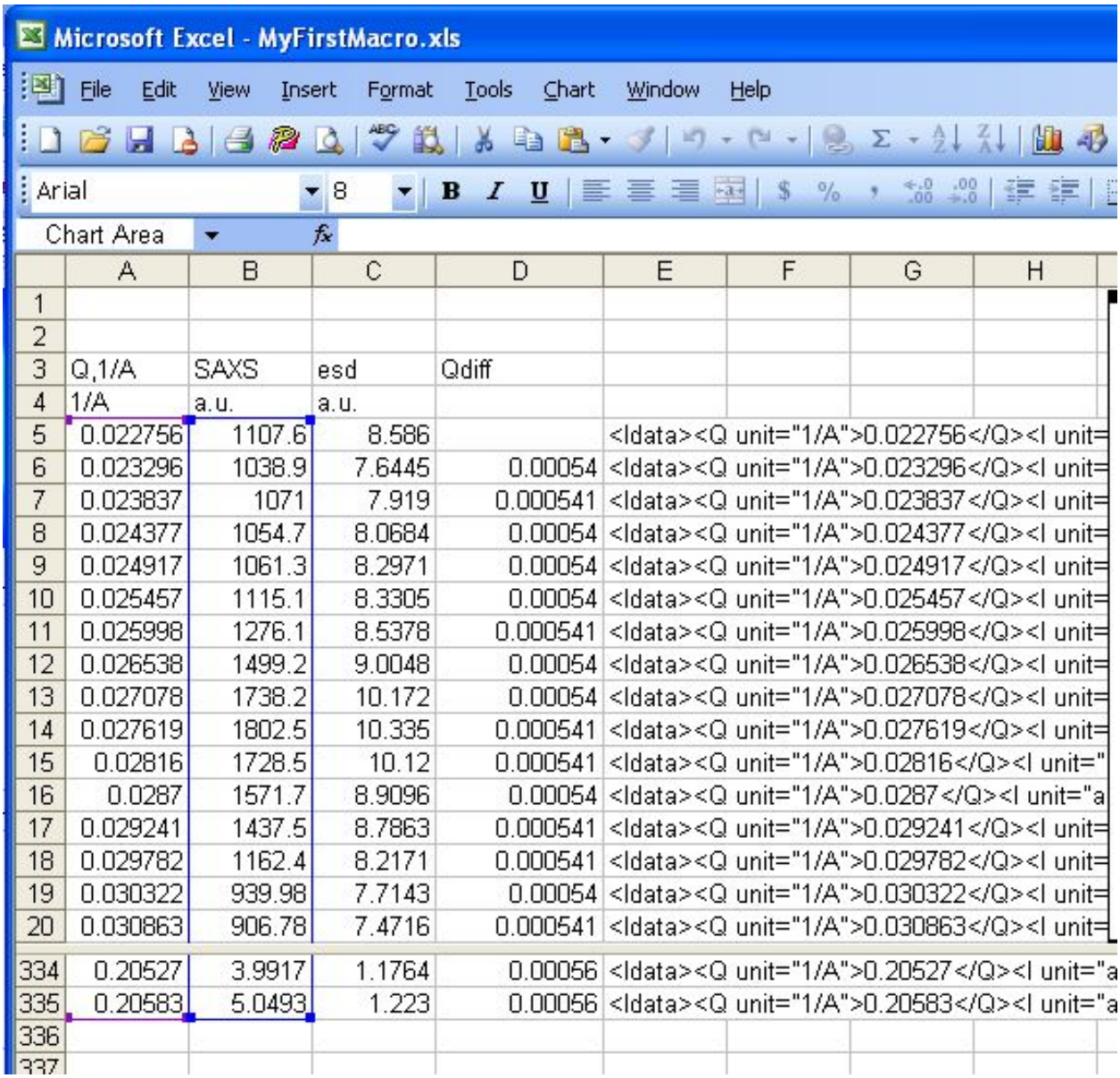
At present, the tutorial section consists of two case studies (see the *Case Studies* section), which can serve as examples. The opportunity could not be more ripe for a better tutorial.

1.4 Case Studies

1.4.1 Case Study: Dry Chick Collagen

To demonstrate how to get SAS data into the XML standard format, consider this set of SAXS data collected at the National Synchrotron Light Source, Brookhaven National Laboratory, using a SAXS camera set up temporarily at beam line X6B (operated by the Materials Science Division, Argonne National Lab).

The sample was **dry chick collagen**. (Thanks to Malcolm Capel, NSLS beam line X12C for the sample.)



Microsoft Excel - MyFirstMacro.xls

File Edit View Insert Format Tools Chart Window Help

Arial 8 B I U

Chart Area

	A	B	C	D	E	F	G	H
1								
2								
3	Q,1/A	SAXS	esd	Qdiff				
4	1/A	a.u.	a.u.					
5	0.022756	1107.6	8.586		<Idata><Q unit="1/A">0.022756</Q><I unit=			
6	0.023296	1038.9	7.6445	0.00054	<Idata><Q unit="1/A">0.023296</Q><I unit=			
7	0.023837	1071	7.919	0.000541	<Idata><Q unit="1/A">0.023837</Q><I unit=			
8	0.024377	1054.7	8.0684	0.00054	<Idata><Q unit="1/A">0.024377</Q><I unit=			
9	0.024917	1061.3	8.2971	0.00054	<Idata><Q unit="1/A">0.024917</Q><I unit=			
10	0.025457	1115.1	8.3305	0.00054	<Idata><Q unit="1/A">0.025457</Q><I unit=			
11	0.025998	1276.1	8.5378	0.000541	<Idata><Q unit="1/A">0.025998</Q><I unit=			
12	0.026538	1499.2	9.0048	0.00054	<Idata><Q unit="1/A">0.026538</Q><I unit=			
13	0.027078	1738.2	10.172	0.00054	<Idata><Q unit="1/A">0.027078</Q><I unit=			
14	0.027619	1802.5	10.335	0.000541	<Idata><Q unit="1/A">0.027619</Q><I unit=			
15	0.02816	1728.5	10.12	0.000541	<Idata><Q unit="1/A">0.02816</Q><I unit="			
16	0.0287	1571.7	8.9096	0.00054	<Idata><Q unit="1/A">0.0287</Q><I unit="a			
17	0.029241	1437.5	8.7863	0.000541	<Idata><Q unit="1/A">0.029241</Q><I unit=			
18	0.029782	1162.4	8.2171	0.000541	<Idata><Q unit="1/A">0.029782</Q><I unit=			
19	0.030322	939.98	7.7143	0.00054	<Idata><Q unit="1/A">0.030322</Q><I unit=			
20	0.030863	906.78	7.4716	0.000541	<Idata><Q unit="1/A">0.030863</Q><I unit=			
334	0.20527	3.9917	1.1764	0.00056	<Idata><Q unit="1/A">0.20527</Q><I unit="a			
335	0.20583	5.0493	1.223	0.00056	<Idata><Q unit="1/A">0.20583</Q><I unit="a			
336								
337								

Figure 1.15: collagen SAXS in Excel table

The raw data was collected on a linear position-sensitive detector and reduced to columns of Q, I, and $Idev$ (estimated standard deviation of I).

The only metadata available for this data (without resorting to digging through piles of old notebooks) was obtained from the headers of two files:

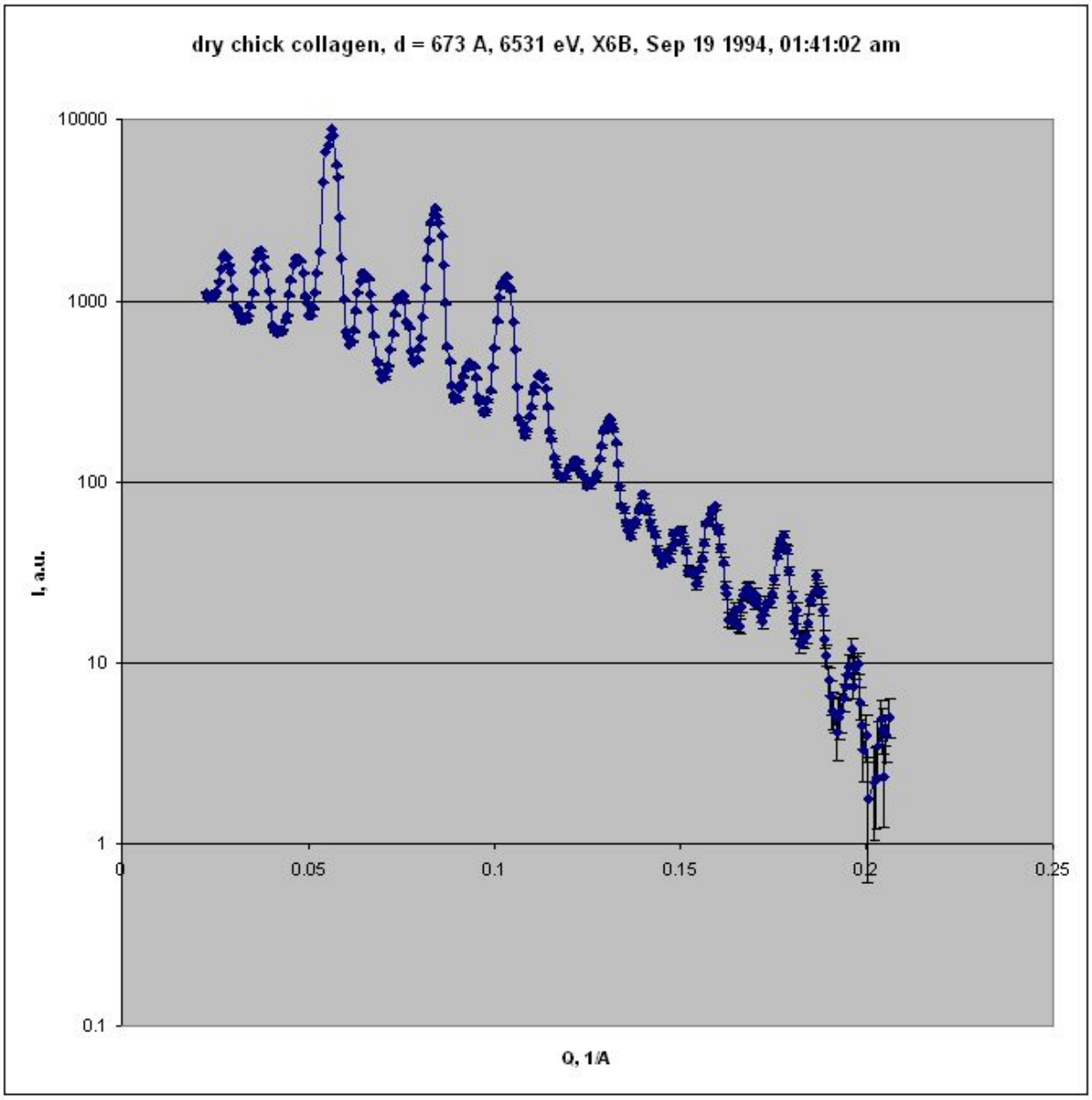


Figure 1.16: collagen SAXS in Excel chart (log-log)

The header lines from *COLLAGEN.ASC* (`download`):

```
Sep 19 1994     01:41:02 am     Elt: 00090 Seconds
ID: No spectrum identifier defined
Memory Size: 8192 Chls  Conversion Gain: 1024  Adc Offset: 0000 Chls
```

The header lines from *collagen.saxs* (`download`):

```
dry chick collagen, d = 673 A
6531 eV, X6B
```

There is enough information to fulfill the minimum requirements of the 1D standard file format and also make an excellent example of a minimal canSAS reduced 1-D SAS data file in XML.

Create the XML data file

The procedure to create the XML data file by hand is described next.

Make the basic XML file

It is easiest to copy a template rather than start from an empty file. Copy the `cansas1d.xml` file (http://www.cansas.org/svn/1dwg/trunk/examples/cansas1d.xml) into your working directory and rename it to *collagen.xml*.

Modify *collagen.xml*

It is easier to see the metadata in the XML file before you enter the SAXS data into the file. With the brief metadata available, most of the other lines in *cansas1d.xml* can be eliminated. This will result in a file that looks like the next example.

collagen.xml with metadata but before data lines are added

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="example.xsl" ?>
<SASroot version="1.1"
        xmlns="urn:cansas1d:1.1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:cansas1d:1.1
            http://www.cansas.org/formats/1.1/cansas1d.xsd"
    >
    <SASentry>
    <Title>dry chick collagen, d = 673 A, 6531 eV, X6B</Title>
        <Run />
        <SASdata>
            <!-- Idata lines will go here -->
        </SASdata>
        <SASsample>
            <ID>dry chick collagen, d = 673 A, 6531 eV, X6B</ID>
        </SASsample>
        <SASinstrument>
```

```
            <name>X6B, NSLS, BNL</name>
            <SASsource>
                <radiation>X-ray synchrotron</radiation>
                <wavelength unit="A">
                    1.898   <!-- = 12398/6531 -->
                </wavelength>
            </SASsource>
            <SAScollimation />
            <SASdetector>
                <name>X6B PSD</name>
            </SASdetector>
        </SASinstrument>
        <SASnote>
            Sep 19 1994     01:41:02 am     Elt: 00090 Seconds
            ID: No spectrum identifier defined
            Memory Size: 8192 Chls  Conversion Gain: 1024  Adc Offset: 0000 Chls

            dry chick collagen, d = 673 A
            6531 eV, X6B
        </SASnote>
    </SASentry>
</SASroot>
```

Prepare the SAXS data

Microsoft Excel is used here to convert the table of SAXS data into the required lines of XML for the standard. Some may prefer to use a cell formula but here, we develop a bit of Excel Macro code to clarify our procedure.

Using Excel macros to reformat the SAXS data

Within Excel, with the SAXS data in columns as shown in the Excel table above, let's define the macros for our use. In Excel, type *<alt><F11>* to open the macro editing window.

Microsoft Excel macro to format the *Idata* lines

```
Function XML_tag(tag, attr, content) As String
  If attr = "" Then
    XML = "<" & tag & ">"
  Else
  XML = "<" & tag & " " & attr & ">"
  End If
  XML = XML & content
  XML = XML & "</" & tag & ">"
  XML_tag = XML
End Function

Function SAS_Idata_tag(element, unit, content) As String
  XML = XML_tag(element, "unit=""" & unit & """", content)
  SAS_Idata_tag = XML
End Function

Function Idata_tag(Q, Q_unit, I, I_unit, Idev, Idev_unit) As String
  XML = SAS_Idata_tag("Q", Q_unit, Q)
  XML = XML & SAS_Idata_tag("I", I_unit, I)
```

```
    XML = XML & SAS_Idata_tag("Idev", Idev_unit, Idev)
    Idata_tag = XML_tag("Idata", "", XML)
End Function
```

Your window will look similar to this one when you copy/paste the above example code: (Yes, my spreadsheet is called *MyFirstMacro.xls*)

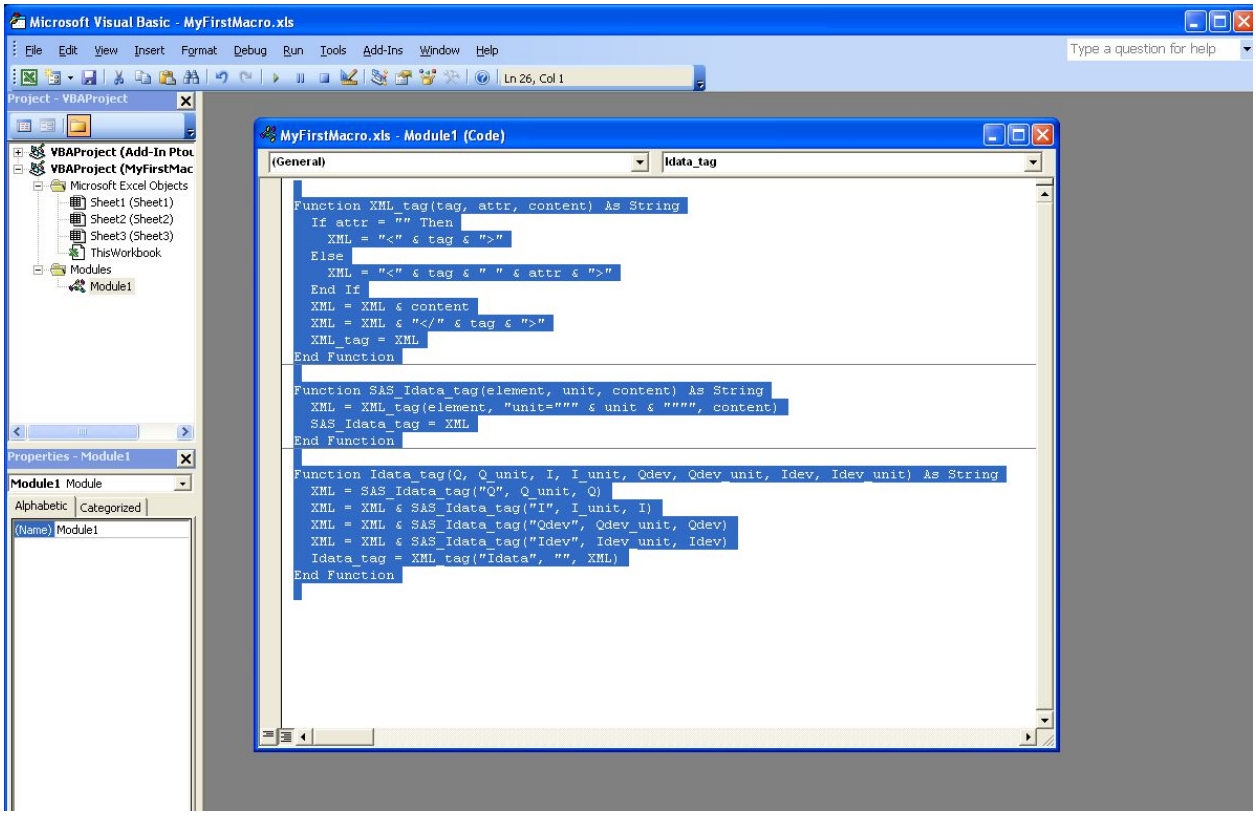


Figure 1.17: case study: Collagen, SAXS data in Excel chart

Now close the macro editing window and return to the SAXS data in the spreadsheet.

construct the *Idata* lines in XML

Move to spreadsheet cell *E5* and enter this formula:

```
=IDATA_tag(A5,$A$4,B5,$B$4,C5,$C$4)
```

Copy it down all rows in column **E** through cell *E335*.

Select cells E5:E335 and copy to clipboard, then paste into *collagen.xml* document inside the *SASdata* element where you see the XML comment.

Final Result

A nicely-formatted display version of the final result can be viewed through the TRAC repository:

http://www.cansas.org/trac/browser/1dwg/trunk/cs_collagen_full.xml

Validate the file

So you think you have an XML file? Let's validate it using the procedure from the documentation. All the instructions are in the *Validation of XML against the Schema* section. No sense in repeating them here.

References

All files are available at:

http://www.cansas.org/trac/browser/1dwg/trunk/examples/collagen/

1.4.2 Case Study: AF1410 Steel

Note: This case study has not yet been written up.

data file: `cs_af1410.xml` http://www.cansas.org/svn/1dwg/trunk/examples/cs_af1410.xml

The data file contains multiple *SASentry* elements that pertain to different samples treated at different conditions in a time series. Each *SASentry* element contains two *SASdata* sections that correspond to sector averages from the two-dimensional SANS data. Since the samples had been subjected to a 1.6T magnetic field to clear the scattering from magnetic domain boundaries in one direction, the sector average for that direction has scattering dominated by purely nuclear scattering moments. The other *SASdata* section has scattering due to both nuclear and magnetic scattering moments.

Also see the publication: A.J. Allen, D. Gavillet, J.R. Weertman, "Small-Angle Neutron Scattering Studies of Carbide Precipitation in Ultrahigh-Strength Steels," *Acta Metall* **41** (1993) 1869-1884.

1.5 Examples

Various topics have been considered or presented in considering this standard. Some are described below.

1.5.1 Example XML Data Files

This section presents two examples of XML Data Files adhering to the cansas1d:1.1 standard.

The first file (*data-simple.xml*) is a basic example and the second file (*cansas1d.xml*) uses almost all the allowed elements. In each, though, most of the data has been removed to clarify the structure.

data-simple.xml

The example data file `data-simple.xml` shows just the basic elements of the cansas1d:1.1 standard. Only a single data point has been shown to more clearly show the other structure. The data file is actually an excerpt from the *bimodal-test1.xml* (http://www.cansas.org/trac/browser/1dwg/trunk/bimodal-test1.xml) example file in the main distribution.

```
1  <?xml version="1.0"?>
2  <?xml-stylesheet type="text/xsl" href="ascii3col.xsl" ?>
3  <SASroot version="1.1"
4          xmlns="urn:cansas1d:1.1"
5          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6          xsi:schemaLocation="urn:cansas1d:1.1
7              http://www.cansas.org/formats/1.1/cansas1d.xsd"
8          >
9          <SASentry>
10                 <Title>SAS bimodal test1</Title>
11                 <Run>1992</Run>
12                 <SASdata>
13                         <Idata>
14                                 <Q unit="1/A">0.0040157139</Q>
15                                 <I unit="1/cm">3497.473</I>
16                                 <Idev unit="1/cm">90.72816</Idev>
17                         </Idata>
18                         <Idata>
19                                 <Q unit="1/A">0.0045408653</Q>
20                                 <I unit="1/cm">3340.003</I>
21                                 <Idev unit="1/cm">84.95314</Idev>
22                         </Idata>
23                 </SASdata>
24                 <SASsample>
25                         <ID>bimodal-test1</ID>
26                 </SASsample>
27                 <SASinstrument>
28                         <name>simulated SAS calculation</name>
29                         <SASsource>
30                                 <radiation>artificial</radiation>
31                                 <wavelength unit="A">1.00</wavelength>
32                         </SASsource>
33                         <SAScollimation/>
34                         <SASdetector>
35                                 <name>calculation</name>
36                         </SASdetector>
37                 </SASinstrument>
38                 <SASprocess>
39                         <name>create the SAS data</name>
40                         <date>1992-01-31</date>
41                         <term name="shape">spheres</term>
42                         <term name="contrast" unit="1/cm^4">100E20</term>
43                         <term name="Background" unit="1/cm">0.1</term>
44                         <term name="sMult" unit="cts/cm">1000.0</term>
45                         <term name="sNoise" unit="fraction">0.25</term>
46                         <SASprocessnote/>
47                 </SASprocess>
48                 <SASnote/>
49         </SASentry>
50 </SASroot>
```

The stylesheet (*ascii3col.xsl*) identified on line 2 of *data-simple.xml* is a very basic XSLT. When *data-simple.xml* is opened in a browser (from a directory containing both *data-simple.xml* and *ascii3col.xsl*), the result looked like this:

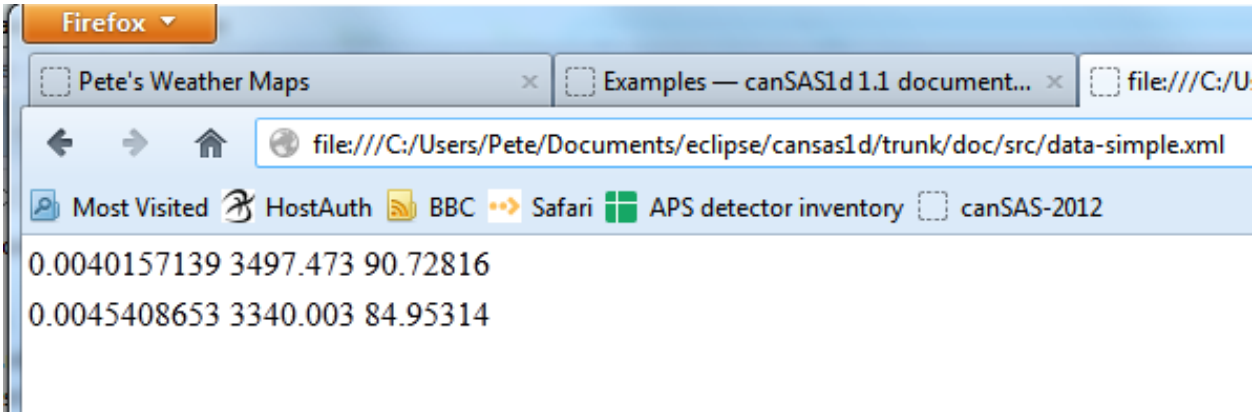


Figure 1.18: View of *data-simple.xml* in a browser **after** XSLT transformation using *ascii3col.xsl*.

cansas1d.xml

The example data file `cansas1d.xml` (http://www.cansas.org/trac/browser/1dwg/trunk/examples/cansas1d.xml) shows examples of most of the elements of the cansas1d:1.1 standard. Only a single data point has been provided here to more clearly show the other structure in the data file.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="cansas1d.xsl" ?>
<SASroot version="1.1"
  xmlns="urn:cansas1d:1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:cansas1d:1.1 http://www.cansas.org/formats/1.1/cansas1d.xsd"
  >
  <SASentry>
    <Title></Title>
    <Run></Run>
    <SASdata>
      <Idata>
          <Q unit="1/A">0.02</Q>
          <I unit="1/cm">1000</I>
          <Idev unit="1/cm">3</Idev>
          <Qdev unit="1/A">0.01</Qdev>
          <Qmean unit="1/A"><!-- Qmean is optional --></Qmean>
          <Shadowfactor><!-- Shadowfactor is optional --></Shadowfactor>
      </Idata>
    </SASdata>
    <SASsample>
      <ID>SI600-new-long</ID>
      <thickness unit="mm">1.03</thickness>
      <transmission>0.327</transmission>
      <temperature unit="C">0.0000</temperature>
      <position>
          <x unit="mm">10.00</x>
          <y unit="mm">0.00</y>
      </position>
      <orientation>
          <roll unit="degree">22.5<!-- was: sample_orientation --></roll>
          <pitch unit="degree">0.020<!-- was: sample_offset_angle --></pitch>
      </orientation>
      <details>
```

```
        <!-- was: sample_prep -->
        http://chemtools.chem.soton.ac.uk/projects/blog/blogs.php/bit_id/2720
      </details>
    </SASsample>
    <SASinstrument>
      <name>canSAS instrument</name>
      <SASsource>
        <radiation>neutron</radiation>
        <beam_size>
          <x unit="mm">12.00</x>
          <y unit="mm">12.00</y>
        </beam_size>
        <beam_shape>disc</beam_shape>
        <wavelength unit="A">6.00</wavelength>
        <wavelength_min unit="nm">0.22</wavelength_min>
        <wavelength_max unit="nm">1.00</wavelength_max>
        <wavelength_spread unit="percent">
          14.3
        </wavelength_spread>
      </SASsource>
      <SAScollimation>
        <aperture name="source" type="radius">
          <size>
            <x unit="mm">50</x>
          </size>
          <distance unit="m">11.000<!-- was: distance_coll --></distance>
        </aperture>
        <aperture name="sample" type="radius">
           <size>
             <x unit="mm">0</x>
           </size>
        </aperture>
      </SAScollimation>
      <SASdetector>
        <name>fictional hybrid</name>
        <SDD unit="m">
          <!-- sample-to-detector distance -->
          4.150
        </SDD>
        <orientation>
          <roll unit="degree">0.00</roll>
          <pitch unit="degree">0.00</pitch>
          <yaw unit="degree">0.00</yaw>
        </orientation>
        <beam_center>
          <x unit="mm">322.64</x>
          <y unit="mm">327.68</y>
        </beam_center>
        <pixel_size>
          <x unit="mm">5.00</x>
          <y unit="mm">5.00</y>
        </pixel_size>
      </SASdetector>
    </SASinstrument>
    <SASprocess>
      <name>spol</name>
      <date>04-Sep-2007 18:35:02</date>
      <term name="radialstep" unit="mm">10.000</term>
```

```
      <term name="sector_width" unit="degree">180.0</term>
      <term name="sector_orient" unit="degree">0.0</term>
      <term name="MASK_file">USER:MASK.COM</term>
      <SASprocessnote>
          AvA1 0.0000E+00 AsA2 1.0000E+00 XvA3 1.0526E+03 XsA4
          5.2200E-02 XfA5 0.0000E+00
      </SASprocessnote>
      <SASprocessnote>
          S... 13597 0 2.26E+02 2A 5mM 0%D2O Sbak 13594 0 1.13E+02
          H2O Buffer
      </SASprocessnote>
      <SASprocessnote>V... 13552 3 1.00E+00 H2O5m</SASprocessnote>
    </SASprocess>
    <SASprocess>
      <name>NCNR-IGOR</name>
      <date>03-SEP-2006 11:42:47</date>
      <description />
      <term name="average_type">Circular</term>
      <term name="SAM_file">SEP06064.SA3_AJJ_L205</term>
      <term name="BKD_file">SEP06064.SA3_AJJ_L205</term>
      <term name="EMP_file">SEP06064.SA3_AJJ_L205</term>
      <term name="DIV_file">SEP06064.SA3_AJJ_L205</term>
      <term name="MASK_file">SEP06064.SA3_AJJ_L205</term>
      <term name="ABS:TSTAND">1</term>
      <term name="ABS:DSTAND" unit="mm">1</term>
      <term name="ABS:IZERO">230.09</term>
      <term name="ABS:XSECT" unit="mm">1</term>
      <SASprocessnote/>
    </SASprocess>
    <SASnote />
  </SASentry>
</SASroot>
```

Other Example Data Files

There are many example data files in the repository. They may be viewed with a WWW browser, [21] downloaded, or the entire directory may be checked out to create a local copy for you.

```
>>> svn co http://www.cansas.org/svn/1dwg/tags/v1.1/examples cansas1d-examples
```

- 1998spheres.xml
- bimodal-test1.xml
- cansas1d-template.xml
- cansas1d.xml
- cs_af1410.xml
- cs_collagen.xml
- cs_collagen_full.xml
- cs_rr_polymers.xml
- gc14-dls-i22.xml
- GLASSYC_C4G8G9_no_TL.xml

[21] http://www.cansas.org/svn/1dwg/tags/v1.1/examples (or http://www.cansas.org/svn/1dwg/trunk/examples)

- GLASSYC_C4G8G9_w_TL.xml
- ill_sasxml_example.xml
- ISIS_SANS_Example.xml
- isis_sasxml_example.xml
- r586.xml
- r597.xml
- s81-polyurea.xml
- samdata_WITHTX.xml
- W1W2.XML
- xg009036_001.xml

1.5.2 Example XML Stylesheets

This section presents examples of XML Stylesheets useful for the cansas1d:1.1 standard. XML Stylesheets (XSLT) are used to transform XML documents into other documents such as XML documents, xhtml documents, or even ASCII text. XML stylesheets also can be used to extract metadata from XML files.

ascii3col.xsl

The *ascii3col.xsl* stylesheet displays all the *Idata* blocks in a cansas1d:1.1 file in 3-column ASCII form. Be careful using this stylesheet on files with multiple *SASdata* or *SASentry* blocks since this stylesheet assumes there is only one of each of these. While it is the most common case to have only one of each, some of the examples have multiple data sets. This stylesheet will concatenate all of the *Idata*.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cs="urn:cansas1d:1.1"
        xsi:schemaLocation="urn:cansas1d:1.1 http://www.cansas.org/formats/1.1/cansas1d.xsd"
>
<xsl:template match="/">
<html>
<table>
<xsl:for-each select="cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata">
<tr>
<td><xsl:value-of select="cs:Q"/></td>
<td><xsl:value-of select="cs:I"/></td>
<td><xsl:value-of select="cs:Idev"/></td>
</tr>
</xsl:for-each>
</table>
</html>
</xsl:template>
</xsl:stylesheet>
```

cansas1d.xsl

The `cansas1d.xsl` (at about 500 lines, it is too large to print here) is the standard XSL stylesheet for cansas1d:1.1 files. It shows all available *SASdata* and metadata, separated by the different *SASentry* blocks.

1.6 Language Bindings

Bindings (import/export drivers) and other software support have been created and contributed. These are listed here by the language or software environment.

1.6.1 FORTRAN

The development of the FORTRAN language, so beloved of scientists, pre-dates the development of XML. And it shows. FORTRAN is not a language that manipulates strings with ease, and this makes parsing XML decidedly awkward. So unless you *really* must use FORTRAN, you are probably better off with C/C++ (or something else more 'modern'), see for example Daniel Veillard's LIBXML2 library at http://xmlsoft.org/ or Frank van den Berghen's parser at http://www.applied-mathematics.net/tools/xmlParser.html.

If you have to use a dialect earlier than FORTRAN-90 (F90), then the chances are you will have to code your own parser.

Software Development Kits

For later dialects, there are some SDKs available on the Web:

F90

- *XMLPARSE* - by Arjen Markus at http://xml-fortran.sourceforge.net/
- *FoX* - by Toby White others at http://uszla.me.uk/space/software/FoX/

F95

- *XML* - by Mart Rentmeester at http://nn-online.org/code/xml/

canSAS 1-D SAS XML v1.0 and v1.1 support

Steve King <*stephen.king@stfc.ac.uk*> (ISIS) has provided a F77 routine that will read cansas1d v1.0 and v1.1 XML files.

http://www.cansas.org/trac/browser/1dwg/trunk/fortran/SASXML_G77.F

1.6.2 IgorPro

An import tool (binding) for IgorPro has been created (*cansasXML.ipf*). You can check out the IgorPro working directory from the SVN server (see instructions below).

To use the *canSASxml.ipf* procedure, you must have the XMLutils XOP IGOR plugin installed. See the *Usage Notes* section below.

Note: Note that this tool is not a true binding [22] in that the structure of the XML file is not replicated in IgorPro data structures. This tool reads the vectors of 1-D SAS data (*Q*, *I*, ...) into IgorPro waves (*Qsas*, *Isas*, ...). The tool also reads most of the metadata into an IgorPro textWave for use by other support in IgorPro.

Note: Note that the code described here is *not a complete user interface*. (See further comments below.) It is expected that this code will be called by a graphical user interface routine and that routine will handle the work of copying the loaded SAS data in IgorPro from the *root:Packages:CS_XMLreader* data folder to the destination of choice (including any renaming of waves as desired).

---

file http://www.cansas.org/trac/browser/1dwg/trunk/IgorPro/cansasXML.ipf

author Pete R. Jemian <jemian@anl.gov>

date 2013-03-29

version 1.12 (**requires** latest XMLutils XOP – see below)

purpose Implement an IgorPro file reader to read the canSAS 1-D reduced SAS data in XML files that adhere to the cansas1d:1.1 standard.

URL

TRAC http://www.cansas.org/trac/browser/1dwg/trunk/IgorPro/cansasXML.ipf

SVN http://www.cansas.org/svn/1dwg/trunk/IgorPro/cansasXML.ipf

requires

- IgorPro: http://www.wavemetrics.com
- *XMLutils* XOP: http://www.igorexchange.com/project/XMLutils
- minimum requirement: IGOR.5.04.x-1.x-dev (circa 2008-Aug-22)

Checkout of support code in Subversion

Subversion (http://subversion.tigris.org/) is a program for managing software versions. There are command line and GUI clients for a variety of operating systems. We won't recommend any here but will show the command lines necessary.

XMLutils XOP

The *XMLutils* XOP, written by Andrew Nelson (ANSTO), is hosted on the IgorExchange (http://www.igorexchange.com/).

One good location to place the checked out *XMLutils* directory is in the *Wavemetrics* directory, next to the *Igor Pro Folder*.

Here is the subversion checkout command:

```
svn co svn://svn.igorexchange.com/packages/XMLutils/ XMLutils
```

To retrieve an updated version of this support in the future, go into the *XMLutils* directory (created above) and type either of these commands:

```
svn update
svn up
```

[22] For example, see *data binding* from http://en.wikipedia.org/wiki/Binding_%28computer_science%29

This will check the repository and update local files as needed. If the installer program was updated, you'll need to run the new installer program. It is not necessary to uninstall first.

The installer executables contained in the download will do all the installation for you. They will place the XOP in the folder */User Procedures/motofit/XMLutils*, and create a shortcut/alias to the plugin in *./Igor Extensions*. Packages from other facilities should place the XOP there as well.

cansasXML.ipf

Check out the canSAS 1d SAS XML reader from the subversion repository:

```
svn checkout http://www.cansas.org/svn/1dwg/trunk cansas-1dwg
```

This will download lots of extra files. The file of interest is in the IgorPro directory and is called *cansasXML.ipf*

To retrieve an updated version of this support in the future, go into the *cansas-1dwg* directory (created above) and type the command:

```
svn update
```

This will check the repository and update files as needed.

Installation

1. **License and Install the *IgorPro* application** (should have already done this step by now)
2. Quit *IgorPro* if it is running
3. **Download *XMLutils* XOP. Either checkout from subversion (see above) or, with a** web browser, visit http://svn.igorexchange.com/viewvc/packages/XMLutils/trunk/
4. Install *XMLutils* XOP by double-clicking the installer for your operating system.
5. **Download *cansasXML.ipf*. Either checkout from subversion (see above) or, with** a web browser, copy *cansasXML.ipf* from the on-line subversion repository. (http://www.cansas.org/svn/1dwg/trunk/IgorPro/cansasXML.ipf)
6. **Copy *cansasXML.ipf* file to *...WavemetricsIgor Pro FolderUser Procedures*** (or file system equivalent)
7. Then, you should be able to restart *IgorPro* and progress from there.

Usage Notes

To use the *canSASxml.ipf* procedure, you must have the *XMLutils* XOP IgorPro plugin installed. This may be downloaded from the IgorExchange Project site. There are installer executables contained in the download that will do all the installation for you. Each installer will place the XOP in the folder *...Wavemetrics:Igor Pro Folder:User Procedures:motofit:XMLutils*, and create a shortcut/alias to the plugin in *...Wavemetrics:Igor Pro Folder:Igor Extensions*.

What it does

Given an XML file, **CS_XmlReader(fileName)** attempts to open the file and read its contents as if it conformed to the canSAS XML standard for reduced 1-D SAS data (cansas1d:1.1, also known as SASXML). If the file is found to be non-conforming, then *CS_XmlReader(fileName)* returns with an error code (show below), otherwise it returns *0*, indicating *no error*. All data read by this code is left in the IgorPro data folder *root:Packages:CS_XMLreader* for pickup by the calling routine. (Two examples are provided to show how a routine might retrieve the data.)

After opening the XML file (with a file identifier *fileID*), control is passed to *CS_1i_parseXml(fileID)* which then walks through the XML elements. For each *SASentry* in the file, a new data folder is created with the name derived from the *Title* element (or best effort determination). Efforts are taken to avoid duplication of data folder names (using standard IgorPro routines). For *SASentry* elements that contain more than one *SASdata* element, a *SASdata* folder is created for each. The corresponding $I(Q)$ is placed in that subfolder. When only one *SASdata* is found, the $I(Q)$ data is placed in the main *Title* folder.

data columns Each column of data in the *SASdata/Idata/** table is placed into a single IgorPro wave. At present, the code does not check for non-standard data columns.(The capability is built into the code but is deactivated at present).

metadata Additional metadata is collected into a single text wave (*metadata*) where the first column is an identifier (or *key*) and the second identifier is the *value*. Only those keys with non-empty values are retained in the metadata table.

> **Caution:** The *values* are not checked for characters that may cause trouble when placed in a wave note. This will be the responsibility of the calling routine to *clean these up* if the need arises.

The code checks for most metadata elements and will check for repeated elements where the standard permits.

Here is an example of the metadata for the *Case Study: Dry Chick Collagen*.

metadata for the *cs_collagen_full.xml* case study

row *i*	key: *metadata[i][0]*	value: *metadata[i][1]*
0	xmlFile	*cs_collagen_full.xml*
1	namespace	`cansas1d:1.1`
2	*Title*	`dry chick collagen, d = 673 A, 6531 eV, X6B`
3	*Run*	`Sep 19 1994 01:41:02 am`
4	*SASsample/ID*	`dry chick collagen, d = 673 A, 6531 eV, X6B`
5	*SASinstrument/name*	`X6B, NSLS, BNL`
6	*SASinstrument/SASsource/radiation*	`X-ray synchrotron`
7	*SASinstrument/SASsource/wavelength*	`1.898`
8	*SASinstrument/SASsource/wavelength/@unit*	`A`
9	*SASinstrument/SASdetector/@name*	`X6B PSD`

10	*SASnote*	`Sep 19 1994 01:41:02 am El`
		`ID: No spectrum identifier defined`
		`Memory Size: 8192 Chls Conversion`
		
		`dry chick collagen, d = 673 A`
		`6531 eV, X6B`

XML foreign namespace elements These are ignored at this time.

XML namespace and header The routine does a *best-efforts* check to ensure that the given XML file conforms to the required *XML file header*. If you take a minimalist view (*a.k.a.* a shortcut), it is likely that your file may be refused by this and other readers. Pay particular attention to UPPER and lower case in the text **cansas1d:1.1** as this is a **key component** used to index through the XML file.

XML stylesheet processing-instruction is not generated The *XMLutils XOP* package does not provide a method to insert the prescribed XML stylesheet processing-instruction into the XML data file.

```
<?xml-stylesheet type=text/xsl href=example.xsl ?>
```

If this processing-instruction is desired, it must be added to each XML data file by other methods such as use of a text editor or application of an XSLT transformation.

Important Functions

These are the important FUNCTIONS in the *cansasXML.ipf* code.

CS_XmlReader(fileName) reads the named XML file and and loads SAS data

prj_grabMyXmlData() demonstration function to show a usage example

prjTest_cansas1d() demonstration function to show a usage example

Example test case

Here is an example running the test routine *prjTest_cansas1d()*.

```
*prjTest_cansas1d()
XMLopenfile: File(path) to open doesn't exist, or file can't be opened
elmo.xml either not found or cannot be opened for reading
    Completed in 0.00669666 seconds
XMLopenfile: XML file was not parseable
cansasXML.ipf: failed to parse XML
    Completed in 0.0133704 seconds
root element is not SASroot with valid canSAS namespace
    Completed in 0.0134224 seconds
bimodal-test1.xml               identified as: cansas1d:1.1 XML file
    Title: SAS bimodal test1
    Completed in 0.068654 seconds
root element is not SASroot with valid canSAS namespace
    Completed in 0.0172572 seconds
root element is not SASroot with valid canSAS namespace
    Completed in 0.0123102 seconds
root element is not SASroot with valid canSAS namespace
    Completed in 0.00930118 seconds
ISIS_SANS_Example.xml           identified as: cansas1d:1.1 XML file
    Title: standard can 12mm SANS
    Completed in 0.0410387 seconds
W1W2.xml                identified as: cansas1d:1.1 XML file
    Title: standard can 12mm SANS
    Title: TK49 standard 12mm SANS
    Completed in 0.0669074 seconds
ill_sasxml_example.xml          identified as: cansas1d:1.1 XML file
    Title: ILL-D22 example: 7D1 2mm
    Completed in 0.0332752 seconds
isis_sasxml_example.xml                 identified as: cansas1d:1.1 XML file
    Title: LOQ TK49 Standard 12mm C9
    Completed in 0.0388868 seconds
r586.xml                identified as: cansas1d:1.1 XML file
    Title: ILL-D11 example1: 2A 5mM 0%D2O
    Completed in 0.0213737 seconds
r597.xml                identified as: cansas1d:1.1 XML file
    Title: ILL-D11 example2: 2A 5mM 0%D2O
    Completed in 0.0221894 seconds
xg009036_001.xml                identified as: cansas1d:1.1 XML file
```

```
    Title: det corrn 5m
    Completed in 0.0286721 seconds
cs_collagen.xml                  identified as: cansas1d:1.1 XML file
    Title: dry chick collagen, d = 673 A, 6531 eV, X6B
    Completed in 0.0296247 seconds
cs_collagen_full.xml             identified as: cansas1d:1.1 XML file
    Title: dry chick collagen, d = 673 A, 6531 eV, X6B
    Completed in 0.0751836 seconds
cs_af1410.xml            identified as: cansas1d:1.1 XML file
    Title: AF1410-10 (AF1410 steel aged 10 h)
    Title: AF1410-8h (AF1410 steel aged 8 h)
    Title: AF1410-qu (AF1410 steel aged 0.25 h)
    Title: AF1410-cc (AF1410 steel aged 100 h)
    Title: AF1410-2h (AF1410 steel aged 2 h)
    Title: AF1410-50 (AF1410 steel aged 50 h)
    Title: AF1410-20 (AF1410 steel aged 20 h)
    Title: AF1410-5h (AF1410 steel aged 5 h)
    Title: AF1410-1h (AF1410 steel aged 1 h)
    Title: AF1410-hf (AF1410 steel aged 0.5 h)
    Completed in 0.338425 seconds
XMLopenfile: File(path) to open doesn't exist, or file can't be opened
cansas1d-template.xml either not found or cannot be opened for reading
    Completed in 0.00892823 seconds
1998spheres.xml                  identified as: cansas1d:1.1 XML file
    Title: 255 nm PS spheres
    Title: 460 nm PS spheres
    Completed in 2.87649 seconds
XMLopenfile: File(path) to open doesn't exist, or file can't be opened
does-not-exist-file.xml either not found or cannot be opened for reading
    Completed in 0.00404549 seconds
cs_rr_polymers.xml               identified as: cansas1d:1.1 XML file
    Title: Round Robin Polymer A
    Title: Round Robin Polymer B
    Title: Round Robin Polymer C
    Title: Round Robin Polymer D
    Completed in 0.0943477 seconds
s81-polyurea.xml                 identified as: cansas1d:1.1 XML file
    Title: S7 Neat Polyurea
    Completed in 0.0361616 seconds
```

IgorPro Graphical User Interface

At least two groups are working on graphical user interfaces that use the canSAS 1-D SAS XML format binding to IgorPro. The GUIs are intended to be used with their suites of SAS analysis tools (and hide the details of using this support code from the user).

Note: There is no support yet for writing the data back into the canSAS format. Several details need to be described, and these are being collected on the discussion page for the XML format

---

Irena tool suite

Jan Ilavsky's **Irena** tool suite for IgorPro has a GUI to load the data found in the XML file. Refer to http://usaxs.xor.aps.anl.gov/staff/ilavsky/irena.htm for more details.

1.6.3 Java JAXB

A Java binding for the cansas1d:1.1 standard has been auto-created using the JAXB tools from Oracle (formerly Sun, see below for more on JAXB) using the *cansas1d.xsd* XML Schema. See the *Downloading* section below.

Using the Java Binding

The basics of the binding are these java statements. First, associate a JAXB context with the canSAS namespace URI

```
jc = JAXBContext.newInstance("org.cansas.cansas1d");
```

Next, create an object to read XML data into Java data objects created by JAXB from the canSAS XML Schema.

```
Unmarshaller unmarshaller = jc.createUnmarshaller();
```

Next, open an XML file from local storage.

```
InputStream in = new FileInputStream("a/data/file.xml);
```

Next, load the XML data into a Java data structure

```
xmlJavaData = (JAXBElement<SASrootType>) unmarshaller.unmarshal(in);
```

Next, get the SASroot object

```
SASrootType sasroot = xmlJavaData.getValue();
```

With a *SASroot* object, one can iterate over the *SASentry* groups and, for instance, print the *Title* string:

```
for ( SASentryType entry : sasroot.getSASentry() ) {
        System.out.printf("SASentry Title:  %s\n", entry.getTitle());
}
```

Full example

An example that uses the binding is provided in the java support and is available for direct `download` or may be viewed using a web browser: http://www.cansas.org/trac/browser/1dwg/trunk/java/ant-eclipse/src/org/cansas/cansas1d/demo/Reader.java

Output from *Reader.java* is:

```
class: org.cansas.cansas1d.demo.Reader
SVN ID: $Id: binding-java-jaxb.rst 321 2013-03-29 22:10:56Z prjemian $


File: ./resources/cansas1d/cs_collagen.xml
SASentry elements: 1
SASentry
Title:  dry chick collagen, d = 673 A, 6531 eV, X6B
#Runs:  1
Run@name:
Run:    Sep 19 1994     01:41:02 am
#SASdata:       1
SASdata@name:
#points:        125

the end.
```

```
File: ./resources/cansas1d/1998spheres.xml
SASentry elements: 2
SASentry
Title:  255 nm PS spheres
#Runs:  1
Run@name:
Run:    scan2.dat, scan 5
#SASdata:       1
SASdata@name:
#points:        1824

SASentry
Title:  460 nm PS spheres
#Runs:  1
Run@name:
Run:    scan1.dat, scan 67
#SASdata:       1
SASdata@name:
#points:        3689

the end.


File: cannot_find_this.xml
File not found: cannot_find_this.xml
```

example: how to retrieve $I(Q)$

This is a slightly longer example. Look near line 75 for this code:

```
SASdataType sasdata = sasroot.getSASentry().getSASdata()
// ...
Qsas[i] = sasdata.getIdata().get(i).getQ().getValue();
Isas[i] = sasdata.getIdata().get(i).getI().getValue();
```

to see the operations that unwind the data into usable *double[]* vectors. Pretty straightforward although there is lots of interesting, yet unnecessary, diagnostic output. Here is a table that describes the items in the line just shown:

java item	description
sasdata	*SASdataType* object
getIdata()	amongst the */SASdata/Idata* tuples ...
get(i)	... pick the *Idata* tuple from row *i*.
getQ()	Just the */SASdata/Idata/Q*
getValue()	and specifically the value, not the unit

Downloading

Resources (JAR files and documentation) for the Java binding may be found in the canSAS subversion *Software repositories (for cansas1d:1.1 standard)*.

cansas1d-#.#.jar JAR file to add to your CLASSPATH in order to use this binding. Adheres to canSAS 1D standard version #.#.

cansas1d-#.#-javadoc.jar http://www.cansas.org/svn/1dwg/tags/v#.#/java/cansas1d-#.#-javadoc.jar

Use this JAR file if you want to add the source code documentation as tooltips to your editor, such as eclipse. Note that this file is compatible with any ZIP program and can be unzipped to provide a directory with all the documentation as a set of HTML pages. Start with the *index.html* page. Adheres to canSAS 1D standard version #.#.

cansas1d-#.#-sources.jar JAR file of the source code. Note that this is *just* the source code tree and not the full project development tree for the Java (JAXB) API. Adheres to canSAS 1D standard version #.#.

source code (for developers) http://www.cansas.org/trac/browser/1dwg/trunk/java/ant-eclipse

canSAS Development project subversion repository for the Java binding. Only use this if you want to participate as a code developer of this binding.

JAXB: Questions and Answers

Q What is *JAXB*?

A Java Architecture for XML Binding (http://java.sun.com/developer/technicalArticles/WebServices/jaxb)

Q Wow! Is it available for other languages?

A Ask Google. *JAXB* is for Java. (http://java.sun.com/developer/technicalArticles/WebServices/jaxb)

For example: http://www.devx.com/ibm/Article/20261

Q How do I pull out the $I(Q)$ data?

A See Java code fragment *above*

Q Has JAXB been useful?

A **Very useful.** Since an XML Schema was defined, JAXB was very useful to create a Java binding automatically. Then, *javadoc* was able to auto-generate the basic documentation as HTML and *pdfdoclet* was able to auto-generate the documentation in a PDF file.

(re)building the JAXB code

To build the java files with JAXB from the XSD Schema, refer to the JAXB documentation: [23]

Here are the steps taken (this time). Note that to use *xjc*, you'll need the full JDK, not just a JVM or JRE.

```
[jemian@gov,250,v1.1]$ mkdir -p java/ant-eclipse/src/org/cansas
[jemian@gov,251,v1.1]$ xjc -d java/ant-eclipse/src/org/cansas cansas1d.xsd
[jemian@gov,262,_1]$ mv *.java ..
[jemian@gov,263,_1]$ cd ../
[jemian@gov,264,cansas1d]$ rmdir _1
edit package line in *.java to read::

        package org.cansas.cansas1d;
```

1.6.4 PHP

A demonstration of writing the cansas1d:1.1 XML data format can be found in the *XmlWriter* form. See the *Converting data into the XML format: XmlWriter* section for more details.

[23] JAXB: http://docs.oracle.com/cd/E17802_01/webservices/webservices/docs/2.0/tutorial/doc/JAXBUsing.html

1.6.5 Python

The Python binding to read canSAS 1-D data files has been implemented using the *gnosis utils*. It is very easy to use, once you have the *gnosis utils* installed. XML files are read by the `readCanSasFile()` routine from the *cansas1d* support file.

version	namespace
1.0	*cansas1d/1.0*
1.1	*urn:cansas1d:1.1*

If either namespace or version don't match, an exception is raised.

If the version and namespace match this table, the data from the file is mapped into Python objects, as shown by the examples below. Otherwise, an appropriate exception is raised.

Interactive example using *cansas1d*

Here is an interactive example showing how to read a canSAS 1D data file using the canSAS 1D Python binding:

```
>>> from cansas1d import readCanSasFile
>>> sasxml = readCanSasFile('bimodal-test1.xml')
>>> print sasxml.SASentry.Title
<Title id="34b0470" >
>>> print sasxml.SASentry.Title.PCDATA
SAS bimodal test1
>>> print len(sasxml.SASentry.SASdata.Idata)
91
>>> print sasxml.SASentry.SASsample.ID.PCDATA
bimodal-test1
>>> source = sasxml.SASentry.SASinstrument.SASsource
>>> print source.radiation.PCDATA, source.wavelength.PCDATA, source.wavelength.unit
artificial 1.00 A
```

Interactive Example using *gnosis.xml.objectify*

Here is an interactive example showing how to read a canSAS 1D data file using *gnosis.xml.objectify*:

```
>>> from gnosis.xml.objectify import XML_Objectify
>>> sasxml = XML_Objectify('bimodal-test1.xml').make_instance()
>>> print sasxml.SASentry.Title.PCDATA
SAS bimodal test1
>>> sasxml.SASentry.Run.PCDATA
1992
>>> sasxml.SASentry.SASinstrument.name.PCDATA
simulated SAS calculation
>>> data0 = sasxml.SASentry.SASdata.Idata[0]
>>> print data0.Q.unit,  data0.I.unit
1/A 1/cm
>>> print data0.Q.PCDATA,  data0.I.PCDATA,  data0.Idev.PCDATA
0.0040157139 3497.473 90.72816
```

example: how to retrieve $I(Q)$

briefly (ignoring any exception handling):

```
>>> from cansas1d import readCanSasFile
>>> sasxml = readCanSasFile('bimodal-test1.xml')
>>> Q = [float(i.Q.PCDATA) for i in sasxml.SASentry.SASdata.Idata]
>>> I = [float(i.I.PCDATA) for i in sasxml.SASentry.SASdata.Idata]
```

cansas1d.py source code documentation

Here is the source code and documentation of the *cansas1d.py* Python binding. This file (and an example that calls this file and prints more data from the file) may be found in the canSAS *subversion* repository. read canSAS 1-D XML data files (either v1.0 or v1.1)

requires gnosis.xml.objectify # easy_install -U gnosis

basic use in a program:

```
import cansas1d
try:
    sasxml = cansas1d.readCanSasFile(xmlFile)
except cansas1d.Exception_canSAS_namespace, answer:
    print "wrong XML namespace:", answer
    return
except cansas1d.Exception_canSAS_version, answer:
    print "wrong version string:", answer
    return
```

Copyright (c) 2013, UChicago Argonne, LLC This file is distributed subject to a Software License Agreement found in the file LICENSE that is included with this distribution.

exception cansas1d.**Exception_canSAS_namespace**
Bases: exceptions.Exception

canSAS XML file namespace

exception cansas1d.**Exception_canSAS_version**
Bases: exceptions.Exception

version string of the canSAS standard

cansas1d.**readCanSasFile**(*xmlFile*)
open a canSAS XML data file as a gnosis file object

Parameters **xmlFile** (*str*) – name of canSAS 1D XML data file

Returns gnosis object with XML data structure

Raises

- **Exception_canSAS_namespace** – if namespace does not match
- **Exception_canSAS_version** – if version does not match

gnosis utils

The Python binding to read canSAS 1-D data files has been implemented using the gnosis utilities. [24] Specifically, the *gnosis.xml.objectify* package turns arbitrary XML documents into Python objects. Since *gnosis.xml.objectify* is not aware of the XML Schema, it is necessary to check that the file read is a canSAS 1D file. Matching of the namespace and version should be sufficient to accept a canSAS 1D file. Appropriate exceptions are raised if the file does not pass these tests.

[24] gnosis utilities: http://freecode.com/projects/gnosisxml/

You can install the gnosis utilities if you have the *distutils* package using:

```
easy_install -U gnosis
```

1.7 The Intensity Problem

The intensity (see *SASdata*) is permitted in three different forms:

1. **absolute units:** $d\Sigma/d\Omega(Q)$ differential cross-section per unit volume per unit solid angle (typical unit: 1/cm)
2. **absolute units:** $d\sigma/d\Omega(Q)$ differential cross-section per unit atom per unit solid angle (typical unit: cm^2)
3. **arbitrary units:** $I(Q)$ usually a ratio of two detectors but unit is meaningless (typical unit: a.u.)

This presents a few problems for analysis software to sort out when reading the data. Fortunately, it is possible to analyze the unit attribute to decide which type of intensity is being reported and make choices at the time the file is read. But this is an area for consideration and possible improvement.

One problem arises with software that automatically converts data into some canonical units used by that software. The software should not convert units between these three types of intensity indiscriminately.

A second problem is that when arbitrary units are used, then the set of possible analytical results is restricted. With such units, no meaningful volume fraction or number density can be determined directly from $I(Q)$.

1.8 XML Help

Listed below are various references useful in learning XML and related topics.

XML eXtensible Markup Language

- http://www.w3schools.com/xml/
- http://www.w3.org/XML/
- http://en.wikipedia.org/wiki/XML
- http://www.zvon.org/xxl/XPathTutorial/General/examples.html

XSL (or **XSLT**) eXtensible Stylesheet Language

- http://www.w3schools.com/xsl/
- http://www.w3.org/Style/XSL/
- http://en.wikipedia.org/wiki/Extensible_Stylesheet_Language
- http://en.wikipedia.org/wiki/XSLT

XPath XPath is a language for finding information in an XML document

- http://www.w3schools.com/xpath/
- http://www.w3.org/Style/XSL/
- http://en.wikipedia.org/wiki/XPath

Schema An XML Schema describes the structure of an XML document

- http://www.w3schools.com/schema/
- http://www.w3.org/XML/Schema
- http://en.wikipedia.org/wiki/XSD

XML Namespaces An XML namespace is used for providing uniquely-named elements and attributes in an XML instance.

- http://www.zvon.org/xxl/NamespaceTutorial/Output
- http://en.wikipedia.org/wiki/XML_namespaces
- http://www.w3schools.com/XML/xml_namespaces.asp
- http://en.wikipedia.org/wiki/Uniform_resource_identifier
- http://www.usingxml.com/Basics/XmlNamespaces

XML Foreign Elements Inclusion of elements, at select locations, that are not defined by the cansas1d.xsd XML Schema

- http://books.xmlschemata.org/relaxng/relax-CHP-11-SECT-4.html
- http://www.w3.org/TR/SVG/extend.html
- http://www.google.com/search?q=XML+foreign+elements

1.9 Downloads

1.9.1 Current Release

Note: See *CHANGES* history below and repository tickets: http://www.cansas.org/trac/report/6?sort=ticket&asc=1

Version 1.1 released 2013-03-29, http://www.cansas.org/trac/ticket/23

```
svn checkout http://www.cansas.org/svn/1dwg/tags/v1.1 cansas1dwg-1.1
```

Documentation

HTML: http://www.cansas.org/formats/canSAS1d/1.1/doc/

PDF: http://www.cansas.org/svn/1dwg/tags/v1.1/doc/cansas-1d-1_1-manual.pdf

1.9.2 Older Releases

Version 1.0 released 2009-05-12 as no changes were committed since January 2009. Use this command to checkout the tagged release:

```
svn checkout http://www.cansas.org/svn/1dwg/tags/v1.0 cansas1dwg-1.0
```

Documentation

PDF: http://www.cansas.org/svn/1dwg/tags/v1.1/doc/cansas-1d-1_0-manual.pdf

1.10 CHANGES

1.10.1 v1.1

2013-03-29 http://www.cansas.org/trac/ticket/23

1.10.2 v1.0

2009-05-12 tagged version canSAS1d/1.0 (http://www.cansas.org/trac/changeset/68)

1.11 License

```
INFORMATION, DATA, APPARATUS, PRODUCT, OR PROCESS DISCLOSED, OR
REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS.
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


THIRD-PARTY LICENSES

THIS SOFTWARE MAY INCLUDE OR USE THIRD-PARTY SOFTWARE THAT HAVE OTHER
LICENSE AGREEMENTS, NOTICES, OR TERMS AND CONDITIONS.  THESE AGREEMENTS
SHOULD BE INCLUDED WITH ANY THIRD-PARTY PACKAGES THAT ARE DISTRIBUTED
WITH THIS SOFTWARE AND SHOULD ALSO BE AVAILABLE ON THE RESPECTIVE
THIRD-PARTY WEB SITES. IT IS YOUR OBLIGATION TO READ AND ACCEPT ALL SUCH
TERMS AND CONDITIONS PRIOR TO USE OF THE CONTENT.
```

1.12 Authors and Contributors

Pete Jemian <*jemian@anl.gov*>, Advanced Photon Source, Argonne, IL, USA

Peter Boesecke <*boesecke@esrf.eu*>, ESRF, Grenoble, France

Paul Butler <*butlerpd@udel.edu*>, CNRS, NIST, Gaithersburg, MD, USA

Ron Ghosh <*reghosh@gmail.com*>, London, UK

Maja Hellsing <*maja.hellsing@physics.uu.se*>, University of Uppsala, Uppsala, Sweden

Jan Ilavsky <*Ilavsky@aps.anl.gov*>, Advanced Photon Source, Argonne, IL, USA

Andrew Jackson <*andrew.jackson@esss.se*>, ESSS, Lund, Sweden

Stephen King <*stephen.king@stfc.ac.uk*>, ISIS, Rutherford Appleton Laboratory, Didcot, UK

Ken Littrell <*littrellkc@ornl.gov*>, HIFR, ORNL, Oak Ridge, TN, USA

Duncan McGillivray <*d.mcgillivray@auckland.ac.nz*>, University of Auckland, New Zealand

Andrew Nelson <*andyfaff@gmail.com*>, ANSTO, Lucas Heights NSW, Australia

Lionel Porcar <*porcar@ill.eu*>, ESRF, Grenoble, France

Adrian Rennie <*adrian.rennie@fysik.uu.se*>, University of Uppsala, Uppsala, Sweden

Tobias Richter <*Tobias.Richter@diamond.ac.uk*>, Diamond Light Source Ltd., Didcot, UK

Sarah Rogers <*sarah.rogers@stfc.ac.uk*>, ISIS, Rutherford Appleton Laboratory, Didcot, UK

Yuya Shinohara <*yuya@k.u-tokyo.ac.jp*>, University of Tokyo, Tokyo, Japan

Masaaki Sugiyama <*sugiyama@rri.kyoto-u.ac.jp*>, Kyoto Univeristy, Kyoto, Japan

---

This document was created March 29, 2013.

- *Downloads*
- *CHANGES*
- *License*
- *Authors and Contributors*

PYTHON MODULE INDEX

INDEX

Symbols

<any>, *see* element, {any}

A

absolute intensity, *see* intensity, absolute

B

C

E

Y