canSAS 1-D Data Format, v1.1

COLLABORATORS

	TITLE : canSAS 1-D Data Format, v1.1	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Pete R. Jemian	August 24, 2012	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
v1.0	2009-09-25	converted from mediawiki to DocBook/5.0	PRJ
v1.1	2012-09-01	v1.1 release	PRJ

Contents

List of Figures

List of Tables

Preface

canSAS 1-D Data Format, v1.0

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. It 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[1] describes the members, timelines, and current status. There is a discussion[2] page for some matters that preceded this revision.

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[3] differ, the XML Schema will be deemed to have the most correct description of the standard.

[1] http://www.smallangles.net/wgwiki/index.php/1D_Data_Formats_Working_Group
[2] http://www.smallangles.net/wgwiki/index.php/Talk:1D_Data_Formats_Working_Group
[3] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d.xsd

Chapter 1

Overview

1.1 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[1] 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.

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.

1.1.1 Status

Version 1.0 was tagged from the subversion repository on 2009-05-12 as no changes were committed since January 2009. Use this command to checkout the tagged release.

Example 1.1 Checkout tagged release from subversion repository.

```
svn checkout http://svn.smallangles.net/svn/canSAS/1dwg/tags/v1.0 cansas1dwg-1.0
```

Version 1.1 was tagged from the subversion repository on 2012-09-01 including changes indicated on the TRAC site. Tickets closed: 17, 18, 20, 21, 22. See http://svn.smallangles.net/trac/canSAS/report/6?sort=ticket&asc=1

Example 1.2 Checkout tagged release from subversion repository.

```
svn checkout http://svn.smallangles.net/svn/canSAS/1dwg/tags/v1.1 cansas1dwg-1.1
```

1.2 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.

[1] http://www.smallangles.net/canSAS

1.2.1 Overview

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 titled Example XML Data Files 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[2] page and its discussion[3] page. A glossary defining the details about each specific field (XPath string, XML elements and attributes) is provided.

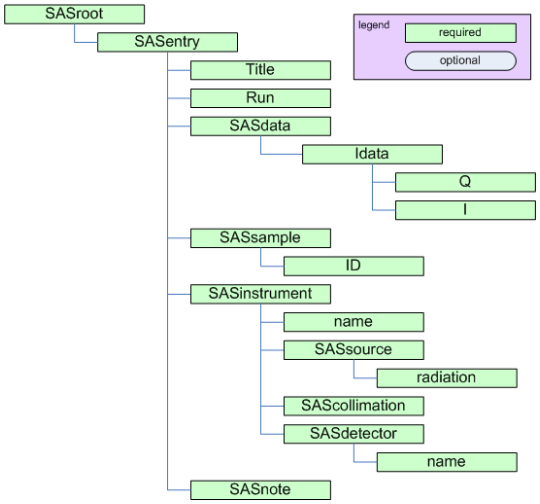


Figure 1.1: block diagram of minimum elements required for cansas1d/1.1 standard

- 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.)

Example 1.3 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://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansas1d.xsd">
```

Table 1.1: 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 `cansas1d/1.1` element can be used at this point
SASdata	this is where the reduced 1-D SAS data is stored
`Idata`	a single data point in the dataset

[2] http://www.smallangles.net/wgwiki/index.php/1D_Data_Formats_Working_Group
[3] http://www.smallangles.net/wgwiki/index.php/Talk:1D_Data_Formats_Working_Group

Table 1.1: (continued)

Element	Description
SAStransmission_spectrum	this is where any transmission spectra may be stored
Tdata	a single data point in the transmission spectrum
{any}	any cansas1d/1.1 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	for each processing or analysis step
SASnote	anything at all

1.3 Rules

1. cansas1d/1.1 XML data files will adhere to the standard if they can successfully validate against the established XML Schema (canSAS1d.xsd).
2. Q=(4 \pi / \lambda) \sin(\theta)

   where \lambda is the wavelength of the radiation

   and 2\theta is the angle through which the detected radiation has been scattered.

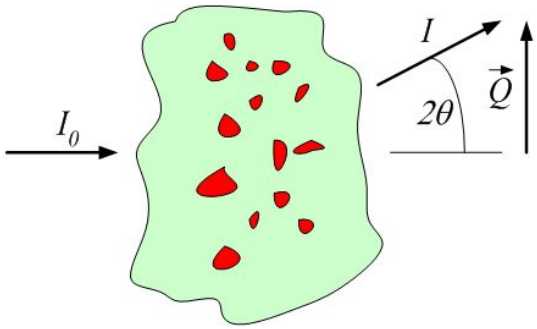
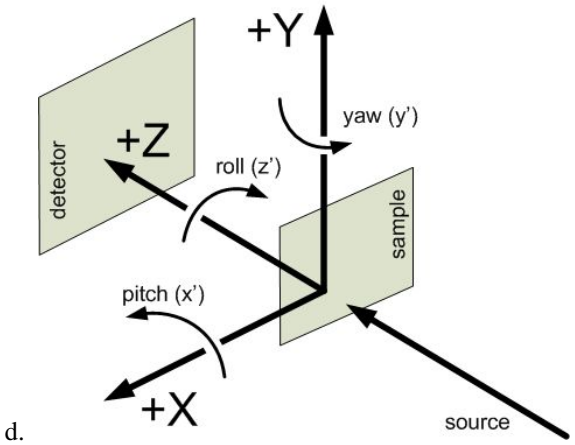


Figure 1.2: definition of Q geometry for small-angle scattering

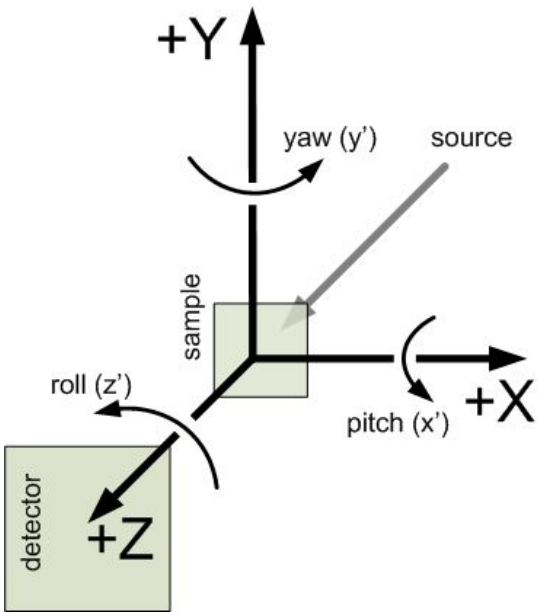
3. units to be given in standard SI abbreviations (eg, m, cm, mm, nm, K) with the following exceptions:
   a. um=micrometres
   b. C=celsius
   c. A=Angstroms
   d. percent=%.
   e. fraction
   f. a.u.=arbitrary units
   g. none=no units are relevant (such as dimensionless)
4. where reciprocal units need to be quoted the format shall be "1/abbreviation"
5. when raised to a power, use similar to "A^3" or "1/m^4" (and not "A3" or "A**3" or "m-4")
6. axes:

a. z is along the flight path (positive value in the direction of 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)

d.



as viewed from source



as viewed from detector

Figure 1.3: definition of translation and orientation geometry as viewed from the detector towards the source

7. orientation (angles) describes one-axis rotations (rotations about multiple axes require more information):
   a. roll is about z
   b. pitch is about x
   c. yaw is about y
8. Unicode characters MUST NOT be used

Note
Can UNICODE characters be permitted in v1.1 standard?

9. Binary data is not supported

1.4 Compatibility of Geometry Definitions

Note: translation and orientation geometry used by canSAS 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

The translation and orientation geometry definitions used here are different than those used by SHADOW[4] where the y and z axes are swapped and the direction of x is changed.

1.5 Converting data into the XML format

The *canSAS/xmlWriter*[5] 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.

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.

1.6 Documentation and Definitions

1.6.1 XML Schema

The cansas1d.xsd XML Schema[6] defines the rules for the XML file format[7][8] and is used to validate any XML file for adherence to the format.

1.6.2 XML Stylesheets

XML stylesheets (also known as XSLT)[9] can be used to extract metadata or to convert into another file format. The default canSAS stylesheet cansasxml-html.xsl[10] 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). (See the excellent write-up by Steve King, ISIS,[11] for an example.) 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) stylesheet in the same directory will produce a WWW page with the SAS data and selected metadata.

1.6.3 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 (command: 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

[4] http://www.nanotech.wisc.edu/shadow
[5] http://www.smallangles.net/canSAS/xmlWriter/
[6] http://www.w3schools.com/xsd
[7] TRAC: http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d.xsd
[8] SVN: http://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansas1d.xsd
[9] http://www.w3schools.com/xsl/
[10] http://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansasxml-html.xsl
[11] http://www.isis.rl.ac.uk/archive/LargeScale/LOQ/xml/cansas_xml_format.pdf

  - 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: 111 $
```

1.6.4 Examples and Case Studies

- Basic example:[12] 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:[13] Simulated SAS data (with added noise) calculated from model bimodal size distribution to test size distribution analysis routines.
- Glassy Carbon Round Robin:[14] Samples of a commercial glassy carbon measured at several facilities worldwide.
- SAXS data from dry chick collagen illustrates the minimum information necessary to meet the requirements of the standard format
- SANS data from AF1410 steel:[15] 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).
- `cansas1d-template.xml`:[16] This 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.

1.6.4.1 XML layout for multiple experiments

Each experiment is described with a single `SASentry` element. The fragment below shows how multiple experiments can be included in a single XML file. Full examples of canSAS XML files with multiple experiments include:

- ISIS LOQ SANS instrument:[17] multiple data sets.
- AF1410 steel SANS contrast variation study from NIST:[18] 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).

Here is a brief sketch of how a file would be arranged with multiple SASentry elements and multiple SASdata elements.

[12] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d.xml
[13] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/bimodal-test1.xml
[14] http://www.smallangles.net/wgwiki/index.php/Glassy_Carbon_Round_Robin
[15] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/examples/af1410/
[16] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d-template.xml
[17] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/W1W2.XML
[18] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/examples/af1410/cs_af1410.xml

Example 1.4 Brief sketch of a file with multiple SASentry and SASdata blocks.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="cansasxml-html.xsl" ?>
<SASroot version="1.1"
      xmlns="cansas1d/1.1"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="cansas1d/1.1 http://svn.smallangles.net/svn/canSAS/1dwg/trunk/ ↩
          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>
```

1.6.5 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 section XML Help for more help with XML foreign elements.

There is an example that demonstrates the use of a foreign namespace:[19] 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 for an example of arranging the content in `SASprocessnote` to avoid the use of foreign namespace elements.

1.6.6 Support tools for Visualization & Analysis software

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

[19] http://svn.smallangles.net/trac/canSAS/browser/1dwg/data/Glassy%20Carbon/ISIS/GLASSYC_C4G8G9_withTL.xml

- **FORTRAN**: See the section titled Fortran binding.
- **IgorPro**: See the section titled IgorPro binding.
- **Java**: See the section titled Java JAXB binding.
- **Microsoft Excel**: Support for Microsoft Excel is provided through the default canSAS stylesheet cansasxml-html.xsl. The ISIS LOQ instrument has provided an excellent description[20] of how to import data from the cansas1d/1.1 format into Excel. Also note that the old WWW site[21] may still be available.
- **PHP**: The canSAS/xmlWriter is implemented in PHP[22] and writes a cansas1d/1.1 data file given three-column ASCII data as input. (PHP source)[23] The code uses DomDocument[24] 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.
- **Python**: See the section titled Python binding.
- **XSLT** (useful in a web browser) is described later in the section titled Example XML Stylesheets.

1.6.7 Software repositories (for cansas1d/1.1 standard)

- **TRAC**: `http://svn.smallangles.net/trac/canSAS/browser/1dwg/tags/v1.0`
- **Subversion**: `http://svn.smallangles.net/svn/canSAS/1dwg/tags/v1.0`

1.7 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`[25] XSD file into form and press `<continue validation>`.
5. check the results

[20] http://www.isis.stfc.ac.uk/instruments/loq/loq2470.html
[21] http://www.isis.rl.ac.uk/LargeScale/LOQ/loq.htm
[22] http://www.php.net
[23] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/php/xmlWriter/index.php
[24] http://www.php.net/DomDocument
[25] http://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansas1d.xsd

Chapter 2

cansas1d/1.1 Specification

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.

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 `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>
```

2.1 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

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

Type

Type may be either of

header

Elements of type *header* describe the required XML header lines. Without questions, use the header in the section titled Required XML Header.

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

container

Elements of type *container* have subelements but no text for themselves. 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).

Description

Description provides useful information about this element.

Attributes

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

```
attributename="value"
```

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

2.1.1 Required XML Header

Example 2.1 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://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansas1d.xsd">
```

Table 2.1:

Name	Type	Occurrence	Description	Attributes
xml declaration	header	[1..1]	`<?xml version="1.0"?>`	`version="1.0"`
stylesheet	header	[0..1]	`<?xml-stylesheet type="text/xsl" href="example.xsl" ?>` Declares *example.xsl* (needs to be in the local directory) is the default stylesheet of an XML visualization tool. Change *example.xsl* to indicate a different stylesheet in the local directory.[3] [4]	`type="text/xsl" href="example.xsl"`

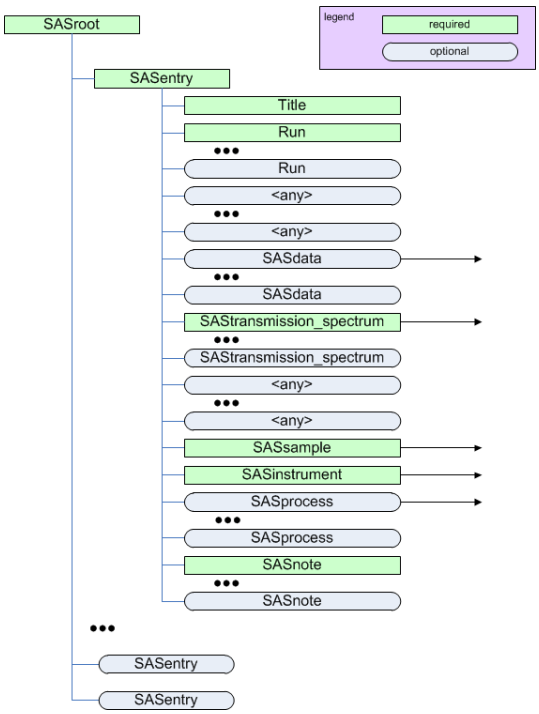
[2] http://www.w3schools.com/xml/xml_syntax.asp

Table 2.1: (continued)

SASroot	container	[1..1]	The canSAS reduced 1-D SAS data (cansas1d/1.1) will be in the `SASroot` database. (This is similar to `NXroot` used by NeXus.[5])	• `version="1.0"` is required to identify the cansas1d/1.1 standard for SAS data. • `xmlns` sets the default namespace URI for all elements (with no prefix) in this file. • `xmlns:xsi` sets `xsi` as the prefix for any elements from the governing XML Schema and defines the namespace URI to use with the `xsi:` element prefix. • `xsi:schemaLocation` associates a suggested URL (where the cansas1d/1.1 XML Schema might be found) with the default namespace string.

2.1.2 SASroot element

- parent: XML header



The SASroot element

Figure 2.1: The SASroot element

[3] Refer to `http://www.w3schools.org/xsl` W3 Schools XSLT Help for assistance in constructing XSLT files.
[4] XML rules actually allow for multiple stylesheet declarations. Explore this possibility as your own adventure.

[5] `http://www.nexusformat.org/NXroot`

Table 2.2:

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`

2.1.3 SASentry element

- parent: SASroot

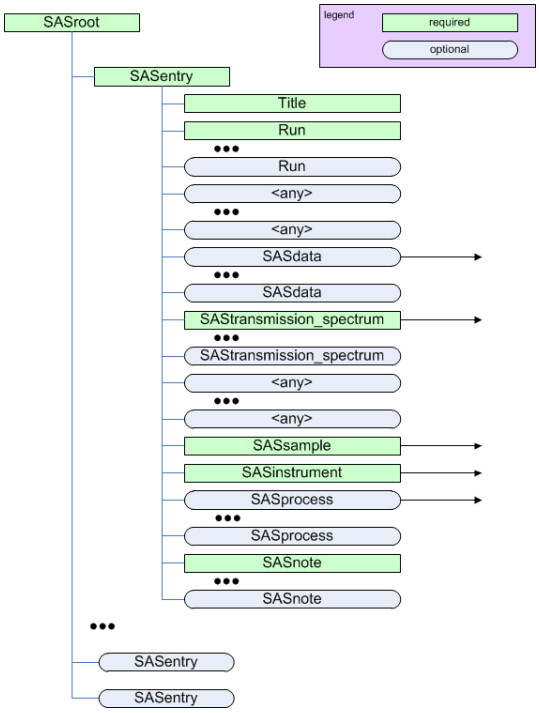


Figure 2.2: The SASentry element

Table 2.3:

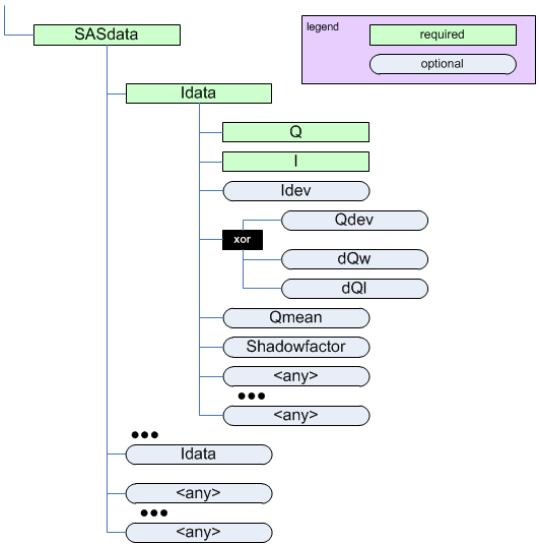
Name	Type	Occurrence	Description	Attributes
`Title`	string	[1..1]	Title of this `SASentry`.	

Table 2.3: (continued)

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}.)	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. See {any} for more details.	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. See {any} for more details.	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}

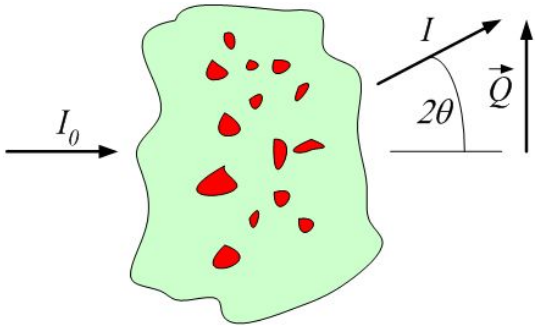
2.1.4 SASdata element

- parent: SASentry



The SASdata element

Figure 2.3: The SASdata element



Q geometry

Figure 2.4: Q geometry

2.1.4.1 SASdata

Table 2.4:

Name	Type	Occurrence	Description	Attributes
`Idata`	container	[1..inf]	Idata describes a single SAS data point.	

2.1.4.2 Idata

Table 2.5:

Name	Type	Occurrence	Description	Attributes

Table 2.5: (continued)

Q	floating-point number	[1..1]	$Q=(4 \pi / \lambda) \sin(\theta)$ where λ is the wavelength of the radiation and 2θ is the angle through which the detected radiation has been scattered. The unit attribute is required. See rules for units for acceptable values. Either 1/A or 1/nm are typical.	unit={unit}
I	floating-point number	[1..1]	Intensity of the detected radiation. The unit attribute is required. See the section about the rules for acceptable values. One possibility might be 1/cm for absolute units when the intensity describes a *differential cross-section per unit volume per unit solid angle*. Be aware that there are different types of intensity used in small-angle scattering that may be reported (see the section titled The Intensity Problem). 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.	unit={unit}
Idev	floating-point number	[0..1]	Estimated uncertainty (usually standard deviation) of I. The unit attribute is required. See rules for units for acceptable values. One possibility might be 1/cm.	unit={unit}
Qdev	floating-point number	[0..1]	Estimated uncertainty (usually standard deviation) of Q. (optional: see note below on usage) The unit attribute is required. See rules for units for acceptable values. Either 1/A or 1/nm are typical.	unit={unit}
dQw	floating-point number	[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. (optional: see note below on usage). The unit attribute is required. See rules for units for acceptable values. Either 1/A or 1/nm are typical.	unit={unit}
dQl	floating-point number	[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. (optional: see note below on usage) The unit attribute is required. See rules for units for acceptable values. Either 1/A or 1/nm are typical.	unit={unit}

Table 2.5: (continued)

`Qmean`	floating-point number	[0..1]	Mean value of `Q` for this datum. Useful when describing data that has been binned from higher-resolution or from area detectors.The `unit` attribute is required. See rules for units for acceptable values. Either `1/A` or `1/nm` are typical.	`unit={unit}`
`Shadowfactor`	floating-point number	[0..1]	Describes the adjustment due to the beam stop penumbra.	**Tip** There is no unit attribute.
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d/1.1 standard can be placed at this point. See {any} for more details.	xmlns:{foreign-prefix}={foreign-namespace}

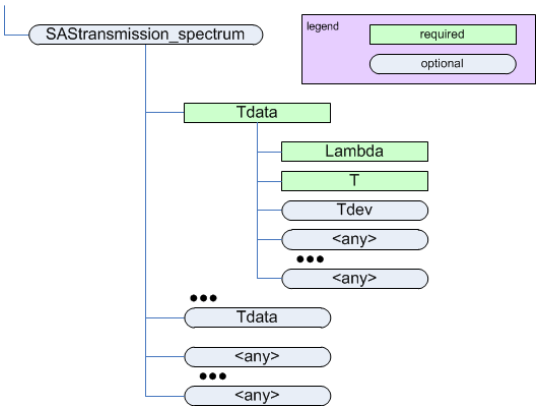
Note
When an optional element (`Idev`, `Qdev`, ...) is used, it must be given in every `Idata` within the enclosing `SASdata`.

Note
If either `dQw` or `dQl` are used, then `Qdev` is not permitted to be used.

Note
The `Shadowfactor` element definition needs revision. NIST?

2.1.5 SAStransmission_spectrum element

- parent: SASentry



The SAStransmission_spectrum element

Figure 2.5: 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:

Table 2.6:

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

2.1.5.1 SAStransmission_spectrum

Table 2.7:

Name	Type	Occurrence	Description	Attributes
`Tdata`	container	[1..inf]	Tdata describes a single transmission spectrum data point.	

2.1.5.2 Tdata

Table 2.8:

Name	Type	Occurrence	Description	Attributes
`Lambda`	floating-point number	[1..1]	Wavelength of the radiation bin. The `unit` attribute is required. See rules for units for acceptable values. Either `A` or `nm` are typical.	`unit={unit}`
`T`	floating-point number	[1..1]	Transmission value (I/Io).	`unit={unit}`
`Tdev`	floating-point number	[0..1]	Estimated uncertainty (usually standard deviation) of `T`. The `unit` attribute is required. See rules for units for acceptable values. One possibility might be `1/cm`.	`unit={unit}`
{any}	container	[0..inf]	Any element(s) not defined in the cansas1d/1.1 standard can be placed at this point. See {any} for more details.	xmlns:{foreign-prefix}={foreign-namespace}

Note
When an optional element (`Tdev`) is used, it must be given in every `Tdata` within the enclosing `SAStransmission_spectrum`.

2.1.6 SASsample element

- parent: SASentry

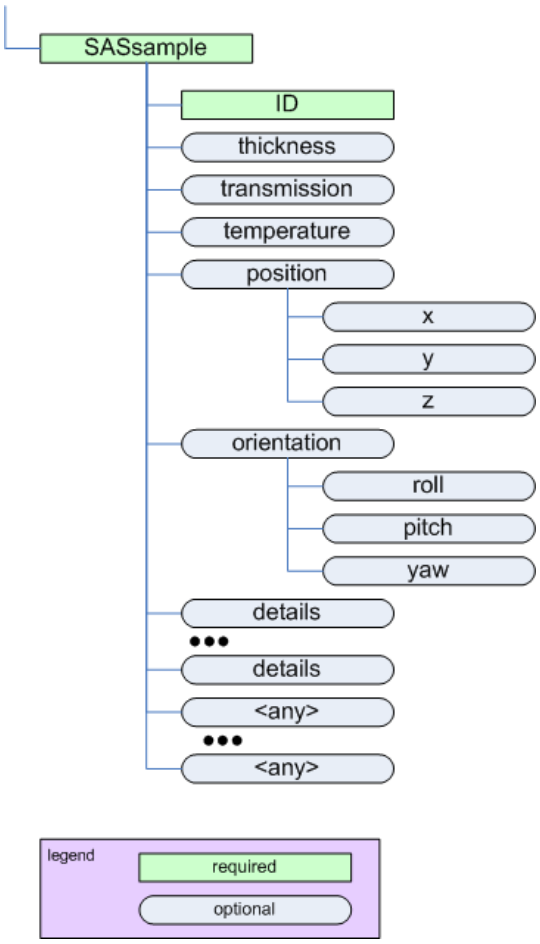
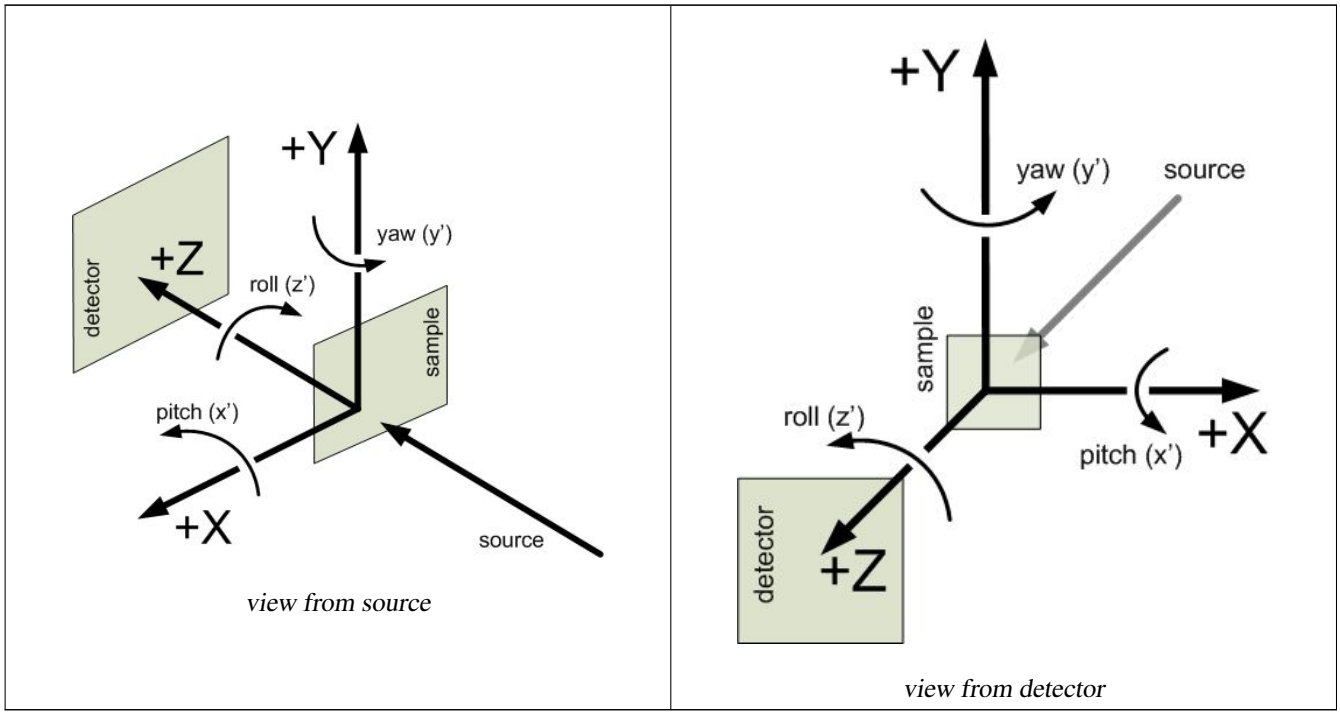


Figure 2.6: The SASsample element

Table 2.9:

Name	Type	Occurrence	Description	Attributes
`ID`	string	[1..1]	Text string that identifies this sample.	
`thickness`	floating-point number	[0..1]	Thickness of this sample. Must specify the unit as an attribute.	`unit={unit}`
`transmission`	floating-point number	[0..1]	Transmission (1-attenuation) of this sample. Express this as a fraction, not as a percentage. NOTE: there is no unit attribute.	
`temperature`	floating-point number	[0..1]	Temperature of this sample. Must specify the unit as an attribute.	`unit={unit}`
`position`	container	[0..1]	Location in X, Y, and Z of the sample.	
`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. See {any} for more details.	`xmlns:{foreign-prefix}=-{foreign-namespace}`

2.1.6.1 geometry



2.1.6.2 position

Table 2.10:

Name	Type	Occurrence	Description	Attributes
x	floating-point number	[0..1]	Position of the sample in X. The `unit` attribute is required. See the section about the rules for acceptable values.	`unit={unit}`
y	floating-point number	[0..1]	Position of the sample in Y. The `unit` attribute is required. See the section about the rules for acceptable values.	`unit={unit}`
z	floating-point number	[0..1]	Position of the sample in Z. The `unit` attribute is required. See the section about the rules for acceptable values. **Tip** While Z dimension is allowed by the standard (provided by use of a standard element in the XML Schema), it does not make sense for small-angle scattering.	`unit={unit}`

2.1.6.3 orientation

Table 2.11:

Name	Type	Occurrence	Description	Attributes

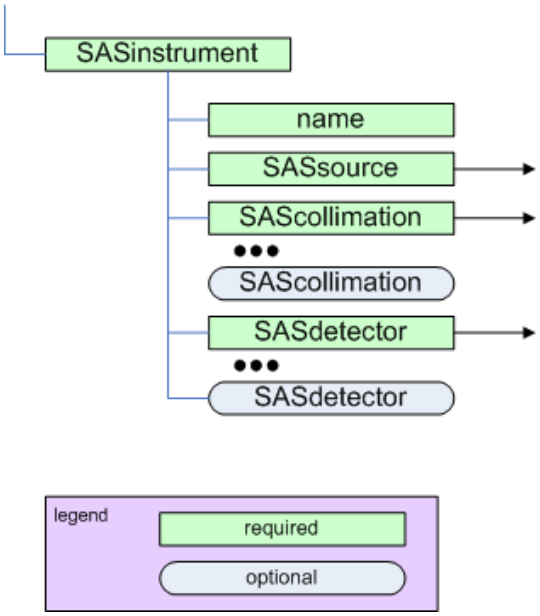
Table 2.11: (continued)

roll	floating-point number	[0..1]	Rotation about the Z axis (roll). The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}
pitch	floating-point number	[0..1]	Rotation about the X axis (pitch). The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}
yaw	floating-point number	[0..1]	Rotation about the Y axis (yaw). The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}

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.

2.1.7 SASinstrument element

- parent: SASentry



The SASinstrument element

Figure 2.7: The SASinstrument element

Table 2.12:

Name	Type	Occurrence	Description	Attributes
name	string	[1..1]	Text string that identifies the name of this instrument.	

Table 2.12: (continued)

SASsource	container	[1..1]	Text string that identifies the name of this source of radiation.	`name={name}`
SAScollimation	container	[1..inf]	Text string that identifies the name of this instrument collimation.	`name={name}`
SASdetector	container	[1..inf]	Text string that identifies the name of this detector.	

2.1.8 SASsource element

- parent: SASinstrument

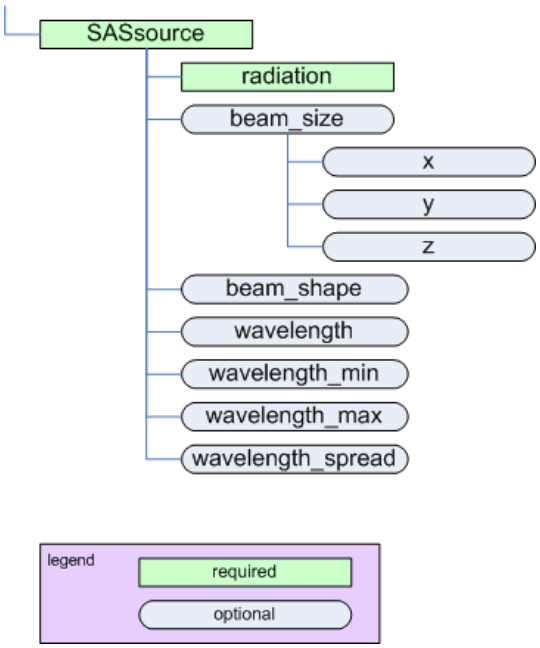


Figure 2.8: The SASsource element

Table 2.13:

Name	Type	Occurrence	Description	Attributes

Table 2.13: (continued)

radiation	string	[1..1]	Name of the radiation used. For maximum compatibility with NeXus, use one of the names defined by either *NeXus* NXsource/type • 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 *NeXus* NXsource/probe • neutron • x-ray • muon • electron	
beam_size	container	[0..1]	Physical dimension of the beam (incident on the sample). Note: If beam is round, just use X dimension. Note: While Z dimension is allowed by the standard, it does not make sense for small-angle scattering.	name={name}
beam_shape	string	[0..1]	Text description of the shape of the beam (incident on the sample).	
wavelength	floating-point number	[0..1]	wavelength (λ) of radiation incident on the sample.	unit={unit}
wavelength_min	floating-point number	[0..1]	Some facilities specify wavelength using a range. The minimum of such a range is given by wavelength_min.	unit={unit}
wavelength_max	floating-point number	[0..1]	Some facilities specify wavelength using a range. The maximum of such a range is given by wavelength_max.	unit={unit}
wavelength_spread	floating-point number	[0..1]	Some facilities specify the width of the wavelength spectrum. The width of such a range is given by wavelength_spread.	unit={unit}

2.1.8.1 beam_size

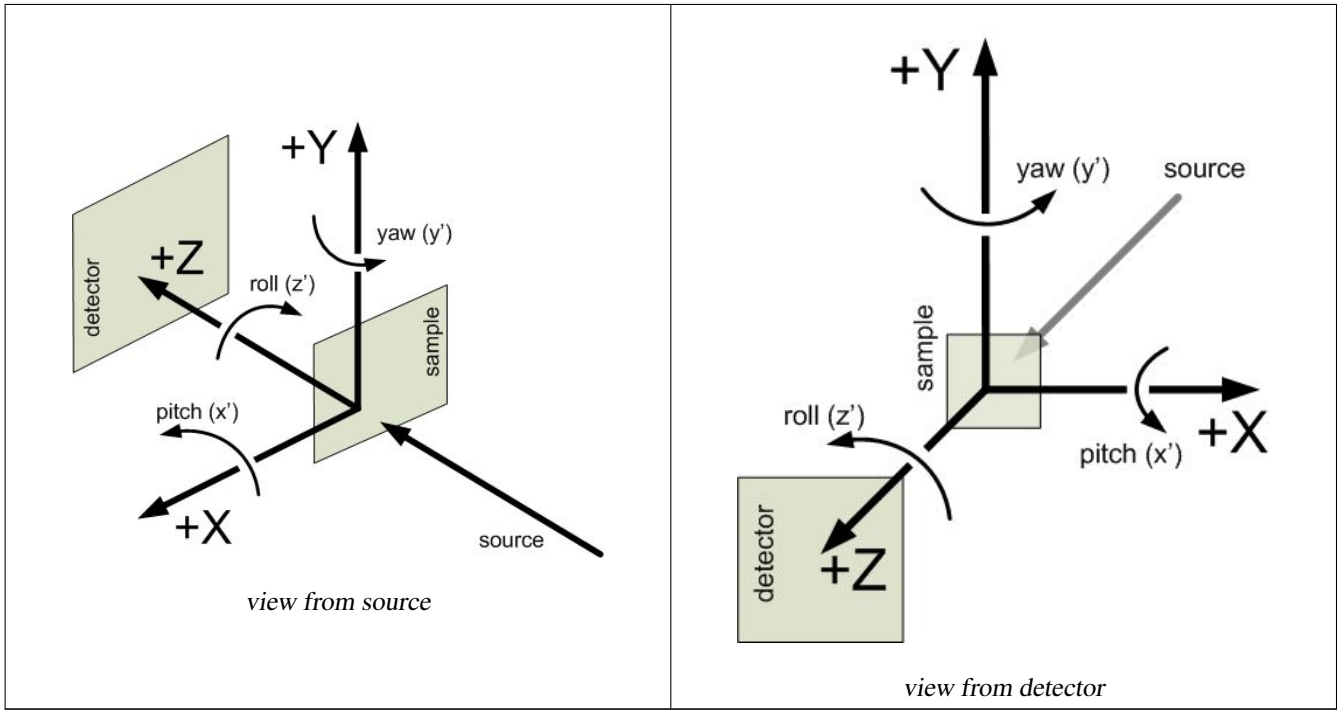
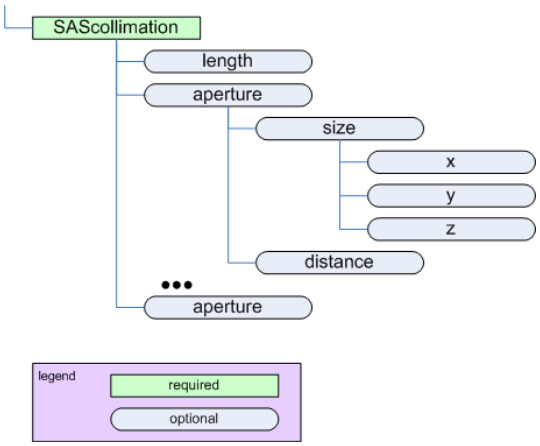


Table 2.14:

Name	Type	Occurrence	Description	Attributes
x	floating-point number	[0..1]	Dimension of the beam in X. The `unit` attribute is required. See the section about the rules for acceptable values.	`unit={unit}`
y	floating-point number	[0..1]	Dimension of the beam in Y. The `unit` attribute is required. See the section about the rules for acceptable values.	`unit={unit}`
z	floating-point number	[0..1]	Dimension of the beam in Z. The `unit` attribute is required. See the section about the rules for acceptable values. **Tip** While Z dimension is allowed by the standard (provided by use of a standard element in the XML Schema), it does not make sense for small-angle scattering.	`unit={unit}`

2.1.9 SAScollimation element

- parent: SASinstrument



The SAScollimation element

Figure 2.9: The SAScollimation element

2.1.9.1 SAScollimation

Table 2.15:

Name	Type	Occurrence	Description	Attributes
`length`	floating-point number	[0..1]	Amount/length of collimation inserted (on a SANS instrument)	`unit={unit}`
`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 aperture (pinhole, 4-blade slit, Soller slit, ...).	`name={type}`

2.1.9.2 aperture

Table 2.16:

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

2.1.9.3 size

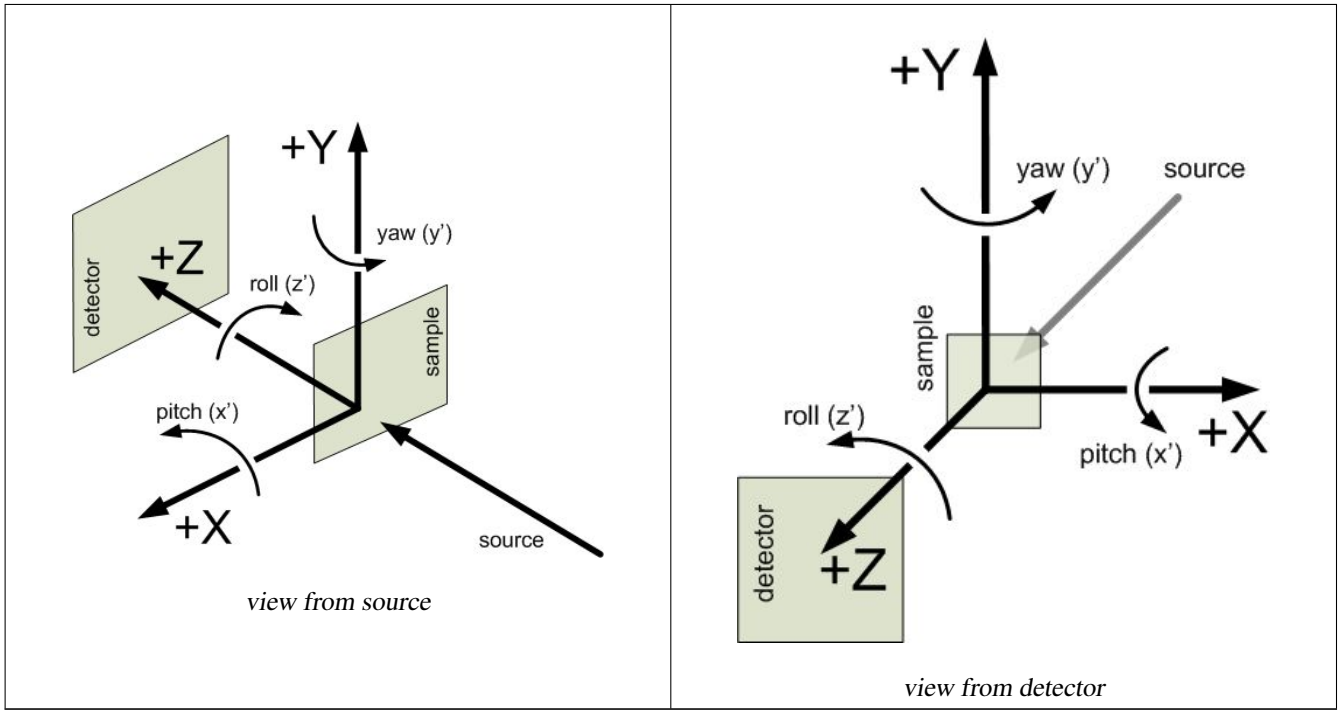
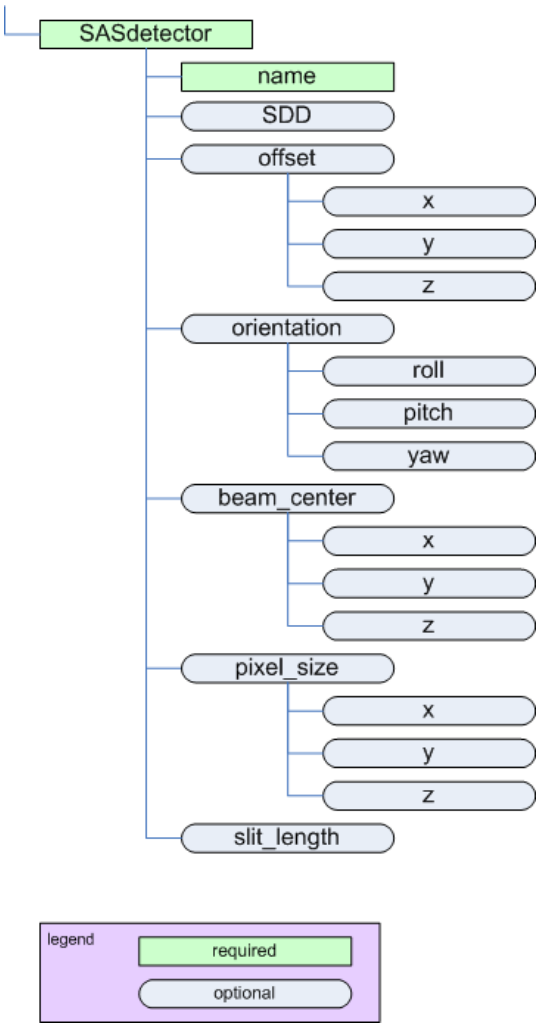


Table 2.17:

Name	Type	Occurrence	Description	Attributes
x	floating-point number	[0..1]	Dimension of the collimation in X. The `unit` attribute is required. See the section about the rules for acceptable values.	`unit={unit}`
y	floating-point number	[0..1]	Dimension of the collimation in Y. The `unit` attribute is required. See the section about the rules for acceptable values.	`unit={unit}`
z	floating-point number	[0..1]	Dimension of the collimation in Z. The `unit` attribute is required. See the section about the rules for acceptable values. **Tip** While Z dimension is allowed by the standard (provided by use of a standard element in the XML Schema), it does not make sense for small-angle scattering.	`unit={unit}`

2.1.10 SASdetector element

- parent: SASinstrument



The SASdetector element

Figure 2.10: The SASdetector element

2.1.10.1 SASdetector

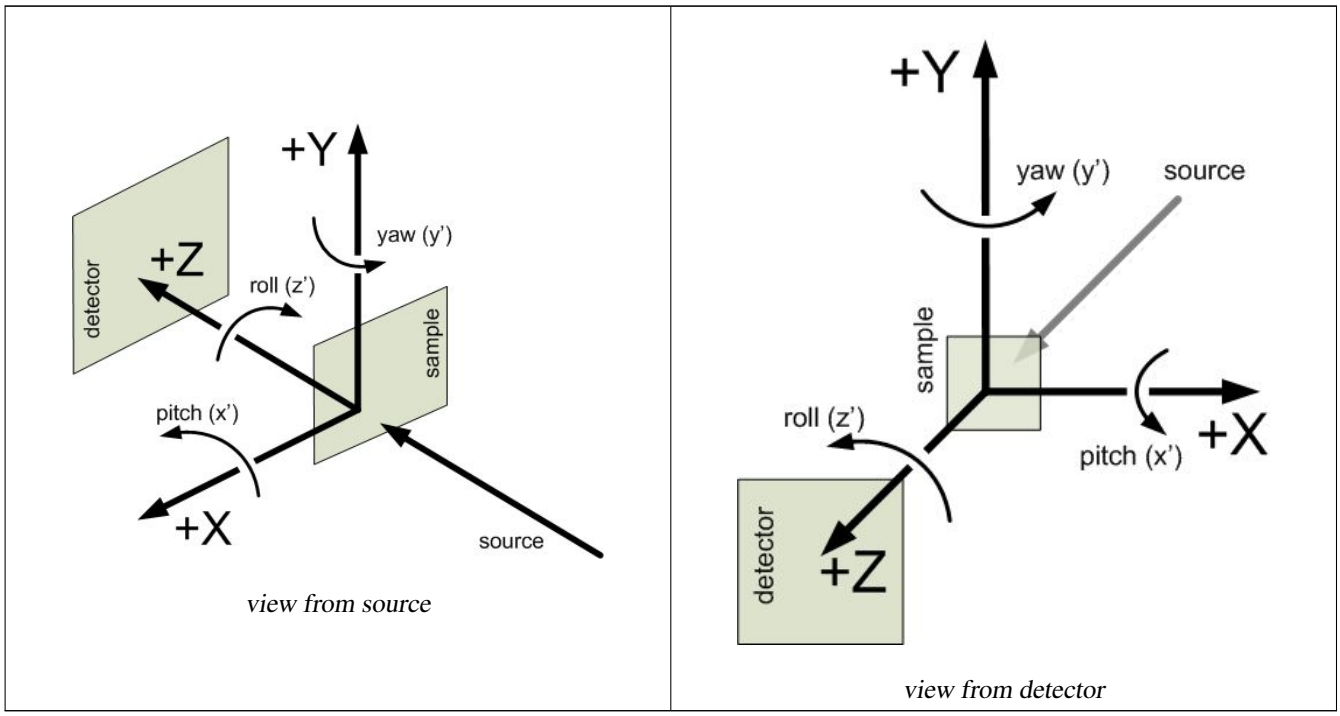
Table 2.18:

Name	Type	Occurrence	Description	Attributes
`name`	string	[1..1]	Text string that identifies the name of this detector.	
`SDD`	floating-point number	[0..1]	Distance between sample and detector.	`unit={unit}`
`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 and Y (and Z if necessary).	
`pixel_size`	container	[0..1]	Size of detector pixels in X and Y (and Z if necessary).	

Table 2.18: (continued)

slit_length	floating-point number	[0..1]	Slit length of the instrument for this detector. This is expressed in the same units as Q (reciprocal space units).	unit={unit}

2.1.10.2 geometry



2.1.10.3 offset

Table 2.19:

Name	Type	Occurrence	Description	Attributes
x	floating-point number	[0..1]	Offset of the detector position in X. The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}
y	floating-point number	[0..1]	Offset of the detector position in Y. The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}

Table 2.19: (continued)

z	floating-point number	[0..1]	Offset of the detector position in Z. The unit attribute is required. See the section about the rules for acceptable values. **Tip** While Z dimension is allowed by the standard (provided by use of a standard element in the XML Schema), it does not make sense for small-angle scattering.	unit={unit}

2.1.10.4 orientation

Table 2.20:

Name	Type	Occurrence	Description	Attributes
roll	floating-point number	[0..1]	Rotation about the Z axis (roll). The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}
pitch	floating-point number	[0..1]	Rotation about the X axis (pitch). The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}
yaw	floating-point number	[0..1]	Rotation about the Y axis (yaw). The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}

2.1.10.5 beam_center

Table 2.21:

Name	Type	Occurrence	Description	Attributes
x	floating-point number	[0..1]	Position of the beam center on the detector in X. The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}
y	floating-point number	[0..1]	Position of the beam center on the detector in Y. The unit attribute is required. See the section about the rules for acceptable values.	unit={unit}

Table 2.21: (continued)

z	floating-point number	[0..1]	Position of the beam center on the detector in Z. The `unit` attribute is required. See the section about the rules for acceptable values. **Tip** While Z dimension is allowed by the standard (provided by use of a standard element in the XML Schema), it does not make sense for small-angle scattering.	`unit={unit}`

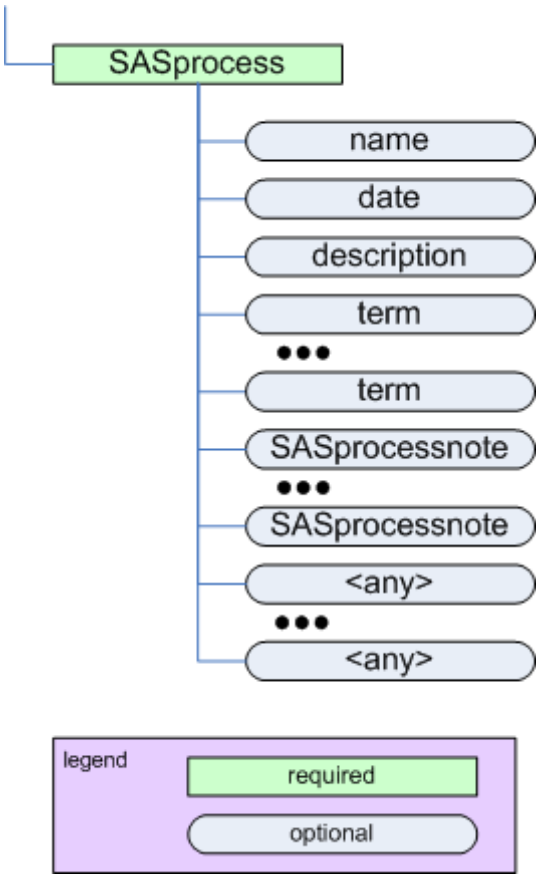
2.1.10.6 pixel_size

Table 2.22:

Name	Type	Occurrence	Description	Attributes
x	floating-point number	[0..1]	Size of a detector pixel in X. The `unit` attribute is required. See the section about the rules for acceptable values.	`unit={unit}`
y	floating-point number	[0..1]	Size of a detector pixel in Y. The `unit` attribute is required. See the section about the rules for acceptable values.	`unit={unit}`
z	floating-point number	[0..1]	Size of a detector pixel in Z. The `unit` attribute is required. See the section about the rules for acceptable values. **Tip** While Z dimension is allowed by the standard (provided by use of a standard element in the XML Schema), it does not make sense for small-angle scattering.	`unit={unit}`

2.1.11 SASprocess element

- parent: SASentry



The SASprocess element

Figure 2.11: The SASprocess element

Table 2.23:

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. Use a format which is easily machine-readable such as yyyy-mm-dd hh:mm:ss The format for the date string may be specified at a later date.	
`description`	string	[0..1]	Optional description for this data processing or analysis step.	
`term`	string	[0..inf]	This is used to specify 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}`
SASprocessnote	container	[1..inf]	This element is used to describe 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. See {any} for more details.	`xmlns:{foreign-prefix}={foreign-namespace}`

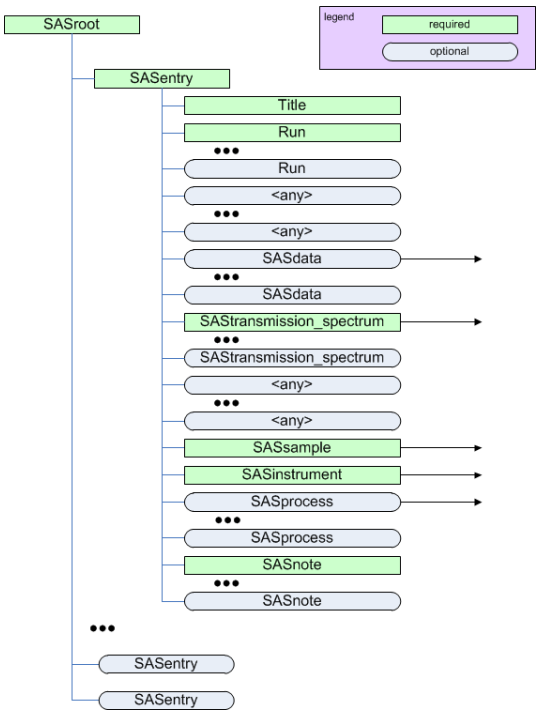
2.1.11.1 SASprocessnote element

Table 2.24:

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. See {any} for more details.	`xmlns:{foreign-prefix}=-{foreign-namespace}`

2.1.12 SASnote element

- parent: SASentry



The SASroot element

Figure 2.12: The SASroot element

Table 2.25:

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. See {any} for more details.	`xmlns:{foreign-prefix}=-{foreign-namespace}`

2.1.13 `{any}` element

Table 2.26:

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}`

Chapter 3

cansas1d/1.1 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, which can serve as examples. The opportunity is ripe for a better tutorial.

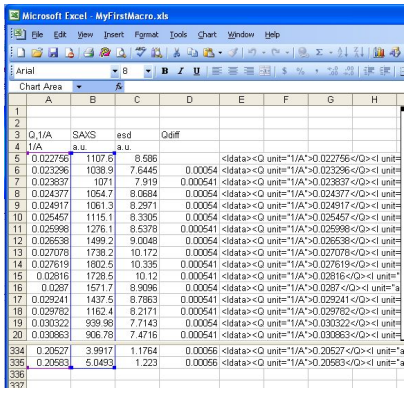
3.1 Case Studies

3.1.1 Case Study: Dry Chick Collagen

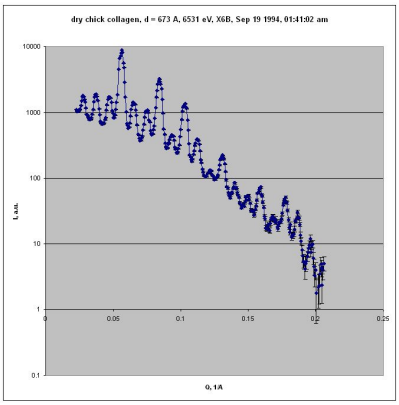
3.1.1.1 Overview

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.)



collagen SAXS in Excel table



collagen SAXS in Excel chart (log-log)

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 two

file headers: *collagen.asc* [1] and *collagen.saxs* [2] as shown.

Example 3.1 First few lines from file *collagen.asc*

```
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
```

Example 3.2 Full listing of file *collagen.saxs*

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

But, 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.

3.1.1.2 Procedure

3.1.1.2.1 make the basic XML file

It is easiest to copy a template rather than start from an empty file. Copy the *cansas1d.xml*[3] file into your working directory and rename it to `collagen.xml`.

3.1.1.2.2 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.

[1] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/examples/collagen/COLLAGEN.ASC
[2] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/examples/collagen/collagen.saxs
[3] http://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansas1d.xml

Example 3.3 `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="cansas1d/1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="cansas1d/1.1 http://svn.smallangles.net/svn/canSAS/1dwg/trunk/ ↩
        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>
```

3.1.1.2.3 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.

3.1.1.2.3.1 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.

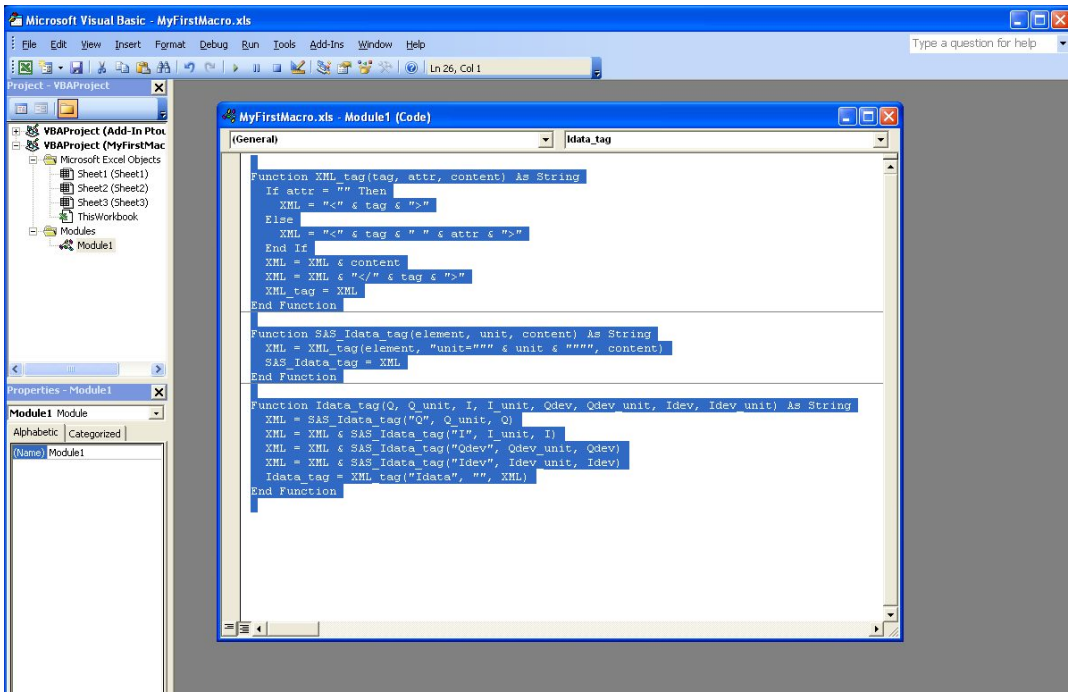
Example 3.4 Microsoft Excel macro to format the `Idata` lines.

```
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*)



case study: Collagen, SAXS data in Excel chart

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

3.1.1.2.3.2 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.

3.1.1.3 Final Result

A nicely-formatted display version of the final result can be viewed through the TRAC repository: *http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cs_collagen_full.xml*

3.1.1.4 Validate your file

So you think you have an XML file. Let's validate it using the procedure from the documentation.All the instructions are on the documentation page. No sense in repeating them here.

3.1.1.5 References

All files are available at *http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/examples/collagen/*

3.1.2 Case Study: AF1410 steel

3.1.2.1 Overview

Note

This case study has not yet been written up. For now, see the data file (http://svn.smallangles.net/svn/canSAS/1dwg/trunk/examples/af1410/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 magentic field to clear the scattering from magentic 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.

Chapter 4

Bindings and Software Support

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

4.1 Fortran binding

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* have to 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.

4.1.1 Software Development Kits

For later dialects, there are some SDK's 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/
- For F95:
  - XML - by Mart Rentmeester at http://nn-online.org/code/xml/

4.1.2 canSAS 1-D SAS XML v1.0 support

Steve King[mailto:s.m.king@rl.ac.uk] (ISIS) has provided a F77 routine (*SASXML_G77.F*) that will read CanSAS XML v1.0 files.

4.2 IgorPro binding

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 below.

Note
Note that this tool is not a true binding 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

cansasXML.ipf

author

Pete R. Jemian <jemian@anl.gov>

date

2009-09-02

version

1.11 (**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://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/IgorPro/cansasXML.ipf

SVN: http://svn.smallangles.net/svn/canSAS/1dwg/trunk/IgorPro/cansasXML.ipf

requires

IgorPro: http://www.wavemetrics.com `XMLutils XOP`[1] (minimum requirement: IGOR.5.04.x-1.x-dev, 2008-Aug-22)

4.2.1 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.

4.2.1.1 XMLutils XOP

The XMLutils XOP, written by Andrew Nelson (ANSTO), is hosted on the IgorExchange.[2]

One good location to place the checked out XMLutils directory is in the Wavemetrics directory, next to the Igor Pro Folder.[3]

In the future, to retrieve an updated version of this support, go into the XMLutils directory (created above) and type the command:

```
svn update
```

[1] http://www.igorexchange.com/project/XMLutils
[2] http://www.igorexchange.com/
[3] svn co svn://svn.igorexchange.com/packages/XMLutils/ XMLutils

(or just

```
svn up
```

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.

4.2.1.2 cansasXML.ipf

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

```
svn checkout http://svn.smallangles.net/svn/canSAS/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

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

```
svn update
```

This will check the repository and update files as needed.

4.2.2 Installation

1. License and Install IgorPro (should have already been done 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 you 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.[4]
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

4.2.3 Usage Notes

To use the `canSASxml.ipf` procedure, you must have the XMLutils XOP IGOR 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.

[4] http://svn.smallangles.net/svn/canSAS/1dwg/trunk/IgorPro/cansasXML.ipf

4.2.4 What it does

Given an XML file, **CS_XmlReader(fileName)** attempts to open the fileand 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` that indicates `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 XMLelements. 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 and 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.

4.2.4.1 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).

4.2.4.2 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 `cs_collagen_full.xml` case study:

Table 4.1: 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 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`

4.2.4.3 XML foreign namespace elements

These are ignored at this time.

4.2.4.4 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/lower case in the text **cansas1d/1.1** as this is a **key component** used toindex through the XML file.

4.2.4.5 XML stylesheet processing-instruction is not generated

The *XMLutils* 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.

4.2.5 List of Functions

These are (most of) the FUNCTIONS in the cansasXML.ipf code. The only functions of interest are **CS_XmlReader(fileName)** which reads the named XML file and and loads SAS data and the two demonstration functions **prj_grabMyXmlData()** and **prjTest_cansas1d()** that together show a usage example.

- **CS_XmlReader(fileName)**: open a canSAS 1-D reduced SAS XML data file
- input: *fileName* (string) name of canSAS XML file (can include file system path name to file)
- returns:
  - 0 successful
  - -1: XML file not found
  - -2: root element is not SASroot with valid canSAS namespace
  - -3: SASroot version is not 1.0
  - -4: no SASentry elements (NOT USED NOW)
  - -5: XOPutils needs upgrade
- CS_1i_parseXml(fileID): **This is what guides the work**, given a file ID returned from **XMLOpenFile()**, parses that file for SAS data and metadata (1i in the function name signifies this is a function that supports INPUT from version 1.0 XML files)
- CS_1i_getOneSASdata(fileID, Title, SASdataPath) : harvest the data and metadata in the specific SASdata element
- CS_1i_getOneVector(file,prefix,XML_name,Igor_name) : harvest just one column (vector) of data
- CS_1i_GetReducedSASdata(fileID, SASdataPath) : grab the data and put it in the working data folder
- CS_1i_locateTitle(fileID, SASentryPath) : determine the title for this experiment
- CS_appendMetaData(fileID, key, xpath, value) : queries XML file for **xpath**. If **value** is not empty, appends it to **metadata** where *last* is the new last row: metadata[last][0]=key; metadata[last][1]=value
- CS_buildXpathStr(prefix, value) : this function can be used only with very simple XPath constructions
- CS_cleanFolderName(proposal) : given a proposal string, returns a candidate folder name for immediate use

- CS_findElementIndex(matchStr) : looks for element index in structure *W_ElementList* returned from call to **XmlElemList(fileID)**
- CS_getDefaultNamespace(fileID) : returns the string containing the default namespace for the XML file
- CS_registerNameSpaces() : Builds a table of all namespaces used in the XML file and appends **W_ElementList** with full namespace-xpath string for each element.
- CS_simpleXmlListXpath(fileID, prefix, value) : Calls **XMLlistXpath()** with proper namespace prefix attached.
- CS_simpleXmlWaveFmXpath(fileID, prefix, value) : Calls **XMLwaveFmXpath()** with proper namespace prefix attached.
- CS_updateWaveNote(wavName, key, value) : adds (or replaces) definition of *key*=*value* in the wave note of *wavName*
- CS_XmlStrFmXpath(fileID, prefix, value) : Calls **XmlStrFmXpath()** with proper namespace prefix attached. Trims the result string.
- CS_XPath_NS(simpleStr) : this function adds namespace info as necessary to simpleStr (an XPath)
- TrimWS(str) : Calls **TrimWSL(TrimWSR(str))**
- TrimWSL(str) : Trims white space from left (leading) end of **str**
- TrimWSR(str) : Trims white space from right (trailing) end of **str**
- prjTest_cansas1d() : Demonstration function that calls **CS_XmlReader(fileName)** for many of the test data sets.
- prj_grabMyXmlData() : Demonstration function that moves loaded data from root:Packages:CS_XMLreader to a user's data folder. (In this *example*, that folder is root:PRJ_canSAS.)
- testCollette() : Demonstration function that reads an ISIS/LOQ file and copies the data to the root folder a la COLLETE

4.2.6 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
```

4.2.7 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

4.2.7.1 Irena tool suite

Jan Ilavsky's **Irena**[5] tool suite for IgorPro has a GUI to load the data found in the XML file. Refer to the WWW site for more details.

4.3 Java (JAXB) binding for the cansas1d/1.1 standard

A Java binding for the cansas1d/1.1 standard has been auto-created using the JAXB tools from Sun (see below for more on JAXB) using the `cansas1d.xsd` XML Schema. Resources (JAR files and documentation) for the Java binding may be found in the canSAS subversion repository [6]

`http://svn.smallangles.net/svn/canSAS/1dwg/tags/v1.0/java`

canSAS subversion repository tagged release directory for the Java binding. Use resources from this directory in your development projects.

`cansas1d-1.0.jar`

This is the JAR file to add to your CLASSPATH in order to use this binding.

`cansas1d-1.0-javadoc.jar`

Use this JAR file if you want to add the javadoc documentation as tooltips to your editor, such as eclipse. (auto-generated from the project source code using maven2) 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.

`cansas1d-1.0-sources.jar`

JAR file of the source code. (auto-generated from the project source code using maven2) Note that this is *just* the source code tree and not the full project development tree for the Java (JAXB) API.[7]

`cansas1d-1.0.pdf`

PDF file of the javadoc source code documentation. (auto-generated from the project source code using pdfdoclet)

`http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/java/maven/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.

4.3.1 Example_canSAS_Reader.java: example usage in JAVA

An example (Example_canSAS_Reader.java) has been constructed[8] to show how to read a cansas1d/1.1 XML file using the Java API.

In short, these are the important two lines:

```
CanSas1dType reader = new CanSas1dType();
```

and

```
SASrootType sasRoot = reader.open(xmlFile);
```

where `String xmlFile;` is the name of the XML file to be read. You will also need these imports

[5] http://usaxs.xor.aps.anl.gov/staff/ilavsky/irena.htm
[6] http://svn.smallangles.net/svn/canSAS/1dwg/tags/v1.0/java
[7] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/java/maven/eclipse/
[8] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/java/maven/eclipse/src/main/java/org/scatteringsw/reader/Example_canSAS_Reader.java

```
        import javax.xml.bind.JAXBException;

        import net.smallangles.cansas1d.CanSas1dType;
        import net.smallangles.cansas1d.SASdataType;
        import net.smallangles.cansas1d.SASentryType;
        import net.smallangles.cansas1d.SASrootType;
        import net.smallangles.cansas1d.SASentryType.Run;
```

Also, since `CanSas1dType.open(xmlFile)` can throw a `JAXBException`, you should use a `try{} catch {}` clause. See the source code for the example.

Here is a Java class that shows how to use the JAXB binding. Use this with any of the test data supplied with the cansas-1d-standard directory (above). By default, it shows the two samples in the `1998spheres.xml` example file.[9] (You'll have to get the directory paths right until this documentation improves.)

```
1  package org.scatteringsw.reader;
2
3  //############ SVN repository information ###################
4  //# $Date: 2009-11-29 22:37:21 -0600 (Sun, 29 Nov 2009) $
5  //# $Author: prjemian $
6  //# $Revision: 168 $
7  //# $URL: http://svn.smallangles.net/svn/canSAS/1dwg/trunk/doc/src/Example_canSAS_Reader. ↩
       java $
8  //# $Id: Example_canSAS_Reader.java 168 2009-11-30 04:37:21Z prjemian $
9  //############ SVN repository information ###################
10
11 import java.util.List;
12
13 import javax.xml.bind.JAXBException;
14
15 import net.smallangles.cansas1d.CanSas1dType;
16 import net.smallangles.cansas1d.SASdataType;
17 import net.smallangles.cansas1d.SASentryType;
18 import net.smallangles.cansas1d.SASrootType;
19 import net.smallangles.cansas1d.SASentryType.Run;
20
21 /**
22  * Demonstrate how to use the Java binding for the
23  * cansas1d/1.0 standard XML format for reduced
24  * small-angle scattering data.
25  *
26  */
27 public class Example_canSAS_Reader {
28
29   /**
30    * @param args
31    */
32   public static void main(String[] args) {
33     CanSas1dType reader = new CanSas1dType();
34
35     try {
36       // open the XML file and load contents into a Java data structure
37       SASrootType sasRoot = reader.open("java-test.xml");
38
39       // canSAS XML file is now loaded in memory
40       int numEntries = sasRoot.getSASentry().size();
41       System.out.println("SASentry elements: " + numEntries);
42
43       for (int i = 0; i < numEntries; i++ ) {
44         System.out.println("SASentry");
```

[9] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/1998spheres.xml

```
45
46          SASentryType sasEntry = sasRoot.getSASentry().get(i);
47          System.out.printf("Title:\t%s\n", sasEntry.getTitle());
48
49          List<SASentryType.Run> runs = sasEntry.getRun();
50          System.out.printf("#Runs:\t%d\n", runs.size());
51
52          for( int j = 0; j < runs.size(); j++ ) {
53            Run run = (Run) runs.get(j);
54            System.out.printf("Run@name:\t%s\n", run.getName());
55            System.out.printf("Run:\t%s\n", run.getValue());
56          }
57
58          List<SASdataType> datasets = sasEntry.getSASdata();
59          System.out.printf("#SASdata:\t%d\n", sasEntry.getSASdata().size());
60
61          for( int j = 0; j < runs.size(); j++ ) {
62            SASdataType sasData = (SASdataType) datasets.get(j);
63            System.out.printf("SASdata@name:\t%s\n", sasData.getName());
64            System.out.printf("#points:\t%d\n", sasData.getIdata().size());
65          }
66          System.out.println();
67        }
68
69        System.out.println("the end.");
70
71      } catch (JAXBException e) {
72        e.printStackTrace();
73        System.out.printf("Could not open (unmarshall) XML file: %s\n", xmlFile);
74      }
75    }
76
77  }
```

4.3.2 example: how to retrieve I(Q)

This is a slightly longer example. Look for the line (near line 75) that has

```
Qsas[i] = sdt.getIdata().get(i).getQ().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.

- sdt : SASdataType object
- getIdata() : /SASdata/Idata
- get(i) : /SASdata/Idata[i]
- getQ() : /SASdata/Idata/Q
- getValue() : /SASdata/Idata/Q (value, not the unit)

4.3.2.1 GetSASdata.java

```
1  /*
2   *   ############ SVN repository information ####################
3   *   # $Date: 2009-11-29 22:37:21 -0600 (Sun, 29 Nov 2009) $
4   *   # $Author: prjemian $
```

```
 *   # $Revision: 168 $
 *   # $HeadURL: http://svn.smallangles.net/svn/canSAS/1dwg/trunk/doc/src/GetSASdata.java $
 *   # $Id: GetSASdata.java 168 2009-11-30 04:37:21Z prjemian $
 *   ############ SVN repository information ###################
 */
package jlake;

import java.io.File;
import java.util.List;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBElement;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;

import net.smallangles.cansas1d.SASdataType;
import net.smallangles.cansas1d.SASdetectorType;
import net.smallangles.cansas1d.SASentryType;
import net.smallangles.cansas1d.SASinstrumentType;
import net.smallangles.cansas1d.SASrootType;
import net.smallangles.cansas1d.SASentryType.Run;


/**
 * Load the SAS data for desmearing by Jemian's ``lake'' program.
 */
public class GetSASdata {


  private static SASrootType sasRoot;   // SAS data (from cansas1d/1.0 XML file)
  private static double[] Qsas;     // input Q
  private static double[] Isas;     // input I (slit-smeared)
  private static double[] Idev;     // input Idev (slit-smeared)
  private static double[] Ismr;     // calculated I slit-smeared
  private static double[] Idsm;     // calculated I desmeared
  private static double[] IdsmDev;    // calculated Idev desmeared
  private static double   slit_length;


  /**
   * @param xmlPropertyFile
   */
  public GetSASdata(String xmlDataFile)
  {
    // load SAS data into memory
    try {
      sasRoot = (SASrootType) loadXML("cansas1d", xmlDataFile);
    } catch (JAXBException e) {
      e.printStackTrace();
      System.out.println("ERROR: Cannot find or interpret SAS XML data file:\t" +  ↩
          xmlDataFile);
      return;
    }

    // SAS data are loaded
    // grab the SAS data to be desmeared
    int entryIndex = 0; // /SASentry[1] : unit base in XML, 0 base in Java
    int dataIndex = 0;  // SASdata[1]
    int detectorIndex = 0;  // SASdetector[1]
    SASentryType entry = (SASentryType) sasRoot.getSASentry().get(entryIndex);
    SASdataType sdt = (SASdataType) entry.getSASdata().get(dataIndex);
    if (sdt.getName().trim().compareTo("slit-smeared") != 0) {
```

```
      System.out.println("selected SASdata element must start: <SASdata name=\"slit-smeared \">");
      // throw something (an exception) here?
      return;
    }
    int numPoints = sdt.getIdata().size();
    Qsas = new double[numPoints]; // input Q
    Isas = new double[numPoints]; // input I (slit-smeared)
    Idev = new double[numPoints]; // input Idev (slit-smeared)
    for (int i = 0; i < numPoints; i++) {
      Qsas[i] = sdt.getIdata().get(i).getQ().getValue();
      Isas[i] = sdt.getIdata().get(i).getI().getValue();
      Idev[i] = sdt.getIdata().get(i).getIdev().getValue();
    }
    Ismr = new double[numPoints];   // calculated I slit-smeared
    Idsm = new double[numPoints];   // calculated I desmeared
    IdsmDev = new double[numPoints];  // calculated Idev desmeared
    SASinstrumentType instrument = (SASinstrumentType) entry.getSASinstrument();
    SASdetectorType detector = (SASdetectorType) instrument.getSASdetector().get( detectorIndex);
    slit_length = detector.getSlitLength().getValue();
  }


  /**
   * @param (String) pkg Java package containing XML Schema bound to Java data structures
   * @param (String) xmlFile XML file to be opened
   * @return (Object) root object of Java data structure from XML file
   * @throws JAXBException
   */
  @SuppressWarnings("unchecked")
  private static Object loadXML(String pkg, String xmlFile) throws JAXBException {
    // use the $(pkg) schema that is bound to a Java structure
    JAXBContext jc = JAXBContext.newInstance(pkg);
    Unmarshaller unmarshaller = jc.createUnmarshaller();
    // open the XML file into a Java data structure
    Object obj = (Object) ((JAXBElement<Object>) unmarshaller
        .unmarshal(new File(xmlFile))).getValue();
    return obj;
  }


  /**
   * @param dt (DesmearingType) Desmearing properties
   * @param sasRoot (SASrootType) SAS data from XML file
   */
  public void inputReporter()
  {
    System.out.println("dataFile:\t" + dt.getDataFile().trim());
    System.out.printf("dataset selected:\t/SASroot/SASentry[%d]/SASdata[%d]\n",
        dt.getEntryIndex(), dt.getDataIndex());
    System.out.printf("detector selected:\t/SASroot/SASentry[%d]/SASinstrument/SASdetector [%d]\n",
        dt.getEntryIndex(), dt.getDataIndex(), dt.getDetectorIndex());
    System.out.println("extrapolation_form:\t" + dt.getExtrapolationForm().trim());
    System.out.println("x_start_extrapolation_evaluation:\t" + dt. getXStartExtrapolationEvaluation().getValue());
    System.out.println("x_start_extrapolation_evaluation unit:\t" + dt. getXStartExtrapolationEvaluation().getUnit());
    System.out.println("iterations:\t" + dt.getIterations());
    System.out.println("iterative_weight_method:\t" + dt.getIterativeWeightMethod().trim()) ;
```

```
122
123     System.out.println("#---------------------------------------");
124
125     int numEntries = sasRoot.getSASentry().size();
126     System.out.println("SASentry elements: " + numEntries);
127     for( int i = 0; i < numEntries; i++ ) {
128       System.out.println("SASentry");
129       SASentryType entry = sasRoot.getSASentry().get(i);
130       System.out.printf("Title:\t%s\n", entry.getTitle());
131       List<SASentryType.Run> runs = entry.getRun();
132       System.out.printf("#Runs:\t%d\n", runs.size());
133       for( int j = 0; j < runs.size(); j++ ) {
134         Run run = (Run) runs.get(j);
135         System.out.printf("Run@name:\t%s\n", run.getName());
136         System.out.printf("Run:\t%s\n", run.getValue());
137       }
138       List<SASdataType> datasets = entry.getSASdata();
139       System.out.printf("#SASdata:\t%d\n", datasets.size());
140       for( int j = 0; j < datasets.size(); j++ ) {
141         SASdataType sdt = (SASdataType) datasets.get(j);
142         System.out.printf("SASdata@name:\t%s\n", sdt.getName());
143         System.out.printf("#points:\t%d\n", sdt.getIdata().size());
144       }
145       List<SASdetectorType> detectors = entry.getSASinstrument().getSASdetector();
146       System.out.printf("#SASdetector:\t%d\n", detectors.size());
147       for( int j = 0; j < detectors.size(); j++ ) {
148         SASdetectorType det = (SASdetectorType) detectors.get(j);
149         System.out.printf("SASdata@name:\t%s\n", det.getName());
150         try {
151           System.out.printf("SDD:\t%g\t(%s)\n", det.getSDD()
152               .getValue(), det.getSDD().getUnit());
153         } catch (Exception e) {
154           System.out.println("SDD:\tundefined");
155         }
156         try {
157           System.out.printf("slit_length:\t%g\t(%s)\n", det
158               .getSlitLength().getValue(), det.getSlitLength()
159               .getUnit());
160         } catch (Exception e) {
161           System.out.println("slit_length:\tundefined");
162         }
163       }
164       System.out.println();
165     }
166   }
167
168   /**
169    * @param args
170    */
171   public static void main(String[] args) {
172     // load test desmearing properties and data into memory
173     GetSASdata sas = new GetSASdata("test.xml");
174     sas.inputReporter();
175     System.out.println("the end.");
176   }
177
178
179   /**
180    * @return the sasRoot
181    */
182   public SASrootType getSasRoot() {
183     return sasRoot;
```

```
184   }
185
186   /**
187    * @param sasRoot the sasRoot to set
188    */
189   public void setSasRoot(SASrootType sasRoot) {
190     GetDesmearingInfo.sasRoot = sasRoot;
191   }
192
193   /**
194    * @return the qsas
195    */
196   public double[] getQsas() {
197     return Qsas;
198   }
199
200   /**
201    * @param qsas the qsas to set
202    */
203   public void setQsas(double[] qsas) {
204     Qsas = qsas;
205   }
206
207   /**
208    * @return the isas
209    */
210   public double[] getIsas() {
211     return Isas;
212   }
213
214   /**
215    * @param isas the isas to set
216    */
217   public void setIsas(double[] isas) {
218     Isas = isas;
219   }
220
221   /**
222    * @return the idev
223    */
224   public double[] getIdev() {
225     return Idev;
226   }
227
228   /**
229    * @param idev the idev to set
230    */
231   public void setIdev(double[] idev) {
232     Idev = idev;
233   }
234
235   /**
236    * @return the ismr
237    */
238   public double[] getIsmr() {
239     return Ismr;
240   }
241
242   /**
243    * @param ismr the ismr to set
244    */
245   public void setIsmr(double[] ismr) {
```

```
246     Ismr = ismr;
247   }
248
249   /**
250    * @return the idsm
251    */
252   public double[] getIdsm() {
253     return Idsm;
254   }
255
256   /**
257    * @param idsm the idsm to set
258    */
259   public void setIdsm(double[] idsm) {
260     Idsm = idsm;
261   }
262
263   /**
264    * @return the idsmDev
265    */
266   public double[] getIdsmDev() {
267     return IdsmDev;
268   }
269
270   /**
271    * @param idsmDev the idsmDev to set
272    */
273   public void setIdsmDev(double[] idsmDev) {
274     IdsmDev = idsmDev;
275   }
276
277   /**
278    * @return
279    */
280   public double getSlitLength() {
281     return slit_length;
282   }
283
284 }
```

4.3.2.2 java-test.xml

`java-test.xml` is an example cansas1d/1.1 XML data file (derived from the standard test file for the `lake` desmearing code).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?xml-stylesheet type="text/xsl" href="example.xsl" ?>
3  <!--
4      ############ SVN repository information ####################
5      # $Date: 2012-08-11 17:30:57 -0500 (Sat, 11 Aug 2012) $
6      # $Author: prjemian $
7      # $Revision: 233 $
8      # $HeadURL: http://svn.smallangles.net/svn/canSAS/1dwg/trunk/doc/src/java-test.xml $
9      # $Id: java-test.xml 233 2012-08-11 22:30:57Z prjemian $
10     ############ SVN repository information ####################
11 -->
12 <SASroot version="1.1"
13     xmlns="cansas1d/1.1"
14     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
15     xsi:schemaLocation="cansas1d/1.0 http://svn.smallangles.net/svn/canSAS/1dwg/trunk/ ↩
           cansas1d.xsd"
```

```
    >
    <SASentry>
        <Title>(severely abbreviated version of the) standard test dataset for Lake desmearing routine</Title>
        <Run>Run</Run>
        <SASdata name="slit-smeared">
            <Idata><Q unit="1/A">0.000371484</Q><I unit="1/cm">211554</I><Idev unit="1/cm">1874.86</Idev></Idata>
            <Idata><Q unit="1/A">0.000510348</Q><I unit="1/cm">172441</I><Idev unit="1/cm">505.368</Idev></Idata>
            <Idata><Q unit="1/A">0.0006516</Q><I unit="1/cm">139041</I><Idev unit="1/cm">373.826</Idev></Idata>
            <Idata><Q unit="1/A">0.000798956</Q><I unit="1/cm">110665</I><Idev unit="1/cm">299.932</Idev></Idata>
            <Idata><Q unit="1/A">0.000931629</Q><I unit="1/cm">91165.3</I><Idev unit="1/cm">255.102</Idev></Idata>
            <Idata><Q unit="1/A">0.00107907</Q><I unit="1/cm">73855.7</I><Idev unit="1/cm">218.547</Idev></Idata>
            <Idata><Q unit="1/A">0.00121794</Q><I unit="1/cm">60799.2</I><Idev unit="1/cm">191.38</Idev></Idata>
            <Idata><Q unit="1/A">0.00168963</Q><I unit="1/cm">33573.8</I><Idev unit="1/cm">130.321</Idev></Idata>
            <Idata><Q unit="1/A">0.00227932</Q><I unit="1/cm">17602</I><Idev unit="1/cm">82.5591</Idev></Idata>
            <Idata><Q unit="1/A">0.00325199</Q><I unit="1/cm">7394.04</I><Idev unit="1/cm">50.0284</Idev></Idata>
            <Idata><Q unit="1/A">0.00608234</Q><I unit="1/cm">1305.37</I><Idev unit="1/cm">3.52884</Idev></Idata>
            <Idata><Q unit="1/A">0.00891261</Q><I unit="1/cm">464.949</I><Idev unit="1/cm">2.1601</Idev></Idata>
            <Idata><Q unit="1/A">0.0113893</Q><I unit="1/cm">241.462</I><Idev unit="1/cm">1.7191</Idev></Idata>
            <Idata><Q unit="1/A">0.0151631</Q><I unit="1/cm">126.664</I><Idev unit="1/cm">1.46166</Idev></Idata>
            <Idata><Q unit="1/A">0.0264845</Q><I unit="1/cm">53.7715</I><Idev unit="1/cm">1.27858</Idev></Idata>
            <Idata><Q unit="1/A">0.0378057</Q><I unit="1/cm">43.2662</I><Idev unit="1/cm">1.25507</Idev></Idata>
            <Idata><Q unit="1/A">0.0472402</Q><I unit="1/cm">39.9715</I><Idev unit="1/cm">1.25581</Idev></Idata>
            <Idata><Q unit="1/A">0.0566745</Q><I unit="1/cm">38.9685</I><Idev unit="1/cm">1.25924</Idev></Idata>
            <Idata><Q unit="1/A">0.0996003</Q><I unit="1/cm">38.8772</I><Idev unit="1/cm">1.27929</Idev></Idata>
            <Idata><Q unit="1/A">0.144883</Q><I unit="1/cm">37.5137</I><Idev unit="1/cm">1.30765</Idev></Idata>
            <Idata><Q unit="1/A">0.190162</Q><I unit="1/cm">39.1407</I><Idev unit="1/cm">1.33442</Idev></Idata>
            <Idata><Q unit="1/A">0.224119</Q><I unit="1/cm">38.589</I><Idev unit="1/cm">1.35581</Idev></Idata>
        </SASdata>
        <SASsample>
            <ID>ID</ID>
        </SASsample>
        <SASinstrument>
            <name>calculated</name>
            <SASsource>
                <radiation>model</radiation>
            </SASsource>
            <SAScollimation/>
            <SASdetector>
                <name>calculated</name>
```

```
                <slit_length unit="1/A">0.08</slit_length>
            </SASdetector>
        </SASinstrument>
        <SASnote/>
    </SASentry>
</SASroot>
```

4.3.3 JAXB

- Question : What is JAXB?
- Answer : Java Architecture for XML Binding (JAXB): [10]
- Wow! : Is it available for other languages?
- Answer : Ask Google. JAXB is for Java. (example) [11]
- Question : How do I pull out the *I(Q)* data?
- Answer : see fragment above (gets data for desmearing)
- Question : Has JAXB been useful?
- Answer : *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.

4.4 Python binding

Specific *support* for the cansas1d/1.1 data standard in Python is being developed by *NIST/NCNR* as part of their contribution to the *DANSE* project.

Here are some terse instructions to get you started:

Example 4.1

```
svn co http://danse.us/trac/sans/browser/trunk/DataLoader DataLoader
cd DataLoader
python setup.py install
```

The *release notes* have a list of the dependencies.

4.4.1 Comments

Other constructive suggestions (that predate the NIST/NCNR support) have been gathered on this page.

[10] http://java.sun.com/developer/technicalArticles/WebServices/jaxb/
[11] http://www.devx.com/ibm/Article/20261

4.4.2 gnosis.xml.objectify

The GnosisUtils [12] offer a method to read any XML file into Python data structures. This utility does not validate the XML against a specific XML Schema which can be both good (flexible, especially when XML Foreign Namespace elements are used) and not so good (XML content not guaranteed to be valid *by the rules*).

A quick test of this turned up an acceptable result in that it was able to read several of the canSAS test XML files, including those with foreign namespaces. And ***it was very easy.*** (Especially with some help from http://www.xml.com/pub/a/2003/07/02/py-xml.html)

Here is a quick example.

4.4.2.1 installation

Here is the condensed installation (without all that output) steps. Your system may have gnosis already installed. You may also need sysAdmin privileges. You may need ...

Example 4.2 Gnosis installation instructions

```
cd /tmp
wget http://freshmeat.net/redir/gnosisxml/22028/url_tgz/Gnosis_Utils-1.2.2.tar.gz
tar xzf Gnosis_Utils-1.2.2.tar.gz
cd Gnosis_Utils-1.2.2/
python setup.py install_all
```

4.4.2.2 quick test in Python

Here is the Python code (without all that output) (called `python-test.py`):

Example 4.3 python-test.py

```
import gnosis.xml.objectify

sasxml = gnosis.xml.objectify.XML_Objectify('bimodal-test1.xml').make_instance()
print sasxml.SASentry.Title.PCDATA
print sasxml.SASentry.Run.PCDATA
print sasxml.SASentry.SASinstrument.name.PCDATA
print sasxml.SASentry.SASdata.Idata[0].Q.unit,  sasxml.SASentry.SASdata.Idata[0].I.unit
print sasxml.SASentry.SASdata.Idata[0].Q.PCDATA,  sasxml.SASentry.SASdata.Idata[0].I.PCDATA ↩
    ,  sasxml.SASentry.SASdata.Idata[0].Idev.PCDATA
```

Example 4.4 output from `python-test.py`

```
[Pete@regitte,2441,cansas1dwg-regitte]$ ./python-test.py
SAS bimodal test1
1992
simulated SAS calculation
1/A 1/cm
0.0040157139 3497.473 90.72816
```

[12] http://freshmeat.net/projects/gnosisxml/

Example 4.5 full session output

```
[Pete@regitte,2429,/tmp]$ cd /tmp
/tmp
[Pete@regitte,2430,/tmp]$ wget http://freshmeat.net/redir/gnosisxml/22028/url_tgz/ ↩
    Gnosis_Utils-1.2.2.tar.gz
--11:43:16--  http://freshmeat.net/redir/gnosisxml/22028/url_tgz/Gnosis_Utils-1.2.2.tar.gz
           => 'Gnosis_Utils-1.2.2.tar.gz'
Resolving freshmeat.net... 66.35.250.168
Connecting to freshmeat.net|66.35.250.168|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://www.gnosis.cx/download/Gnosis_Utils.More/Gnosis_Utils-1.2.2.tar.gz [ ↩
    following]
--11:43:16--  http://www.gnosis.cx/download/Gnosis_Utils.More/Gnosis_Utils-1.2.2.tar.gz
           => 'Gnosis_Utils-1.2.2.tar.gz'
Resolving www.gnosis.cx... 64.41.64.172
Connecting to www.gnosis.cx|64.41.64.172|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 287,989 (281K) [application/x-tar]

100%[=========================================================================================================
     287,989       --.--K/s

11:43:16 (2.47 MB/s) - 'Gnosis_Utils-1.2.2.tar.gz' saved [287989/287989]

[Pete@regitte,2431,/tmp]$ tar xzf Gnosis_Utils-1.2.2.tar.gz
[Pete@regitte,2432,/tmp]$ cd Gnosis_Utils-1.2.2/
/tmp/Gnosis_Utils-1.2.2
[Pete@regitte,2433,Gnosis_Utils-1.2.2]$ python setup.py install_all
[Pete@regitte,2434,Gnosis_Utils-1.2.2]$ cd ~/workspace/cansas1dwg-regitte
[Pete@regitte,2435,cansas1dwg-regitte]$ python
Python 2.5.1 (r251:54863, May 18 2007, 16:56:43)
[GCC 3.4.4 (cygming special, gdc 0.12, using dmd 0.125)] on cygwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import gnosis.xml.objectify
>>> sasxml = gnosis.xml.objectify.XML_Objectify('bimodal-test1.xml').make_instance()
>>> print sasxml.SASentry.Title.PCDATA
SAS bimodal test1
>>> print sasxml.SASentry.Run.PCDATA
1992
>>> print sasxml.SASentry.SASinstrument.name.PCDATA
simulated SAS calculation
>>> print sasxml.SASentry.SASdata.Idata[0].Q.unit
1/A
>>> print sasxml.SASentry.SASdata.Idata[0].I.unit
1/cm
>>> print sasxml.SASentry.SASdata.Idata[0].Q.PCDATA,  sasxml.SASentry.SASdata.Idata[0].I. ↩
    PCDATA,  sasxml.SASentry.SASdata.Idata[0].Idev.PCDATA
0.0040157139 3497.473 90.72816
```

4.4.2.3 Conclusion: OK

This has the promise of being a useful approach to reading this format in Python. Now, how to write back out... ?

4.4.3 generateDS.py

generateDS.py (http://www.rexx.com/~dkuhlman/, http://www.rexx.com/~dkuhlman/generateDS.html) can build a binding (map the structure of the XML file directly into a Python data structure) for Python from an XML Schema. However, the cansas1d/1.1

XML schema (cansas1d.xsd) does not seem to fit the model.It seems, for now, that `generateDS-1.12a` fails on a certain `annotate` line.

```
[Pete@regitte,2402,cansas1dwg-regitte]$ python ~/generateDS-1.12a/generateDS.py -p CS1D_ -o ↩
    cansas1d.py -s cansas1dsubs.py cansas1d.xsd
Traceback (most recent call last):
  File "/home/Pete/generateDS-1.12a/generateDS.py", line 3997, in <module>
    main()
  File "/home/Pete/generateDS-1.12a/generateDS.py", line 3993, in main
    processIncludes, superModule=superModule)
  File "/home/Pete/generateDS-1.12a/generateDS.py", line 3909, in parseAndGenerate
    root.annotate()
AttributeError: 'NoneType' object has no attribute 'annotate'
```

4.4.3.1 Conclusion: not ready yet

Either the canSAS standard (by means of the cansas1d.xsd XML Schema) is not ready or *generateDS.py* does not cover the XML Schema requirements we have at this time.Either way, this is not a viable tool to use now. (2008-05-16)

4.4.4 Other suggestions

- http://www.devx.com/ibm/Article/20261
- http://mail.python.org/pipermail/xml-sig/2002-April/007559.html
- http://pywebsvcs.sourceforge.net/

Appendix A

The Intensity Problem

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

- **absolute units:** differential cross-section per unit volume per unit solid angle (typical unit: 1/cm)
- **absolute units:** differential cross-section per unit atom per unit solid angle (typical unit: cmˆ2)
- **arbitrary units:** 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 (such as no meaningful volume fraction or number density can be determined directly from I(Q)).

Appendix B

Examples

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

B.1 Example XML Data Files

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

B.1.1 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` example file in the main distribution. The stylesheet in `data-simple.xml` is very basic [1] data file.

[1] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/bimodal-test1.xml

Example B.1 *data-simple.xml*

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="ascii3col.xsl" ?>
<SASroot version="1.1"
    xmlns="cansas1d/1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="cansas1d/1.1 http://svn.smallangles.net/svn/canSAS/1dwg/trunk/ ↩
        cansas1d.xsd"
    >
  <SASentry>
    <Title>SAS bimodal test1</Title>
    <Run>1992</Run>
    <SASdata>
      <Idata><Q unit="1/A">0.0040157139</Q><I unit="1/cm">3497.473</I><Idev unit="1/cm"> ↩
          90.72816</Idev></Idata>
      <Idata><Q unit="1/A">0.0045408653</Q><I unit="1/cm">3340.003</I><Idev unit="1/cm"> ↩
          84.95314</Idev></Idata>
    </SASdata>
    <SASsample>
      <ID>bimodal-test1</ID>
    </SASsample>
    <SASinstrument>
      <name>simulated SAS calculation</name>
      <SASsource>
        <radiation>artificial</radiation>
        <wavelength unit="A">1.00</wavelength>
      </SASsource>
      <SAScollimation/>
      <SASdetector>
        <name>calculation</name>
      </SASdetector>
    </SASinstrument>
    <SASprocess>
      <name>create the SAS data</name>
      <date>1992-01-31</date>
      <term name="shape">spheres</term>
      <term name="contrast" unit="1/cm^4">100E20</term>
      <term name="Background" unit="1/cm">0.1</term>
      <term name="sMult" unit="cts/cm">1000.0</term>
      <term name="sNoise" unit="fraction">0.25</term>
      <SASprocessnote/>
    </SASprocess>
    <SASnote/>
  </SASentry>
</SASroot>
```

B.1.2 cansas1d.xml

The example data file `cansas1d.xml` shows examples of most of the elements of the cansas1d/1.1 standard. Only a single data point has been shown to more clearly show the other structure. [2]

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="cansasxml-html.xsl" ?>
<SASroot version="1.1"
  xmlns="cansas1d/1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="cansas1d/1.1 http://svn.smallangles.net/svn/canSAS/1dwg/trunk/ ↩
        cansas1d.xsd"
```

[2] http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d.xml

```
    >
  <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>
```

B.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.

B.2.1 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`.

Example B.2 *ascii3col.xsl*

```
<?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="cansas1d/1.0"
xsi:schemaLocation="cansas1d/1.0 http://svn.smallangles.net/svn/canSAS/1dwg/trunk/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>
```

B.2.2 cansasxml-html.xsl

The `cansasxml-html.xsl` is the standard XSL stylesheet for cansas1d/1.1 files. It shows all available `SASdata` and metadata, separated by the different `SASentry` blocks.

```
<?xml version="1.0"?>

<!--
############ SVN repository information ###################
# $Date: 2012-08-24 13:20:08 -0500 (Fri, 24 Aug 2012) $
# $Author: prjemian $
# $Revision: 236 $
# $HeadURL: http://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansasxml-html.xsl $
# $Id: cansasxml-html.xsl 236 2012-08-24 18:20:08Z prjemian $
############ SVN repository information ###################

Purpose:
  This stylesheet is used to translate cansas1d/1.0
  XML data files into a display form for viewing
```

```
  in a web browser such as Firefox or Internet Explorer.

Usage:
  xsltproc cansasxml-html.xsl datafile.xml > datafile.html
  (or include it as indicated at the documentation site
  http://www.smallangles.net/wgwiki/index.php/cansas1d_documentation)
-->

<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:cs="cansas1d/1.0"
  xmlns:fn="http://www.w3.org/2005/02/xpath-functions"
  xmlns:ues="urn:efficiency:spectrum"
  xmlns:ums="urn:monitor:spectrum"
  xmlns:uts="urn:transmission:spectrum"
  >

  <xsl:template match="/">
<!-- DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/ ↩
    xhtml1/DTD/xhtml1-transitional.dtd" -->
    <html>
      <head>
        <title>SAS data in canSAS 1-D format</title>
      </head>
      <body>
        <h1>SAS data in canSAS 1-D format</h1>
        <small>generated using <TT>cansasxml-html.xsl</TT> from canSAS</small>
        <BR />
        <table border="2">
          <tr>
            <th bgcolor="lavender">canSAS 1-D XML version:</th>
            <td><xsl:value-of select="cs:SASroot/@version" /></td>
          </tr>
          <tr>
            <th bgcolor="lavender">number of entries:</th>
            <td><xsl:value-of select="count(cs:SASroot/cs:SASentry)" /></td>
          </tr>
          <xsl:if test="count(/cs:SASroot//cs:SASentry)>1">
            <!-- if more than one SASentry, make a table of contents -->
            <xsl:for-each select="/cs:SASroot//cs:SASentry">
              <tr>
                <th bgcolor="lavender">SASentry-<xsl:value-of select="position()" /></th>
                <td>
                  <a href="#SASentry-{generate-id(.)}">
                    <xsl:if test="@name!=''">(<xsl:value-of select="@name" />)</xsl:if>
                    <xsl:value-of select="cs:Title" />
                  </a>
                </td>
                <xsl:if test="count(cs:SASdata)>1">
                  <td>
                    <!-- if more than one SASdata, make a local table of contents -->
                    <xsl:for-each select="cs:SASdata">
                      <xsl:if test="position()>1">
                        <xsl:text> | </xsl:text>
                      </xsl:if>
                      <a href="#SASdata-{generate-id(.)}">
                        <xsl:choose>
                          <xsl:when test="cs:name!=''">
                            <xsl:value-of select="cs:name" />
                          </xsl:when>
                          <xsl:when test="@name!=''">
                            <xsl:value-of select="@name" />
```

```
                        </xsl:when>
                        <xsl:otherwise>
                          SASdata<xsl:value-of select="position()" />
                        </xsl:otherwise>
                      </xsl:choose>
                    </a>
                  </xsl:for-each>
                </td>
              </xsl:if>
            </tr>
          </xsl:for-each>
        </xsl:if>
      </table>
      <xsl:apply-templates  />
      <hr />
      <small><center>$Id: cansasxml-html.xsl 236 2012-08-24 18:20:08Z prjemian $</center> ↩
          </small>
    </body>
  </html>
</xsl:template>

<xsl:template match="cs:SASroot">
  <xsl:for-each select="cs:SASentry">
    <hr />
    <br />
    <a id="#SASentry-{generate-id(.)}"  name="SASentry-{generate-id(.)}" />
    <h1>SASentry<xsl:value-of select="position()" />:<xsl:if
      test="@name!=''">(<xsl:value-of select="@name" />)</xsl:if>
      <xsl:value-of select="cs:Title" /></h1>
    <xsl:if test="count(cs:SASdata)>1">
      <table border="2">
        <caption>SASdata contents</caption>
        <xsl:for-each select="cs:SASdata">
          <tr>
            <th>SASdata-<xsl:value-of select="position()" /></th>
            <td>
              <a href="#SASdata-{generate-id(.)}">
                <xsl:choose>
                <xsl:when test="@name!=''">
                    <xsl:value-of select="@name" />
                  </xsl:when>
                  <xsl:otherwise>
                    SASdata<xsl:value-of select="position()" />
                  </xsl:otherwise>
                </xsl:choose>
              </a>
            </td>
          </tr>
        </xsl:for-each>
      </table>
    </xsl:if>
    <br />
    <table border="2">
      <tr>
        <th>SAS data</th>
        <th>Selected Metadata</th>
      </tr>
      <tr>
        <td valign="top">
          <xsl:apply-templates  select="cs:SASdata" />
          <xsl:apply-templates  select="cs:SASsample/ues:efficiency_ratio" />
          <xsl:apply-templates  select="cs:SASsample/ums:monitor_spectrum" />
```

```
            <xsl:apply-templates  select="cs:SASsample/uts:transmission_spectrum" />
          </td>
          <td valign="top">
            <table border="2">
              <tr bgcolor="lavender">
                <th>name</th>
                <th>value</th>
                <th>unit</th>
              </tr>
              <tr>
                <td>Title</td>
                <td><xsl:value-of select="cs:Title" /></td>
                <td />
              </tr>
              <tr>
                <td>Run</td>
                <td><xsl:value-of select="cs:Run" /></td>
                <td />
              </tr>
              <tr><xsl:apply-templates  select="run" /></tr>
              <xsl:apply-templates  select="cs:SASsample" />
              <xsl:apply-templates  select="cs:SASinstrument" />
              <xsl:apply-templates  select="cs:SASprocess" />
              <xsl:apply-templates  select="cs:SASnote" />
            </table>
          </td>
        </tr>
      </table>
    </xsl:for-each>
  </xsl:template>

  <xsl:template match="cs:SASdata">
    <a id="#SASdata-{generate-id(.)}"  name="SASdata-{generate-id(.)}" />
    <table border="2">
      <caption><xsl:if
        test="@name!=''"><xsl:value-of select="@name" /></xsl:if> (<xsl:value-of
        select="count(cs:Idata)" /> points)</caption>
      <tr bgcolor="lavender">
        <xsl:for-each select="cs:Idata[1]/*">
          <th>
            <xsl:value-of select="name()" />
            <xsl:if test="@unit!=''"> (<xsl:value-of select="@unit" />)</xsl:if>
          </th>
        </xsl:for-each>
      </tr>
      <xsl:for-each select="cs:Idata">
        <tr>
          <xsl:for-each select="*">
            <td><xsl:value-of select="." /></td>
          </xsl:for-each>
        </tr>
      </xsl:for-each>
    </table>
  </xsl:template>

  <xsl:template match="cs:SASsample">
    <tr>
      <td>SASsample</td>
      <td><xsl:value-of select="@name" /></td>
      <td />
    </tr>
    <xsl:for-each select="*">
```

```
      <xsl:choose>
        <xsl:when test="name()='position'">
          <xsl:apply-templates select="." />
        </xsl:when>
        <xsl:when test="name()='orientation'">
          <xsl:apply-templates select="." />
        </xsl:when>
        <xsl:when test="name()='efficiency_ratio'">
          <!-- xmlns="urn:efficiency:spectrum" -->
        </xsl:when>
        <xsl:when test="name()='monitor_spectrum'">
          <!-- xmlns="urn:monitor:spectrum" -->
        </xsl:when>
        <xsl:when test="name()='transmission_spectrum'">
          <!-- xmlns="urn:transmission:spectrum" -->
        </xsl:when>
        <xsl:otherwise>
          <tr>
            <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
            <td><xsl:value-of select="." /></td>
            <td><xsl:value-of select="@unit" /></td>
          </tr>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>

  <xsl:template match="cs:SASinstrument">
    <tr>
      <td>SASinstrument</td>
      <td><xsl:value-of select="cs:name" /></td>
      <td><xsl:value-of select="@name" /></td>
    </tr>
    <xsl:for-each select="*">
      <xsl:choose>
        <xsl:when test="name()='SASsource'"><xsl:apply-templates select="." /></xsl:when>
        <xsl:when test="name()='SAScollimation'"><xsl:apply-templates select="." /></ ↩
            xsl:when>
        <xsl:when test="name()='SASdetector'"><xsl:apply-templates select="." /></xsl:when>
        <xsl:when test="name()='name'" />
        <xsl:otherwise>
          <tr>
            <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
            <td><xsl:value-of select="." /></td>
            <td><xsl:value-of select="@unit" /></td>
          </tr>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>

  <xsl:template match="cs:SASsource">
    <tr>
      <td><xsl:value-of select="name()" /></td>
      <td><xsl:value-of select="@name" /></td>
      <td />
    </tr>
    <xsl:for-each select="*">
      <xsl:choose>
        <xsl:when test="name()='beam_size'"><xsl:apply-templates select="." /></xsl:when>
        <xsl:otherwise>
          <tr>
```

```
260             <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
261             <td><xsl:value-of select="." /></td>
262             <td><xsl:value-of select="@unit" /></td>
263           </tr>
264         </xsl:otherwise>
265       </xsl:choose>
266     </xsl:for-each>
267   </xsl:template>
268
269   <xsl:template match="cs:beam_size">
270     <tr>
271       <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
272       <td><xsl:value-of select="@name" /></td>
273       <td />
274     </tr>
275     <xsl:for-each select="*">
276       <tr>
277         <td><xsl:value-of select="name(../..)" />_<xsl:value-of select="name(..)" />_< ↩
              xsl:value-of select="name()" /></td>
278         <td><xsl:value-of select="." /></td>
279         <td><xsl:value-of select="@unit" /></td>
280       </tr>
281     </xsl:for-each>
282   </xsl:template>
283
284   <xsl:template match="cs:SAScollimation">
285     <xsl:for-each select="*">
286       <xsl:choose>
287         <xsl:when test="name()='aperture'"><xsl:apply-templates select="." /></xsl:when>
288         <xsl:otherwise>
289           <tr>
290             <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
291             <td><xsl:value-of select="." /></td>
292             <td><xsl:value-of select="@unit" /></td>
293           </tr>
294         </xsl:otherwise>
295       </xsl:choose>
296     </xsl:for-each>
297   </xsl:template>
298
299   <xsl:template match="cs:aperture">
300     <tr>
301       <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
302       <td><xsl:value-of select="@name" /></td>
303       <td><xsl:value-of select="@type" /></td>
304     </tr>
305     <xsl:for-each select="*">
306       <xsl:choose>
307         <xsl:when test="name()='size'"><xsl:apply-templates select="." /></xsl:when>
308         <xsl:otherwise>
309           <tr>
310             <td><xsl:value-of select="name(../..)" />_<xsl:value-of select="name(..)" />_< ↩
                  xsl:value-of select="name()" /></td>
311             <td><xsl:value-of select="." /></td>
312             <td><xsl:value-of select="@unit" /></td>
313           </tr>
314         </xsl:otherwise>
315       </xsl:choose>
316     </xsl:for-each>
317   </xsl:template>
318
319   <xsl:template match="cs:size">
```

```
320     <tr>
321       <td><xsl:value-of select="name(../..)" />_<xsl:value-of select="name(..)" />_< ↩
              xsl:value-of select="name()" /></td>
322       <td><xsl:value-of select="@name" /></td>
323       <td />
324     </tr>
325     <xsl:for-each select="*">
326       <tr>
327         <td><xsl:value-of select="name(../../..)" />_<xsl:value-of select="name(../..)" />_ ↩
                <xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
328         <td><xsl:value-of select="." /></td>
329         <td><xsl:value-of select="@unit" /></td>
330       </tr>
331     </xsl:for-each>
332   </xsl:template>
333
334   <xsl:template match="cs:SASdetector">
335     <tr>
336       <td><xsl:value-of select="name()" /></td>
337       <td><xsl:value-of select="cs:name" /></td>
338       <td><xsl:value-of select="@name" /></td>
339     </tr>
340     <xsl:for-each select="*">
341       <xsl:choose>
342         <xsl:when test="name()='name'" />
343         <xsl:when test="name()='offset'"><xsl:apply-templates select="." /></xsl:when>
344         <xsl:when test="name()='orientation'"><xsl:apply-templates select="." /></xsl:when>
345         <xsl:when test="name()='beam_center'"><xsl:apply-templates select="." /></xsl:when>
346         <xsl:when test="name()='pixel_size'"><xsl:apply-templates select="." /></xsl:when>
347         <xsl:otherwise>
348           <tr>
349             <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
350             <td><xsl:value-of select="." /></td>
351             <td><xsl:value-of select="@unit" /></td>
352           </tr>
353         </xsl:otherwise>
354       </xsl:choose>
355     </xsl:for-each>
356   </xsl:template>
357
358   <xsl:template match="cs:orientation">
359     <tr>
360       <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
361       <td><xsl:value-of select="@name" /></td>
362       <td />
363     </tr>
364     <xsl:for-each select="*">
365       <tr>
366         <td><xsl:value-of select="name(../..)" />_<xsl:value-of select="name(..)" />_< ↩
                xsl:value-of select="name()" /></td>
367         <td><xsl:value-of select="." /></td>
368         <td><xsl:value-of select="@unit" /></td>
369       </tr>
370     </xsl:for-each>
371   </xsl:template>
372
373   <xsl:template match="cs:position">
374     <tr>
375       <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
376       <td><xsl:value-of select="@name" /></td>
377       <td />
378     </tr>
```

```
379     <xsl:for-each select="*">
380       <tr>
381         <td><xsl:value-of select="name(../..)" />_<xsl:value-of select="name(..)" />_< ↩
              xsl:value-of select="name()" /></td>
382         <td><xsl:value-of select="." /></td>
383         <td><xsl:value-of select="@unit" /></td>
384       </tr>
385     </xsl:for-each>
386   </xsl:template>
387
388   <xsl:template match="cs:offset">
389     <tr>
390       <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
391       <td><xsl:value-of select="@name" /></td>
392       <td />
393     </tr>
394     <xsl:for-each select="*">
395       <tr>
396         <td><xsl:value-of select="name(../..)" />_<xsl:value-of select="name(..)" />_< ↩
              xsl:value-of select="name()" /></td>
397         <td><xsl:value-of select="." /></td>
398         <td><xsl:value-of select="@unit" /></td>
399       </tr>
400     </xsl:for-each>
401   </xsl:template>
402
403   <xsl:template match="cs:beam_center">
404     <tr>
405       <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
406       <td><xsl:value-of select="@name" /></td>
407       <td />
408     </tr>
409     <xsl:for-each select="*">
410       <tr>
411         <td><xsl:value-of select="name(../..)" />_<xsl:value-of select="name(..)" />_< ↩
              xsl:value-of select="name()" /></td>
412         <td><xsl:value-of select="." /></td>
413         <td><xsl:value-of select="@unit" /></td>
414       </tr>
415     </xsl:for-each>
416   </xsl:template>
417
418   <xsl:template match="cs:pixel_size">
419     <tr>
420       <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
421       <td><xsl:value-of select="@name" /></td>
422       <td />
423     </tr>
424     <xsl:for-each select="*">
425       <tr>
426         <td><xsl:value-of select="name(../..)" />_<xsl:value-of select="name(..)" />_< ↩
              xsl:value-of select="name()" /></td>
427         <td><xsl:value-of select="." /></td>
428         <td><xsl:value-of select="@unit" /></td>
429       </tr>
430     </xsl:for-each>
431   </xsl:template>
432
433   <xsl:template match="cs:term">
434     <tr>
435       <td><xsl:value-of select="@name" /></td>
436       <td><xsl:value-of select="." /></td>
```

```
437       <td><xsl:value-of select="@unit" /></td>
438     </tr>
439   </xsl:template>
440
441   <xsl:template match="cs:SASprocessnote" mode="standard">
442     <tr>
443       <td><xsl:value-of select="name()" /></td>
444       <td><xsl:value-of select="." /></td>
445       <td><xsl:value-of select="@name" /></td>
446     </tr>
447   </xsl:template>
448
449   <xsl:template match="cs:SASprocess">
450     <tr>
451       <td><xsl:value-of select="name()" /></td>
452       <td><xsl:value-of select="cs:name" /></td>
453       <td><xsl:value-of select="@name" /></td>
454     </tr>
455     <xsl:for-each select="*">
456       <xsl:choose>
457         <xsl:when test="name()='name'" />
458         <xsl:when test="name()='term'"><xsl:apply-templates select="." /></xsl:when>
459         <xsl:when test="name()='SASprocessnote'">
460           <xsl:choose>
461             <xsl:when test="../@name='Indra'">
462               <!-- Customization for APS USAXS metadata -->
463               <xsl:apply-templates select="." mode="Indra"/>
464             </xsl:when>
465             <xsl:otherwise><xsl:apply-templates select="." mode="standard"/></xsl:otherwise ↩
                >
466           </xsl:choose>
467         </xsl:when>
468         <xsl:otherwise>
469           <tr>
470             <td><xsl:value-of select="name(..)" />_<xsl:value-of select="name()" /></td>
471             <td><xsl:value-of select="." /></td>
472             <td />
473           </tr>
474         </xsl:otherwise>
475       </xsl:choose>
476     </xsl:for-each>
477   </xsl:template>
478
479   <xsl:template match="cs:SASnote">
480     <tr>
481       <td><xsl:value-of select="name()" /></td>
482       <td><xsl:value-of select="." /></td>
483       <xsl:if test="@name=''">
484         <td/>
485       </xsl:if>
486       <xsl:if test="@name!=''">
487         <td><xsl:value-of select="@name" /></td>
488       </xsl:if>
489     </tr>
490   </xsl:template>
491
492   <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->
493   <!-- +++++++++++  Customizations  ++++++++++++ -->
494   <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->
495
496   <xsl:template match="cs:SASprocessnote" mode="Indra">
497     <!--
```

```
498       Customization for APS USAXS metadata
499       These will be IgorPro wavenote variables
500     -->
501     <xsl:for-each select="cs:APS_USAXS">
502       <!-- ignore any other elements at this point -->
503       <tr>
504         <td bgcolor="lightgrey"><xsl:value-of select="name(..)" /></td>
505         <td bgcolor="lightgrey"><xsl:value-of select="name()" /></td>
506         <td bgcolor="lightgrey"><xsl:value-of select="@name" /></td>
507       </tr>
508       <xsl:for-each select="*">
509         <tr>
510           <td><xsl:value-of select="name()" /></td>
511           <td><xsl:value-of select="." /></td>
512           <td><xsl:value-of select="@name" /></td>
513         </tr>
514       </xsl:for-each>
515     </xsl:for-each>
516   </xsl:template>
517
518   <xsl:template match="ues:efficiency_ratio">
519     <!--
520       Customization for ISIS wavelength-dependent data
521     -->
522     <table border="2">
523       <caption>
524         wavelength-dependent detector efficiency spectrum
525       </caption>
526       <tr bgcolor="lavender">
527         <xsl:for-each select="ues:data[1]/*">
528           <th>
529             <xsl:value-of select="name()" />
530             <xsl:if test="@unit!=''"> (<xsl:value-of select="@unit" />)</xsl:if>
531           </th>
532         </xsl:for-each>
533       </tr>
534       <xsl:for-each select="ues:data">
535         <tr>
536           <xsl:for-each select="*">
537             <td><xsl:value-of select="." /></td>
538           </xsl:for-each>
539         </tr>
540       </xsl:for-each>
541     </table>
542   </xsl:template>
543
544   <xsl:template match="ums:monitor_spectrum">
545     <!--
546       Customization for ISIS wavelength-dependent data
547     -->
548     <table border="2">
549       <caption>
550         wavelength-dependent monitor spectrum
551       </caption>
552       <tr bgcolor="lavender">
553         <xsl:for-each select="ums:data[1]/*">
554           <th>
555             <xsl:value-of select="name()" />
556             <xsl:if test="@unit!=''"> (<xsl:value-of select="@unit" />)</xsl:if>
557           </th>
558         </xsl:for-each>
559       </tr>
```

```
      <xsl:for-each select="ums:data">
        <tr>
          <xsl:for-each select="*">
            <td><xsl:value-of select="." /></td>
          </xsl:for-each>
        </tr>
      </xsl:for-each>
    </table>
  </xsl:template>

  <xsl:template match="uts:transmission_spectrum">
    <!--
      Customization for ISIS wavelength-dependent data
    -->
    <table border="2">
      <caption>
        wavelength-dependent transmission spectrum
      </caption>
      <tr bgcolor="lavender">
        <xsl:for-each select="uts:data[1]/*">
          <th>
            <xsl:value-of select="name()" />
            <xsl:if test="@unit!=''"> (<xsl:value-of select="@unit" />)</xsl:if>
          </th>
        </xsl:for-each>
      </tr>
      <xsl:for-each select="uts:data">
        <tr>
          <xsl:for-each select="*">
            <td><xsl:value-of select="." /></td>
          </xsl:for-each>
        </tr>
      </xsl:for-each>
    </table>
  </xsl:template>

</xsl:stylesheet>
```

Appendix C

Glossary

Note
This Glossary might not be provided in future versions of this manual.

This section provides a glossary defining the details about each specific field (XPath string, XML elements and attributes) in the cansas1d/1.1 standard.

- Each term (element or attribute) is listed by its *XPath* in the XML file. The `cs:` prefix is defined by a `xmlns:cs="cansas1d/1.1"` namespace attribute listed in the XML header.
- Elements are shown below sorted by their XPath. In the XML file, the order of elements is defined by the XML Schema. An example is given in the file: `cansas1d-template.xml`
- Each term in the standard is shown with a comment embedded.
- The comment indicates
  - the name of the element,
  - how many times the element can be used,
    - [0..1] : element is optional but can only appear once within enclosing element
    - [1..1] : element is required and can only appear once within enclosing element
    - [1..inf] : element is required but can appear as many times as needed within enclosing element
    - [0..inf] : element is optional and can appear as many times as needed within enclosing element
    - [] : element is optional, number of appearances within enclosing element is not specified
  - and a short description.
- When shown in the template below with the /@unit XPath, the unit attribute is required.

C.1 Listed by full XPath reference

`/cs:SASroot`

[1..1] The canSAS reduced 1-D SAS data will be in the SASroot database. This is similar to the root element of a NeXus file (`NXroot`).

`/cs:SASroot/@version`

[1..1] version="1.0" Required attribute to indicate the version of the standard to which this XML document is encoded.

/cs:SASroot/cs:SASentry

[1..inf] A single SAS scan is reported in a SASentry. This is similar to `NXentry` used by NeXus. A `SASentry` can use an optional `name` attribute to provide a string for this SASentry.

/cs:SASroot/cs:SASentry/@name

[0..1] Optional string attribute to identify this particular `SASentry`. Use of the string associated with the `name` attribute is not defined by this standard.

/cs:SASroot/cs:SASentry/<any>

[0..inf] Provision at this point for any element to be entered that is not part of the canSAS standard. Declare a default namespace declaration such as `xmlns="urn:example-namespace:example-context"` to identify that this is a foreign element (where you replace "`example-namespace`" and "`example-context`" with your own terms).

/cs:SASroot/cs:SASentry/cs:Run

[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 SASinstrument configurations has not yet been defined.

/cs:SASroot/cs:SASentry/cs:Run/@name

[0..1] Optional string attribute to identify this particular SASrun. Use this to associate (correlate) multiple SASdata elements with Run elements. (Give them the same name.)

/cs:SASroot/cs:SASentry/cs:SASdata

[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.)

/cs:SASroot/cs:SASentry/cs:SASdata/@name

[0..1] Optional string attribute to identify this particular SASdata.

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata

[1..inf] Idata describes a single SAS data point.

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:dQl

[0..1] Q resolution perpendicular to the axis of scanning (the low-resolution slit length direction).

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:dQl/@unit

[1..1] Required unit for dQl. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:dQw

[0..1] Q resolution along the axis of scanning (the high-resolution slit width direction).

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:dQw/@unit

[1..1] Required unit for dQw. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:I

[1..1] Intensity of the detected radiation.

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:I/@unit

[1..1] Required unit for I. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:Idev

[0..1] Estimated uncertainty (usually standard deviation) of I. Must specify the unit as an attribute.

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:Idev/@unit

[1..1] Required unit for Idev. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:Q

[1..1] Q = (4 pi / lambda) sin(theta) where lambda is the wavelength of the radiation and 2theta is the angle through which the detected radiation has been scattered.

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:Q/@unit

[1..1] Required unit for Q. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:Qdev

[0..1] Estimated uncertainty (usually standard deviation) of Q. Must specify the unit as an attribute.

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:Qdev/@unit

[1..1] Required unit for Qdev. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:Qmean

[0..1] Mean value of Q for this datum. Must specify the unit as an attribute.

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:Qmean/@unit

[1..1] Required unit for Qmean. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/cs:Shadowfactor

[0..1] Describes the adjustment due to the beam stop penumbra.

Note
The use of Shadowfactor is limited and not generally expected in files except those from the NIST NCNR.

/cs:SASroot/cs:SASentry/cs:SASdata/cs:Idata/<any>

[0..inf] Provision at this point for any element to be entered that is not part of the canSAS standard. Declare a default namespace declaration such as `xmlns="urn:example-namespace:example-context"` to identify that this is a foreign element (where you replace "`example-namespace`" and "`example-context`" with your own terms).

/cs:SASroot/cs:SASentry/cs:SASinstrument

[1..1] Description of the instrument.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:name

[1..1] Name of the instrument.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation

[1..inf] Description of the instrument collimation.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/@name

[0..1] Optional text to describe this collimation element.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture

[0..inf] Slit or aperture.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/@name

[0..1] Optional name for this aperture.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/@type

[1..1] Optional text to describe the type aperture (pinhole, 4-blade slit, Soller slit, ...).

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:distance

[0..1] Distance from this collimation element to the sample.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:distance/@unit

[1..1] distance requires a unit to be specified. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:size

[0..1] Opening dimensions of this aperture.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:size/@name

[1..1] Optional attribute to clarify the name of this beam size.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:size/cs:x

[0..1] Dimension of the aperture in X.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:size/cs:x/@unit

[1..1] Required unit for the dimension of x. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:size/cs:y

[0..1] Dimension of the aperture in Y.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:size/cs:y/@unit

[1..1] Required unit for the dimension of y. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:size/cs:z

[0..1] Dimension of the aperture in Z. While this is allowed by the standard, it does not make much sense for small-angle scattering.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:aperture/cs:size/cs:z/@unit

[1..1] Required unit for the dimension of z. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:length

[0..1] Amount/length of collimation inserted (on a SANS instrument).

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SAScollimation/cs:length/@unit

[1..1] length requires a unit to be specified. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector

[1..inf] Description of a single or composite detector.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:beam_center

[0..1] Center of the beam on the detector in X and Y.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:beam_center/@name

Optional attribute to clarify the name of this detector beam center.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:beam_center/cs:x

[0..1] Center of the beam on the detector in X.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:beam_center/cs:x/@unit

[1..1] Required unit for the dimension of x. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:beam_center/cs:y

[0..1] Center of the beam on the detector in Y.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:beam_center/cs:y/@unit

[1..1] Required unit for the dimension of y. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:beam_center/cs:z

[0..1] Center of the beam on the detector in Z. While this is allowed by the standard, it does not make much sense for small-angle scattering.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:beam_center/cs:z/@unit

[1..1] Required unit for the dimension of z. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:name

[1..1] Name of the detector.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:offset

[0..1] Offset of the detector position in X, Y, and Z.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:offset/@name

Optional attribute to clarify the name of this beam size.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:offset/cs:x

[0..1] Offset of the detector position in X.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:offset/cs:x/@unit

[1..1] Required unit for the dimension of x. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:offset/cs:y

[0..1] Offset of the detector position in Y.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:offset/cs:y/@unit

[1..1] Required unit for the dimension of y. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:offset/cs:z

[0..1] Offset of the detector position in Z. While this is allowed by the standard, it does not make much sense for small-angle scattering.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:offset/cs:z/@unit

[1..1] Required unit for the dimension of z. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:orientation

[0..1] Orientation (rotation) of the detector in roll, pitch, and yaw. Must specify the unit as an attribute.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:orientation/@name

Optional attribute to name this orientation.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:orientation/cs:pitch

[0..1] Optional rotation of the detector about the X axis (pitch).

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:orientation/cs:pitch/@unit

[1..1] Required unit for the dimension of pitch. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:orientation/cs:roll

[0..1] Optional rotation of the detector about the Z axis (roll).

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:orientation/cs:roll/@unit

[1..1] Required unit for the dimension of roll. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:orientation/cs:yaw

[0..1] Optional rotation of the detector about the Y axis (yaw).

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:orientation/cs:yaw/@unit

[1..1] Required unit for the dimension of yaw. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:pixel_size

[0..1] Size of detector pixels in X and Y.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:pixel_size/@name

Optional attribute to clarify the name of this detector pixel size.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:pixel_size/cs:x

[0..1] Size of detector pixels in X.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:pixel_size/cs:x/@unit

[1..1] Required unit for the dimension of x. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:pixel_size/cs:y

[0..1] Size of detector pixels in Y.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:pixel_size/cs:y/@unit

[1..1] Required unit for the dimension of y. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:pixel_size/cs:z

[0..1] Size of detector pixels in Z. While this is allowed by the standard, it does not make much sense for small-angle scattering.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:pixel_size/cs:z/@unit

[1..1] Required unit for the dimension of z. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:SDD

[0..1] Distance between sample and detector. Must specify the unit as an attribute.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:SDD/@unit

[1..1] Required unit for SDD. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:slit_length

[0..1] Slit length of the instrument for this detector. This is expressed in the same units as Q (reciprocal space units). Must specify the unit as an attribute.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASdetector/cs:slit_length/@unit

[1..1] Required unit for the slit length. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource

[1..1] Description of the source of the radiation.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/@name

[0..1] Optional text description of the source of the radiation (incident on the sample). This can be different from /cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:radiation.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:beam_shape

[0..1] Text description of the shape of the beam (incident on the sample).

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:beam_size

[0..1] Physical dimension of the beam (incident on the sample). Note: If beam is round, just use X dimension. Note: While Z dimension is allowed by the standard, it does not make sense for small-angle scattering.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:beam_size/@name

Optional attribute to clarify the name of this beam size.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:beam_size/cs:x

[0..1] Dimension of the beam size in X.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:beam_size/cs:x/@unit

[1..1] Required unit for the dimension of x. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:beam_size/cs:y

[0..1] Dimension of the beam size in Y.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:beam_size/cs:y/@unit

[1..1] Required unit for the dimension of y. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:beam_size/cs:z

[0..1] Dimension of the beam size in Z. While this is allowed by the standard, it does not make much sense for small-angle scattering.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:beam_size/cs:z/@unit

[1..1] Required unit for the dimension of z. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:radiation

[1..1] Name of the radiation used (neutron, X-ray, synchrotron X-ray, Cu Ka X-ray tube, ..."

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:wavelength

[0..1] wavelength of radiation incident on the sample.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:wavelength/@unit

[1..1] wavelength of radiation requires a unit to be specified. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:wavelength_max

[0..1] Some facilities specify wavelength using a range. The maximum of such a range is given by wavelength_max.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:wavelength_max/@unit

[1..1] wavelength_max requires a unit to be specified. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:wavelength_min

[0..1] Some facilities specify wavelength using a range. The minimum of such a range is given by wavelength_min.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:wavelength_min/@unit

[1..1] wavelength_min requires a unit to be specified. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:wavelength_spread

[0..1] Some facilities specify the width of the wavelength spectrum. The extent (or width) of such a range is given by wavelength_spread.

/cs:SASroot/cs:SASentry/cs:SASinstrument/cs:SASsource/cs:wavelength_spread/@unit

[1..1] wavelength_spread requires a unit to be specified. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASnote

[1..inf] Free form description of anything not covered by other elements.

/cs:SASroot/cs:SASentry/cs:SASprocess

[0..inf] Description of a processing or analysis step.

/cs:SASroot/cs:SASentry/cs:SASprocess/@name

[0..1] Optional attribute to provide a name for this SASprocess.

Note

It is redundant with /cs:SASroot/cs:SASentry/cs:SASprocess/cs:name but it is not the same. It should probably be removed.

/cs:SASroot/cs:SASentry/cs:SASprocess/cs:date

[0..1] Optional date for this data processing or analysis step.

> **Note**
> SHOULD WE SPECIFY THE FORMAT FOR THE DATE?

/cs:SASroot/cs:SASentry/cs:SASprocess/cs:description

[0..1] Optional description for this data processing or analysis step.

/cs:SASroot/cs:SASentry/cs:SASprocess/cs:name

[0..1] Optional name for this data processing or analysis step.

/cs:SASroot/cs:SASentry/cs:SASprocess/cs:SASprocessnote

[1..inf] This element is used to describe anything about SASprocess that is not already described.

/cs:SASroot/cs:SASentry/cs:SASprocess/cs:SASprocessnote/<any>

[0..inf] Provision at this point for any element to be entered that is not part of the canSAS standard. Declare a default namespace declaration such as `xmlns="urn:example-namespace:example-context"` to identify that this is a foreign element (where you replace "`example-namespace`" and "`example-context`" with your own terms).

/cs:SASroot/cs:SASentry/cs:SASprocess/cs:term

[0..1] This is used to specify the value of a single variable, parameter, or term related to the SASprocess step.

/cs:SASroot/cs:SASentry/cs:SASsample

[] Description of the sample.

/cs:SASroot/cs:SASentry/cs:SASsample/@name

[0..1] Optional attribute to name this sample. (Should be the same as SASsample/cs:ID)

/cs:SASroot/cs:SASentry/cs:SASsample/<any>

[0..inf] Provision at this point for any element to be entered that is not part of the canSAS standard. Declare a default namespace declaration such as `xmlns="urn:example-namespace:example-context"` to identify that this is a foreign element (where you replace "`example-namespace`" and "`example-context`" with your own terms).

/cs:SASroot/cs:SASentry/cs:SASsample/cs:details

[0..inf] Text string to supply additional sample details.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:ID

[1..1] Text string that identifies this sample.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:orientation

[0..1] Orientation (rotation) of the sample.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:orientation/@name

Optional attribute to name this orientation.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:orientation/cs:pitch

[0..1] Optional rotation of the sample about the X axis (pitch).

/cs:SASroot/cs:SASentry/cs:SASsample/cs:orientation/cs:pitch/@unit

[1..1] Required unit for the dimension of pitch. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASsample/cs:orientation/cs:roll

[0..1] Optional rotation of the sample about the Z axis (roll).

/cs:SASroot/cs:SASentry/cs:SASsample/cs:orientation/cs:roll/@unit

[1..1] Required unit for the dimension of roll. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASsample/cs:orientation/cs:yaw

[0..1] Optional rotation of the sample about the Y axis (yaw).

/cs:SASroot/cs:SASentry/cs:SASsample/cs:orientation/cs:yaw/@unit

[1..1] Required unit for the dimension of yaw. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASsample/cs:position

[0..1] Location in X, Y, and Z of the sample. Must specify the unit as an attribute to each position.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:position/@name

Optional attribute to name this position.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:position/cs:x

[0..1] Location of the sample in X.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:position/cs:x/@unit

[1..1] Required unit for the dimension of x. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASsample/cs:position/cs:y

[0..1] Location of the sample in Y.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:position/cs:y/@unit

[1..1] Required unit for the dimension of y. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASsample/cs:position/cs:z

[0..1] Location of the sample in Z. While this is allowed by the standard, it does not make much sense for small-angle scattering.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:position/cs:z/@unit

[1..1] Required unit for the dimension of z. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASsample/cs:temperature

[0..1] Temperature of this sample. Must specify the unit as an attribute.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:temperature/@unit

[1..1] Required unit for temperature. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASsample/cs:thickness

[0..1] Thickness of this sample. Must specify the unit as an attribute.

/cs:SASroot/cs:SASentry/cs:SASsample/cs:thickness/@unit

[1..1] Required unit for thickness. (See @unit for details.)

/cs:SASroot/cs:SASentry/cs:SASsample/cs:transmission

[0..1] Transmission (1-attenuation) of this sample. Express this as a fraction, not as a percentage. NOTE: there is not "unit" attribute.

/cs:SASroot/cs:SASentry/cs:Title

[1..1] Title of this SASentry.

@unit

Data unit to be given in standard SI abbreviations (e.g., m, cm, mm, nm, K) with the following exceptions: um=micrometres C=celsius A=Angstroms percent=%. fraction a.u.=arbitrary units none=no units are relevant (such as dimensionless)

cs:term

[0..1] This is used to specify the value of a single variable, parameter, or term related to the SASprocess step. This could also be used in a SASnote element to indicate terms not associated with a SASprocess step.

cs:term/@name

[1..1] Name of the term.

cs:term/@unit

[1..1] Unit (string) of the term. (See @unit for details.)

orientation/cs:pitch

[0..1] Rotation about about the X axis. Unit must be specified.

orientation/cs:roll

[0..1] Rotation about about the Z axis. Unit must be specified.

orientation/cs:yaw

[0..1] Rotation about about the Y axis. Unit must be specified.

position/cs:x

[0..1] Translation in the horizontal direction, orthogonal to Y and Z. Positive X direction increases as defined by Y and Z. Unit must be specified.

position/cs:y

[0..1] Translation along the vertical gravitational direction. Positive direction increases upward. Unit must be specified.

position/cs:z

[0..1] Translation along the beam direction. Positive direction increases from source towards detector. Unit must be specified.

roll, pitch, yaw

Coordinates for (roll, pitch, yaw) values representing an orientation or rotation. Unit must be specified for each.

x, y, z

Coordinates for (x, y, z) values representing a position or dimension. Unit must be specified for each.

Appendix D

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 (Transformation)
  - 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**: XML namespaces are 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
- **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

Chapter 5

Index