CanSAS 1-D Data Format Manual, v1.0

COLLABORATORS

	TITLE : CanSAS 1-D Data Format Manual, v1.0	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Pete R. Jemian	September 25, 2009	

REVISION HISTORY

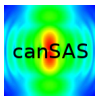
NUMBER	DATE	DESCRIPTION	NAME

Contents

List of Figures

List of Tables

Preface



CanSAS 1-D Data Format Manual, v1.0

Chapter 1

Wiki Manual

1.1 Disclaimer

This description is meant to inform the community how to layout the information within the XML files. However, should the information in this document and the cansas1d/1.0 SAS XML Schema (http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas differ, the XML Schema will be deemed to have the most correct description of the standard.

1.2 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. CanSAS (http://www.smallangles.net/canSAS) identified that a significant need within the SAS community can be satisfied by a robust, self-describing, text-based, standard format to communicate reduced one-dimensional small-angle scattering data, "I(Q)", between users of our facilities. Our goal has been to define such a format that leaves the data file instantly human-readable, editable in the simplest of editors, and importable by simple text import filters in programs that need not recognise advanced structure in the file nor require advanced programming interfaces. The file should contain both the primary data of "I(Q)" and also any other descriptive information (metadata) about the sample, measurement, instrument, processing, or analysis steps.

The cansas1d/1.0 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.

1.2.1 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.

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

1.3 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.

1.3.1 Overview

The basic elements of the cansas1d/1.0 standard are shown in the following table. After an XML header, the root element of the file is [[cansas1d_SASroot | SASroot]] which contains one or more [[cansas1d_SASentry | SASentry]] elements, each of which describes a single experiment (data set, time-slice, step in a series, new sample, etc.). Details of the [[cansas1d_SASentry | SASentry]] element are also shown in the next figure. Refer to the [[cansas1d_block_diagrams | block diagrams]] for alternative depictions. See [http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d.xml cansas1d.xml] for an example XML file. Examples, Case Studies, and other background information are below. More discussion can be found on the [[1D_Data_Formats_Working_Group|canSAS 1D Data Formats Working Group]] page and its [[Talk:1D_Data_Formats_Working_Grou page. Details about each specific field (XPath string, XML elements and attributes) are described on the [[cansas1d_definition_of_terms]] page.

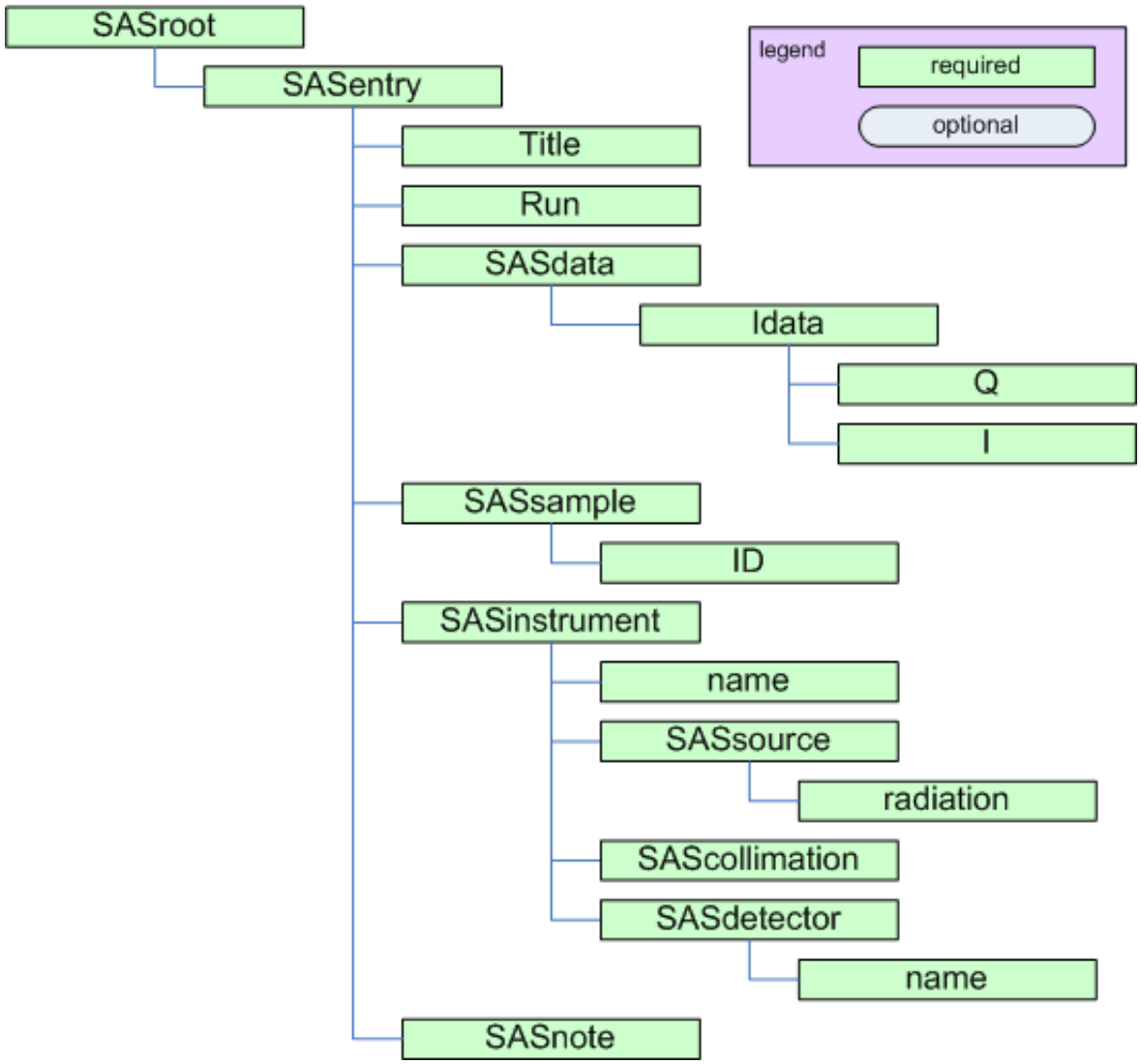


Figure 1.1: block diagram of minimum elements required for cansas1d/1.0 standard

- 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.)
- block diagrams
- cansas1d.xml example XML file

- discussion of this format: basic more
- Seek outside help for XML
- Definition of terms: Details about each specific field (XPath string, XML elements and attributes)

1.3.1.1 Basic elements of the cansas1d/1.0 standard

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
:::[[cansas1d_any | {any}]]	any non-cansas1d/1.0 element can be used at this point
:::[[cansas1d_SASdata | SASdata]]	this is where the reduced 1-D SAS data is stored
::::[[cansas1d_SASdata | Idata]]	a single data point in the dataset
:::[[cansas1d_any | {any}]]	any non-cansas1d/1.0 element can be used at this point
:::[[cansas1d_SASsample | SASsample]]	description of the sample
:::[[cansas1d_SASinstrument | SASinstrument]]	description of the instrument
::::[[cansas1d_SASsource | SASsource]]	description of the source
::::[[cansas1d_SAScollimation | SAScollimation]]	description of the collimation
::::[[cansas1d_SASdetector | SASdetector]]	description of the detector
:::[[cansas1d_SASprocess | SASprocess]]	for each processing or analysis step
:::[[cansas1d_SASnote | SASnote]]	anything at all

1.3.1.2 Required XML file header

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

1.3.2 Rules

1. canSAS1d/1.0 XML data files will adhere to the standard if they can successfully [[cansas1d_documentation#Validation_of_XML_ | validate]] against the established XML Schema ([http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d.xsd cansas1d.xsd])
2. $$Q = (4pi/lambda)sin(theta) \tag{1.1}$$

   where lambda is the wavelength of the radiation

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

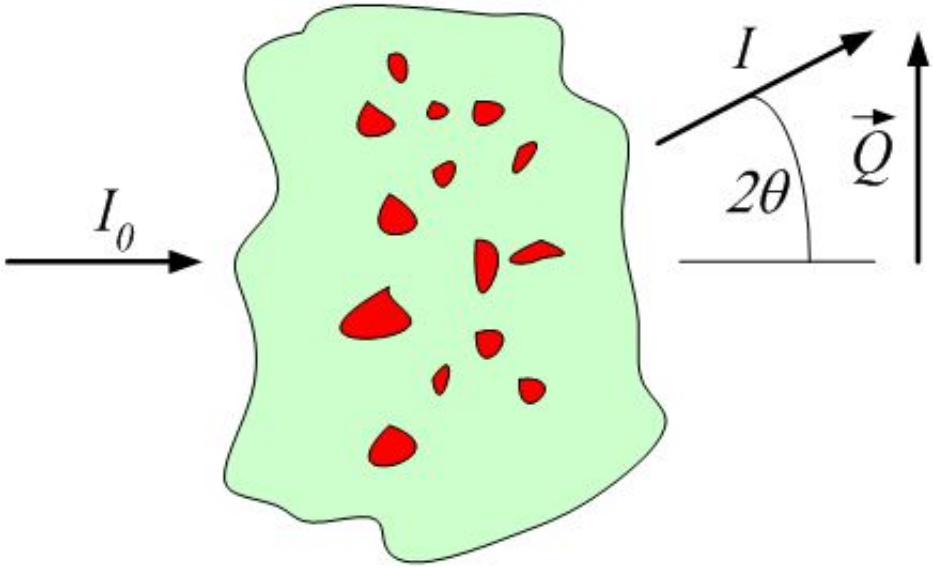
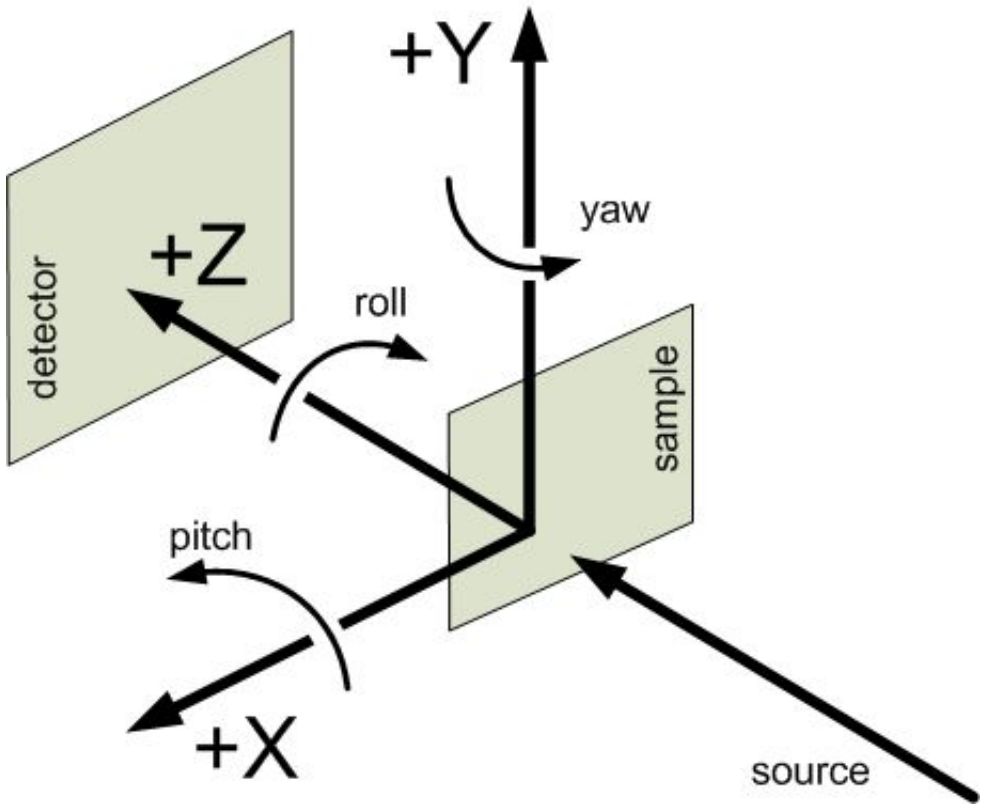


Figure 1.2: definition of Q geometry for small-angle scattering

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 "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.

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

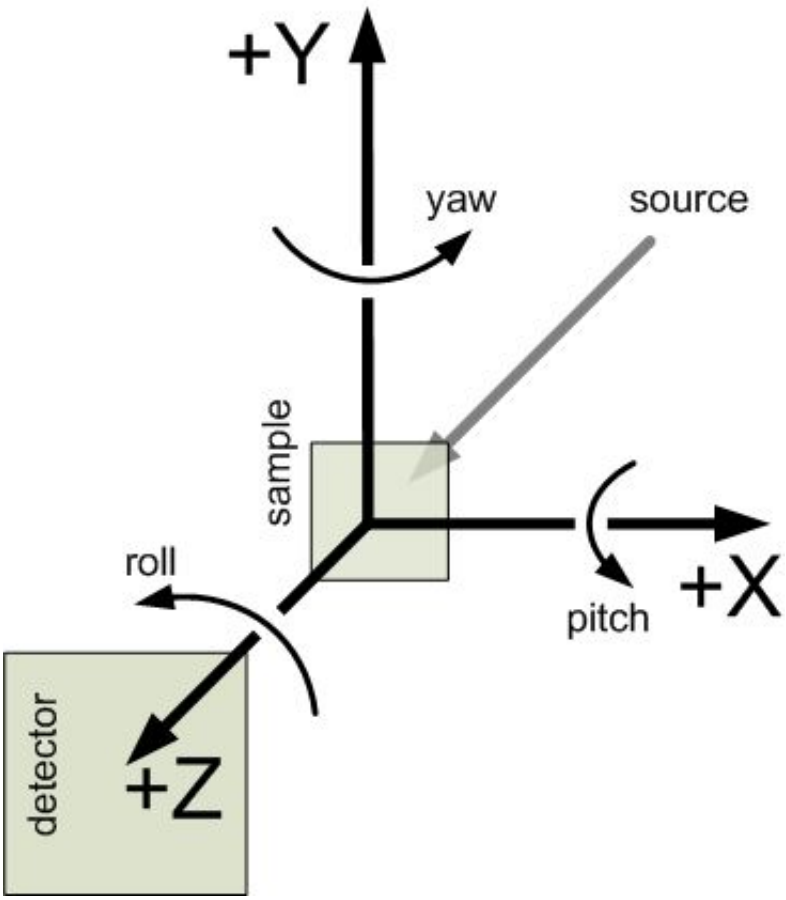


Figure 1.4: definition of translation and orientation geometry as viewed from the source towards the detector

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
9. Binary data is not supported

1.3.3 Compatibility of Geometry Definitions

Note: translation and orientation geometry used by canSAS are consistent with:

- http://en.wikipedia.org/wiki/Cartesian_coordinate_system
- http://en.wikipedia.org/wiki/Right-hand_rule
- http://www.nexusformat.org/Coordinate_Systems
- http://mcstas.risoe.dk/documentation/tutorial/node6.html
- http://webhost5.nts.jhu.edu/reza/book/kinematics/kinematics.htm

The translation and orientation geometry definitions used here are different than those used by "'SHADOW"' (http://www.nanotech.wisc.
where the "y" and "z" axes are swapped and the direction of "x" is changed.

1.3.4 Converting data into the XML format

A WWW form ([http://www.smallangles.net/canSAS/xmlWriter/ canSAS/xmlWriter]) is available to translate three-column ASCII text data into the canSAS1d/1.0 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 <nowiki>Submit</nowiki> 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/1.0 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.

1.4 Documentation and Definitions

- **Documentation**: [[cansas1d_documentation]] (this page)
- **Definitions**: [[cansas1d_definition_of_terms]]
- **Block diagrams**: [[cansas1d_block_diagrams]]

1.4.1 XML Schema

The [http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d.xsd cansas1d.xsd] [http://www.w3schools.com/xsd XML Schema] defines the rules for the XML file format ([http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d.xsd TRAC], [http://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansas1d.xsd SVN]) and is used to validate any XML file for adherence to the format.

1.4.2 XML Stylesheets

- "'cansasxml-html.xsl"': [http://www.w3schools.com/xsl/ XSLT stylesheets] can be used to extract metadata or to convert into another file format. The default canSAS stylesheet [[http://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansasxml-html.xsl cansasxml-html.xsl]] should be copied into each folder with canSAS XML data file(s). It can be used to display the data in a supporting WWW browser (such as Firefox or Internet Explorer) or to import into Microsoft Excel (with the added XML support in Excel). (See the excellent write-up by Steve King, ISIS, at http://www.isis.rl.ac.uk/archive/LargeScale/LOQ/xml/cansas_xml_fo for an example.) By default, MS Windows binds "'*.xml"' files to start Internet Explorer. Double-clicking on a canSAS XML data file with the [http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansasxml-html.xsl "'cansasxml-html.xsl"'] stylesheet in the same directory will produce a WWW page with the SAS data and selected metadata.
- Suggestions for support software that writes canSAS1d/1.0 XML data files:
  - 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
    * 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. For example:

    ```
    # \$Revision: 95 $
    ```

    is version 66 (updated 2009-01-12).

1.4.3 Examples and Case Studies

- [http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d.xml cansas1d.xml] basic example: 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.
- [http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/bimodal-test1.xml bimodal-test1.xml]: Simulated SAS data to test size distribution calculation routines.
- [[Glassy_Carbon_Round_Robin | Glassy Carbon Round Robin]]: Glassy carbon samples measured at several facilities worldwide.
- [[cansas1d_casestudy_collagen | dry chick collagen]]: illustrates the minimum information necessary to meet the requirements of the standard format
- [[cansas1d_casestudy_af1410 | AF1410 steel]]: SANS study using magnetic contrast variation (with multiple samples and multiple data sets for each sample), the files can be viewed from TRAC (no description yet): [http://svn.smallangles.net/trac/canSAS/brows http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/examples/af1410/]
- [http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cansas1d-template.xml cansas1d-template.xml]: 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.

1.4.3.1 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:

- [http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/W1W2.XML ISIS LOQ SANS instrument]
- [http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/cs_af1410.xml NIST SANS data]

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

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

1.4.4 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/1.0 format. Please refer to the references (and others) [[#Help_for_XML | below]] for deeper discussions on foreign elements.

No examples exist.

Note
2009-09, PRJ: Actually, the ISIS glassy carbon data is now an example

---

At present, all examples of canSAS xml files using foreign namespaces have been converted to bring that data into either the ”’SASprocessnote”’ or ”’SASnote”’ elements. Refer to the [http://svn.smallangles.net/trac/canSAS/changeset/47 TRAC changes] for an example of arranging the content in ”’SASprocessnote”’ to avoid the use of foreign namespace elements.

1.4.5 Support tools for Visualization & Analysis software

Support for importing canSAS1d/1.0 files exists for these languages:

Note
Refactor the wiki pages here and link as appropriate.

- **FORTRAN**
- **IgorPro**
- **Java**
- **Microsoft Excel**
- **PHP**
- **Python**
- **XSLT** (useful in a web browser)

1.4.6 Software repositories (for canSAS1d/1.0 standard)

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

1.5 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 [http://svn.smallangles.net/svn/canSAS/1dwg/trunk/cansas1d.xsd cansas1d.xsd] XSD file into form and press <continue validation>
5. check the results

1.6 Help for XML

The various references for help on XML have been moved to their own wiki page: [[XmlHelp]]

Note
chapter="Other" section="XML Help" xml:id="wiki-XML_Help"

Chapter 2

Elements of the CanSAS XML standard

There are various elements (tag names) in the canSAS1d/1.0 standard. Each of these is described below.

2.1 ”{any}” element

Table 2.1:

Name	Type	occurrence	Description	Attributes
”{any}”	container	[0..unbounded]	Any element(s) not defined in the cansas1d/1.0 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

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.

3.1 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.

3.1.1 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/

3.1.2 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.

3.2 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).

Table 3.1: Current status of IgorPro support

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 adheres to the cansas1d/1.0 standard: http://www.smallangles.net/wgwiki/index.php/cansas1d_documentati
URL	TRAC: http://svn.smallangles.net/trac/canSAS/browser/1dwg/trunk/IgorPro/c SVN: http://svn.smallangles.net/svn/canSAS/1dwg/trunk/IgorPro/cansasXM
requires	IgorPro: http://www.wavemetrics.com XMLutils - XOP: http://www.igorexchange.com/project/XMLutils (IGOR.5.04.x-1.x-dev, 2008-Aug-22)

3.2.1 Checkout of support code in Subversion

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

3.2.1.1 XMLutils XOP

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

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

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

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

```
svn update
```

This will check the repository and update files as needed.If the installer program was updated, you;ll need to run the newinstaller 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.

3.2.1.2 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 IgorProdirectory and is called cansasXML.ipf

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

```
svn update
```

This will check the repository and update files as needed.

3.2.2 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, go to http://svn.igorexchange.com/
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 on-line subversion repository: http://svn.smallangles.net/svn/canSAS/1dwg/trunk/IgorPro/cansasXML.ipf
6. Copy cansasXML.ipf file to ...WavemetricsIgor Pro FolderUser Procedures (or file system equivalent)
7. Then, you should be able to restart IgorPro and progress from there

3.2.3 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 ...WavemetricsIgor Pro FolderUser ProceduresmotofitXMLutils, and create a shortcut/alias to the plugin in ...WavemetricsIgor Pro FolderIgor Extensions.

3.2.4 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 forreduced 1-D SAS data (cansas1d/1.0, also known as SASXML). If the file is found to be non-conforming, then **CS_XmlReader(fileName)** returns with anerror 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 createdwith 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 SASdataelement, 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.

3.2.4.1 data columns

Each column of data in the **SASdata/Idata/*** table is placed into a singleIgorPro 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).

3.2.4.2 metadata

Additional metadata is collected into a single text wave (*metadata*) where the first columnis an identifier (or *key*) and the second identifier is the *value*. Onlythose keys with non-empty values are retained in the metadata table.**CAUTION:** The *values* are not checked for characters that may cause trouble whenplaced in a wave note. This will be the responsibility of the calling routineto *clean these up* if the need arises.

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

Here is an example of the metadata for the ”cs_collagen_full.xml” case study:

Table 3.2: metadata for the ”cs_collagen_full.xml” case study

i (*row*)	metadata[*i*][0] (*key*)	metadata[*i*][1] (*value*)
0	xmlFile	cs_collagen_full.xml
1	namespace	cansas1d/1.0
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 dry chick collagen, d = 673 A 6531 eV, X6B`

3.2.4.3 XML foreign namespace elements

These are ignored at this time.

3.2.4.4 XML namespace and header

The routine does a *best-efforts* check to ensure that the givenXML file conforms to the [[cansas1d_documentation#Required_XML_file_h lrequired 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/1.0** as this is a **key component** used toindex through the XML file.

3.2.4.5 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.

3.2.5 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 demonstrationfunctions **prj_grabMyXmlData()** and **prjTest_cansas1d()** 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

3.2.6 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/1.0 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/1.0 XML file
    Title: standard can 12mm SANS
    Completed in 0.0410387 seconds
W1W2.xml     identified as: cansas1d/1.0 XML file
    Title: standard can 12mm SANS
    Title: TK49 standard 12mm SANS
    Completed in 0.0669074 seconds
ill_sasxml_example.xml     identified as: cansas1d/1.0 XML file
    Title: ILL-D22 example: 7D1 2mm
    Completed in 0.0332752 seconds
isis_sasxml_example.xml      identified as: cansas1d/1.0 XML file
    Title: LOQ TK49 Standard 12mm C9
    Completed in 0.0388868 seconds
r586.xml     identified as: cansas1d/1.0 XML file
    Title: ILL-D11 example1: 2A 5mM 0%D2O
    Completed in 0.0213737 seconds
r597.xml     identified as: cansas1d/1.0 XML file
    Title: ILL-D11 example2: 2A 5mM 0%D2O
    Completed in 0.0221894 seconds
xg009036_001.xml     identified as: cansas1d/1.0 XML file
    Title: det corrn 5m
    Completed in 0.0286721 seconds
cs_collagen.xml      identified as: cansas1d/1.0 XML file
    Title: dry chick collagen, d = 673 A, 6531 eV, X6B
    Completed in 0.0296247 seconds
cs_collagen_full.xml     identified as: cansas1d/1.0 XML file
    Title: dry chick collagen, d = 673 A, 6531 eV, X6B
    Completed in 0.0751836 seconds
cs_af1410.xml      identified as: cansas1d/1.0 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/1.0 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/1.0 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/1.0 XML file
    Title: S7 Neat Polyurea
    Completed in 0.0361616 seconds
```

3.2.7 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 detailsof 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 theseare being collected on the discussion page for the XML format

3.2.7.1 Irena tool suite

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

Chapter 4

Other matters

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

4.1 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