# ADIOS Visualization Schema 1.1 User's Manual

**December 2013**

# ADIOS VISUALIZATION SCHEMA 1.1 USER'S MANUAL

Prepared for the
Office of Science
U.S. Department of Energy

S. Klasky, Q. Liu, N. Podhorszki, D. Pugmire, R. Tchoua, J. Mu

Dec. 2013

Prepared by

OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6070
managed by
UT-BATTELLE, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

# Contents

## Figures

## Acknowledgments

The ADIOS Visualization Schema has been designed by the ADIOS team in collaboration with Ken Moreland at Sandia National Laboratories.

# 1 Introduction

## 1.1 Goals

The dramatic increase of computing and storage resources in the last few decades has allowed scientists to run more complex simulations and generate huge amounts of data. This phenomenon has also changed the composition and the collaboration in teams working around simulations. In addition to traditional scientists – we call application scientists – teams with access to supercomputers need experts in various disciplines including applied mathematics for development of algorithms, I/O and visualization specialists. To extract insight from simulation results, these researchers need to efficiently share knowledge and data without the burden of acquiring expertise in domain outside of their field of training. The goal of the visualization schema is to lower the barrier of entry of application scientists into computing fields; specifically it is to bridge the gap between them and the *big data* visualization experts.

ADIOS already bridges such a gap between application scientists and I/O scientists. While the ADIOS team is researching and developing a state of the art I/O system for super computers it provides easy-to-use, high-level application program interfaces (APIs) so that application scientists can easily adapt the ADIOS library and produce science without diving too deeply into computer configuration and skills. We build on this model to allow users to indicate how they want and need to visualize their data.

## 1.2 What Is ADIOS Visualization Schema?

In the teams we have experience with; scientists often write their own analysis scripts or utilize a collection of scripts from the community. Their improvised visualization skills are not always scalable, hence the need for group members who are computer scientists. Using visualization tools such as VisIt, ParaView, AVS , etc. requires a certain level of expertise as well as information on the code data structures. Visualization experts need to know where and how to access the data to be visualized. They need metadata about files and physical variables to be visualized while an understanding of the science is not essential. Visualization scientists speak in terms of scalars, vectors, arrays and meshes for instance. This is information that application scientists are familiar with and can point to in their code without knowing protocols and methods expected by common visualization tools. Therefore there is a need for translation between the language of the application scientist and the visualization expert. Our solution is the design and implementation of the ADIOS Visualization Schema.

Unlike other attempts to solve this gap of knowledge between application and visualization scientists, the ADIOS Visualization Schema does not focus on the connecting popular self-describing data formats to popular visualization tools. The use of self-describing data does not insure standard organization, or

consistent naming convention of particular visualization elements. Therefore, straightforward visualization of the data by visualization collaborators often requires a closer connection to the science and the scientist who wrote the data. Similarly the development of visualization plugins in popular tools like VisIt and ParaView requires investment of time and energy from both application and visualization scientists. This may seem like a reasonable approach as long as codes, platforms and tools do not evolve, however this is rarely the case in bleeding edge research and development.

The key to the ADIOS Visualization schema approach is the persistent focus on collaboration and separation of expertise. Indeed the schema focuses on the visual understanding of the data: content that can be processed by scientists' eyes and brains. Changes in the visual representation of a variable are no longer tied to codes and developers but rather to the data itself. Application scientists can describe and modify the representation of variables in the ADIOS XML file without understanding the protocol and methods to visualize it in any specific software. They can share scientific content with colleagues without knowledge of their preferred tools and naming conventions. The visualization experts can adapt readers based on scientists' changes without understanding the corresponding changes in the simulation code. Moreover instead of having a reader for each code, programs such as VisIt and Paraview will require only a single reader to visualize any ADIOS data file. Finally, by embedding the schema in ADIOS, users get performance and flexibility of I/O as an added bonus.

## 1.3 Future ADIOS Visualization Schema Goals

One of the main goals of the next ADIOS Visualization Schema version will be to explorer and support more codes including Adaptive Mesh Refinement (AMR) codes. Another goal is to survey and implement the use of expressions in the schema, allowing for more sophisticated description of visualization and analysis in the schema. A long term subsequent goal is to describe as much of a simulation monitoring and analysis workflow in the schema. We plan to research and develop ways for users to describe pieces of, or entire workflows used by scientists to run, monitor and analyze their results using the schema.

## 1.4 Implementation

Following the ADIOS model the schema is implemented in two-parts: the XML configuration file and the APIs. We also provide No-XML APIs for codes which do not or cannot use the XML configuration file. A detailed description of the APIs with XML example is presented in this manual. Using the XML or no-XML APIs, descriptions of mesh and variables are inserted in the ADIOS BP file format as will be illustrated in the examples and figures below.

# 2 XML Configuration

## 2.1 Overview

XML tags and attributes provide a convenient way to organize data in a human readable way. XML is also an intrinsic part of ADIOS is designed to allow users to store as much metadata as they can in an external configuration file. It is already the case that if users change the way they write data, they can simply edit their ADIOS XML file. We now extend this approach by giving the scientists a way to describe the mesh structure, variables that compose the mesh and related variables. The goal is to require minimal information from a domain scientist as possible and still be able to visualize the data on popular HPC tools. When data representation changes in the code, or new variables are added, users do not have to change their code, they simply shuffle variables and meshes around in the XML. Visualization software then interprets the changes and accurately displays the data produced by the simulation.

The main new tags in the ADIOS XML file are the mesh tag and the children XML tags that vary with the mesh type (uniform, rectilinear, structured or unstructured).

## 2.2 Version

To indicate the schema version use the schema-version XML attribute in the adios-config XML tag. The currently expected value is 1.1, see Figure 1 below.

```
<adios-config host-language="Fortran" schema-version="1.1">
```

Figure 1: Schema version ADIOS XML attribute.

The corresponding ADIOS attributes inserted in the binary files are:

```
string   /adios_schema/version_major                    attr  = "1"
string   /adios_schema/version_minor                    attr  = "1"
```

Figure 2: Schema version ADIOS BP attributes

## 2.3 Variables

To associate a variable with a mesh it is to be displayed on, insert a mesh XML attribute wish a mesh name of your choice. Use unique mesh names (corresponding to unique mesh tags) within the same adios-config.xml.

#### 2.3.1.1 Assigning meshes to variables

```
<var name="Var1" mesh="meshname" gwrite="var1" type="float"
dimensions="iter,num_points"/>
```

Figure 3: Variables on meshes (ADIOS XML attribute)

In ADIOS BP file (output for BP list of bpls command):

```
string    /Var1/adios_schema                                    attr   = "meshname"
```

Figure 4: Variables on meshes (ADIOS BP attribute)

#### 2.3.1.2 Other descriptive options for variables

##### 2.3.1.2.1 Centering

The centering of the variable value onto the mesh is indicated using the attribute center as shown below.

```
<var name="Var1" mesh="meshname" center="node" gwrite="var1" type="float"
dimensions="iter,num_points"/>
```

Figure 5: Variables centering (ADIOS XML attribute)

In ADIOS BP file (output for BP list of bpls command):

```
string    /Var1/adios_schema/centering                          attr   = "node"
```

Figure 6: Variables centering (ADIOS BP attribute)

##### 2.3.1.2.2 Time

The time concept is useful to specify at the mesh level to indicate whether a mesh varies with time, how many time steps there are etc. However the schema allows for these nuances to be set at the variable level for more flexibility.

###### *2.3.1.2.2.1 time-steps*

This description can be used in 3 different cases:

- A single number (integer): A single multiple of the mesh time step to indicate that this variable was written at a specific time step of the simulation.
- A range between 2 min/max numbers (integers) in all dimensions: This variable has been written between two particular time steps during the simulation

- Three numbers: Following the XDMF[1] example, this format indicates a start, a stride and a count to indicate that the variable was written at consistent intervals (stride) from a particular time (start) and for a specific number (count) of time.

The time formatting attributes are as follows: time-steps="threshold_t " or time-steps="36,t_limit" or time-steps= “0,1,t_max”. Expected values are strings representing numbers or ADIOS variable names separated by commas.

```
<var name="Var1" mesh="meshname" center=”node” gwrite="cos" type="float"
dimensions="iter,num_points" time-steps= “0,tlimit”/>
```

Figure 7: Schema variable time-steps ADIOS XML attribute

In ADIOS BP file (output for BP list of bpls command):

```
double   /Var1/adios_schema/time-steps-min          attr  = 0
string   /Var1/adios_schema/time-steps-max          attr  = "tlimit"
```

Figure 8: Schema variable time-steps ADIOS BP attributes

#### *2.3.1.2.2.2 time-scale*

This description is used exactly like the time steps except with real numbers. While time steps are integer numbers simulation can output data at specific time scales and probable want to capture that time scale. For example, a simulation writing out data every 1.5 ms of simulation time will probably want to specify the time steps as 0.0000, 0.0015, 0.0030, etc. The formatting is also similar to the time-steps but the expected values are string representing numbers (including doubles) and/or existing variable names from the ADIOS XML configuration file (format: time-scale="0.0015 " or time-scale="0.0,t_limit" or time-scale= “0,0.0015,200”).

```
<var name="Var1" mesh="meshname" center=”node” gwrite="cos" type="float"
dimensions="iter,num_points" time-scale= “0,0.0015, 200”/>
```

Figure 9: Schema variable time-scale ADIOS XML attribute

[1] http://www.xdmf.org/index.php/Main_Page

In ADIOS BP file (output for BP list of bpls command):

```
double   /Var1/adios_schema/time-scale-start          attr  = 0
double   /Var1/adios_schema/time-scale-stride         attr  = 0.0015
double   /Var1/adios_schema/time-scale-count          attr  = 200
```

Figure 10: Schema variable time-scale ADIOS BP attributes

#### 2.3.1.2.2.3 time-format-series

Simulations monitoring programs and workflows output data that is often transformed into images. Combining those images into movies usually requires a consistent naming convention for the image files. Therefore we have added this description in order to quickly describe the padding pattern for output images. If this number is 0, images for a variable will be over-written over time. If this number is 4, then the time-steps for images will be padded with 0 up to 4 digit numbers (varname.0023.png, varname.0024.png, varname.0025.png…). The formatting of the time series is as follows: time-series-format = "4". Allowed values are strings representing one single integer value.

```
<var name="Var1" mesh="meshname" center="node" gwrite="cos" type="float"
dimensions="iter,num_points" time-series-format= "5"/>
```

Figure 11: Schema variable time-series-format ADIOS XML attribute

In ADIOS BP file (output for BP list of bpls command):

```
string   /adios_schema/U/time-series-format          attr  = "2"
```

Figure 12: Schema variable time-series-format ADIOS BP attribute

#### 2.3.1.2.2.4 Hyperslab

Similarly to the hyperslab concept in XDMF we use this concept to indicate a subset of a variable. We use the concept of start, stride and count in all dimensions of a variable to identify a subset of a dataset. The formatting of the hyperslab is as follows: hyperslab="0 0, 1 stride2,30 nphi" (hyperslab= "start1 start2, stide1 stride2, count1, count2". Allowed values: strings representing numbers (integers for array indices) or variable names separated by a space for each start, stride and count values; start, stride and count values are separated by commas. Figure 13 and 14 illustrate the special case.

```
<var name="Var1" mesh="mesh4" type="double" dimensions="npoints"
hyperslab="0,32"/>
<var name="Var2" mesh="mesh5" type="double" dimensions="npoints"
hyperslab="0,1,32"/>
<var name="Var3" mesh="mesh6" type="double" dimensions="nxpoints"
hyperslab="15"/>
```

Figure 13: Schema variable hyperslab ADIOS XML attribute

In ADIOS BP file (output for BP list of bpls command):

```
string   /Var1/adios_schema                    attr  = "mesh4"
string   /Var1/adios_schema/min                attr  = "0"
string   /Var1/adios_schema/max                attr  = "32"

string   /Var2/adios_schema                    attr  = "mesh5"
string   /Var2/adios_schema/start              attr  = "0"
string   /Var2/adios_schema/stride             attr  = "1"
string   /Var2/adios_schema/count              attr  = "32"

string   /Var3/adios_schema                    attr  = "mesh6"
string   /Var3/adios_schema/singleton          attr  = "15"
```

Figure 14: Schema variable hyperslab ADIOS BP attribute

Note 1: The format above is for a 1D array. The start, stride and count input should have as many values as the variable separated by a space.

Note 2: We use the symbol ":" to indicate a superset. We have encountered cases where users defined 1D arrays in the ADIOS XML config file but wrote values for this array several times during one time step (multiple planes); the mesh provided in this case was a 2D mesh. Therefore we allow for users to indicate that they will write an additional dimension for certain variables. This would allow the reader to process all additional points, lines, or planes creating extended variable dataset if necessary. We do so using the special ":" symbol. See example below for more details.

```
<var name="dpot" type="real*8" dimensions="1,nnode" mesh="xgc.mesh"
hyperslab=":"/>
```

Figure 15: Schema hyperslab special case ADIOS XML attribute

In ADIOS BP file (output for BP list of bpls command):

```
string   /dpot/adios_schema                    attr  = "xgc.mesh"
string   /dpot/adios_schema/singleton          attr  = ":"
```

Figure 16: Schema hyperslab special case ADIOS BP attribute

In the above example, our reader gets the extra dimensions and creates planes for the dpot variable naming each plane with an extension number: dpot-0000, dpot-0001….dpot000n.

## 2.4 Meshes

The main mesh tag requires name and type attributes. The mesh topology (shape) and geometry (coordinates) information varies with the mesh type. The main difference between the mesh types is the amount of information required to describe it. First we describe some basic concepts that apply to all mesh types.

#### 2.4.1.1 Time

The time-varying attribute is optional and will be equal to "no" by default assuming that the mesh does not change over time. Other time attributes are similar to the variables time attributes: time-steps, time-scale, and time –series-format. When the mesh contains these attributes, all variables on the mesh are assumed to follow the same timing patterns unless if they have their own time xml attributes. The time-steps and time-scale attributes may point to time variables in the ADIOS XML configuration file. While the time-format-series attribute value would be a string representing an integer for padding and formatting output image files. These attributes are added to the mesh tags similarly to the way they are added to the variable tags.

```
<mesh name="xgc.mesh" type="uniform" time-varying="no" time-steps= "0,1,99"
time-scale = "0,0.0015,99" time-series-format="4"/>
```

Figure 17: Schema mesh time ADIOS XML attributes

```
string   /adios_schema/xgc.mesh/time-varying          attr  = "no"
double   /adios_schema/xgc.mesh/time-steps-start      attr  = 0
double   /adios_schema/xgc.mesh/time-steps-stride     attr  = 1
double   /adios_schema/xgc.mesh/time-steps-count      attr  = 99
double   /adios_schema/xgc.mesh/time-scale-start      attr  = 0
double   /adios_schema/xgc.mesh/time-scale-stride     attr  = 0.0015
double   /adios_schema/xgc.mesh/time-scale-count      attr  = 99
```

Figure 18: Schema mesh time ADIOS BP attributes

#### 2.4.1.2 Mesh variables and variables on meshes

Let us consider an ADIOS XML file containing one mesh there will be three types of variables in the file.

- Variables composing the mesh
- Variables on the mesh
- Variables without a mesh

Also, while a simulation run will have only one xml file it may contain several ADIOS groups and output several files. Therefore the schema gives the option to pin-point the mesh variables' locations. Variables composing the mesh maybe in describe in a separate ADIOS group and written in a separate file.

- Mesh group – group containing mesh variables
- Mesh file – file where mesh variables are written

Note 3: Therefore variable to be displayed on top of mesh "xgc.mesh" could be stored in ADIOS grou "field3d", while variables composing this mesh could be stored in a separate group called: "diagnosis.mesh". The mesh variables may subsequently be stored in a different file ("xgc.mesh.bp") than the output variables to be displayed on that mesh (xgc.3d.00001.bp). The file and group attributes are specified as follows: file="xgc.mesh.bp" group="xgc.mesh" and expected values are:

- Group: strings representing other ADIOS group names in the file.
- File: strings for output file name containing the mesh composing the mesh

```
<mesh name="xgc.mesh" type="unstructured" file="xgc.mesh.bp" time-varying="no"/>

<mesh name="xgc.mesh" type="unstructured" group="diagnostic.mesh" time-varying="no"/>
```

Figure 19: Schema mesh file and group ADIOS XML attributes

In ADIOS BP file (output for BP list of bpls command):

```
string   /adios_schema/xgc.mesh/mesh-file          attr  = "xgc.mesh.bp"
string   /adios_schema/xgc.mesh/mesh-group         attr  = "diagnostic.mesh"
```

Figure 20: Schema mesh file and group ADIOS BP attributes

#### 2.4.1.3 Uniform Mesh

The uniform mesh is the most structured case of the structured meshes and requires the least amount of information. If no details are provided reader should assume a uniformly divided line, rectangle (2D) or box (3D) (coordinates range from 0 to number of points-1 in all available dimensions).

```
<mesh name="mesh1" type="uniform" time-varying="no">
```

Figure 21: Schema basic uniform mesh declaration

Available XML children of the uniform mesh tag are:

- Dimensions – optional. Default will be the dimensions of the variables assigned to this mesh
- Origin – optional. Default will be {0, [0,0], [0,0,0] …} depending on the mesh dimensions
- Maximum – optional. Default will be the number of points minus the origins
- Spacing – optional. Default will be one

Example uniform mesh description:

```
<var name="V1" mesh="mesh3" gwrite="t" gread="t" type="double"
dimensions="NX1,NY1"/>

<var name="V2" mesh="mesh4" gwrite="t" gread="t" type="double"
dimensions="NX2,NY2"/>

<mesh name="mesh3" type="uniform" time-varying="no">
        <dimensions value="D1,50" />
        <origin value="-2,O2" />
        <spacing value="S1" />
</mesh>

<mesh name="mesh4" type="uniform" time-varying="no">
        <dimensions value="40,D2" />
        <origin value="O1,-1" />
        <maximum value="M1,600" />
</mesh>
```

Figure 22: Schema uniform mesh ADIOS XML description

These XML children are all optional; i.e. the default for this mesh will be a uniform line that goes from 0 to the number of points minus one in each variable dimension. Users can provide an origin and a spacing, or and origin and a

maximum for example. Depending on the mesh dimensions and on what is provided the reader will deduct the other mesh characteristics. Any of these XML children can be equated to a variable name and the reader will get the values from the corresponding variable in ADIOS.

The examples above result in the following attributes being inserted in the ADIOS binary output file:

```
string     /V1/adios_schema                          attr  = "mesh3"

string     /V2/adios_schema                          attr  = "mesh4"

string     /adios_schema/mesh3/type                  attr  = "uniform"
string     /adios_schema/mesh3/time-varying          attr  = "no"
string     /adios_schema/mesh3/dimensions0           attr  = "D1"
double     /adios_schema/mesh3/dimensions1           attr  = 50
integer    /adios_schema/mesh3/dimensions-num        attr  = 2
double     /adios_schema/mesh3/origins0              attr  = -2
string     /adios_schema/mesh3/origins1              attr  = "O2"
integer    /adios_schema/mesh3/origins-num           attr  = 2
string     /adios_schema/mesh3/spacings0             attr  = "S1"
integer    /adios_schema/mesh3/spacings-num          attr  = 1

string     /adios_schema/mesh4/type                  attr  = "uniform"
string     /adios_schema/mesh4/time-varying          attr  = "no"
double     /adios_schema/mesh3/dimensions0           attr  = 40
string     /adios_schema/mesh3/dimensions1           attr  = "D2"
integer    /adios_schema/mesh3/dimensions-num        attr  = 2
string     /adios_schema/mesh3/origins0              attr  = "O1"
double     /adios_schema/mesh3/origins1              attr  = -1
integer    /adios_schema/mesh3/origins-num           attr  = 2
string     /adios_schema/mesh4/maximums0             attr  = "M1"
double     /adios_schema/mesh4/maximums1             attr  = 600
integer    /adios_schema/mesh4/maximums-num          attr  = 2
```

Figure 23: Schema uniform mesh ADIOS BP attributes

#### 2.4.1.4 Rectilinear Mesh

The rectilinear mesh is a special case of the structured mesh: the dimensions indicate the connectivity and only the coordinates of axis nodes need to be known. The number of coordinates per node can be determined from the value of the dimensions of the axis' variables. We use coordinates-single-var and coordinates-multi-var tags to indicate whether there is one single multi-dimensional variable for all axes or one variable per axis.

```
<mesh name="mesh1" type="rectilinear" time-varying="no">
```

Figure 24: Schema basic rectilinear mesh declaration

Available XML children of the rectilinear mesh tags are:

- Dimensions – optional
- Coordinates of axis (coordinates-single-var/coordinates-multi-var) – required

Example of rectilinear mesh XML description and corresponding ADIOS BP binary file listing:

```
<var name="V1" mesh="mesh3" gwrite="t" gread="t" type="double"
dimensions="NX1,NY1,NZ1"/>

<var name="V2" mesh="mesh4" gwrite="t" gread="t" type="double"
dimensions="NX2,NY2"/>

<mesh name="mesh3" type="rectilinear" time-varying="no">
        <dimensions value="D0,D1,D2" />
        <coordinates-single-var value="XYZ3" />
</mesh>

<mesh name="mesh4" type="rectilinear" time-varying="no">
        <dimensions value="30,D2"/>
        <coordinates-multi-var value="X4,Y4" />
</mesh>
```

Figure 25: Schema rectilinear mesh ADIOS XML description

```
string   /V1/adios_schema                              attr  = "mesh3"
string   /V2/adios_schema                              attr  = "mesh4"

string   /adios_schema/mesh3/type                      attr  = "rectilinear"
string   /adios_schema/mesh3/time-varying              attr  = "no"
string   /adios_schema/mesh3/dimensions0               attr  = "D0"
string   /adios_schema/mesh3/dimensions1               attr  = "D1"
string   /adios_schema/mesh3/dimensions2               attr  = "D2"
integer  /adios_schema/mesh3/dimensions-num            attr  = 3
string   /adios_schema/mesh3/coords-single-var         attr  = "XYZ3"

string   /adios_schema/mesh4/type                      attr  = "rectilinear"
string   /adios_schema/mesh4/time-varying              attr  = "no"
double   /adios_schema/mesh4/dimensions0               attr  = 30
string   /adios_schema/mesh4/dimensions1               attr  = "D2"
integer  /adios_schema/mesh4/dimensions-num            attr  = 2
string   /adios_schema/mesh4/coords-multi-var0         attr  = "X4"
string   /adios_schema/mesh4/coords-multi-var1         attr  = "Y4"
integer  /adios_schema/mesh4/coords-multi-var-num      attr  = 2
```

Figure 26: Schema rectilinear mesh ADIOS BP attributes

#### 2.4.1.5 Structured Mesh

For a structured mesh the dimensions are the rows, columns and planes (connectivity); the points specify the coordinates for each node while nspace indicates the number of expected coordinates per node. In this case the coordinates for each node is required to construct the mesh since there is no uniformity in the placement of the nodes. Points can be given through one or multiple variables {C} or {X,Y,Z} where C is a 1D array with interleaved coordinates {x,y,z,x,y,z…x,y,z} . When using points-multi-var, nspace can be derived from the number of variables provided. When using points-single-var and a 1D variable, nspace is required to correctly section the 1d array. If the variable is multi-dimensional, the reader should be able to assign each dimension to a coordinate axis.

```
<mesh name="mesh1" type="structured" time-varying="no">
```

Figure 27: Schema basic structured mesh declaration

Available XML children of the structured mesh tag are:

- Dimensions – required
- Points (points-single-var/points-multi-var) – required
- nspace – optional

Example of structured mesh XML description:

```
<var name="V1" mesh="mesh3" gwrite="t" gread="t" type="double"
dimensions="NX1,NY1"/>

<var name="V2" mesh="mesh4" gwrite="t" gread="t" type="double"
dimensions="NX2,NY2"/>

<mesh name="mesh3" type="structured" time-varying="no">
        <dimensions value="D1,D2"/>
        <points-multi-var value="X1,Y1" />
</mesh>

<mesh name="mesh4" type="structured" time-varying="no">
<dimensions value="30,D2"/>
        <nspace value="D0" />
        <points-single-var value="Z" />
</mesh>
```

Figure 28: Schema structured mesh ADIOS XML description

The above examples insert the following descriptive attributes in the ADIOS output files:

```
string    /V1/adios_schema                          attr  = "mesh3"

string    /V2/adios_schema                          attr  = "mesh4"

string    /adios_schema/mesh3/type                  attr  = "structured"
string    /adios_schema/mesh3/time-varying          attr  = "no"
string    /adios_schema/mesh3/dimensions0           attr  = "D1"
string    /adios_schema/mesh3/dimensions1           attr  = "D2"
integer   /adios_schema/mesh3/dimensions-num        attr  = 2
string    /adios_schema/mesh3/points-multi-var0     attr  = "X1"
string    /adios_schema/mesh3/points-multi-var1     attr  = "Y1"
integer   /adios_schema/mesh3/points-multi-var-num  attr  = 2

string    /adios_schema/mesh4/type                  attr  = "structured"
string    /adios_schema/mesh4/time-varying          attr  = "no"
double    /adios_schema/mesh4/dimensions0           attr  = 30
string    /adios_schema/mesh4/dimensions1           attr  = "D2"
integer   /adios_schema/mesh4/dimensions-num        attr  = 2
string    /adios_schema/mesh4/points-single-var     attr  = "Z"
```

Figure 29: Schema structured mesh ADIOS BP attributes

#### 2.4.1.6 Unstructured Mesh

The unstructured mesh requires the most information. It is the only mesh type that requires cell information explicitly. It does not assume any regularity in the placement of nodes. Users must provide the number, shape and data (connection list) for the cell information. Unstructured meshes can include different type of cells. For example an unstructured mesh may contain triangle cells and rectangular cells. Therefore the user also needs to specify whether the cells are uniform or mixed and provide cell information for each set of cells.

```
<mesh name="mesh1" type="unstructured" time-varying="no">
```

Figure 30: Schema basic unstructured mesh declaration

Available XML children of the rectilinear mesh tags are:

- nspace – optional
- number-of-points – optional
- Coordinates of axis (coordinates-single-var/coordinates-multi-var) – required
- uniform cell tag (count, type and data) – required unless mixed cell tag is present

- mixed cell tag (count, type and data) – required unless uniform cell tag is present

Example of unstructured mesh XML description:

```
<var name="V1" center =“point” mesh="mesh5" gwrite="t" gread="t" type="double"
dimensions="NX1,NY1"/>

<var name="V2" center =“node”  mesh="mesh6" gwrite="t" gread="t" type="double"
dimensions="NX2,NY2"/>
<var name="V3" center =“cell” mesh="mesh7" gwrite="t" gread="t" type="double"
dimensions="NX3,NY3,NZ3"/>
<var name="V4" center =“point” mesh="mesh8" gwrite="t" gread="t" type="double"
dimensions="NX4,NY4,NZ4"/>

<mesh name="mesh5" type="unstructured" time-varying="no">
        <points-single-var value="P1" />
        <nspace value="nsp" />
        <number-of-points value="25" />
        <mixed-cells count="100,NC2" data="C1,C2" type="tri,quad" />
</mesh>

<mesh name="mesh6" type="unstructured" time-varying="no">
        <points-single-var value="P1" />
        <nspace value="nsp" />
        <uniform-cells count="num_cells" data="cells" type="quad" />
</mesh>

<mesh name="mesh7" type="unstructured" time-varying="no">
        <points-multi-var value="X1,Y1,Z1" />
        <uniform-cells count="num_cells" data="C1" type="prism" />
</mesh>

<mesh name="mesh8" type="unstructured" time-varying="no">
        <points-multi-var value="X2,Y2,Z2" />
        <mixed-cells count="150,30" data="C1,C2" type="tet,pyr" />
</mesh>
```

Figure 31: Schema unstructured mesh ADIOS XML description

The above examples correspond to the ADIOS BP attributes below:

```
string   /V1/adios_schema                    attr  = "mesh5"
string   /V1/adios_schema/centering          attr  = "point"
string   /V2/adios_schema                    attr  = "mesh6"
string   /V2/adios_schema/centering          attr  = "node"
string   /V3/adios_schema                    attr  = "mesh7
string   /V3/adios_schema/centering          attr  = "cell""
string   /V4/adios_schema                    attr  = "mesh8
string   /V4/adios_schema/centering          attr  = "point"
```

```
string   /adios_schema/mesh5/type                    attr  = "unstructured"
string   /adios_schema/mesh5/time-varying            attr  = "no"

string   /adios_schema/mesh5/nspace                  attr  = "nsp"
string   /adios_schema/mesh5/points-single-var       attr  = "P1"
integer  /adios_schema/mesh5/ncsets                  attr  = 2
string   /adios_schema/mesh5/ccount0                 attr  = 100
string   /adios_schema/mesh5/cdata0                  attr  = "C1"
string   /adios_schema/mesh5/ctype0                  attr  = "triangle"
string   /adios_schema/mesh5/ccount1                 attr  = "NC2"
string   /adios_schema/mesh5/cdata1                  attr  = "C2"
string   /adios_schema/mesh5/ctype1                  attr  = "quad"

string   /adios_schema/mesh6/type                    attr  = "unstructured"
string   /adios_schema/mesh6/time-varying            attr  = "no"
string   /adios_schema/mesh6/nspace                  attr  = "nsp"
string   /adios_schema/mesh6/points-single-var       attr  = "P1"
integer  /adios_schema/mesh6/ncsets                  attr  = 1
string   /adios_schema/mesh6/ccount                  attr  = “num_cells”
string   /adios_schema/mesh6/cdata                   attr  = "C1"
string   /adios_schema/mesh6/ctype                   attr  = "quad"

string   /adios_schema/mesh7/type                    attr  = "unstructured"
string   /adios_schema/mesh7/time-varying            attr  = "no"
string   /adios_schema/mesh7/points-multi-var0       attr  = "X1"
string   /adios_schema/mesh7/points-multi-var1       attr  = "Y1"
string   /adios_schema/mesh7/points-multi-var2       attr  = "Z1"
double   /adios_schema/mesh7/points-multi-var-num    attr  = 3
integer  /adios_schema/mesh7/ncsets                  attr  = 1
string   /adios_schema/mesh7/ccount                  attr  = “num_cells”
string   /adios_schema/mesh7/cdata                   attr  = "C1"
string   /adios_schema/mesh7/ctype                   attr  = "prism"

string   /adios_schema/mesh8/type                    attr  = "unstructured"
string   /adios_schema/mesh8/time-varying            attr  = "no"
string   /adios_schema/mesh8/points-multi-var0       attr  = "X2"
string   /adios_schema/mesh8/points-multi-var1       attr  = "Y2"
string   /adios_schema/mesh8/points-multi-var2       attr  = "Z2"
integer  /adios_schema/mesh8/points-multi-var-num    attr  = 3
integer  /adios_schema/mesh8/ncsets                  attr  = 2
string   /adios_schema/mesh8/ccount0                 attr  = 150
string   /adios_schema/mesh8/cdata0                  attr  = “C1”
string   /adios_schema/mesh8/ctype0                  attr  = "tetrahedron"
string   /adios_schema/mesh8/ccount1                 attr  = 30
string   /adios_schema/mesh8/cdata1                  attr  = "C2"
string   /adios_schema/mesh8/ctype1                  attr  = "pyramid"
```

Figure 32: Schema unstructured mesh ADIOS BP attributes

Note 4: Note that nspace is mostly needed when providing 1D array to the points-single-var tag in order to correctly decompose the 1D array.

Note 5: Currently expected cell types (long and short forms) are listed in table below:

Table 1: ADIOS Visualization Schema cell types

| Long form | Short form |
|---|---|
| Line | line |
| Triangle | triangle |
| Quadrilateral | quad |
| Hexahedron | hex |
| Prism | prism |
| Tetrahedron | tet |
| Pyramid | pyr |

# 3 NO-XML API

## 3.1 Overview

As mentioned in the ADIOS manual, ADIOS provides an option of writing data without loading an XML configuration file. This set of APIs is designed to cater to output data, which is not definable from the start of the simulation; such as an adaptive code. Using the no-XML API allows users to change their IO setup at runtime in a dynamic fashion. This section discusses the details of no-XML ADIOS Visualization Schema API's. These APIs insert the appropriate schema ADIOS BP attributes into the simulation output files. They routines follow the same principles as the basic no-XML ADIOS APIs and are mostly based on or around the ***adios_define_attribute*** API call. First start by defining the ADIOS Visualization Schema version using the ***adios_define_schema_version*** API, then assign mesh to variables using the **adios_define_var_mesh** API and finally further describe variables and meshes using the other APIs. The no-XML APIs achieve the same functionality as the XML configuration and the corresponding API. Table 2 presents the full list of APIs with a brief description of each API followed by a description of their inputs in the next section.

Table 2: List of all no-XML APIs

| API | Short description |
|---|---|
| **adios_define_schema_version** | Defines the schema version |
| **adios_define_var_mesh** | Assigns a mesh to a variable |
| **adios_define_var_centering** | Defines the variable centering on the mesh nodes (or points), cells or edges |
| **adios_define_var_timesteps** | Defines the variable time steps |
| **adios_define_var_timescale** | Define the variable time scale |
| **adios_define_var_timeseriesformat** | Defines the variable time series format or padding pattern for images |
| **adios_define_var_hyperslab** | Defines a variable hyper slab (sub set or super set) |
| **adios_define_mesh_timeSteps** | Define the time steps at the mesh level (for all variables on that mesh) |
| **adios_define_mesh_timeScale** | Define the time scale at the mesh level (for all variables on that mesh) |
| **adios_define_mesh_timeSeriesFormat** | Define the time series formatting at the mesh level (for all variables on that mesh) |
| **adios_define_mesh_group** | Indicates where (which ADIOS group) mesh variables are stored |
| **adios_define_mesh_file** | Indicates where mesh variables are written |
| **adios_define_mesh_uniform** | Defines a uniform mesh |
| **adios_define_mesh_rectilinear** | Defines a rectilinear mesh |
| **adios_define_mesh_structured** | Defines a structured mesh |
| **adios_define_mesh_unstructured** | Define an unstructured mesh |

## 3.2 Version

***int adios_define_schema_version(struct adios_group_struct * new_group, char * schema_version);***

Figure 33: Define schema version

Input:

- pointer to the ADIOS group structure
- string containing the version (ex: "1.0")

## 3.3 Variables

### 3.3.1 Assigning meshes to variables

***int adios_define_var_mesh(int64_t ptr_new_group, char * varname, char * varpath, char * meshname);***

Figure 34: Assign mesh to variable

Input:

- pointer to the ADIOS group structure
- string containing the variable name
- string containing the variable path
- string containing the mesh name

### 3.3.2 More options for variables:

#### 3.3.2.1 Centering: (node/point, cell or edge)-centered

***int adios_define_var_centering(int64_t ptr_new_group, char * varname, char * varpath, char * centering);***

Figure 35: Define variable centering

Input:

- pointer to the ADIOS group structure
- string containing the variable name
- string containing the variable path

- string containing the centering information (node or point, cell or edge)

#### 3.3.2.2 Time

##### 3.3.2.2.1 time-steps

***int adios_define_var_timesteps (const char * timesteps, struct adios_group_struct * new_group, const char * varname, const char * varpath);***

Figure 36: Defining variable time steps

Input:

- string containing time step
- ADIOS group structure
- string containing the variable name
- string containing the variable path

##### 3.3.2.2.2 time-scale

***int adios_define_var_timescale (const char * timescale, struct adios_group_struct * new_group, const char * varname, const char * varpath);***

Figure 37: Defining variable time steps

Input:

- string containing time scale
- ADIOS group structure
- string containing the variable name
- string containing the variable path

##### 3.3.2.2.3 time-format-series

***int adios_define_var_timeseriesformat (const char * timeseries, struct adios_group_struct * new_group, const char * varname, const char * varpath);***

Figure 38: Defining the time series format

Input:

- string containing time series format (integers)

- ADIOS group structure
- string containing the variable name
- string containing the variable path

#### 3.3.2.2.4 Hyperslab

***int adios_define_var_hyperslab ( const char * hyperslab,struct adios_group_ struct * new_group, const char * varname, const char * varpath);***

Figure 39: Defining variable hyperslabs

Input:

- string containing hyperslab (a number, a range or 3 numbers)
- ADIOS group structure
- string containing the variable name
- string containing the variable path

## 3.4 Meshes

### 3.4.1 Time

***int adios_define_mesh_timeSteps (const char * timesteps, struct adios_group_struct * new_group, const char * name);***

Figure 40: Defining mesh time steps

Input:

- string containing time steps (a integer, a range of integers in all dimensions or 3 integers)
- ADIOS group structure
- string containing the mesh name

***int adios_define_mesh_timeScale (const char * timescale, struct adios_group_struct * new_group, const char * name);***

Figure 41: Defining mesh time scale

Input:

- string containing time steps (a numbers, a range of numbers in all dimensions or 3 numbers)
- ADIOS group structure
- string containing the mesh name

***int adios_define_mesh_timeSeriesFormat (const char * timeseries, struct adios_group_struct * new_group, const char * name);***

Figure 42: Defining the mesh time series format

Input:

- string containing the number of padding zeros (time-series-format)
- ADIOS group structure
- string containing the mesh name

### 3.4.2 Mesh file and mesh group

***int adios_define_mesh_group(int64_t ptr_new_group, char * name, char * group);***

Figure 43: Defining mesh group

Input:

- Pointer to the ADIOS group structure
- String containing the mesh name
- string containing the ADIOS group name

***int adios_define_mesh_file(int64_t ptr_new_group, char * name, char * file);***

Figure 44: Defining mesh file

Input:

- Pointer to the ADIOS group structure
- String containing the mesh name
- string containing the file where mesh variables are written

### 3.4.3 Uniform Mesh

***int adios_define_mesh_uniform (char * dimensions, char * origin, char * spacing, char * maximum, struct adios_group_struct * new_group , const char * name);***

Figure 45: Defining a uniform mesh

Input:

- String containing the mesh dimensions in ADIOS format
- String containing the mesh origins (in all dimensions)
- String containing the mesh spacings (between points in all dimensions)
- String containing the mesh maximums (in all dimensions)
- ADIOS group structure
- String containing mesh name

### *3.4.4 Rectilinear Mesh*

***int adios_define_mesh_rectilinear (char * dimensions, char * coordinates,struct adios_group_struct * new_group,const char * name);***

Figure 46: Defining a rectilinear mesh

Input:

- String containing the mesh dimensions in ADIOS format
- String containing the variable name(s) pointing to the mesh coordinates
- ADIOS group structure
- String containing mesh name

### *3.4.5 Structured Mesh*

***int adios_define_mesh_structured(char * dimensions, char * nspace, char * points, struct adios_group_struct * new_group, char * name);***

Figure 47: Defining a structured mesh

Input:

- String containing the mesh dimensions in ADIOS format

- String containing nspace (an integer or the variable pointing to an integer)
- String containing the variable name(s) pointing to the mesh coordinates
- ADIOS group structure
- String containing mesh name

### 3.4.6 Unstructured Mesh

***int adios_define_mesh_unstructured(char *nspace, char *npoints, char *points, char * data, char * count, char * type, struct adios_group_struct * new_group, const char * name);***

Figure 48: Defining an unstructured mesh

Input:

- String containing nspace (an integer or the variable pointing to an integer)
- String containing the variable name(s) pointing to the mesh coordinates
- String containing the variable name(s) pointing to the mesh cell data (arrays)
- String containing numbers or variable names(s) pointing to the mesh cell counts
- String containing cell types or variable names(s) pointing to the mesh cell types (refer to short and long form of cell types in table 1)
- ADIOS group structure
- String containing mesh name