

VMAP

Virtual Material Modelling in Manufacturing

STANDARD SPECIFICATIONS



Version no.: 0.4.0
Edited by: Fraunhofer SCAI - Priyanka Gulati

Current Version is for BETA testing only.



List of Authors

4a Engineering	Bernhard Jilka
Wittmann Battenfeld GmbH	Peter Reithhofer
MSC Software Belgium S.A.	Filipp Pühringer
Convergent Manufacturing Technologies Inc.	<i>undefined</i>
Audi AG	Anthony Floyd
Dr. Reinold Hagen Stiftung/Hagen Engineering GmbH	Tim Bergmann
DYNAmore GmbH	Olaf Bruch
EDAG Engineering GmbH	Patrick Michels
ESI Software Germany GmbH	Christian Liebold
Fraunhofer SCAI	Tolga Usta
inuTech GmbH	Lukasz Lasek
Karlsruhe Institute of Technology (KIT)	Sebastian Müller
Kautex Maschinenbau GmbH	André Öckerath
NAFEMS Deutschland, Österreich, Schweiz GmbH	Klaus Wolf
RIKUTEC Richter Kunststofftechnik GmbH & Co. KG	Priyanka Gulati
Robert Bosch GmbH	Giannoula Mitrou
Simcon kunststofftechnische Software GmbH	Niels Sondergaard
Delft University of Technology	Constantin Krauß
DevControl B.V.	Luise Kärger
In Summa Innovations b.v.	<i>undefined</i>
KE-Works	Gino Duffett
Material innovation institute M2i	Daniel Grotenburg
MSC Softwar Benelux	Joachim Strauch
Philips	Matthias De Monte
Reden BV	Liyona Bonakdar
University of Groningen	Max Mades
BETA CAE System Internation AG	Jilt Sietsma
e-Xstream engineering	Cor de Vries
Sintratec	Maarten Oudendijk
	Bastiaan Beijer
	Anouar Krairi
	Jesus Mediavilla
	Pieter Vosbeek
	Harm Kooiker
	Edwin Lamers
	Lambert Russcher
	Anthony Vakis
	George Mokios
	Thanasis Fassas
	Laurent Adam
	Christian von Burg



License of this document

Copyright © 2017-2020 VMAP Project Partners and
2020 onwards VMAP Standards Community

For more details about VMAP Standards Community please visit
<https://www.vmap.eu.com/community/>.

If you have any feedback or queries, send them to info@vmap.eu.com

How to Use This Booklet

The aim of this book is to help you understand the VMAP Standard Specifications. If you have never heard about VMAP, then please read the VMAP General Information. It explains the VMAP Project and provides an overview of the VMAP Use Cases. This book introduces the ongoing VMAP specifications. For the CAE Tool Engineers and Software Engineers and Application Engineers, this book provides ongoing specifications and implementation specifications.

Overview of Booklet Structure

This VMAP information is divided into two complimentary documents: VMAP General Information and VMAP Standard Specification Documentation.

VMAP General Information contains:

- Chapter 1 introduces and explains the VMAP Standard, the guiding idea and the definition.
- Chapter 2 throws light on State of the Art.
- Chapter 3 provides a brief account of the requirement analysis, which led to the inception of VMAP.
- Chapter 4 introduces the Software Architecture of VMAP and the output technology used by VMAP.
- Chapter 5 describes the Use Cases that were used to demonstrate the usefulness and capacity of the VMAP Standards.

VMAP Standard Specification Documentation contains:

- Chapter 1 introduces the Software Architecture of VMAP and the output technology used by VMAP.
- Chapter 2 shows how to start using the API.
- Chapter 3 describes the relationship among the C++ structures defined in VMAP Standard I/O Library.
- Chapter 4 gives an account of the VMAP Standard I/O Library or VMAP Standard API.
- Chapter 5 contains information on compiling the VMAP Standard API.
- Chapter 6 provides a possibility to implement your own VMAP I/O Library. This chapter should be used carefully, since the **Nomenclature** and structure used by VMAP is explained in detail. It is essential to follow this **Nomenclature** and structure to get the correct VMAP Standard file.
- Chapter 7 shows the snapshots from the HDF5 Viewer of a standard VMAP .h5 file.



- Chapter 8 further elaborates on the specifications. It describes the standard VMAP Element definitions, which are already part of the factory, and how to define one of your own elements.
- Chapter 9 further elaborates on the specifications with standard VMAP Integration Type definitions and how to define one of your own integration type.
- Chapter 10 provides some basic tutorials on how to use the VMAP Standard API.
- Chapter 11 defines simple test cases which could be used by a developer or an end user.

Target Audiences

To be able to use the VMAP Documentation efficiently, prior knowledge of modelling and simulation is required. The user should have a hands on experience of any CAE Tool, or has at least the basic knowledge about Finite Element Analysis. Users and Developers may have different needs so we try to list the relevant documentation parts in the table below.

VMAP Documentation Chapters of Interest	
CAE Tool End Users to understand the VMAP Standard background, format and testing.	
VMAP General Information	VMAP Standards Document
1	11
2	
3	
4	
5	
CAE Tool Developers to understand and implement VMAP Standard API within their own software tool.	
VMAP General Information	VMAP Standards Document
1	2
4	4
	7
	8
	9
	10
	11
CAE Tool Developers to implement their own VMAP I/O Library instead of using the VMAP Standard I/O Library. These users should check every detail carefully to implement the correct VMAP Standard, especially noting the implementation of element types and integration types.	
VMAP General Information	VMAP Standards Document
1	6
4	7
	8
	9
	10
	11



Abbreviations

CAE	Computer Aided Engineering
FEM	Finite Element Methods
FEA	Finite Element Analysis
SWIG	Simplified Wrapper and Interface Generator
API	Application Programming Interface

Contents

1	VMAP Software Architecture	15
1.1	VMAP Interface to CAE Tools	16
1.2	SWIG	17
1.3	HDF5 technology	17
2	VMAP Standard API Build	19
2.1	VMAP Header & Source Code Files	19
2.1.1	VMAPFile.h	19
2.1.2	VMAPFile.cxx	19
2.1.3	VMAP.h	19
2.1.4	VMAP.cxx	19
2.1.5	VMAPH5Tools.h	20
2.2	VMAP Standard API Calling Sequence - Creating VMAP .h5 File .	20
2.2.1	Writing System Data	20
2.2.2	Writing an FE Mesh	21
2.2.3	Writing State Variables	22
3	VMAP Standard API - Data Structure Relationship	23
3.1	Data Structure Dependency	23
3.1.1	VMAP Group	23
3.1.2	GEOMETRY Group	23
3.1.3	VARIABLES Group	24
3.1.4	SYSTEM Group	25
3.1.5	POINTS Group	26
3.1.6	ELEMENTS Group	26
3.1.7	MYELEMENTS Data Set	26
3.1.8	ELEMENTTYPES Data Set	27
3.1.9	State Variables Group	27
3.1.10	UNITSYSTEM Attribute	28
4	VMAP Standard API	29
4.1	Namespace Documentation	29
4.2	VMAP Namespace Reference	29
4.3	File Documentation	31
4.4	src/VMAP.cxx File Reference	31
4.5	src/VMAP.h File Reference	32



4.6	src/VMAPDeclspec.h File Reference	33
4.6.1	Macro Definition Documentation	33
4.7	src/VMAPElementTypeFactory.cxx File Reference	33
4.8	src/VMAPElementTypeFactory.h File Reference	33
4.9	src/VMAPException.cxx File Reference	33
4.10	src/VMAPException.h File Reference	34
4.11	src/VMAPFile.cxx File Reference	34
4.12	src/VMAPFile.h File Reference	36
4.13	src/VMAPH5Tools.h File Reference	37
4.13.1	Macro Definition Documentation	37
4.14	src/VMAPIntegrationTypeFactory.cxx File Reference	37
4.15	src/VMAPIntegrationTypeFactory.h File Reference	38
5	Compiling & Linking The API	39
5.1	Overview	39
5.2	Basic Build Requirements for VMAP Standard API	39
5.3	Additional Requirements for VMAP Standard API	40
5.4	Build Using Cmake GUI	40
5.5	VMAP Integration	42
5.5.1	C++ Library	42
5.5.2	Python Interface	42
5.5.3	Java Interface	43
5.5.4	C# Interface	43
6	Implementation Specifications	44
6.1	VMAP Group	44
6.1.1	VERSION Attribute	45
6.2	GEOMETRY Group	46
6.3	<PART-ID> Group	46
6.3.1	NAME Attribute	47
6.4	POINTS Group	47
6.4.1	MYCOORDINATESYSTEM Attribute	48
6.4.2	MYSIZE Attribute	48
6.4.3	MYCOORDINATES Data Set	49
6.4.4	MYIDENTIFIERS Data Set	49
6.5	ELEMENTS Group	50
6.5.1	MYSIZE Attribute	51
6.5.2	MYELEMENTS Data Set	51
6.6	VARIABLES Group	52
6.7	STATE-<n> Group	53
6.7.1	MYSTATENAME Attribute	54
6.7.2	MYTOTALTIME Attribute	54
6.7.3	MYSTATETIME Attribute	54
6.7.4	MYSTATEINCREMENT Attribute	54
6.8	<PART-ID> Group	55

6.8.1	MYSIZE Attribute	55
6.9	TEMPERATURE Group	55
6.9.1	MYCOORDINATESYSTEM Attribute	57
6.9.2	MYDIMENSION Attribute	57
6.9.3	MYENTITY Attribute	58
6.9.4	MYIDENTIFIER Attribute	58
6.9.5	MYINCREMENTVALUE Attribute	59
6.9.6	MYLOCATION Attribute	59
6.9.7	MYMULTIPLICITY Attribute	59
6.9.8	MYTIMEVALUE Attribute	60
6.9.9	MYUNIT Attribute	60
6.9.10	MYVARIABLEDESCRIPTION Attribute	60
6.9.11	MYVARIABLENAME Attribute	61
6.9.12	MYVALUES Data Set	61
6.9.13	MYINTEGRATIONTYPES Data Set	62
6.10	SYSTEM Group	62
6.10.1	COORDINATESYSTEM Data Set	63
6.10.2	ELEMENTTYPES Data Set	65
6.10.3	INTEGRATIONTYPES Data Set	67
6.10.4	METADATA Data Set	68
6.10.5	UNITS Data Set	69
6.10.6	UNITSYSTEM Data Set	70
7	Storage Format	73
7.1	.h5 File View	74
7.2	VMAP Group View	75
7.2.1	VERSION Attribute View	76
7.3	GEOMETRY Group View	76
7.4	<PART-ID> Group View	77
7.5	POINTS Group View	79
7.5.1	MYCOORDINATES Dataset	80
7.5.2	MYIDENTIFIERS Dataset	81
7.6	ELEMENTS Group View	82
7.6.1	MYELEMENTS Dataset	84
7.7	VARIABLES Group View	85
7.8	STATE-<n> Group View	85
7.9	<PART-ID> Group View	86
7.10	STRAIN-CAUCHY Group View	87
7.10.1	MYGEOMETRYIDS Dataset	87
7.10.2	MYVALUES Dataset	89
7.10.3	MYINTEGRATIONTYPES Dataset	90
7.11	SYSTEM Group View	90
7.11.1	COORDINATESYSTEM Dataset	92
7.11.2	ELEMENTTYPES Dataset	93
7.11.3	INTEGRATIONTYPES Dataset	94



7.11.4	METADATA Dataset	95
7.11.5	UNITS Dataset	96
7.11.6	UNITSYSTEM Dataset	97
8	Element Definition Specifications	98
8.1	USER_DEFINED	98
8.2	POINT	99
8.3	LINE_2	99
8.4	LINE_3	100
8.5	LINE_4	100
8.6	TRIANGLE_3	100
8.7	TRIANGLE_4	101
8.8	TRIANGLE_6	101
8.9	QUAD_4	102
8.10	QUAD_8	102
8.11	QUAD_9	103
8.12	TETRAHEDRON_4	104
8.13	TETRAHEDRON_5	104
8.14	TETRAHEDRON_10	105
8.15	TETRAHEDRON_11	105
8.16	PYRAMID_5	106
8.17	PYRAMID_6	106
8.18	PYRAMID_13	107
8.19	WEDGE_6	107
8.20	WEDGE_15	108
8.21	HEXAHEDRON_8	108
8.22	HEXAHEDRON_9	109
8.23	HEXAHEDRON_20	109
8.24	HEXAHEDRON_21	110
8.25	HEXAHEDRON_27	110
8.26	POLYGON	111
8.27	POLYHEDRON	111
9	Integration Type Definition Specifications	112
9.1	USER_DEFINED	112
9.2	GAUSS_1	113
9.3	GAUSS_2	113
9.4	GAUSS_3	113
9.5	GAUSS_4	114
9.6	GAUSS_5	114
9.7	GAUSS_6	114
9.8	GAUSS_7	114
9.9	GAUSS_8	115
9.10	GAUSS_9	115
9.11	GAUSS_10	115

9.12	GAUSS_11	116
9.13	GAUSS_12	116
9.14	GAUSS_13	116
9.15	GAUSS_14	117
9.16	GAUSS_15	117
9.17	GAUSS_16	117
9.18	LOBATTO_1	118
9.19	LOBATTO_2	118
9.20	LOBATTO_3	118
9.21	LOBATTO_4	118
9.22	LOBATTO_5	118
9.23	LOBATTO_6	119
9.24	LOBATTO_7	119
9.25	LOBATTO_8	119
9.26	LOBATTO_9	119
9.27	LOBATTO_10	120
9.28	LOBATTO_11	120
9.29	LOBATTO_12	120
9.30	LOBATTO_13	121
9.31	LOBATTO_14	121
9.32	LOBATTO_15	121
9.33	LOBATTO_16	122
9.34	SIMPSON_1	122
9.35	SIMPSON_3	122
9.36	SIMPSON_5	122
9.37	SIMPSON_7	123
9.38	SIMPSON_9	123
9.39	SIMPSON_11	123
9.40	SIMPSON_13	124
9.41	SIMPSON_15	124
9.42	TRAPEZOIDAL_1	124
9.43	TRAPEZOIDAL_2	124
9.44	TRAPEZOIDAL_3	125
9.45	TRAPEZOIDAL_4	125
9.46	TRAPEZOIDAL_5	125
9.47	TRAPEZOIDAL_6	125
9.48	TRAPEZOIDAL_7	126
9.49	TRAPEZOIDAL_8	126
9.50	TRAPEZOIDAL_9	126
9.51	TRAPEZOIDAL_10	126
9.52	TRAPEZOIDAL_11	127
9.53	TRAPEZOIDAL_12	127
9.54	TRAPEZOIDAL_13	127
9.55	TRAPEZOIDAL_14	128
9.56	TRAPEZOIDAL_15	128

9.57	GAUSS_TRIANGLE_1	128
9.58	GAUSS_TRIANGLE_3	129
9.59	GAUSS_TRIANGLE_4	129
9.60	GAUSS_TRIANGLE_6	129
9.61	GAUSS_QUAD_1	129
9.62	GAUSS_QUAD_4	130
9.63	GAUSS_QUAD_9	130
9.64	NODES_TRIANGLE_3	130
9.65	NODES_TRIANGLE_6	131
9.66	NODES_QUAD_4	131
9.67	NODES_QUAD_8	131
9.68	NODES_QUAD_9	132
9.69	GAUSS_LAYERED_HEXAHEDRON_4	132
9.70	GAUSS_TETRAHEDRON_1	132
9.71	GAUSS_TETRAHEDRON_4	132
9.72	GAUSS_TETRAHEDRON_8	133
9.73	GAUSS_TETRAHEDRON_11	133
9.74	GAUSS_TETRAHEDRON_15	133
9.75	GAUSS_PYRAMID_1	134
9.76	GAUSS_PYRAMID_5	134
9.77	GAUSS_PYRAMID_8	134
9.78	GAUSS_WEDGE_1	135
9.79	GAUSS_WEDGE_2	135
9.80	GAUSS_WEDGE_6	135
9.81	GAUSS_WEDGE_8	135
9.82	GAUSS_WEDGE_9	136
9.83	GAUSS_WEDGE_18	136
9.84	GAUSS_HEXAHEDRON_1	137
9.85	GAUSS_HEXAHEDRON_8	137
9.86	GAUSS_HEXAHEDRON_27	138
9.87	NODES_TETRAHEDRON_4	139
9.88	NODES_TETRAHEDRON_10	139
9.89	NODES_WEDGE_6	139
9.90	NODES_WEDGE_15	140
9.91	NODES_PYRAMID_5	140
9.92	NODES_HEXAHEDRON_8	140
9.93	NODES_HEXAHEDRON_20	141
9.94	NODES_HEXAHEDRON_27	142
9.95	Combined Integration Types	142
9.96	Composite Integration Types	142

10 Tutorials 143

10.1	Creating or Opening a VMAP .h5 File	143
10.1.1	Parameters:	143
10.1.2	Returns:	143



10.2	Create a Group in VMAP File	144
10.2.1	Parameters:	144
10.2.2	Returns:	144
10.3	Get Sub-Groups from VMAP File	144
10.3.1	Parameters:	144
10.3.2	Returns:	145
10.4	Check if a Group exists in VMAP File	145
10.4.1	Parameters:	145
10.4.2	Returns:	145
10.5	Write the Version to VMAP File	145
10.5.1	Parameters:	145
10.5.2	Returns:	146
10.6	Read the Version of VMAP File	146
10.6.1	Parameters:	146
10.6.2	Returns:	146
11	Simple Test Cases	147
11.1	Break Forming of a Metal Bracket	147
11.1.1	General Description of the test case “Break Forming of a Metal Bracket”	147
11.1.2	Software Implementation	149
11.1.3	Results	150
11.1.4	Extension of the test case “Break Forming of a Metal Bracket”	151
	Bibliography	153
	Appendices	154
	Appendix A	154

Chapter 1

VMAP Software Architecture

This chapter explains the VMAP software architecture (Figure 1.1), briefly going through all the layers. The further chapters then focus on each layer in detail.

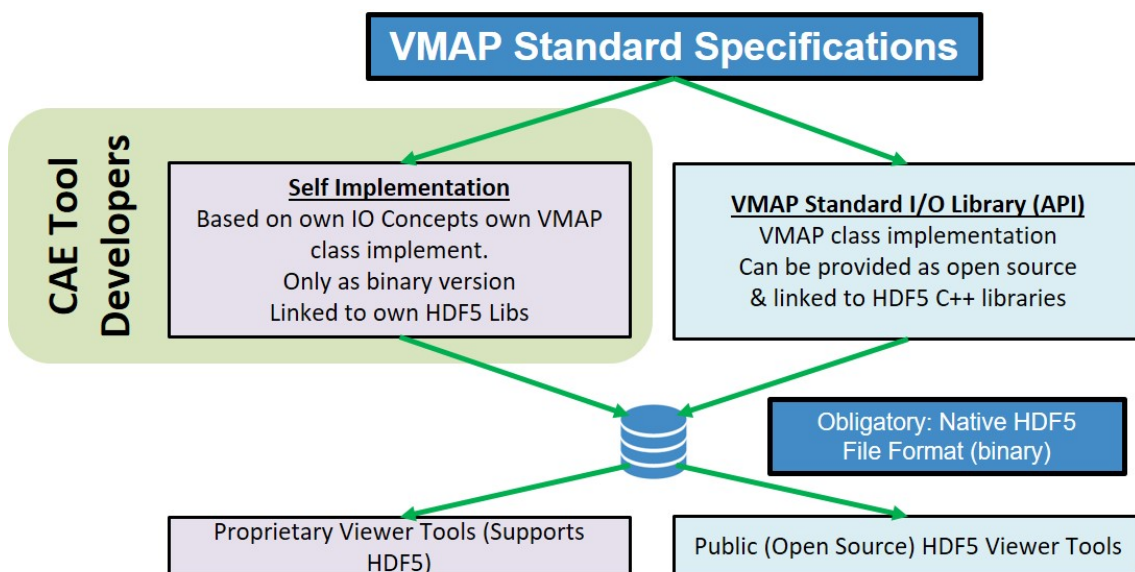


Figure 1.1: VMAP Software Architecture

VMAP Standard Specifications are at the core of the software architecture. VMAP offers two possibilities for any user. First, is to use the VMAP Standard Specifications via the VMAP Standard I/O Library (API) built in C++. The second option is to implement your own VMAP I/O classes using the VMAP Standard Specifications. The only obligation is to use the native HDF5 file format as the output. HDF5 file format is an optimal and apt output option for VMAP because HDF5 Viewer is an open source tool, just like VMAP Standard Specifications are open source. Section 1.3 explains HDF5 Technology in detail.

The VMAP Standard I/O Library or **VMAP Standard API** is explained in detail in chapters 3 & 4. The option to implement your own VMAP I/O Library is explained with schematic diagrams in chapter 6.

1.1 VMAP Interface to CAE Tools

Almost all CAE tools offer API, these API are used by ISVs to build codes. ISV codes written in C++ can be directly linked to the 'VMAP Standard API'. ISV codes written in Python, Java, C# or FORTRAN utilize the 'VMAP Standard API' through a language specific interface. For Python, Java and C# such a language specific interface can be automatically generated using the **Simplified Wrapper and Interface Generator (SWIG)**(Section 1.2). For FORTRAN the language specific interface is possible but must be written manually. Figure 1.2 shows the extended software architecture.

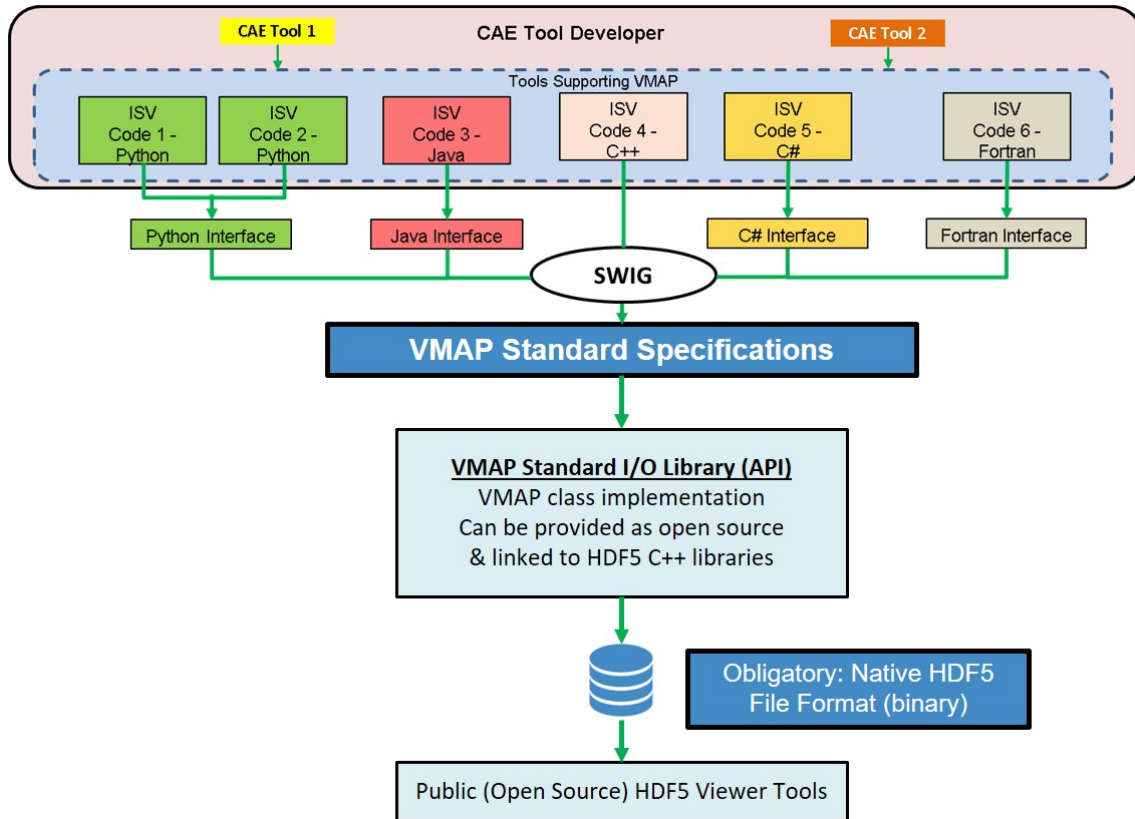


Figure 1.2: Extended VMAP Software Architecture

The VMAP Standard API and its role in a chain CAE simulation process is represented in (Figure 1.3). The image shows two simulations, Blow Moulding simulation carried out using Code A and Cooling simulation carried out using Code B. The cooling simulation requires the output result of the blow moulding simulation. Such a situation arises very often in the industry, where results of one simulation are re-

quired to carry out another simulation. Since, there are multiple CAE tools (Codes) available in the market, each time a combination of tools is used a new specific converter needs to be developed. This is where VMAP Standard comes into the picture, with all CAE tools providing VMAP Standard format as one of the output options, the specific converters will become unnecessary. VMAP Standard will facilitate reusability and thus, time saving. Since VMAP Standard is currently in development phase, the converter is replaced by an external VMAP converter. As the standard is completely formalised, the VMAP Standard API can be directly integrated into the CAE tool.

CAE tools which additionally require a Mapper to map data from Simulation Model A to Simulation Model B, can also have the Mapper integrated with the VMAP Standard API.

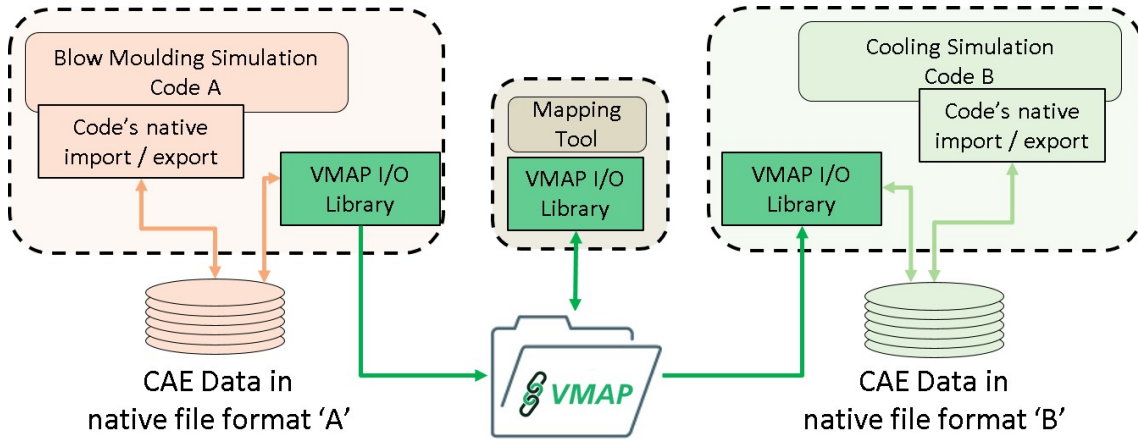


Figure 1.3: VMAP Standard API in CAE chain simulation process

1.2 SWIG

SWIG is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages. SWIG is used with different types of target languages including common scripting languages such as JavaScript, Perl, PHP, Python, Tcl and Ruby. The list of supported languages also includes non-scripting languages such as C#. SWIG is most commonly used to create high-level interpreted or compiled programming environments, user interfaces, and as a tool for testing and prototyping C/C++ software. SWIG is typically used to parse C/C++ interfaces and generate the 'glue code' required for the above target languages to call into the C/C++ code [5]

1.3 HDF5 technology

The VMAP interface and transfer file relies on the HDF5 technology. The Hierarchical Data Format (HDF) implements a model for managing and storing data.

The model includes an abstract data model and an abstract storage model (the data format), and libraries to implement the abstract model and to map the storage model to different storage mechanisms. The HDF5 Library provides a programming interface to a concrete implementation of the abstract models. The library also implements a model of data transfer, an efficient movement of data from one stored representation to another stored representation. The figure below illustrates the relationships between the models and implementations. This chapter explains these models in detail.

The Hierarchical Data Format version 5 (HDF5), is an open source file format that supports large, complex, heterogeneous data. HDF5 uses a "file directory" like structure that allows you to organize data within the file in many different structured ways, as you might do with files on your computer. The HDF5 format also allows for embedding of metadata making it self-describing.

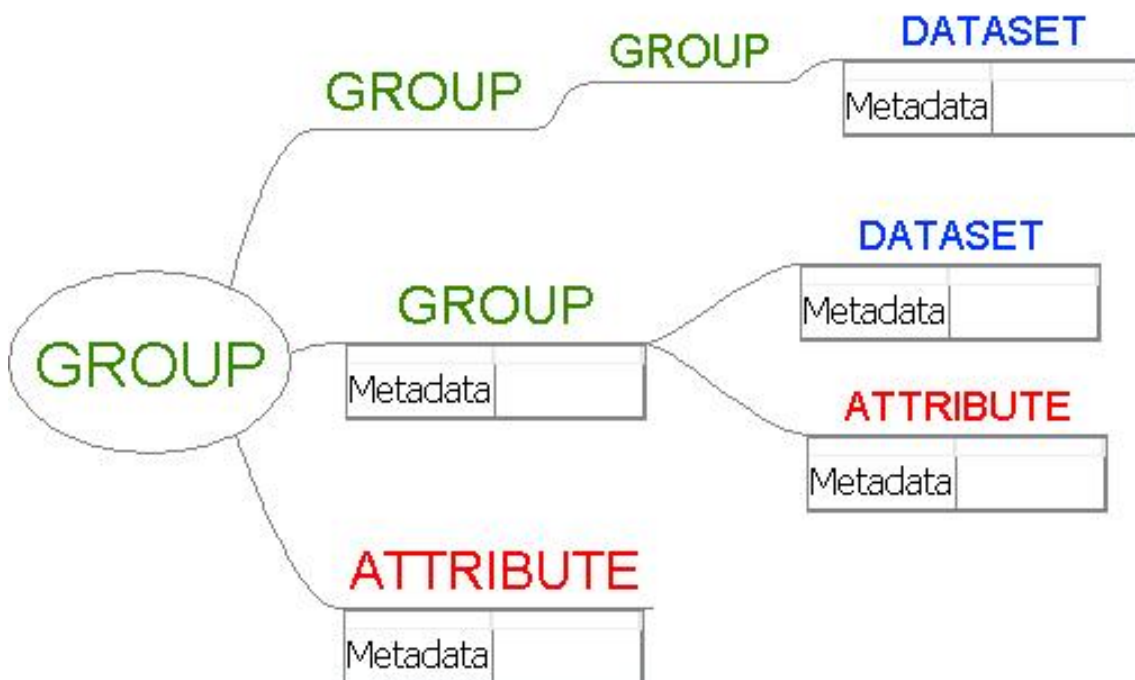


Figure 1.4: HDF5 file format

Chapter 2

VMAP Standard API Build

This chapter outlines the steps to use the VMAP Standard API. The header and source code files provided as VMAP standard are explained in following section.

2.1 VMAP Header & Source Code Files

2.1.1 VMAPFile.h

This header file contains all the read & write functions of VMAP Standard API. These functions are used to read/write from/to the .h5 file. This file only contains the declaration of these functions.

2.1.2 VMAPFile.cxx

This source code file initializes the .h5 file format and defines all the read & write functions declared in VMAPFile.h header file.

2.1.3 VMAP.h

This header file contains all the arguments read and written by the VMAPFile.h. These arguments are defined using **struct** reference, default **constructor** & **destructor** and **get** & **set** functions to assign values to the attributes of the structure. Only the declarations are part of this file, but no definitions.

2.1.4 VMAP.cxx

This source code file initializes the **constructor** & **destructor** defined in VMAP.h header file. It also defines all the **get** & **set** functions of each **struct** reference defined in VMAP.h header file.

2.1.5 VMAPH5Tools.h

This header file defines and declares certain H5 specific tools, which are used by the VMAP I/O.

2.2 VMAP Standard API Calling Sequence - Creating VMAP .h5 File

A VMAP .h5 file is created using `VMAPIFile` class which is defined in `VMAPIFile.h`. Figure 2.1 shows a sample initiation code for VMAP API.

```
#include "H5Cpp.h"
#include "VMAP.h"
#include "VMAPIFile.h"

int main(int argc, char** argv) {
    using namespace H5;
    using namespace VMAP;

    Initialize();
    VMAPIFile vmapFile("/tmp/testfile.h5");
    ...
    ...
    ...
    return 0;
}
```

Figure 2.1: C++ Code to call VMAP API

VMAP Standard API calling sequence is further sub-divided into three categories - writing system data, mesh and state variables to the VMAP .h5 file. All the functions, defined in the following flowcharts, are declared in `VMAPIFile.h` and all the structures are declared in `VMAP.h`.

2.2.1 Writing System Data

The following flowchart (Figure 2.2) shows the calling sequence to write your own VMAP .h5 file with the mandatory system data.

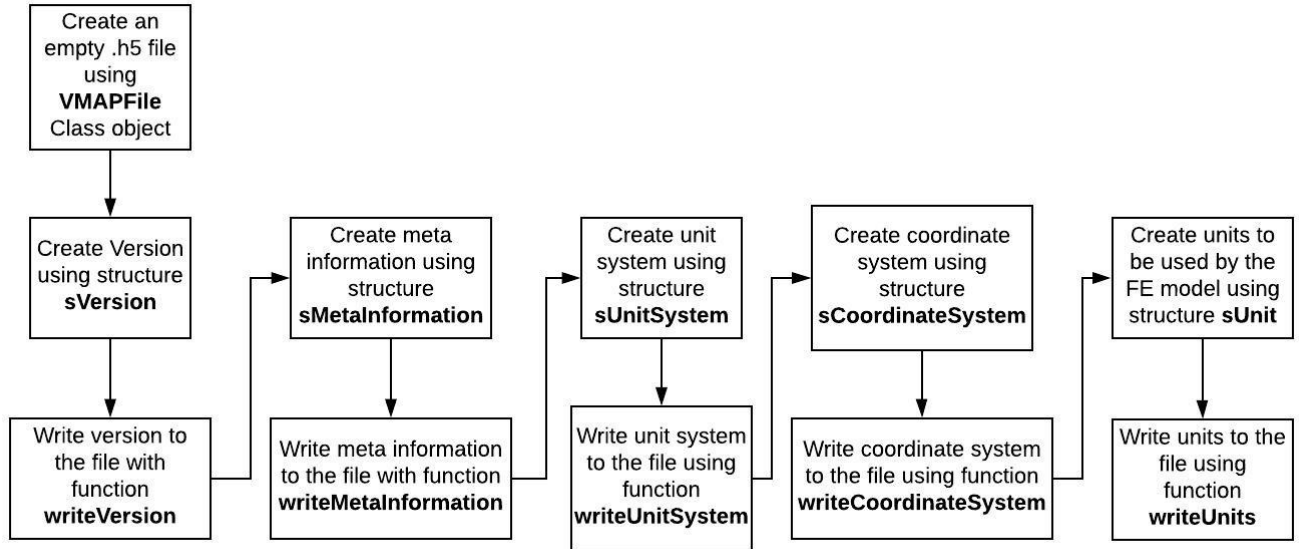


Figure 2.2: Write VMAP .h5 File

2.2.2 Writing an FE Mesh

The following flowchart (Figure 2.3) can be followed to create a FE mesh.

VMAP offers the choice to create an Element Type or use one of the Element Types defined in **VMAPElementTypeFactory.cxx**. The factory offers over 30 different types of 1D, 2D & 3D elements. The specifications of these element types and specification to write your own element type can be found in Chapter 8. Additionally, VMAP offers the possibility to create an Integration Type or use one of the Integration Type defined in **VMAPIntegrationTypeFactory.cxx** (Chapter 9)

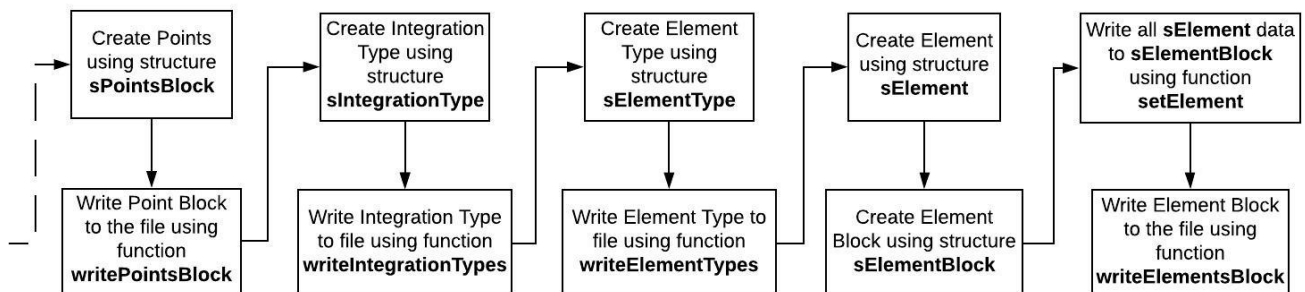


Figure 2.3: Write Mesh to file

It is necessary to define the points block, integration types and element types, to be able to define elements and further build element blocks.

2.2.3 Writing State Variables

A State Variable can be defined Globally, or over one/more Point(s), Element(s), Element Face(s), Integration Point(s). The following flowchart shows the structure and function used for creating and writing State Variables.

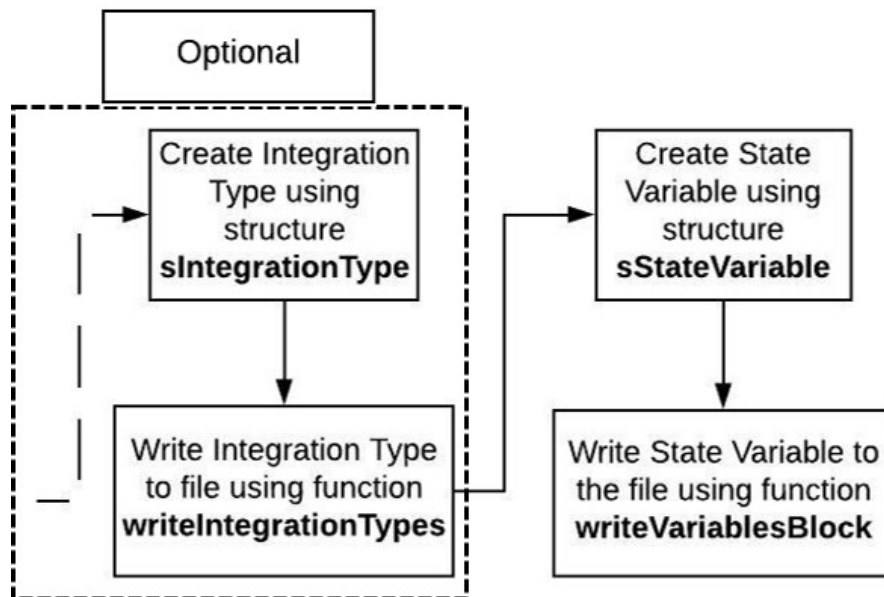


Figure 2.4: Write State Variable

Chapter 3

VMAP Standard API - Data Structure Relationship

This chapter explains the relationship between various structures and classes defined in the VMAP Standard I/O Library. All structures and functions are defined within the namespace VMAP.

3.1 Data Structure Dependency

The data structure dependency is explained in the following sections by means of schematic diagrams. The groups can contain several other groups, datasets and attributes. The VMAP API defines the groups, datasets and attributes with the help of structures defined in C++. In the descriptions below, for all attributes and datasets and some groups, the structure name is defined in brackets e.g. (sStructure).

3.1.1 VMAP Group

VMAP Group in VMAP Standard I/O library contains data from groups - **GEOMETRY**, **MATERIAL**, **VARIABLES** & **SYSTEM** and attribute - **VERSION** (sVersion). Figure 3.1 shows a schematic of the VMAP group and the data it contains.

3.1.2 GEOMETRY Group

The **GEOMETRY** group in VMAP Standard I/O library contains sub-group <PART-ID>, within this lies groups **POINTS** (sPointsBlock) and **ELEMENTS** (sElementBlock). Figure 3.2 shows a schematic of group **GEOMETRY**.

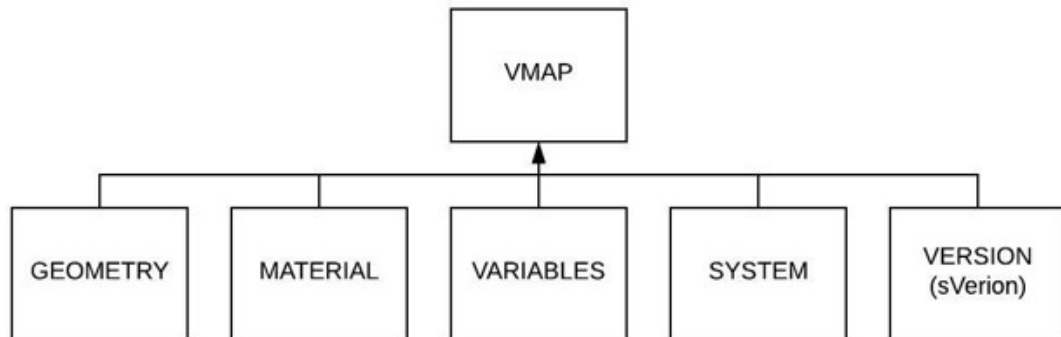


Figure 3.1: VMAP Group Dependency

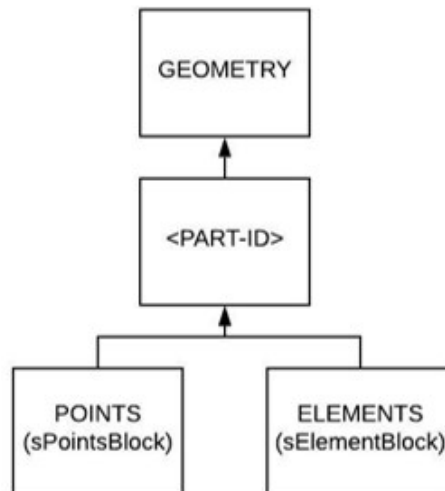


Figure 3.2: GEOMETRY Group Dependency

3.1.3 VARIABLES Group

The **VARIABLES** group in VMAP Standard I/O library contains sub-groups **STATE-*<n>*** and further **<PART-ID>**, within this lies the state variables (**sStateVariable**). Figure 3.3 shows a schematic of group **VARIABLES**.

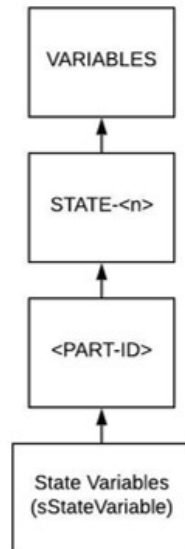


Figure 3.3: VARIABLES Group Dependency

3.1.4 SYSTEM Group

The **SYSTEM** group in VMAP Standard I/O library contains datasets - **METADATA** (sMetaInformation), **UNITSYSTEM** (sUnitSystem), **COORDINATESYSTEM** (sCoordinateSystem), **ELEMENTTYPES** (sElementTypes), **INTEGRATIONTYPES** (sIntegrationTypes) and **UNITS** (sUnit). Figure 3.4 shows a schematic of group **SYSTEM**.

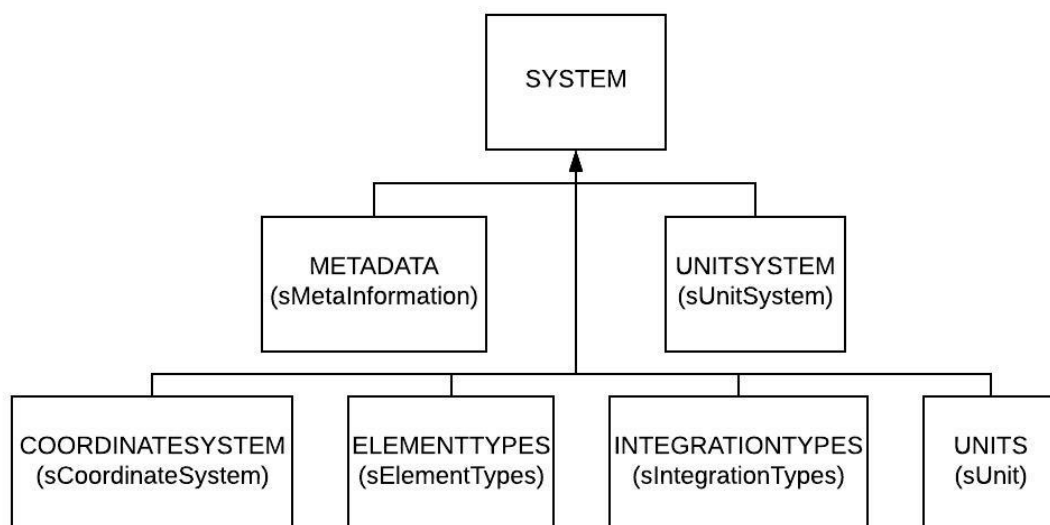


Figure 3.4: SYSTEM Group Dependency

3.1.5 POINTS Group

The POINTS group (sPointsBlock) in VMAP Standard I/O library requires data from the data set COORDINATESYSTEM (sCoordinateSystem). Figure 3.5 shows a schematic of group POINTS.

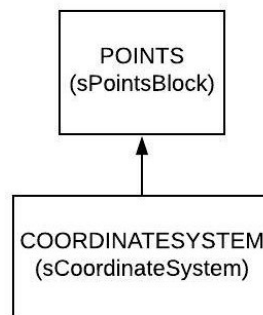


Figure 3.5: POINTS Group Dependency

3.1.6 ELEMENTS Group

The ELEMENTS group (sElementBlock) in VMAP Standard I/O library requires data from data set MYELEMENTS (sElement). Figure 3.6 shows a schematic of group ELEMENTS

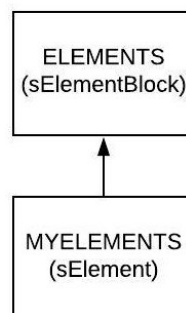


Figure 3.6: ELEMENTS Group Dependency

3.1.7 MYELEMENTS Data Set

The MYELEMENTS data set (sElement) in VMAP Standard I/O library requires data from data sets MATERIALTYPES (sMaterialType), ELEMENTTYPES (sElementType), POINTS->MYIDENTIFIERS (sPointsBlock), COORDINATESYSTEM (sCoordinateSystem). Figure 3.7 shows a schematic of data set MYELEMENTS.

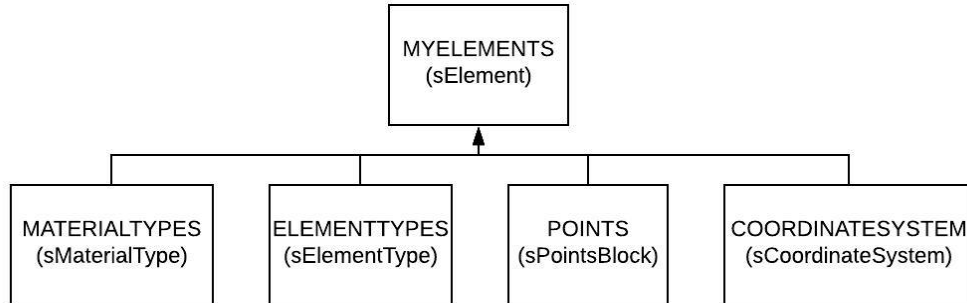


Figure 3.7: MYELEMENTS Data Set Dependency

3.1.8 ELEMENTTYPES Data Set

The **ELEMENTTYPES** data set (**sElementType**) in VMAP Standard I/O library requires data from **INTEGRATIONTYPES** (**sIntegrationType**) and **POINTS** (**sPointsBlock**). Figure 3.8 shows a schematic of data set **ELEMENTTYPES**.

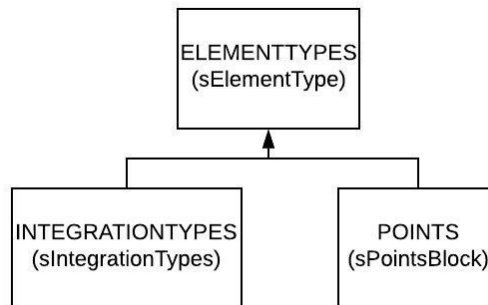


Figure 3.8: ELEMENTTYPES Data Set Dependency

3.1.9 State Variables Group

The State Variables group in VMAP Standard I/O library refers to all the state variables, each state variable (**sStateVariable**) inherits data from **COORDINATESYSTEM** (**sCoordinateSystem**), **UNITS** (**sUnit**), and attribute **MYLOCATION** which could be a/an point, integration point, element or element face. Figure 3.9 shows a schematic of a state variable group.

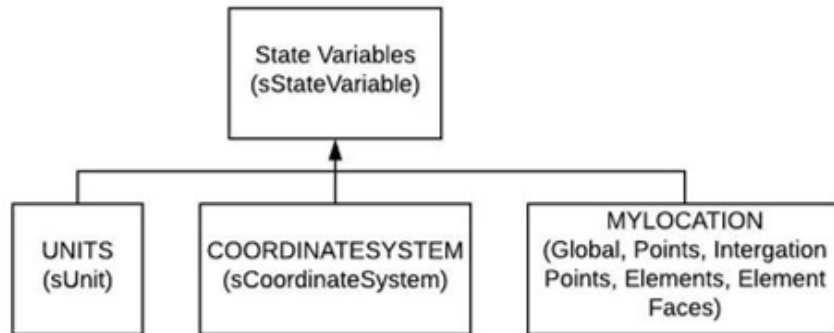


Figure 3.9: State Variable Dependency

Note: Additional Integration Types have to be defined for State Variables, if they do not lie on Integration Points already defined for the Element Type.

3.1.10 UNITSYSTEM Attribute

The UNITSYSTEM attribute (`sUnitSystem`) in VMAP Standard I/O library requires data from Base Unit (`sBaseUnit`). Figure 3.10 shows a schematic of attribute UNITSYSTEM.

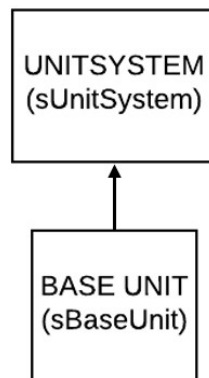


Figure 3.10: UNITSYSTEM Dependency

Chapter 4

VMAP Standard API

4.1 Namespace Documentation

The VMAP namespace is documented in the following section.

4.2 VMAP Namespace Reference

Classes

- struct sVersion
- struct sMetaInformation
- struct sCoordinateSystem
- struct sBaseUnit
- struct sUnitSystem
- struct sUnit
- struct sPointsBlock
- struct sIntegrationType
- struct sElementType
- struct sElement
- struct sElementBlock
- struct sGeometrySet
- struct sStateVariable
- class VMAPElementTypeFactory

Class to generate file VMAP element types.

- class Exception
- class ErrOutOfRange

- class ErrSpaceMismatch
- class ErrSizeMismatch
- class ErrTypeMismatch
- class ErrUnsupported
- class ErrNotImplemented
- class ErrInvalid
- class VMAPFile

Class to read / write a VMAP HDF5 file.

- class VMAPIntegrationTypeFactory

Class to generate file VMAP integration types.

Functions

- template<class tException >
void VMAP_DECLSPEC Assert (const bool condition)
General assertion with exception template.
- template<class tException >
void VMAP_DECLSPEC Assert (const bool condition, const std::string &s)
General assertion with exception template and string.
- H5::CompType H5VersionType (sizeof(sVersion))
- H5::CompType H5MetaInformationType (sizeof(sMetaInformation))
- H5::CompType H5CoordinateSystemType (sizeof(sCoordinateSystem))
- H5::CompType H5BaseUnitType (sizeof(sBaseUnit))
- H5::CompType H5UnitSystemType (sizeof(sUnitSystem))
- H5::CompType H5UnitType (sizeof(sUnit))
- H5::CompType H5IntegrationTypeType (sizeof(sIntegrationType))
- H5::CompType H5ElementTypeType (sizeof(sElementType))
- H5::CompType H5ElementsType (sizeof(sElement))
- void Initialize ()

Method to initialize the VMAP HDF5 data classes.

- herr_t collect_subgroup_names (hid_t loc_id, const char *name, const H5-L_info_t *linfo, void *opdata)
- static void do_dtype (hid_t)
- static void do_dset (hid_t)
- static void do_link (hid_t, char *)
- static void do_attr (hid_t)

- static void do_plist (hid_t)
- static void scan_group (hid_t)
- static void scan_attrs (hid_t)

Variables

- static const int VMAP_VERSION_MAJOR = 0
- static const int VMAP_VERSION_MINOR = 4
- static const int VMAP_VERSION_PATCH = 0
- static H5::VarLenType H5VariableIntType
- static H5::VarLenType H5VariableDoubleType
- static const char * GROUP_VMAP = "/VMAP/"
- static const char * GROUP_SYSTEM = "/VMAP/SYSTEM"
- static const char * GROUP_GEOMETRY = "/VMAP/GEOMETRY"
- static const char * GROUP_VARIABLE = "/VMAP/VARIABLES"
- static const char * ATTRIBUTE_VERSION = "VERSION"
- static const char * ATTRIBUTE_METADATA = "METADATA"
- static const char * ATTRIBUTE_UNITSYSTEM = "UNITSYSTEM"
- static const char * DATASET_UNITS = "UNITS"
- static const char * DATASET_COORDINATESYSTEM = "COORDINATESYSTEM"
- static const char * DATASET_INTEGRATIONTYPES = "INTEGRATIONTYPES"
- static const char * DATASET_ELEMENTTYPES = "ELEMENTTYPES"
- static const char * DATASET_POINTS = "POINTS"
- static const char * DATASET_ELEMENTS = "ELEMENTS"
- static const char * DATASET_GEOMETRYSETS = "GEOMETRYSETS"
- static int INITIALIZATION_FLAG = 0

4.3 File Documentation

All the header and code files are documented in the following sections.

4.4 src/VMAP.cxx File Reference

```
#include "VMAP.h"  
#include "H5Cpp.h"
```

Namespaces

- VMAP

4.5 src/VMAP.h File Reference

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <vector>
#include "H5Cpp.h"
#include "VMAPDeclspec.h"
```

Classes

- struct VMAP::sVersion
- struct VMAP::sMetaInformation
- struct VMAP::sCoordinateSystem
- struct VMAP::sBaseUnit
- struct VMAP::sUnitSystem
- struct VMAP::sUnit
- struct VMAP::sPointsBlock
- struct VMAP::sIntegrationType
- struct VMAP::sElementType
- struct VMAP::sElement
- struct VMAP::sElementBlock
- struct VMAP::sGeometrySet
- struct VMAP::sStateVariable

Namespaces

- VMAP

Variables

- static const int VMAP::VMAP_VERSION_MAJOR = 0
- static const int VMAP::VMAP_VERSION_MINOR = 4
- static const int VMAP::VMAP_VERSION_PATCH = 0

4.6 src/VMAPDeclspec.h File Reference

Macros

- `#define VMAP_DECLSPEC`

4.6.1 Macro Definition Documentation

`#define VMAP_DECLSPEC`

Definition at line 19 of file VMAPDeclspec.h.

4.7 src/VMAPElementTypeFactory.cxx File Reference

```
#include "VMAPElementTypeFactory.h"
#include <vector>
#include <string>
```

Namespaces

- VMAP

4.8 src/VMAPElementTypeFactory.h File Reference

```
#include "VMAP.h"
```

Classes

- class VMAP::VMAPElementTypeFactory
Class to generate file VMAP element types.

Namespaces

- VMAP

4.9 src/VMAPException.cxx File Reference

```
#include <iostream>
#include "VMAPException.h"
```

Namespaces

- VMAP

4.10 src/VMAPException.h File Reference

```
#include <string>
#include <iostream>
#include <exception>
#include "VMAPDeclspec.h"
```

Classes

- class VMAP::Exception
- class VMAP::ErrOutOfRange
- class VMAP::ErrSpaceMismatch
- class VMAP::ErrSizeMismatch
- class VMAP::ErrTypeMismatch
- class VMAP::ErrUnsupported
- class VMAP::ErrNotImplemented
- class VMAP::ErrInvalid

Namespaces

- VMAP

Functions

- template<class tException >
void VMAP_DECLSPEC VMAP::Assert (const bool condition)
General assertion with exception template.
- template<class tException >
void VMAP_DECLSPEC VMAP::Assert (const bool condition, const std::string &s)
General assertion with exception template and string.

4.11 src/VMAPFile.cxx File Reference

```
#include "VMAPFile.h"
```

```
#include "VMAPH5Tools.h"
#include "VMAPEException.h"
#include "hdf5.h"
#include "hdf5_hl.h"
#include <fstream>
#include <sstream>
#include <stdio.h>
#include <stdlib.h>
```

Namespaces

- VMAP

Functions

- H5::CompType VMAP::H5VersionType (sizeof(sVersion))
- H5::CompType VMAP::H5MetaInformationType (sizeof(sMetaInformation))
- H5::CompType VMAP::H5CoordinateSystemType (sizeof(sCoordinateSystem))
- H5::CompType VMAP::H5BaseUnitType (sizeof(sBaseUnit))
- H5::CompType VMAP::H5UnitSystemType (sizeof(sUnitSystem))
- H5::CompType VMAP::H5UnitType (sizeof(sUnit))
- H5::CompType VMAP::H5IntegrationTypeType (sizeof(sIntegrationType))
- H5::CompType VMAP::H5ElementTypeType (sizeof(sElementType))
- H5::CompType VMAP::H5ElementsType (sizeof(sElement))
- void VMAP::Initialize ()

Method to initialize the VMAP HDF5 data classes.

- herr_t VMAP::collect_subgroup_names (hid_t loc_id, const char *name, const H5L_info_t *linfo, void *opdata)

Variables

- static H5::VarLenType VMAP::H5VariableIntType
- static H5::VarLenType VMAP::H5VariableDoubleType
- static const char * VMAP::GROUP_VMAP = "/VMAP/"
- static const char * VMAP::GROUP_SYSTEM = "/VMAP/SYSTEM"
- static const char * VMAP::GROUP_GEOMETRY = "/VMAP/GEOMETRY"
- static const char * VMAP::GROUP_VARIABLE = "/VMAP/VARIABLES"
- static const char * VMAP::ATTRIBUTE_VERSION = "VERSION"

- `static const char * VMAP::ATTRIBUTE_METADATA = "METADATA"`
- `static const char * VMAP::ATTRIBUTE_UNITSYSTEM = "UNITSYSTEM"`
- `static const char * VMAP::DATASET_UNITS = "UNITS"`
- `static const char * VMAP::DATASET_COORDINATESYSTEM = "COORDINATESYSTEM"`
- `static const char * VMAP::DATASET_INTEGRATIONTYPES = "INTEGRATIONTYPES"`
- `static const char * VMAP::DATASET_ELEMENTTYPES = "ELEMENTTYPES"`
- `static const char * VMAP::DATASET_POINTS = "POINTS"`
- `static const char * VMAP::DATASET_ELEMENTS = "ELEMENTS"`
- `static const char * VMAP::DATASET_GEOMETRYSETS = "GEOMETRYSETS"`
- `static int VMAP::INITIALIZATION_FLAG = 0`

4.12 src/VMAPFile.h File Reference

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <vector>
#include "H5Cpp.h"
#include "VMAP.h"
```

Classes

- `class VMAP::VMAPFile`
Class to read / write a VMAP HDF5 file.

Namespaces

- VMAP

Functions

- `void VMAP::Initialize ()`
Method to initialize the VMAP HDF5 data classes.

4.13 src/VMAPH5Tools.h File Reference

```
#include "hdf5.h"
```

Namespaces

- VMAP

Macros

- `#define MAX_NAME 1024`

Functions

- `static void VMAP::do_dtype (hid_t)`
- `static void VMAP::do_dset (hid_t)`
- `static void VMAP::do_link (hid_t, char *)`
- `static void VMAP::do_attr (hid_t)`
- `static void VMAP::do_plist (hid_t)`
- `static void VMAP::scan_group (hid_t)`
- `static void VMAP::scan_attrs (hid_t)`

4.13.1 Macro Definition Documentation

```
#define MAX_NAME 1024
```

Definition at line 17 of file VMAPH5Tools.h.

Referenced by `VMAP::do_attr()`, `VMAP::do_dset()`, `VMAP::do_link()`, `VMAP::do_plist()`, and `VMAP::scan_group()`.

4.14 src/VMAPIntegrationTypeFactory.cxx File Reference

```
#include "VMAPIntegrationTypeFactory.h"  
#include "VMAPException.h"  
#include <vector>  
#include <math.h>
```

Namespaces

- VMAP

4.15 src/VMAPIntegrationTypeFactory.h File Reference

```
#include "VMAP.h"
```

Classes

- class VMAP::VMAPIntegrationTypeFactory
Class to generate file VMAP integration types.

Namespaces

- VMAP

Chapter 5

Compiling & Linking The API

This chapter explains how to link CAE tool APIs developed in Python, C#, C++, Java or FORTRAN to the VMAP Standard API.

5.1 Overview

Figure 5.1 shows partial software architecture to remind the users about the SWIG interface explained in chapter 1.

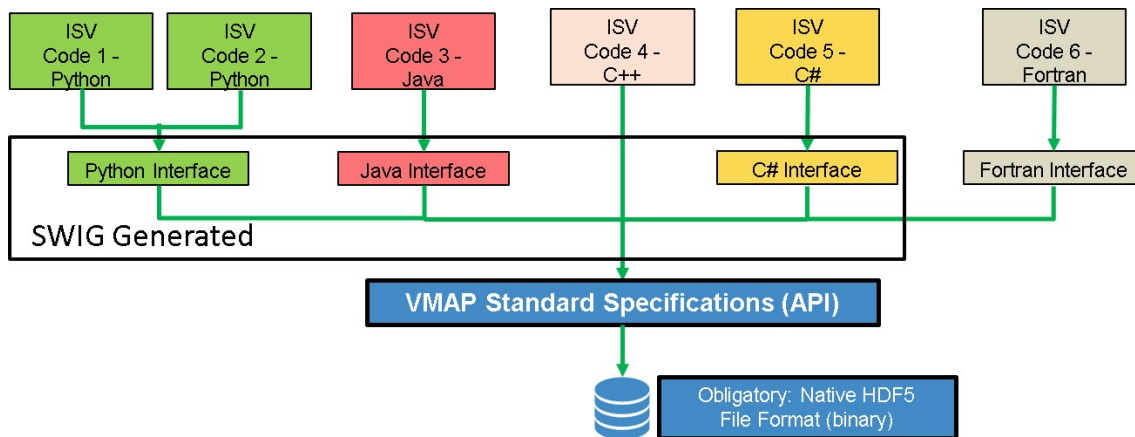


Figure 5.1: SWIG Interface for Linking VMAP Standard API

5.2 Basic Build Requirements for VMAP Standard API

The build requirements for VMAP Standard API are listed below:

1. C++ Compiler
 - Windows: Visual Studio, Intel, Cygwin GCC

- Linux: GCC, Intel
- 2. Third party libraries:
 - Native HDF5 library (with C++ API)
 - Available via HDF5
 - Compiled zlib library to utilize compression
- 3. CMake installation (at least version 3.10.x)
 - CMake

5.3 Additional Requirements for VMAP Standard API

For all code/script formats other than C++, the following is required. #1 is mandatory.

1. SWIG installation (version 2 or higher)
 - SWIG
2. Python installation (for Python Interface)
 - Python
3. Java SDK installation (for Java Interface)
4. C# compiler (for C# Interface)

5.4 Build Using Cmake GUI

1. Open cmake-gui
2. Specify the VMAP source code location
3. Specify the VMAP build location, e.g. in a CMakeFiles sub folder
4. When pressing configure button you have the possibility to select your generator, e.g. Visual Studio 15. See figure 5.2
5. Accept generator with 'Finish' selection
6. Cmake will start to create the project file. See figure 5.3
7. You can uncheck the Java, Python and C# interfaces if unwanted and press 'Configure' again
8. You might get unresolved HDF5 and zlib libraries if cmake is not able to find them

9. Assign them manually per library and 'Configure' until errors are resolved
10. Finally to select 'Generate' button to write the project build generator.

Remarks:

- Building Interfaces requires VMAP build as shared library
- Static HDF libraries are not found properly on Windows (to add manually in cmake-gui)

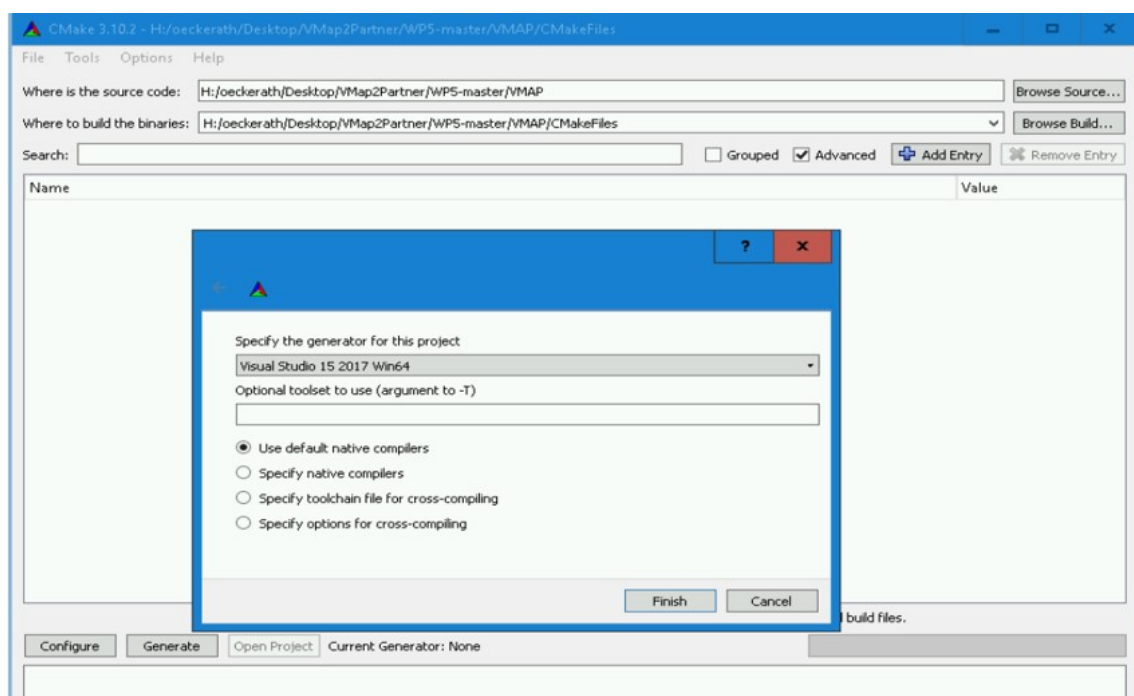


Figure 5.2: Selecting Generator

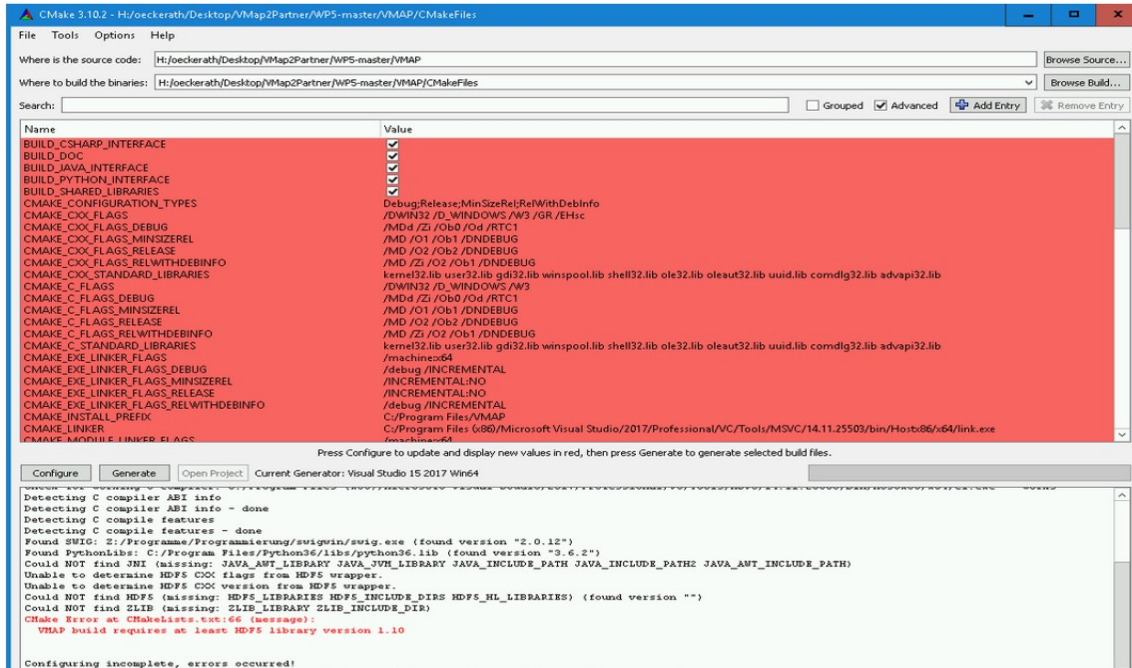


Figure 5.3: Creating Project File

5.5 VMAP Integration

After successfully completing all the previous steps, VMAP integration can be carried out based on the interfaces.

5.5.1 C++ Library

1. Add VMAP 'include' folder to your list of include directories
2. Add libVMAP library to the linker
3. Add HDF5 libraries to the linker
 - libhdf5
 - libhdf5_cpp
 - libhdf5_hl
 - libhdf5_hl_cpp
 - zLib

5.5.2 Python Interface

1. Import module PyVMAP in your Python Code
2. Add libVMAP directory to library path before execution



5.5.3 Java Interface

1. Use `System.loadLibrary("JVMAP")` in your Java Code
2. Add libJVMAP directory to library path before execution

5.5.4 C# Interface

1. Compile all autogenerated *.cs classes to your application
2. Add libVMAP directory to library path before execution

Chapter 6

Implementation Specifications

This chapter outlines the theory to write your own VMAP I/O Library which generates a standard VMAP .h5 output file. The idea of this chapter is to provide users with the flexibility to write their own code without depending on just one standard code. The VMAP .h5 file uses the **same Nomenclature**, for all entities, as described in this chapter. This chapter along with chapter 7 shows the data storage format in a standard VMAP .h5 file. The figures in this chapter explain the hierarchy of data storage in VMAP format. These figures are colour coded with green, red and blue to differentiate among groups, attributes and data sets respectively. Additionally, there are alphabets 'g', 'a' and 'd' as sub-scripts for each of the group, attribute and data set respectively; this facilitates a black-and-white print option without losing the differentiation.

6.1 VMAP Group

VMAP Group has an attribute called **VERSION** and four major groups, which clearly earmark the domains in an FE model. These four groups further consist of groups and data sets along with relevant metadata to comprehensively define an FE model. Tables 6.1 & 6.2 show the **Object Attribute Info** and **General Object Info** of VMAP Group, these tables show information as seen in the HDF5 Viewer.

Number of attributes: 1			
Name	Type	Array Size	Value
VERSION	Compound {...}	Scalar	{...}

Table 6.1: Object Attribute Info

The hierarchical model (from left to right) in Figure 6.1 shows a schematic of the basic VMAP storage structure.

Name:	VMAP
Path:	/
Type:	HDF5 Group
Object Ref:	...

Number of members:	4
Name	Type
GEOMETRY	Group
MATERIAL	Group
VARIABLES	Group
SYSTEM	Group

Table 6.2: General Object Info

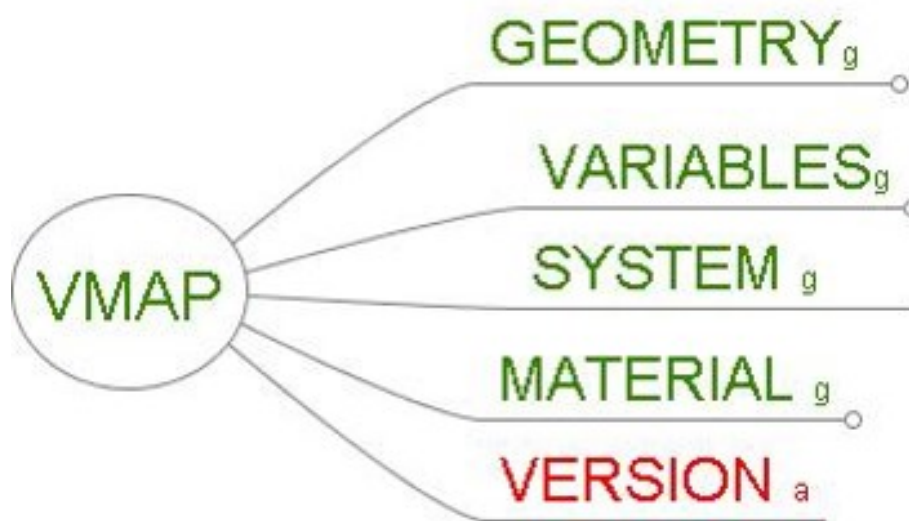


Figure 6.1: VMAP Group

6.1.1 VERSION Attribute

The **VERSION** attribute defines the VMAP version used by the file. The following metadata is associated with **VERSION**:

- **myMajor**: Defines the major version number of VMAP I/O Library.
- **myMinor**: Defines the minor version number of VMAP I/O Library.
- **myPatch**: Defines the patch level number of the VMAP I/O Library.

Table 6.3 shows the data types associated with **VERSION** metadata

Metadata	Data Type
myMajor	Integer
myMinor	Integer
myPatch	Integer

Table 6.3: Data Types of VERSION Metadata

6.2 GEOMETRY Group

GEOMETRY Group stores the geometrical data associated with FE Model, this includes points and elements. Table 6.4 shows only **General Object Info** for group GEOMETRY. This group has no attribute information. As part of the VMAP Standard, the groups associated with GEOMETRY are all the <Part IDs> or <PROPERTY IDs> of an FE Model. <Part IDs> or <PROPERTY IDs> are used synonymously throughout this document and are synonymous throughout the VMAP domain. The points and elements then belong to the respective <Part IDs> or <PROPERTY IDs>. The groups associated with GEOMETRY are shown in Figure 6.3.

Name:	GEOMETRY	Number of members:	n
Path:	/VMAP/	Name	Type
Type:	HDF5 Group	<PART-ID>	Group
Object Ref:	...		

Table 6.4: General Object Info

,where n is the number of Parts/Property-Ids in the FE model.

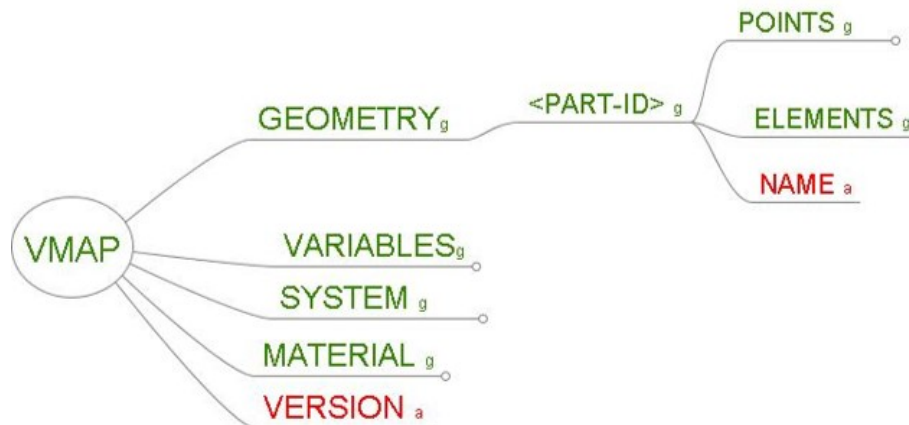


Figure 6.2: GEOMETRY Group

6.3 <PART-ID> Group

<PART-ID> Group stores the points and elements of a particular part/property-id. Tables 6.5 & 6.6 show **Object Attribute Info** and **General Object Info** respectively, for group <PART-ID>. The attribute of <PART-ID> is NAME. As part of the VMAP Standard, the groups associated with <PART-ID> are POINTS & ELEMENTS.

Number of attributes: 1			
Name	Type	Array Size	Value
NAME	String	Scalar	...

Table 6.5: Object Attribute Info

Name:	<PART-ID>	Number of members:	2
Path:	/VMAP/GEOMETRY	Name	Type
Type:	HDF5 Group	POINTS	Group
Object Ref:	...	ELEMENTS	Group

Table 6.6: General Object Info

Note: Within the VMAP Standard Library, there is a create function for creating a group within **GEOMETRY** group with parameters Part-Id and Part-Name. The basis for this group lies in the fact that, all FE solvers have a distinct part number and part name for every part. Hence, the part number and name are directly taken from the solver file. Additionally, this offers the possibility of duplicate point(s) and/or element(s) numbering among various parts.

6.3.1 NAME Attribute

The **NAME** attribute stores the name of the part. The following metadata is associated with **NAME**:

- Value: Defines the name of the part.

Table 6.7 shows the data types associated with **NAME** metadata

Metadata	Data Type
Value	String

Table 6.7: Data Types of NAME Metadata

6.4 POINTS Group

POINTS Group stores the points of an FE model, this includes X,Y,Z coordinates and the unique integral identifier for each point. Tables 6.8 & 6.9 show **Object Attribute Info** and **General Object Info** respectively, for group **POINTS**. The two attributes of **POINTS** group are **MYCOORDINATESYSTEM** & **MYSIZE** and two data sets associated with **POINTS** are **MYCOORDINATES** & **MYIDENTIFIERS**, shown in Figure 6.4.

Number of attributes: 2			
Name	Type	Array Size	Value
MYCOORDINATESYSTEM	Integer	Scalar	...
MYSIZE	unsigned Integer	Scalar	...

Table 6.8: Object Attribute Info

Name:	POINTS	Number of members:	2
Path:	/VMAP/GEOMETRY/<PART-ID>/	Name	Type
Type:	HDF5 Group	MYCOORDINATES	Dataset
Object Ref:	...	MYIDENTIFIERS	Dataset

Table 6.9: General Object Info

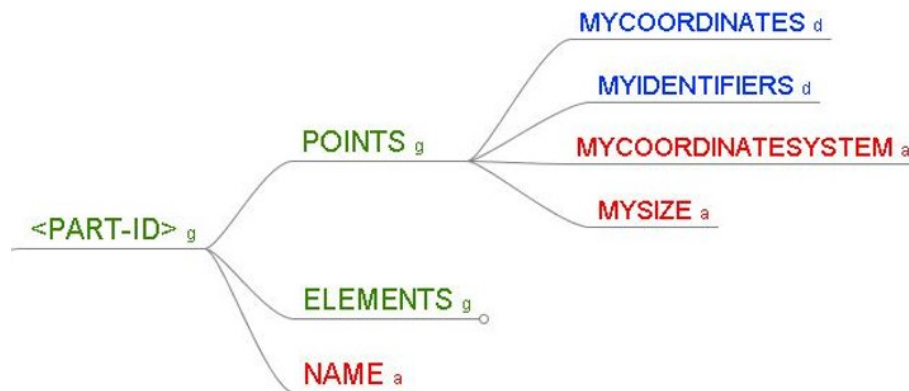


Figure 6.3: POINTS Group

6.4.1 MYCOORDINATESYSTEM Attribute

This attribute contains the coordinate system used by the points.

- Value: Stores the coordinate system reference number. The details of this coordinate system can be found in **SYSTEM ->COORDINATESYSTEM**, see subsection 6.10.1 for more details.

Metadata	Data Type
Value	Integer

Table 6.10: Data Types of MYCOORDINATESYSTEM Metadata

6.4.2 MYSIZE Attribute

This attribute contains the number of points of an FE Model.

- Value: Stores the number of points.

Metadata	Data Type
Value	unsigned Integer

Table 6.11: Data Types of MYSIZE Metadata

6.4.3 MYCOORDINATES Data Set

The MYCOORDINATES data set stores the coordinates of the points in the following storage order:

- X: X coordinate of a point.
- Y: Y coordinate of a point.
- Z: Z coordinate of a point.

In the HDF5 Viewer, the columns are by default named 0, 1 and 2 for X,Y and Z respectively.

Metadata	Data Type
X	Float
Y	Float
Z	Float

Table 6.12: Data Types of MYCOORDINATES Metadata

The **General Object Info** associated with this data set is shown in Table 6.13.

Name:	MYCOORDINATES
Path:	/VMAP/GEOMETRY/<PART-ID>/POINTS/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	MYSIZE x 3
Max Dimension Size(s):	MYSIZE x 3
Data Type:	64-bit floating point

Table 6.13: General Object Info

6.4.4 MYIDENTIFIERS Data Set

The MYIDENTIFIERS data set stores the integral identifiers to points. In the HDF5 Viewer, the column is named by default as 0 for myvalue (shown below in the table).

Metadata	Data Type
myvalue	Integer

Table 6.14: Data Types of MYIDENTIFIERS Metadata

The **General Object Info** associated with this data set is shown in Table 6.15.

Name:	MYIDENTIFIERS
Path:	/VMAP/GEOMETRY/<PART-ID>/POINTS/
Type:	HDF5 Dataset
Object Ref:	..

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	MYSIZE x 1
Max Dimension Size(s):	MYSIZE x 1
Data Type:	32-bit integer

Table 6.15: General Object Info

6.5 ELEMENTS Group

ELEMENTS Group stores the elements of a FE model, this includes identifier, type, coordinate system, material type, element connectivity. Tables 6.16 & 6.17 show **Object Attribute Info** and **General Object Info** respectively, for group **ELEMENTS**. The attribute associated with **ELEMENTS** is **MYSIZE** and data set is **MYELEMENTS**, shown in Figure 6.4.

Number of attributes: 1			
Name	Type	Array Size	Value
MYSIZE	unsigned Integer	Scalar	...

Table 6.16: Object Attribute Info

Name:	ELEMENTS
Path:	/VMAP/GEOMETRY/<PART-ID>/
Type:	HDF5 Group
Object Ref:	...

Number of members: 1	
Name	Type
MYELEMENTS	Dataset

Table 6.17: General Object Info

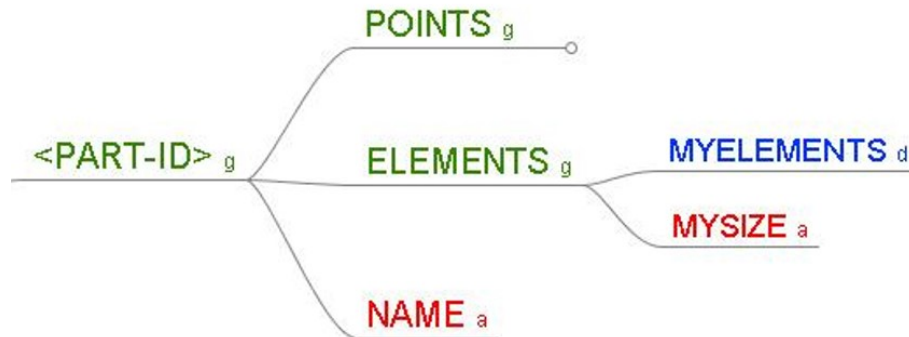


Figure 6.4: ELEMENTS Group

6.5.1 MYSIZE Attribute

This attribute contains the number of elements of an FE Model.

- Value: Stores the number of elements/the size of the data set MYELEMENTS.

Metadata	Data Type
Value	unsigned Integer

Table 6.18: Data Types of MYSIZE Metadata

6.5.2 MYELEMENTS Data Set

The MYELEMENTS data set stores the element details in the following storage order:

- **myIdentifier**: Integral identifier for each element.
- **myElementType**: Stores Element Type reference number. For more details about SYSTEM->ELEMENTTYPES see sub-section 6.10.2.
- **myCoordinateSystem**: Stores the coordinate system reference number. For more details about SYSTEM->COORDINATESYSTEM, see sub-section 6.10.1.
- **myMaterialType**: Stores the material type reference number. For more details about MATERIAL->MATERIALTYPE, see section XX.
- **myConnectivity**: Stores the point number ordering which forms one element. For details about point number ordering see chapter 8.

This is a compound data set.

Metadata	Data Type
myIdentifier	Integer
myElementType	Integer
myCoordinateSystem	Integer
myMaterialType	Integer
myConnectivity	Integer Array

Table 6.19: Data Types of MYELEMENTS Metadata

The **General Object Info** associated with this data set is shown in Table 6.20.

Name:	MYELEMENTS
Path:	/VMAP/GEOMETRY/<PART-ID>/ELEMENTS/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	MYSIZE x 1
Max Dimension Size(s):	MYSIZE x 1
Data Type:	Compound

Table 6.20: General Object Info

6.6 VARIABLES Group

VARIABLES Group stores the all the variables of an FE model, this includes all input & output state variables. Table 6.21 shows only **General Object Info**, since this group has no attribute information. The group(s) associated with **VARIABLES** is/are all the states **STATE-<n>** of the FE-Model. Figure 6.5 shows the complete hierarchical structure of the **VARIABLES** Group.

Name:	VARIABLES	Number of members:	1
Path:	/VMAP/	Name	Type
Type:	HDF5 Group	STATE-<n>	Group
Object Ref:	...		

Table 6.21: General Object Info

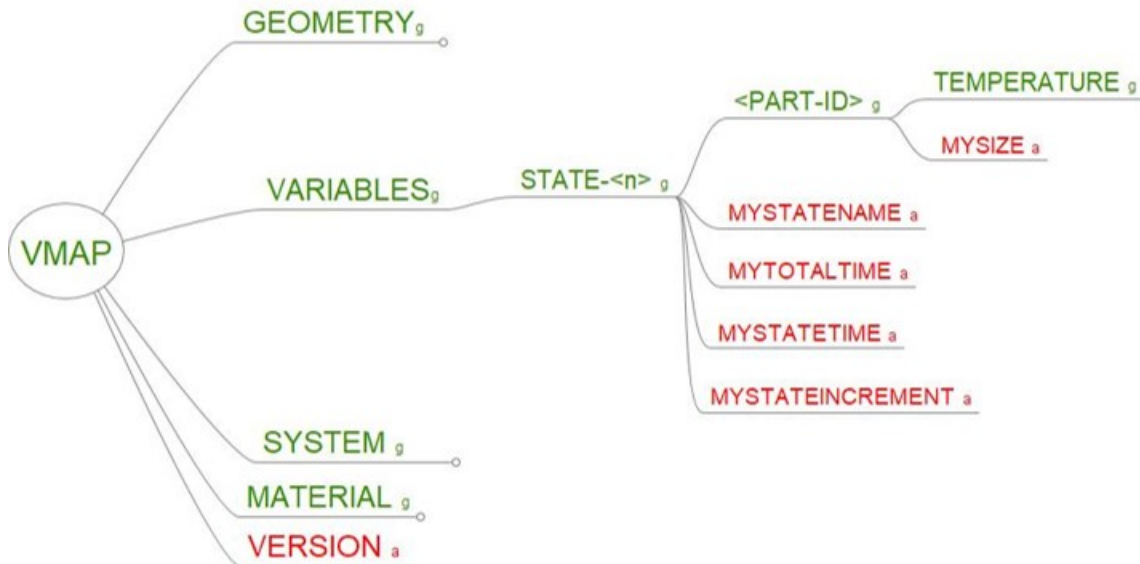


Figure 6.5: VARIABLES Group

6.7 STATE-<n> Group

STATE-<n> Group is defined separate different states of an FE-Model. Here, a **timestep** or a **frame** could be referred as a state. A state is defined as any given interval or frame, where the FE-model has a certain result set, which may or may not change for the next interval or frame. This group contains all the part numbers which have state variables defined for the particular STATE-<n>. Table 6.22 & Table 6.23 show Object Attribute Info and General Object Info respectively. This group has four **optional** attributes to store the state metadata. The group(s) associated with STATE-<n> is/are all the part-ids <PART-ID> with state variables for this state.

Number of attributes: 4			
Name	Type	Array Size	Value
MYSTATENAME	String	Scalar	...
MYTOTALTIME	Double	Scalar	...
MYSTATETIME	Double	Scalar	...
MYSTATEINCREMENT	Integer	Scalar	...

Table 6.22: Object Attribute Info

Name:	VARIABLES
Path:	/VMAP/VARIABLES/
Type:	HDF5 Group
Object Ref:	...

Number of members: n	
Name	Type
<PART-ID>	Group

Table 6.23: General Object Info

, where n refers to the number of Parts that have state variables associated with STATE-<n>.

Note: In VMAP, the STATE-0 is defined as the Input state or the initial state of the FE model.

6.7.1 MYSTATENAME Attribute

This attribute contains the name of the state.

- Value: Stores name of the STATE-<n>.

Metadata	Data Type
Value	String

Table 6.24: Data Types of MYSTATENAME Metadata

6.7.2 MYTOTALTIME Attribute

This attribute contains the total time of the analysis.

- Value: Stores total time of the analysis.

Metadata	Data Type
Value	Double

Table 6.25: Data Types of MYTOTALTIME Metadata

6.7.3 MYSTATETIME Attribute

This attribute contains the time/interval of the state.

- Value: Stores the time/interval of the STATE-<n>.

Metadata	Data Type
Value	Double

Table 6.26: Data Types of MYSTATETIME Metadata

6.7.4 MYSTATEINCREMENT Attribute

This attribute contains the time increment or state increment of the state.

- Value: Stores the time/state increment of the STATE-<n>.

Metadata	Data Type
Value	Integer

Table 6.27: Data Types of MYSTATEINCREMENT Metadata

6.8 <PART-ID> Group

<PART-ID> Group corresponds to the <PART-ID> Group defined for the **GEOMETRY** Group. This means that for a given **STATE-*<n>***, a particular <PART-ID> has state variable(s). Table 6.28 & 6.29 show **Object Attribute Info** and **General Object Info** respectively, for group <PART-ID>. The groups associated with <PART-ID> are all state variables, where each state variable is defined as one group.

Number of attributes: 1			
Name	Type	Array Size	Value
MYSIZE	Integer	Scalar	...

Table 6.28: Object Attribute Info

Name:	<PART-ID>	Number of members: MYSIZE	
Path:	/VMAP/VARIABLES/STATE- <i><n></i> /		
Type:	HDF5 Group	Name	Type
Object Ref:	...	TEMPERATURE	Group

Table 6.29: General Object Info

6.8.1 MYSIZE Attribute

This attribute contains the number of state variables of an FE Model.

- Value: Stores the number of state variables.

Metadata	Data Type
Value	unsigned Integer

Table 6.30: Data Types of MYSIZE Metadata

Section 6.9 explains an example as a State Variable group.

6.9 TEMPERATURE Group

TEMPERATURE Group stores data about the temperature variation over the FE model for a given time stamp. Table 6.31 & 6.32 show **Object Attribute Info** and **General Object Info** respectively, for group **TEMPERATURE**. There are eleven attributes and two data sets associated with **TEMPERATURE** or with any other state variable. The attributes are shown in table 6.31 and explained in the following sub-sections. The two data sets are **MYINTEGRATIONTYPES** & **MYVALUES**. (Figure 6.6)

Number of attributes:	11		
Name	Type	Array Size	Value
MYCOORDINATESYSTEM	Integer	Scalar	...
MYDIMENSION	Integer	Scalar	...
MYENTITY	Integer	Scalar	...
MYIDENTIFIER	Integer	Scalar	...
MYINCREMENTVALUE	Integer	Scalar	...
MYLOCATION	Integer	Scalar	...
MYMULTIPLICITY	Integer	Scalar	...
MYTIMEVALUE	Floating Point	Scalar	...
MYUNIT	Integer	Scalar	...
MYVARIABLEDESCRIPTION	String	Scalar	...
MYVARIABLENAME	String	Scalar	...

Table 6.31: Object Attribute Info

Name:	TEMPERATURE
Path:	/VMAP/RESULT/STATE-<n>/<PART-ID>/
Type:	HDF5 Group
Object Ref:	...

Number of members:	2
Name	Type
MYINTEGRATIONTYPES	Dataset
MYVALUES	Dataset

Table 6.32: General Object Info

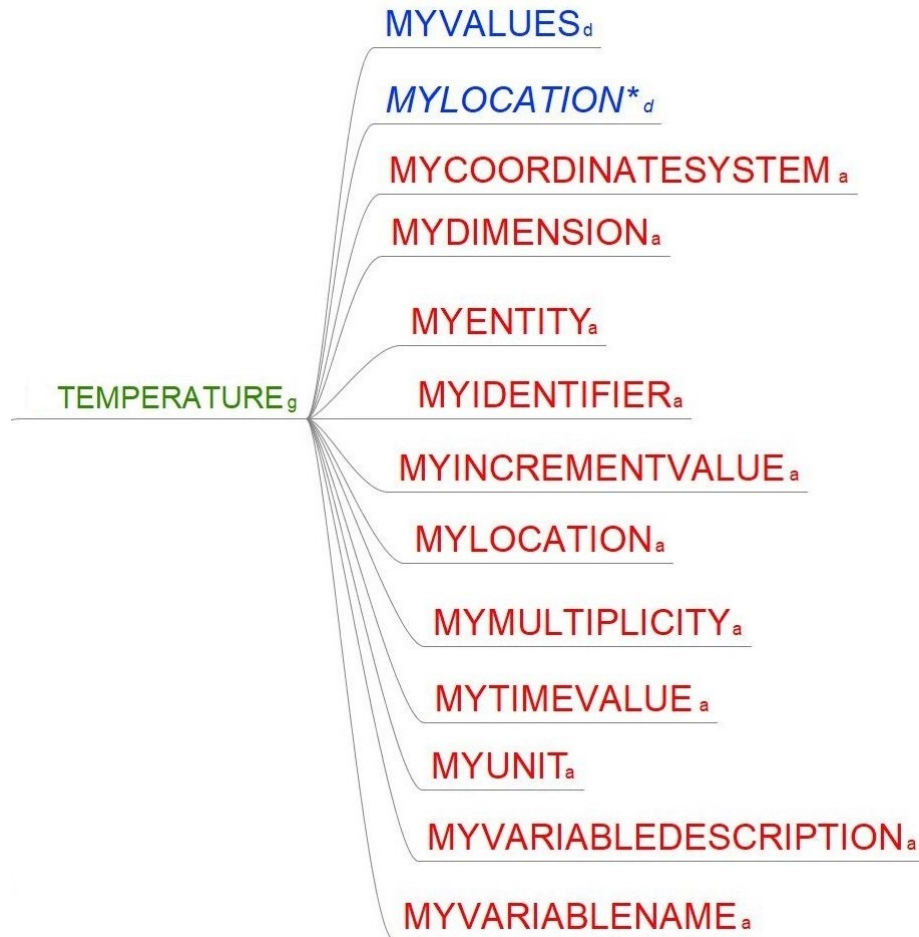


Figure 6.6: TEMPERATURE Group

6.9.1 MYCOORDINATESYSTEM Attribute

Refers to the coordinate system used by the state variable. The details of this coordinate system can be found in **SYSTEM -> COORDINATESYSTEM**, see sub-section 6.10.1 for more details.

- Value: Stores the coordinate system reference number.

Metadata	Data Type
Value	Integer

Table 6.33: Data Types of MYCOORDINATESYSTEM Metadata

6.9.2 MYDIMENSION Attribute

refers to the dimension of the state variable **TEMPERATURE**.

- Value: Stores the reference number of MYDIMENSION.

Metadata	Data Type
Value	Integer

Table 6.34: Data Types of MYDIMENSION Metadata

Table 6.35 shows the available dimensions and their reference numbers in VMAP.

MYDIMENSION	Reference Number
INVALID	0
SCALAR	1
VECTOR	3
2nd Order Plain Tensor Symmetric	4
2nd Order Tensor Symmetric	6
2nd Order Tensor	9
STIFFNESS MATRIX	36
4th Order Tensor Symmetric	45
4th Order Tensor	81

Table 6.35: MYDIMENSION Enumeration

6.9.3 MYENTITY Attribute

Refers to the entity of the state variable **TEMPERATURE**.

- Value: Stores the reference number of MYENTITY

Metadata	Data Type
Value	Integer

Table 6.36: Data Types of MYENTITY Metadata

Table 6.37 shows the available entities and their reference numbers in VMAP

MYENTITY	Reference Number
REAL	1
COMPLEX	2
HAMILTONIAN	4

Table 6.37: MYENTITY Enumeration

6.9.4 MYIDENTIFIER Attribute

Refers to the unique integral identifier for the state variable **TEMPERATURE**.

- Value: Stores the unique identifier.

Metadata	Data Type
Value	Integer

Table 6.38: Data Types of MYIDENTIFIER Metadata

6.9.5 MYINCREMENTVALUE Attribute

Refers to multiple steps over which the state variable is calculated. This attribute is useful when the state variables are not defined over time.

- Value: Stores the step value for the state variable.

Metadata	Data Type
Value	Integer

Table 6.39: Data Types of MYINCREMENTVALUE Metadata

6.9.6 MYLOCATION Attribute

Refers to the location where the state variable **TEMPERATURE** is stored.

- Value: Stores the reference number of **MYLOCATION**.

Metadata	Data Type
Value	Integer

Table 6.40: Data Types of MYLOCATION Metadata

Table 6.41 shows the available locations and their reference numbers in VMAP

MYLOCATION	Reference Number
INVALID	0
GLOBAL	1
NODE	2
ELEMENT	3
INTEGRATION POINT	4
ELEMENT FACE	5

Table 6.41: MYLOCATION Enumeration

6.9.7 MYMULTIPLICITY Attribute

Refers to the number of columns in the **MYVALUES** data set. Majority of the state variables are have multiplicity 1.

- Value: Stores the multiplicity of the **MYVALUES** data set.

Metadata	Data Type
Value	Integer

Table 6.42: Data Types of MYMULTIPLICITY Metadata

Majority of the state variables have MYMULTIPLICITY 1. Sometimes it would be necessary to group identical data types into one value e.g. in LS-DYNA a point can have more than one value when it belongs to more than one element. Set of mathematically identical data types could then be grouped, using multiplicity.

6.9.8 MYTIMEVALUE Attribute

Refers to the time stamp at which the state variable is calculated.

- Value: Stores the time value of the state variable **TEMPERATURE**.

Metadata	Data Type
Value	Floating-Point

Table 6.43: Data Types of MYTIMEVALUE Metadata

For transient analysis and for other time based analyses, each state variable should be stored per MYTIMEVALUE.

6.9.9 MYUNIT Attribute

Refers to the unit used by the state variable. The details of this unit can be found in **SYSTEM ->UNITS**, see sub-section 6.10.5 for more details.

- Value: Stores the reference number of the unit.

Metadata	Data Type
Value	Integer

Table 6.44: Data Types of MYUNIT Metadata

6.9.10 MYVARIABLEDESCRIPTION Attribute

Stores detailed description of the variable. Also refers to Variable Specifications in the following chapters.

- Value: Stores detailed description of the variable based on the source analysis tool.

Metadata	Data Type
Value	String

Table 6.45: Data Types of MYVARIABLEDESCRIPTION Metadata

6.9.11 MYVARIABLENAME Attribute

Stores name of the variable. Also refers to Variable Specifications in the following chapters.

- Value: Stores name of the variable.

Metadata	Data Type
Value	String

Table 6.46: Data Types of MYVARIABLENAME Metadata

6.9.12 MYVALUES Data Set

The MYVALUES data set stores the state variable values. The number of columns can be determined based on MYDIMENSION and MYMULTIPLICITY. In the HDF5 Viewer, the column is named by default as 0 for myvalue (shown in Table 6.47).

Based on the MYLOCATION, the state variable could be defined per **Point**, **Element**, **Element Face**, **Integration Point** or it could be **Global**. The order of the state variables is based on the order of **Point**, **Element**, **Element Face**.

If the state variable is defined per **Integration Point**, then there is **an additional data set** MYINTEGRATIONTYPES, explained in the next sub-section, which stores the reference number to the SYSTEM -> INTEGRATIONTYPES. The values are in the order in which the integration points are defined in the referred integration type.

Metadata	Data Type
myvalue	Floating-Point Array

Table 6.47: Data Types of MYVALUES Metadata

The **General Object Info** associated with this data set is shown in Table 6.48.

Name:	MYVALUES
Path:	/VMAP/VARIABLES/STATE-<n>/<PART-ID>/TEMPERATURE/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	m x 1
Max Dimension Size(s):	m x 1
Data Type:	64-bit floating-point

Table 6.48: General Object Info

m given in Table 6.48 is the length of the MYVALUES array. When,
MYLOCATION = 2 (=POINTS) then m = **MYSIZE** of **POINTS**
MYLOCATION = 3 (=Element) then m = **MYSIZE** of **ELEMENTS**
MYLOCATION = 4 (=Integration Point) then
 $m = \sum (\text{ALL ELEMENT INTEGRATION TYPES})$ (per in-plane, per out-of-plane Integration Point)
MYLOCATION = 5 (=Element Face) then m = SUM of all **ELEMENT FACES**

6.9.13 MYINTEGRATIONTYPES Data Set

This data set exists only when the **MYLOCATION** = 4 (=Integration Point). The data set stores the reference number of the integration type over which the state variable is defined. In the HDF5 Viewer, the column is named by default as 0 for myvalue (shown in Table 6.49).

Metadata	Data Type
myvalue	Floating-Point Array

Table 6.49: Data Types of MYINTEGRATIONTYPES Metadata

The **General Object Info** associated with this data set is shown in Table 6.50.

Name:	MYINTEGRATIONTYPES
Path:	/VMAP/VARIABLES/STATE-<n>/<PART-ID>/TEMPERATURE/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	$m \times 1$
Max Dimension Size(s):	$m \times 1$
Data Type:	32-bit Integer

Table 6.50: General Object Info

m = **MYSIZE** of **ELEMENTS**

6.10 SYSTEM Group

SYSTEM Group stores the system data related to the FE model. This includes six data sets - **COORDINATESYSTEM**, **ELEMENTTYPES** & **INTEGRATIONTYPES**, **METADATA**, **UNITS**

Name:	SYSTEM	Number of members:	6
Path:	/VMAP/	Name	Type
Type:	HDF5 Group	COORDINATESYSTEM	Dataset
Object Ref:	...	ELEMENTTYPES	Dataset
		INTEGRATIONTYPES	Dataset
		METADATA	Dataset
		UNITS	Dataset
		UNITSYSTEM	Dataset

Table 6.51: General Object Info

& UNITSYSTEM. Table 6.51 shows General Object Info, for group SYSTEM. Figure 6.7 shows data sets associated with the SYSTEM group.

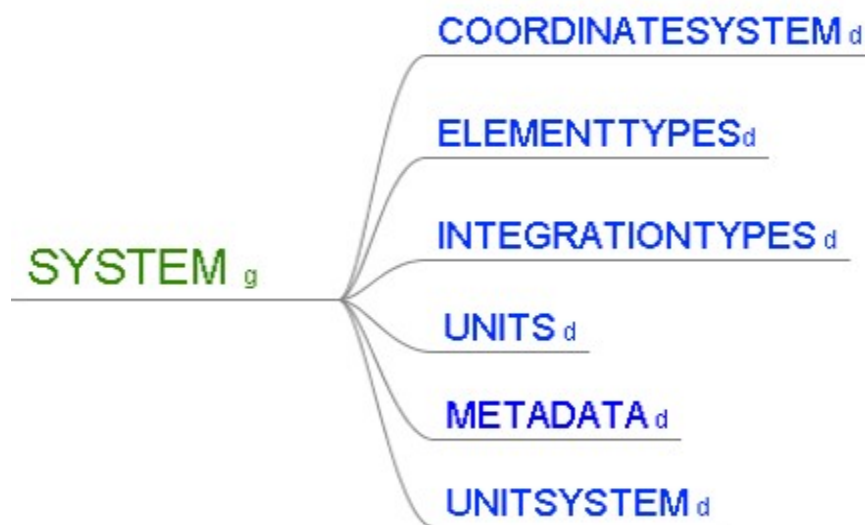


Figure 6.7: SYSTEM Group

6.10.1 COORDINATESYSTEM Data Set

The COORDINATESYSTEM data set stores the global coordinate system used by the FE Model. All the local coordinate systems are stored as state variables in the VARIABLES group. The metadata is stored in the following order:

- **myIdentifier:** Integral identifier for each coordinate system. Best practice involves storing the Global coordinate system with identifier 1.
- **myType:** Stores Coordinate System reference number. For more details about reference number, see Table 6.53.

- **myReferencePoint**: Stores the 3D reference point for system definition, $[0, 0, 0]$ is default.
- **myAxisVector**: Defines up to three vectors describing the coordinate system $[u1, u2, u3, v1, v2, v3, w1, w2, w3]$.

This is a compound data set.

Metadata	Data Type
myIdentifier	Integer
myType	Integer
myReferencePoint	Double Array
myAxisVector	Double Array

Table 6.52: Data Types of COORDINATESYSTEM Metadata

Table 6.53 shows the available coordinate systems and their reference numbers in VMAP.

COORDINATESYSTEM	Reference Number
INVALID	-1
CARTESIAN LEFT HAND	1
CARTESIAN RIGHT HAND	2
NON-ORTHOGONAL	3

Table 6.53: COORDINATESYSTEM Enumeration

The General Object Info associated with this data set is shown in Table 6.54.

Name:	COORDINATESYSTEM
Path:	/VMAP/SYSTEM/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	n x 1
Max Dimension Size(s):	n x 1
Data Type:	Compound

Table 6.54: General Object Info

n refers to number of coordinate systems defined for the FE model.

6.10.2 ELEMENTTYPES Data Set

The **ELEMENTTYPES** data set stores the various types of elements used by the FE Model. The metadata is stored in the following order:

- **myIdentifier**: Integral identifier for each element type used in the FE Model.
- **myTypeName**: Stores the type of the element. VMAP offers a factory of various element types, which can be directly incorporated in the code. Please refer to chapter 8 for more details.
- **myNumberOfNodes**: Stores the number of nodes of the element type.
- **myDimension**: Stores the dimension of the element.
- **myShapeType** : Stores the reference number of the element from the Element Factory library offered in the VMAP Package.
- **myInterpolationType**: Stores the reference number of the interpolation type used by the element type. Please refer to Table 6.57 for more details on Interpolation types.
- **myIntegrationType**: Stores the reference number of the integration type used by the element type. Please refer to sub-section 6.10.3 for more details on Integration types.
- **myNumberofNormalComponents**: Stores the number of normal components of stress or strain tensor for the element type.
- **myNumberofShearComponents**: Stores the number of shear components of stress or strain tensor for the element type.
- **myConnectivity**: Stores the connectivity of the element.
- **myFaceConnectivity**: Stores the face connectivity of the element.

This is a compound data set.

Metadata	Data Type
myIdentifier	Integer
myTypeName	Character Pointer
myNumberOfNodes	Integer
myDimension	Integer
myShapeType	Integer
myInterpolationType	Integer
myIntegrationType	Integer
myNumberOfNormalComponents	Integer
myNumberOfShearComponents	Integer
myConnectivity	Integer Array
myFaceConnectivity	Integer Array

Table 6.55: Data Types of ELEMENTTYPES Metadata

The General Object Info associated with this data set is shown in Table 6.56.

Name:	ELEMENTTYPES
Path:	/VMAP/SYSTEM/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	n x 1
Max Dimension Size(s):	n x 1
Data Type:	Compound

Table 6.56: General Object Info

n refers to number of element types defined for the FE model.

Table 6.57 shows the available interpolation types and their reference numbers in VMAP.

myInterpolationType	Reference Number
CONSTANT	1
LINEAR	2
BILINEAR	3
TRILINEAR	4
QUADRATIC	5
BIQUADRATIC	6
TRIQUADRATIC	7
CUBIC	8
BICUBIC	9
TRICUBIC	10
SPLINE	11

Table 6.57: myInterpolationType Enumeration

6.10.3 INTEGRATIONTYPES Data Set

The INTEGRATIONTYPES data set stores the various types of integration rules used by the FE Model. The metadata is stored in the following order:

- **myIdentifier:** Integral identifier for each integration type used in the FE Model.
- **myTypeName:** Stores the type of the integration rule. VMAP offers a factory of various integration types, which can be directly incorporated in the code. Please refer to chapter 9 for more details.
- **myNumberOfPoints:** Stores the number of points of the integration type.
- **myDimension:** Stores the dimension of abscissa.
- **myOffset:** Stores the shell offset parameter.
- **myAbscissas:** Stores the abscissa of the integration point in local coordinates.
- **myWeights:** Stores the weight of the integration point.
- **mySubTypes:** When a compound integration type is used e.g. Gauss_3 for out-of-plane and Gauss_Triangle_1 for in-plane , then it stores the 2 types together as one single type, *Gauss_Triangle_1* × *Gauss_3*. According to VMAP Standard, the order of compound integration type is,

IN – PLANE × OUT – OF – PLANE

This is a compound data set.

Metadata	Data Type
myIdentifier	Integer
myTypeName	Character Pointer
myNumberOfPoints	Integer
myDimension	Integer
myOffset	Double
myAbscissas	Double Array
myWeights	Double Array
mySubTypes	Integer Array

Table 6.58: Data Types of INTEGRATIONTYPES Metadata

The General Object Info associated with this data set is shown in Table 6.59.

Name:	INTEGRATIONTYPES
Path:	/VMAP/SYSTEM/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	n x 1
Max Dimension Size(s):	n x 1
Data Type:	Compound

Table 6.59: General Object Info

n refers to number of integration types defined for the FE model.

6.10.4 METADATA Data Set

The **METADATA** data set stores all the meta information associated with the FE model. The metadata is stored in the following order:

- **myExporterName**: Stores the name of the tool generating the VMAP file.
- **myFileDate**: Stores the date of file generation.
- **myFileTime**: Stores the time of file generation.
- **myDescription**: Stores the detailed description of file content.
- **myAnalysisType**: Stores the analysis type of the exporter.
- **myUserId**: Stores the name of the user, who generated the VMAP file.

Metadata	Data Type
myExporterName	Character Pointer
myFileDate	Character Pointer
myFileTime	Character Pointer
myDescription	Character Pointer
myAnalysisType	Character Pointer
myUserId	Character Pointer

Table 6.60: Data Types of METADATA Metadata

The General Object Info associated with this data set is shown in Table 6.61.

Name:	METADATA
Path:	/VMAP/SYSTEM/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	6 x 1
Max Dimension Size(s):	6 x 1
Data Type:	Compound

Table 6.61: General Object Info

6.10.5 UNITS Data Set

The UNITS data set stores the various types of units used by the FE Model. The metadata is stored in the following order:

- **myIdentifier**: Integral identifier for each units used in the FE Model.
- **myUnitSymbol**: Stores the unit symbol.
- **myUnitDimension**: Stores a combination of 0s and 1s to form a unit based on the SI unit system defined in section 6.10.6. It is an array with length 7.

This is a compound data set.

Metadata	Data Type
myIdentifier	Integer
myUnitSymbol	Character Pointer
myUnitDimension	Integer Array

Table 6.62: Data Types of UNITS Metadata

The General Object Info associated with this data set is shown in Table 6.63.

Name:	UNITS
Path:	/VMAP/SYSTEM/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	n x 1
Max Dimension Size(s):	n x 1
Data Type:	Compound

Table 6.63: General Object Info

n refers to number of units defined for the FE model.

6.10.6 UNITSYSTEM Data Set

The UNITSYSTEM data set stores the SI unit system. This metadata is stored in the following order:

- **myLengthUnit:** This defines the standard length unit.
 - **myIdentifier:** A unique identifier for length unit is 1
 - **mySIScale:** A scale factor associated with the length unit. Useful to convert the SI system to another user-defined unit system.
 - **mySIShift:** A shift factor associated with the length unit. Useful to convert the SI system to another user-defined unit system.
 - **myUnitSymbol:** Unique SI symbol for the length unit 'mm' - millimeter.
 - **myUnitQuantity:** The name of the unit quantity "LENGTH".
- **myMassUnit:** This defines the standard mass unit.
 - **myIdentifier:** A unique identifier for mass unit is 2
 - **mySIScale:** A scale factor associated with the mass unit. Useful to convert the SI system to another user-defined unit system.
 - **mySIShift:** A shift factor associated with the mass unit. Useful to convert the SI system to another user-defined unit system.
 - **myUnitSymbol:** Unique SI symbol for the mass unit 't' - tonne.
 - **myUnitQuantity:** The name of the unit quantity "MASS".
- **myTimeUnit:** This defines the standard time unit.

- **myIdentifier:** A unique identifier for time unit is 3.
- **mySIScale:** A scale factor associated with the time unit. Useful to convert the SI system to another user-defined unit system.
- **mySIShift:** A shift factor associated with the time unit. Useful to convert the SI system to another user-defined unit system.
- **myUnitSymbol:** Unique SI symbol for the time unit 's' - second.
- **myUnitQuantity:** The name is the unit quantity "TIME".
- **myCurrentUnit:** This defines the standard current unit.
 - **myIdentifier:** A unique identifier for current unit is 4.
 - **mySIScale:** A scale factor associated with the current unit. Useful to convert the SI system to another user-defined unit system.
 - **mySIShift:** A shift factor associated with the current unit. Useful to convert the SI system to another user-defined unit system.
 - **myUnitSymbol:** Unique SI symbol for the current unit 'A' - Ampere.
 - **myUnitQuantity:** The name is the unit quantity "ELECTRIC CURRENT".
- **myTemperatureUnit:** This defines the standard temperature unit.
 - **myIdentifier:** A unique identifier for temperature unit is 5.
 - **mySIScale:** A scale factor associated with the temperature unit. Useful to convert the SI system to another user-defined unit system.
 - **mySIShift:** A shift factor associated with the temperature unit. Useful to convert the SI system to another user-defined unit system.
 - **myUnitSymbol:** Unique SI symbol for the temperature unit 'K' - Kelvin.
 - **myUnitQuantity:** The name is the unit quantity "TEMPERATURE".
- **myAmountOfSubstanceUnit:** This defines the standard amount of substance unit.
 - **myIdentifier:** A unique identifier for amount of substance unit is 6.
 - **mySIScale:** A scale factor associated with the amount of substance unit. Useful to convert the SI system to another user-defined unit system.
 - **mySIShift:** A shift factor associated with the amount of substance unit. Useful to convert the SI system to another user-defined unit system.
 - **myUnitSymbol:** Unique SI symbol for the amount of substance unit 'mol' - mole.
 - **myUnitQuantity:** The name is the unit quantity "AMOUNT OF SUBSTANCE".

- **myLuminousIntensityUnit:** This defines the standard luminous intensity unit.
 - **myIdentifier:** A unique identifier for luminous intensity unit is 7.
 - **mySIScale:** A scale factor associated with the luminous intensity unit. Useful to convert the SI system to another user-defined unit system.
 - **mySIShift:** A shift factor associated with the luminous intensity unit. Useful to convert the SI system to another user-defined unit system.
 - **myUnitSymbol:** Unique SI symbol for the luminous intensity unit 'cd' - candela.
 - **myUnitQuantity:** The name is the unit quantity "LUMINOUS INTENSITY".

This is a compound data set.

Metadata	Data Type
myIdentifier	Integer
mySIScale	Double
mySIShift	Double
myUnitSymbol	Character Pointer
myUnitQuantity	Character Pointer

Table 6.64: Data Types of UNITSYSTEM metadata

The General Object Info associated with this data set is shown in Table 6.65.

Name:	UNITSYSTEM
Path:	/VMAP/SYSTEM/
Type:	HDF5 Dataset
Object Ref:	...

Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	7 x 1
Max Dimension Size(s):	7 x 1
Data Type:	Compound

Table 6.65: General Object Info

Chapter 7

Storage Format

This chapters shows the VMAP format in from HDFView 3.1.0. The HDFView shows the file path at the top. Additionally, there are always two windows for each Group and Dataset 'Object Attribute Info' & 'General Object Info'.

7.1 .h5 File View

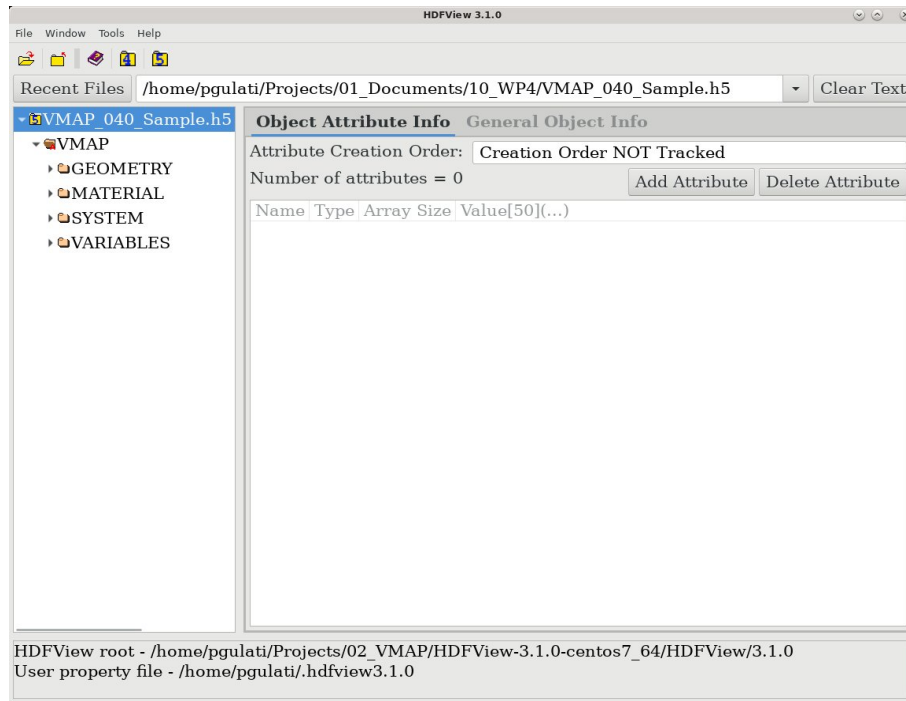


Figure 7.1: .h5 File View - Object Attribute Info

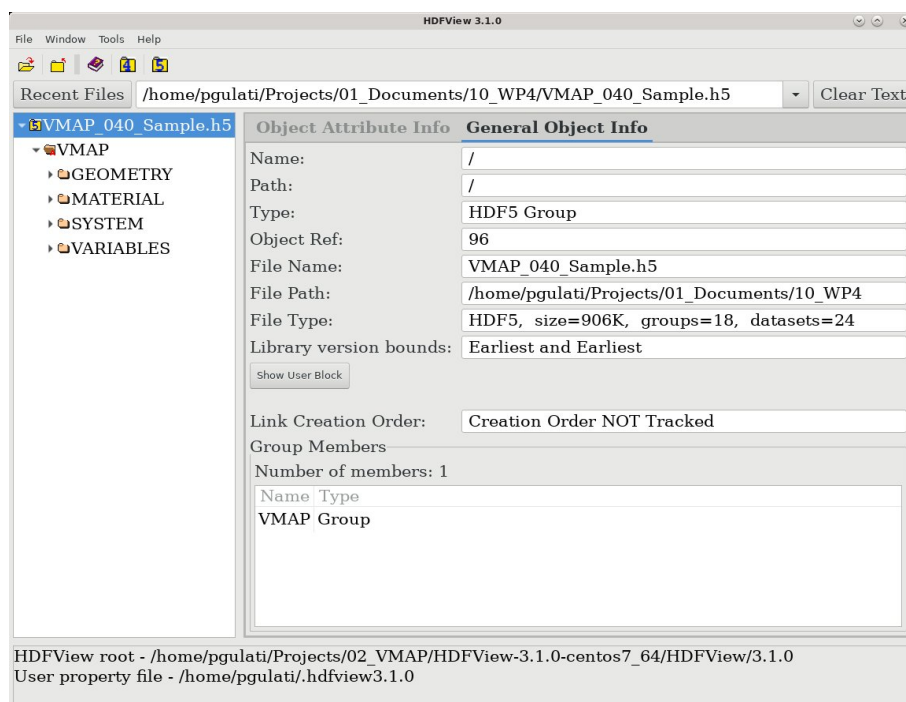


Figure 7.2: .h5 File View - General Object Info

7.2 VMAP Group View

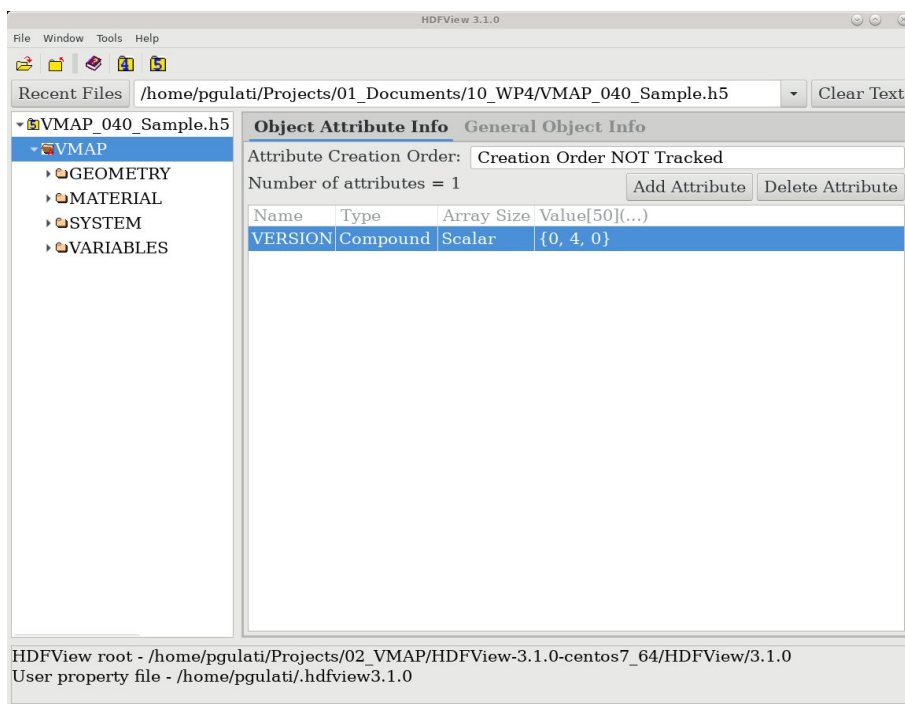


Figure 7.3: VMAP Group View - Object Attribute Info

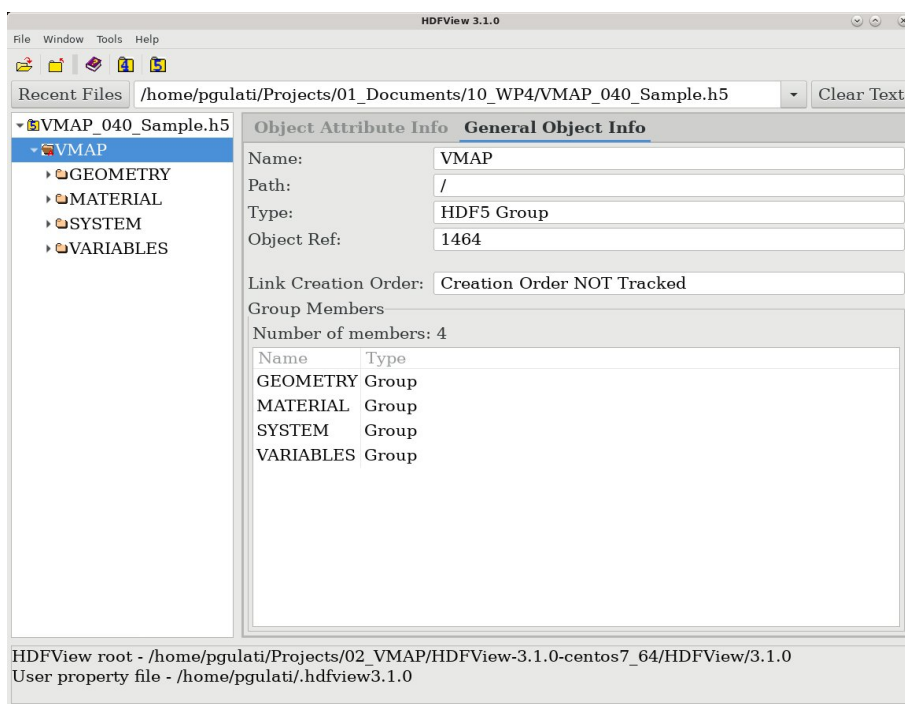


Figure 7.4: VMAP Group View - General Object Info

7.2.1 VERSION Attribute View

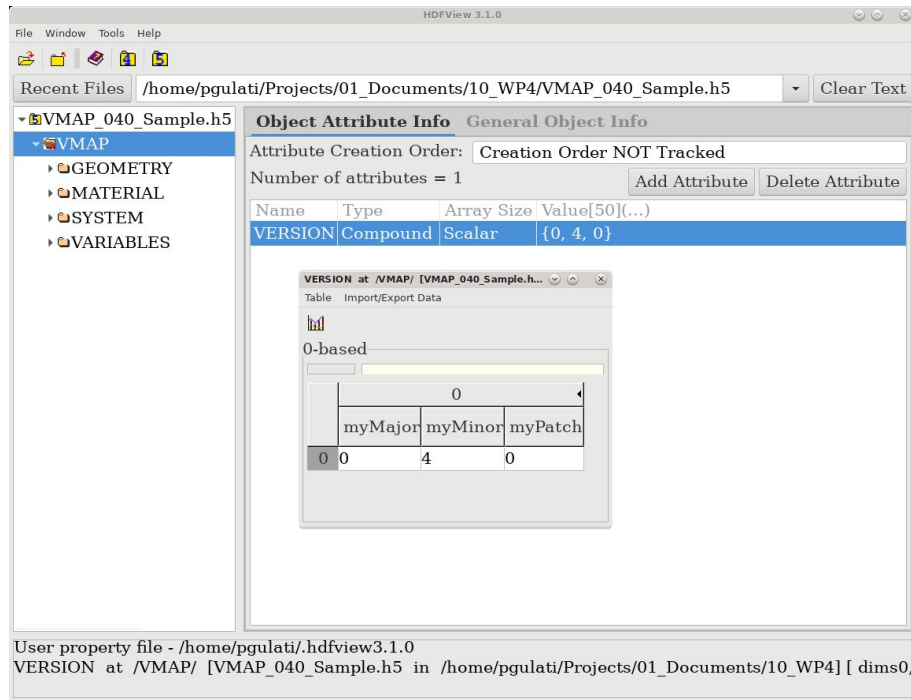


Figure 7.5: VERSION Attribute View - Metadata

7.3 GEOMETRY Group View

GEOMETRY Group has no attributes.

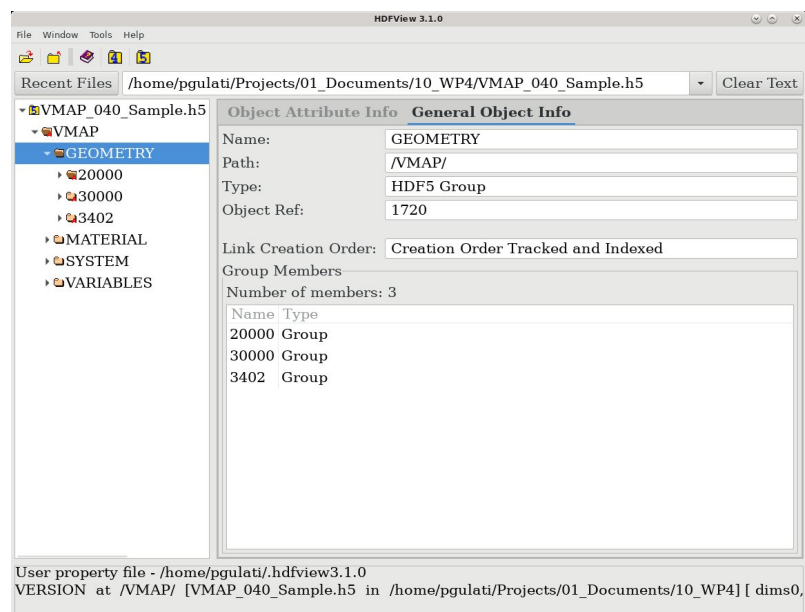


Figure 7.6: GEOMETRY Group View - General Object Info

7.4 <PART-ID> Group View

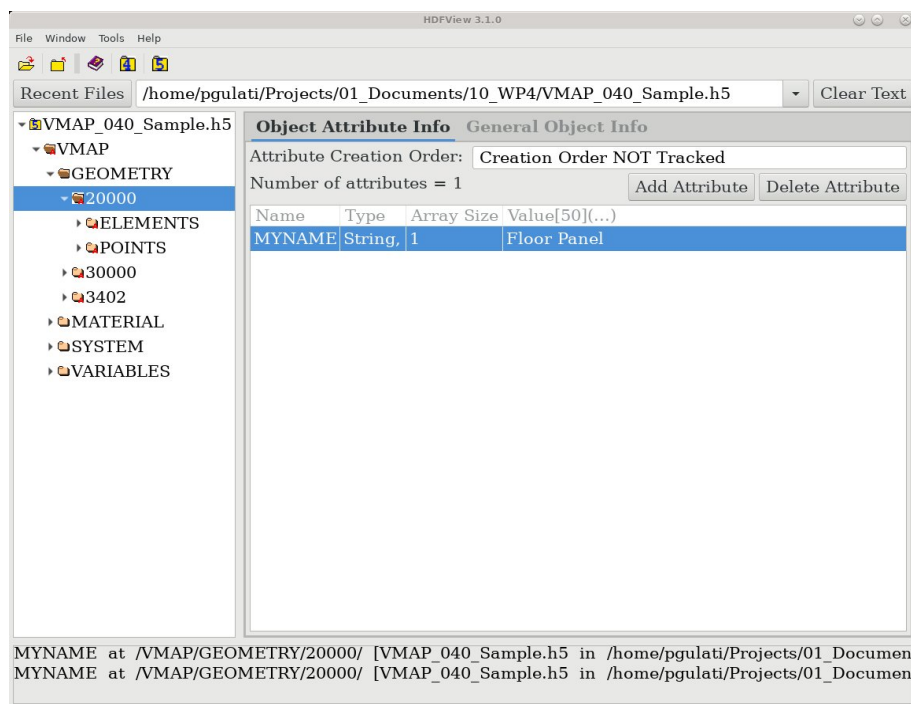


Figure 7.7: <PART-ID> Group View - Object Attribute Info

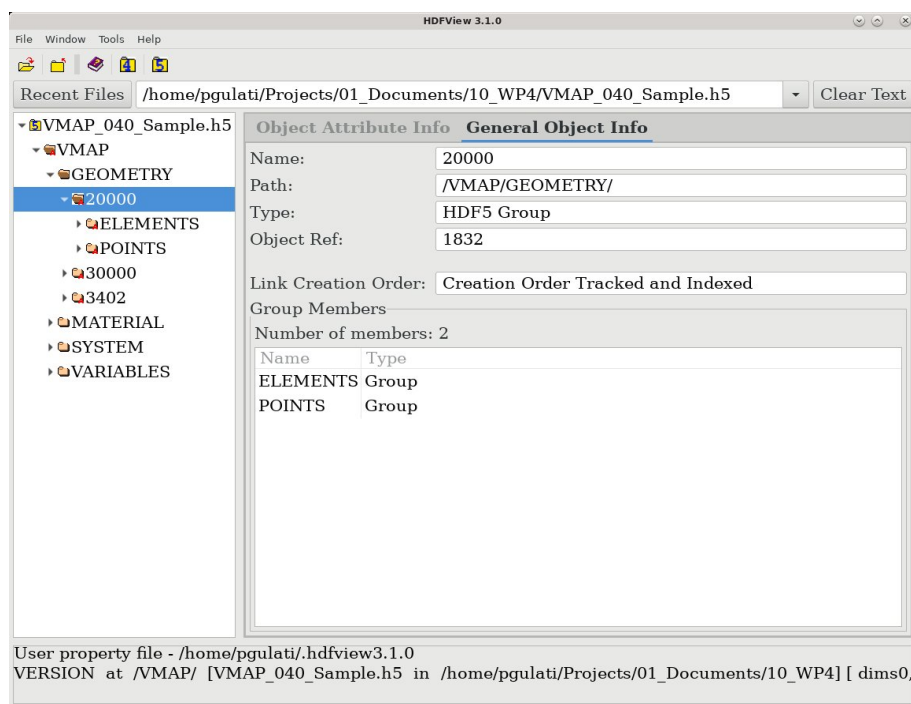


Figure 7.8: <PART-ID> Group View - General Object Info



The attribute MYNAME and its Value is shown in Figure 7.7

7.5 POINTS Group View

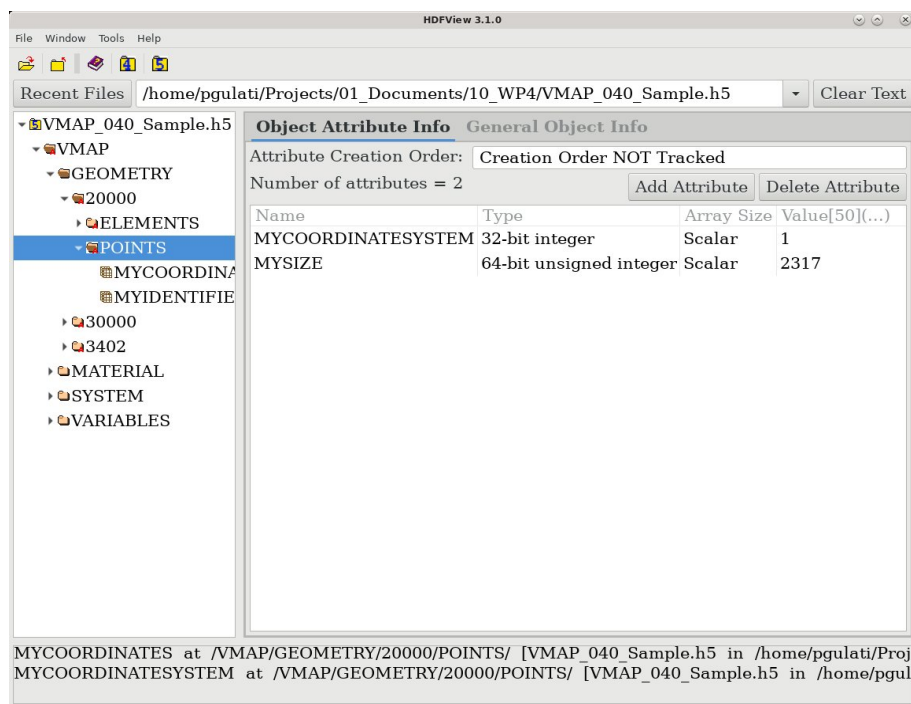


Figure 7.9: POINTS Group View - Object Attribute Info

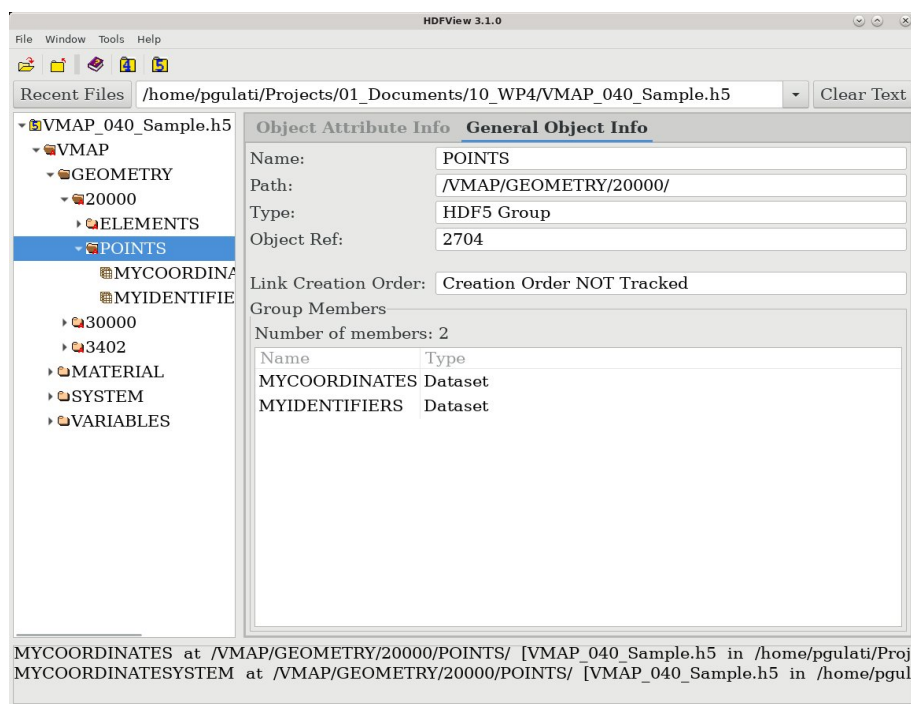


Figure 7.10: POINTS Group View - General Object Info

The attributes MYCOORDINATESYSTEM & MYSIZE are shown in Figure 7.9.

7.5.1 MYCOORDINATES Dataset

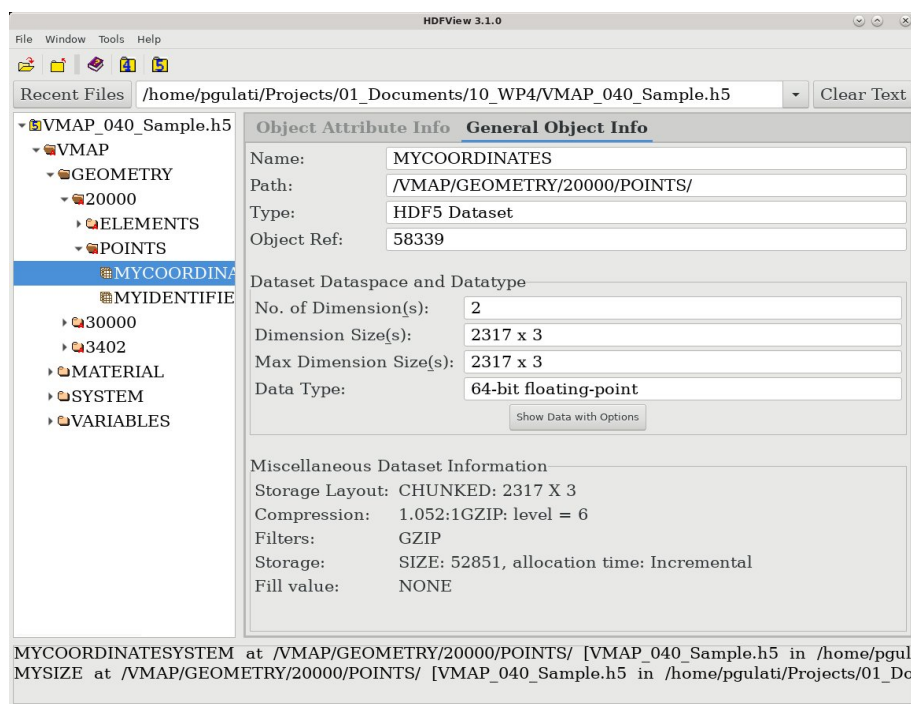


Figure 7.11: MYCOORDINATES Dataset View - General Object Info

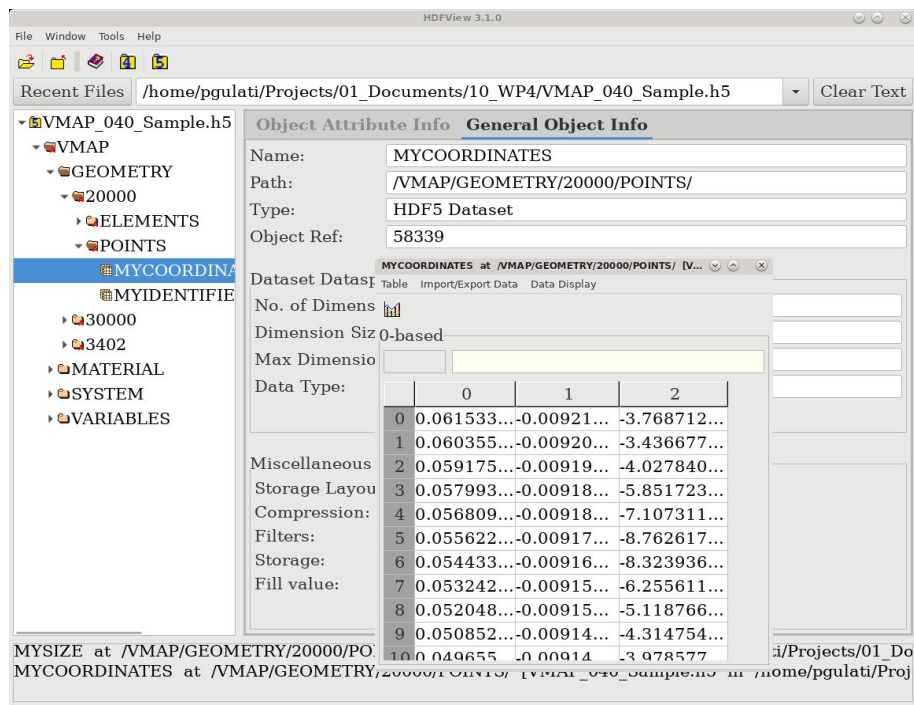


Figure 7.12: MYCOORDINATES Dataset View - Metadata

7.5.2 MYIDENTIFIERS Dataset

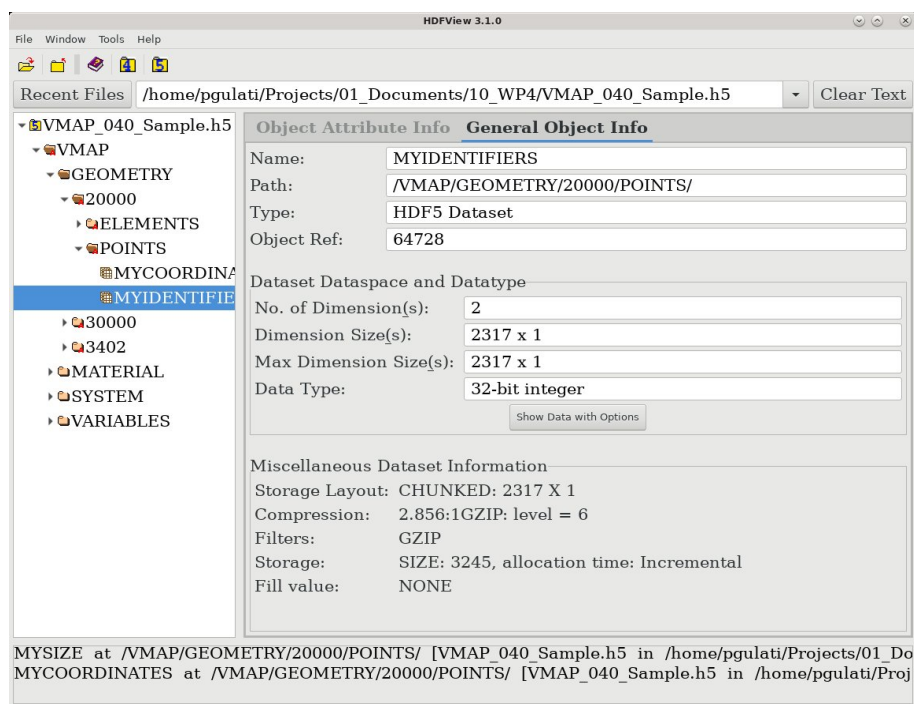


Figure 7.13: MYIDENTIFIERS Dataset View - General Object Info

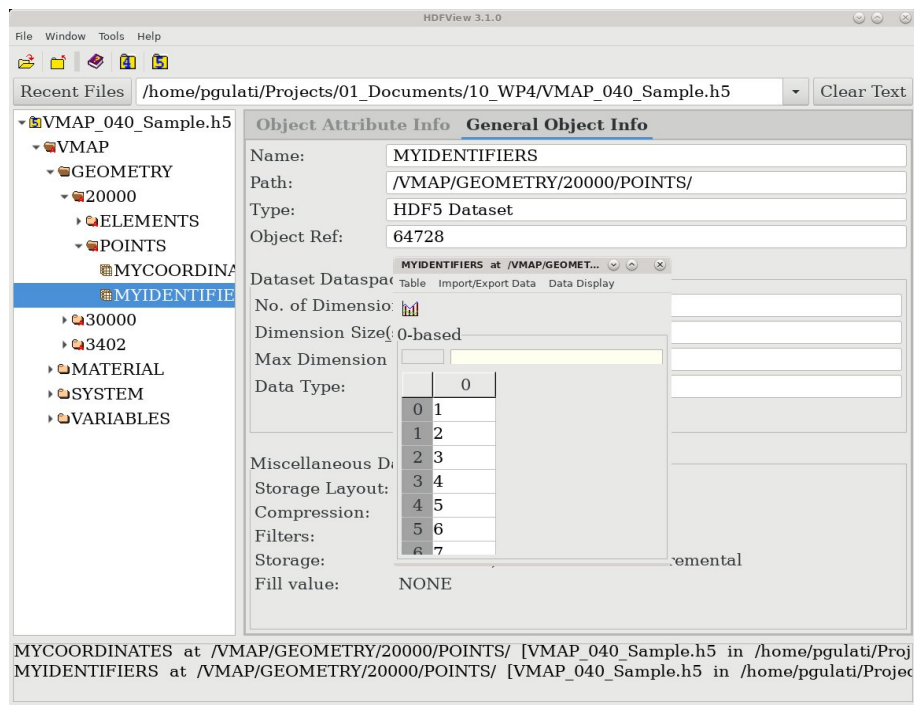


Figure 7.14: MYIDENTIFIERS Dataset View - Metadata

7.6 ELEMENTS Group View

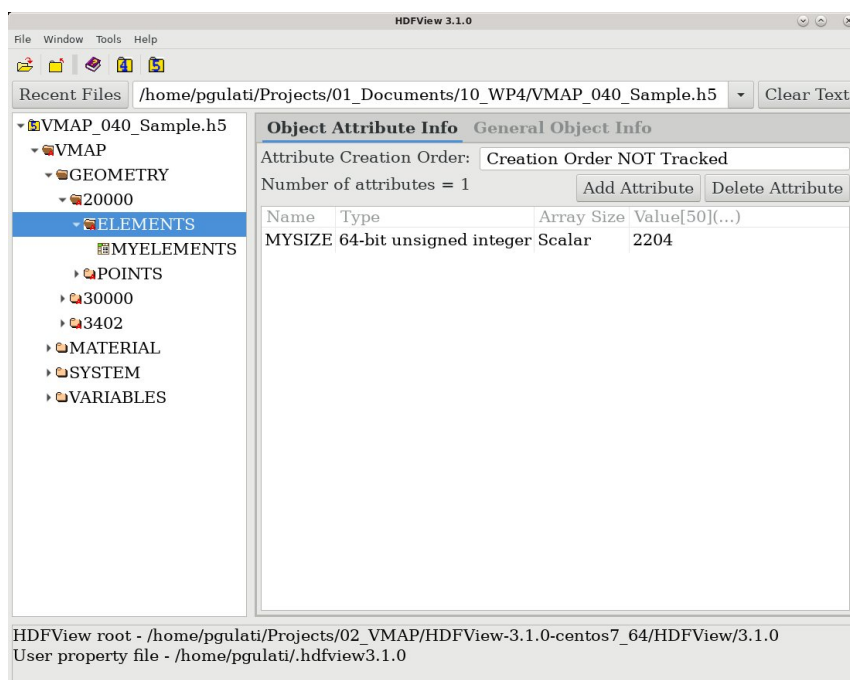


Figure 7.15: ELEMENTS Group View - Object Attribute Info

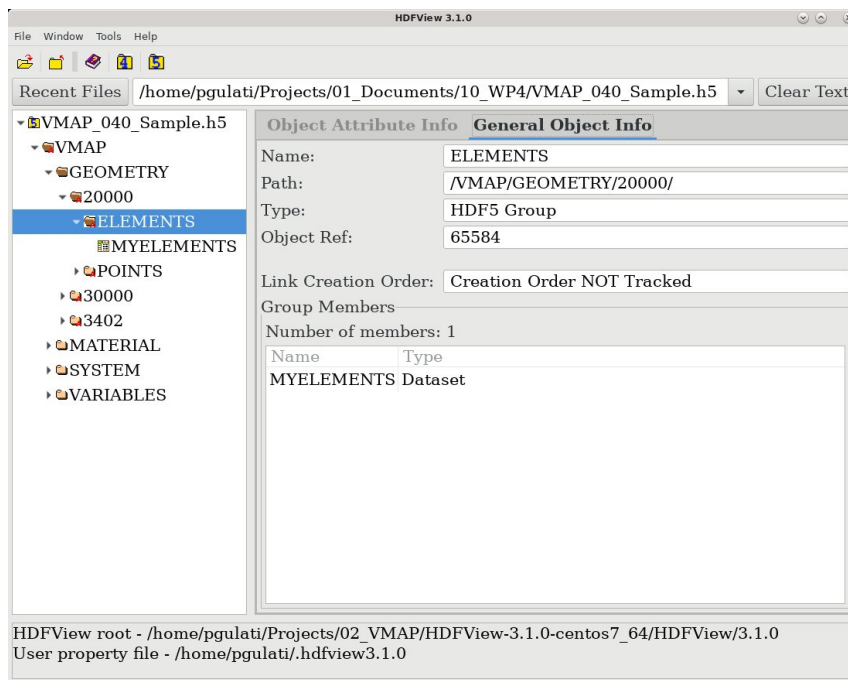


Figure 7.16: ELEMENTS Group View - General Object Info

The attribute MYSIZE is shown in Figure 7.15.

7.6.1 MYELEMENTS Dataset

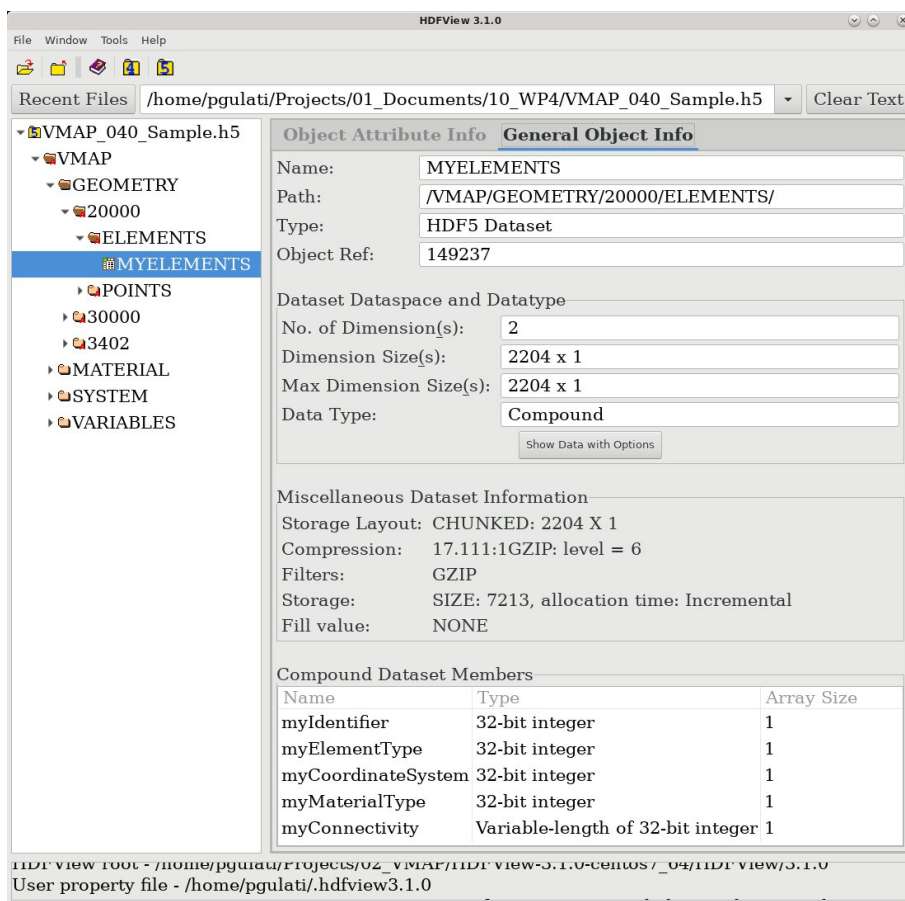


Figure 7.17: MYELEMENTS Dataset View - General Object Info

MYELEMENTS at /VMAP/GEOMETRY/20000/ELEMENTS/ [VMAP_040_Sample.h5 in /home/pgulati/Projects/01_Documents/10_WP4]

Table Import/Export Data

0-based

	myIdentifier	myElementType	myCoordinateSystem	myMaterialType	myConnectivity
0	1	1	1	-1	{(1, 200, 1731, 1730)}
1	2	1	1	-1	{(1730, 1731, 1733, ...}
2	3	1	1	-1	{(200, 199, 1732, 17...}
3	4	1	1	-1	{(1729, 1733, 1736, ...}
4	5	1	1	-1	{(1731, 1732, 1735, ...}
5	6	1	1	-1	{(199, 198, 1734, 17...}

Figure 7.18: MYELEMENTS Dataset View - Metadata

7.7 VARIABLES Group View

RESULT Group has no attributes.

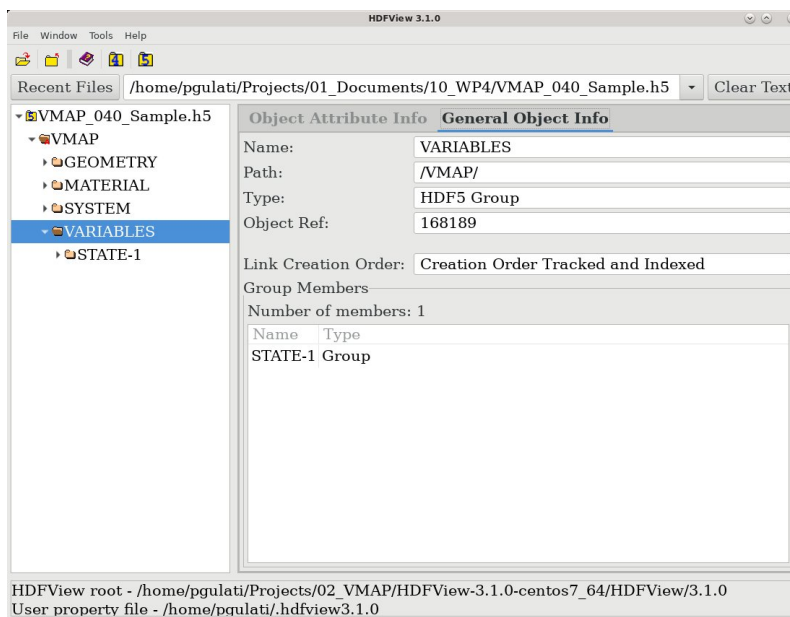


Figure 7.19: RESULT Group View - General Object Info

7.8 STATE-<n> Group View

STATE-<n> Group has no attributes.

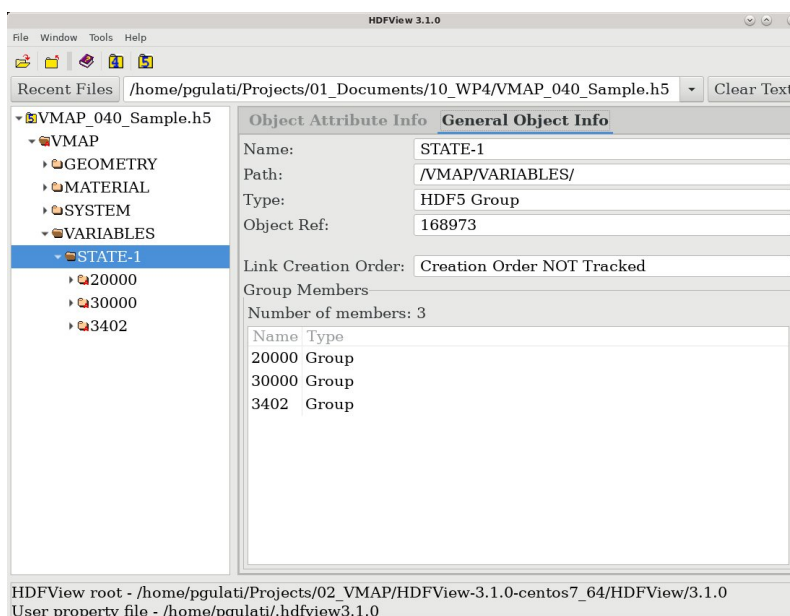


Figure 7.20: STATE-<n> Group View - General Object Info

7.9 <PART-ID> Group View

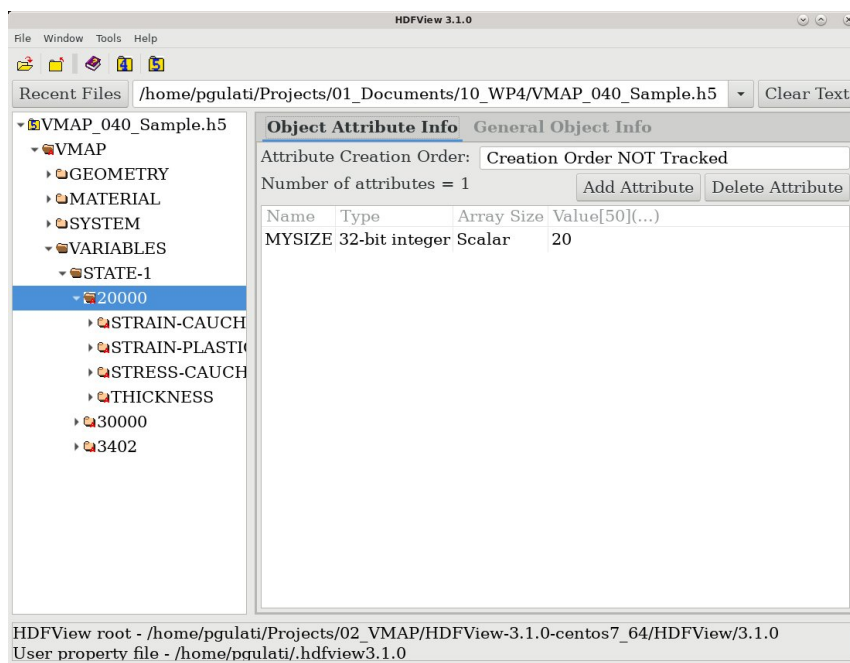


Figure 7.21: <PART-ID> Group View - Object Attribute Info

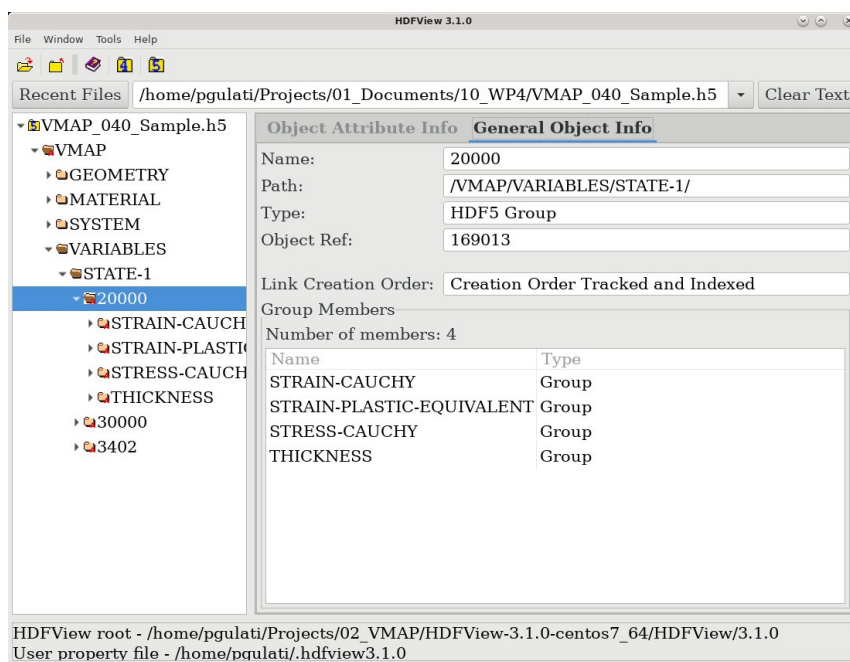


Figure 7.22: <PART-ID> Group View - General Object Info

The attribute MYSIZE is seen in Figure 7.21

7.10 STRAIN-CAUCHY Group View

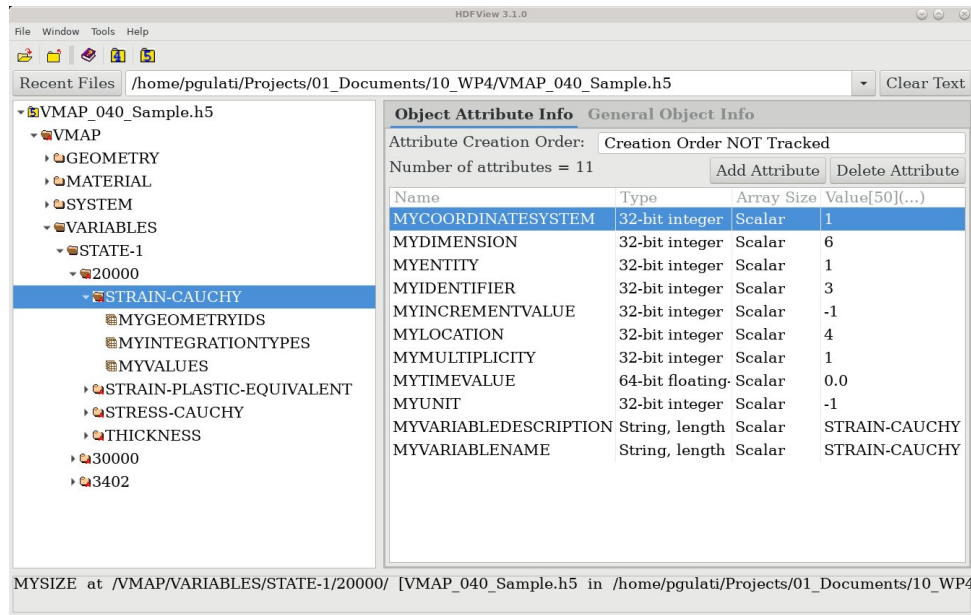


Figure 7.23: STRAIN-CAUCHY Group View - Object Attribute Info

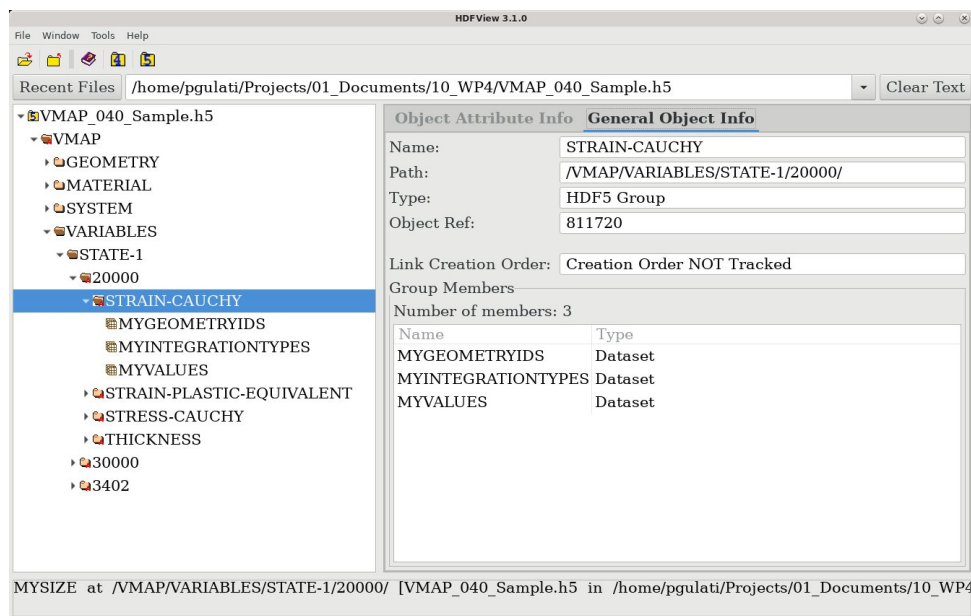


Figure 7.24: STRAIN-CAUCHY Group View - General Object Info

All attribute results can be seen in column Value of figure 7.23.

7.10.1 MYGEOMETRYIDS Dataset

This is an optional dataset.

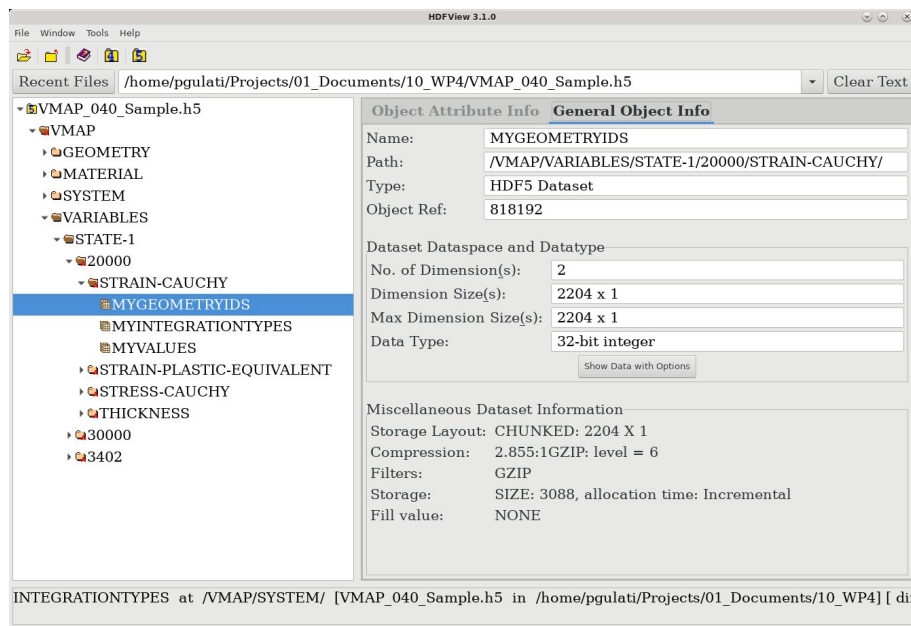


Figure 7.25: MYGEOMETRYIDS Dataset View - General Object Info

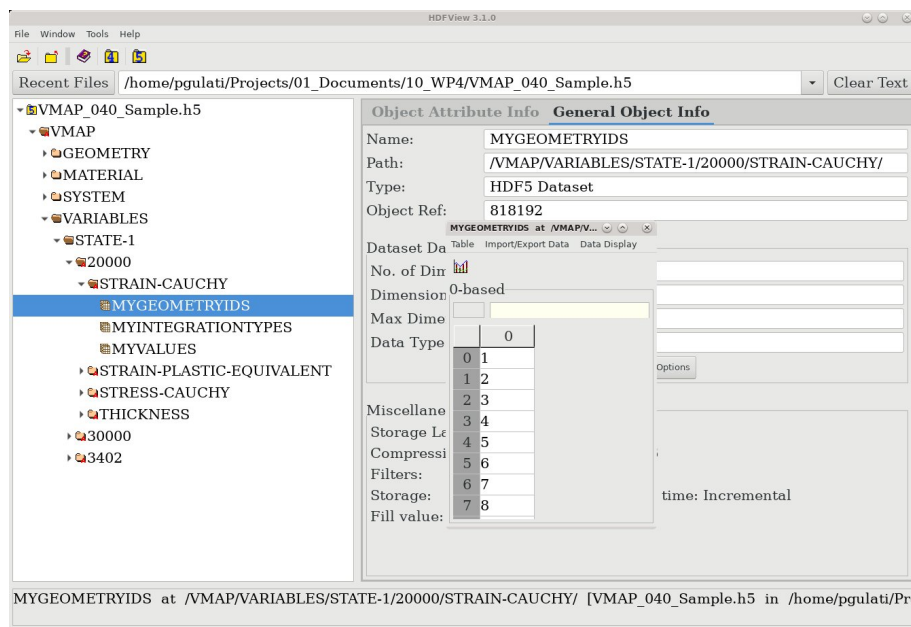


Figure 7.26: MYGEOMETRYIDS Dataset View - Metadata

7.10.2 MYVALUES Dataset

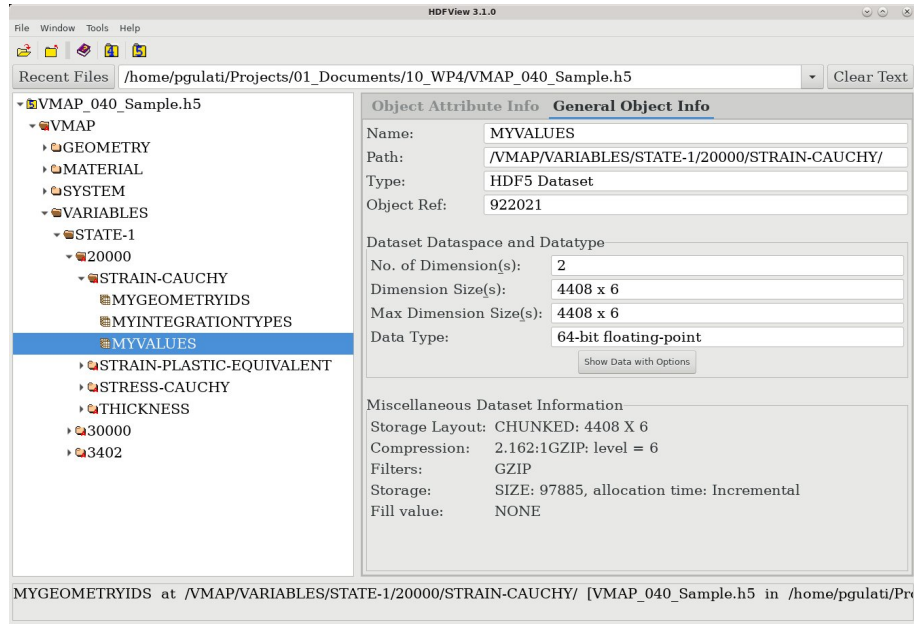


Figure 7.27: MYVALUES Dataset View - General Object Info

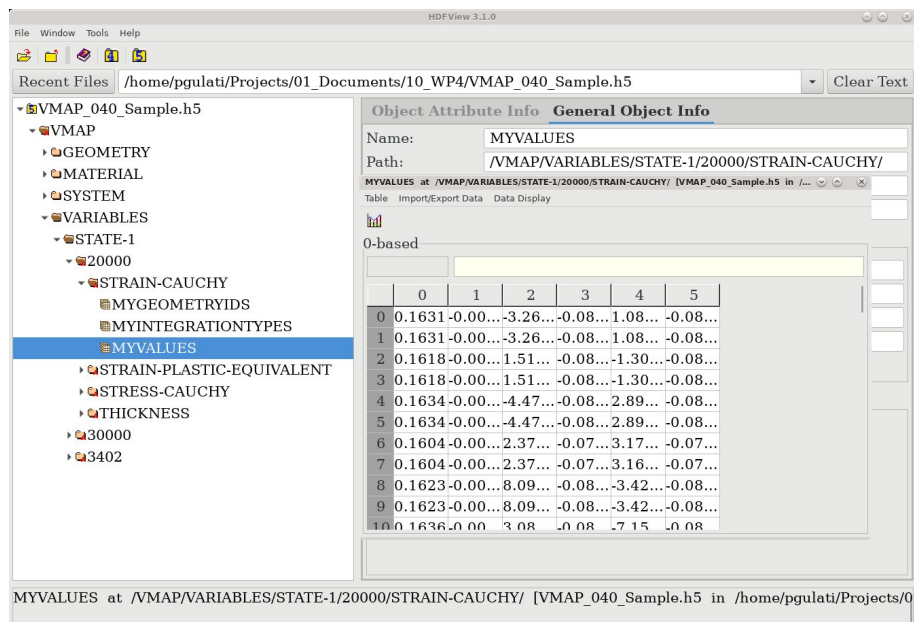


Figure 7.28: MYVALUES Dataset View - Metadata

7.10.3 MYINTEGRATIONTYPES Dataset

This is an optional dataset.

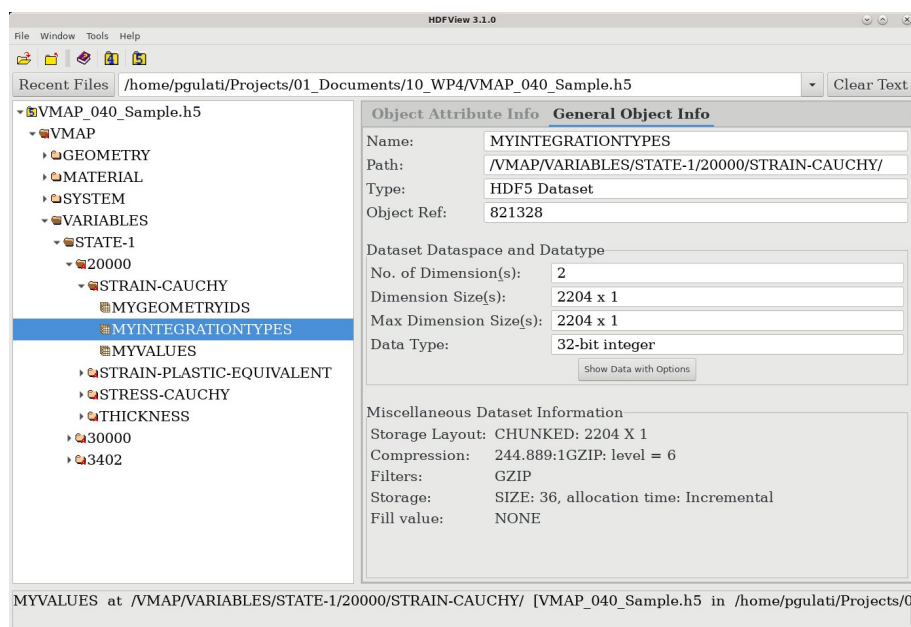


Figure 7.29: MYINTEGRATIONTYPES Dataset View - General Object Info

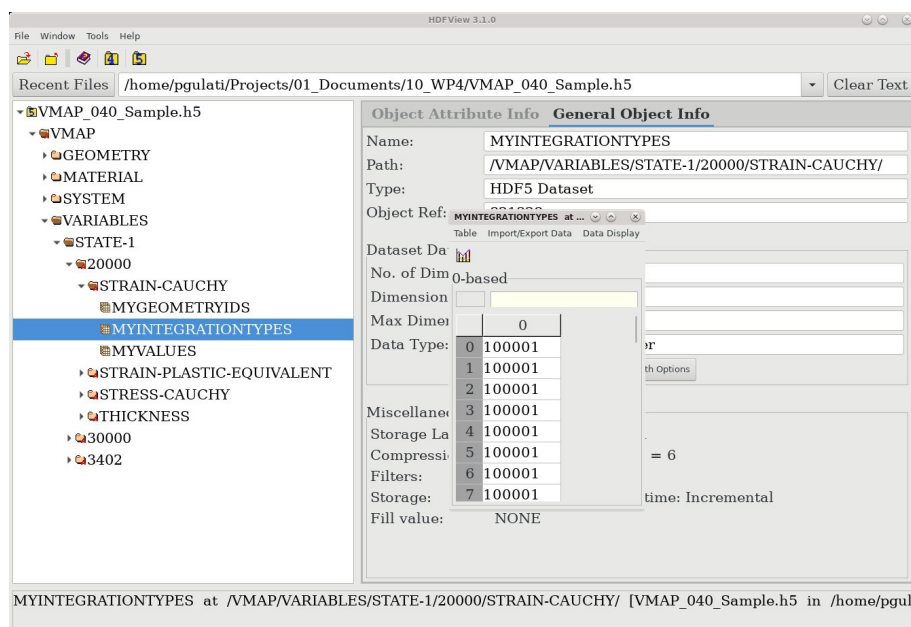


Figure 7.30: MYINTEGRATIONTYPES Dataset View - Metadata

7.11 SYSTEM Group View

SYSTEM Group has no attributes.

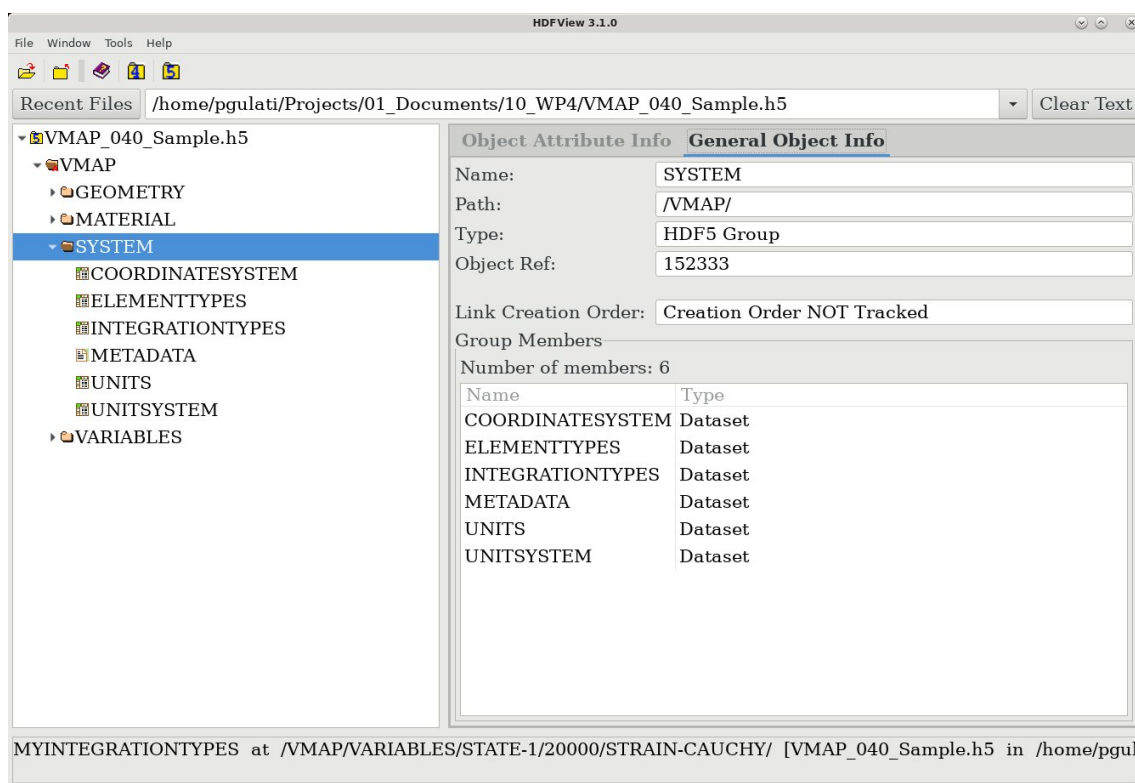


Figure 7.31: SYSTEM Group View - General Object Info

7.11.1 COORDINATESYSTEM Dataset

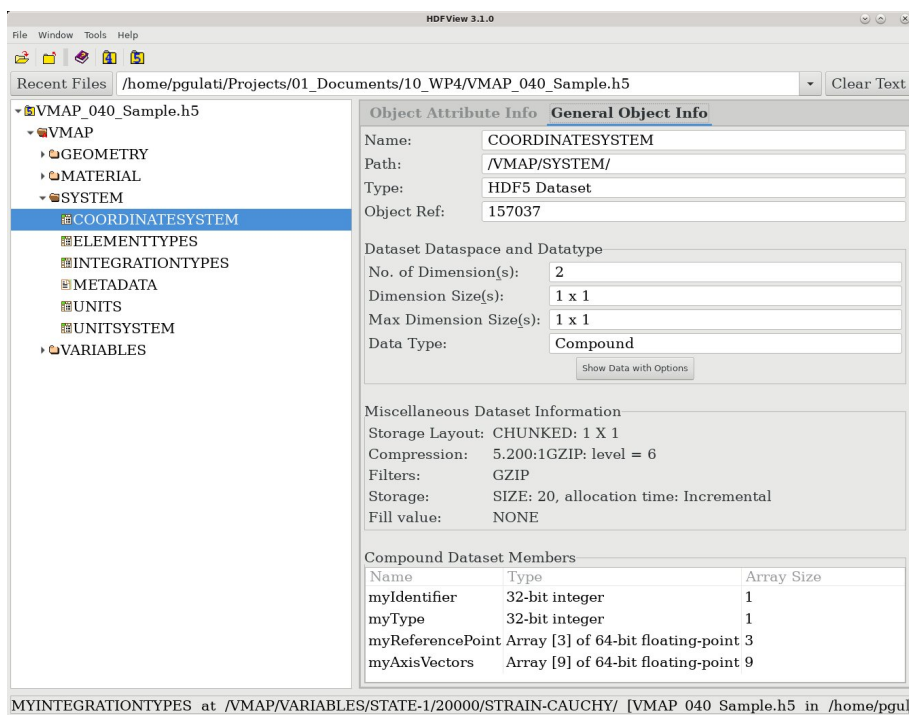


Figure 7.32: COORDINATESYSTEM Dataset View - General Object Info

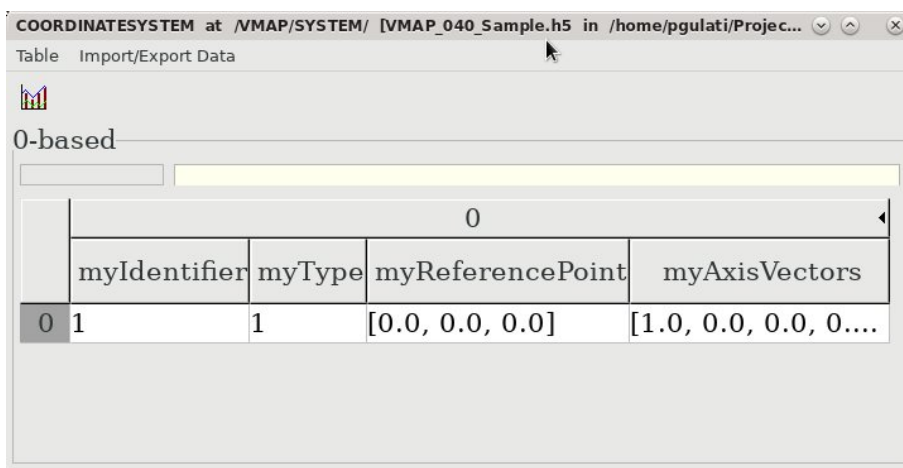


Figure 7.33 shows the COORDINATESYSTEM dataset view displaying the metadata table. The table is titled "0-based" and contains the following data:

	myIdentifier	myType	myReferencePoint	myAxisVectors
0	1	1	[0.0, 0.0, 0.0]	[1.0, 0.0, 0.0, 0....]

Figure 7.33: COORDINATESYSTEM Dataset View - Metadata

7.11.2 ELEMENTTYPES Dataset

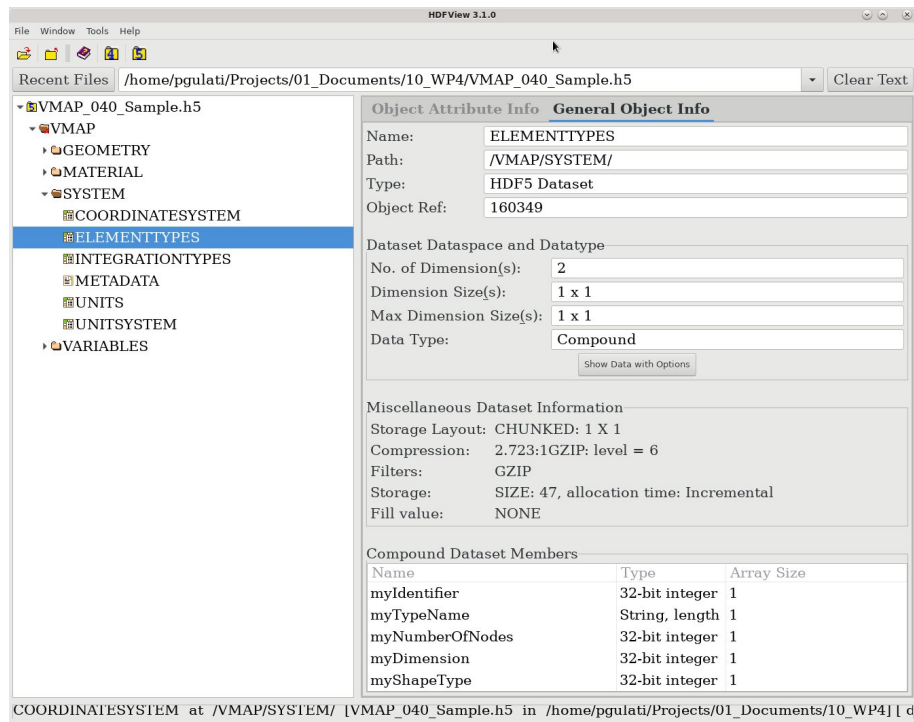


Figure 7.34: ELEMENTTYPES Dataset View - General Object Info

ELEMENTTYPES at /VMAP/SYSTEM/ [VMAP_040_Sample.h5 in /home/pgulati/Projects/01_Doc...

Table Import/Export Data

0-based

	myIdentifier	myTypeName	myNumberOfNodes	m
0	1	VMAP_ELEM_3D_QUAD_4	4	3

Figure 7.35: ELEMENTTYPES Dataset View - Metadata

7.11.3 INTEGRATIONTYPES Dataset

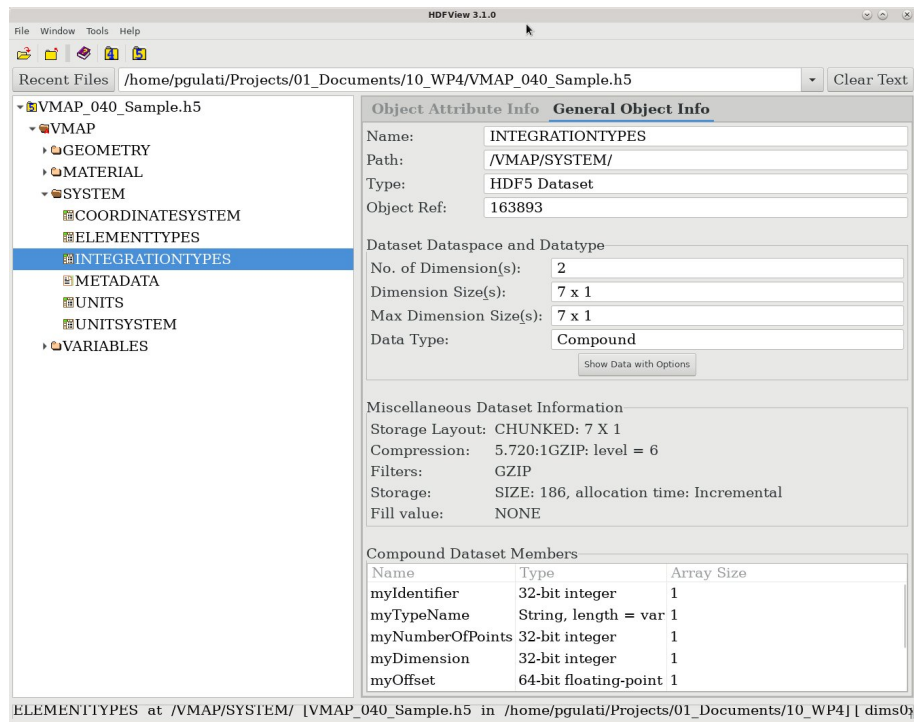


Figure 7.36: INTEGRATIONTYPES Dataset View - General Object Info

Table Import/Export Data

0-based

	myIdentifier	myTypeName	myNumberOfPoints	myDimension	myOffset	myAbscissas	myWeights	mySubTypes
0	100000	VMAP_NODES_QUAD_4	4	2	0.0	{{(-1, -1, 1, ...	{{(1, 1, 1, ...	{0}
1	71	VMAP_GAUSS_QUAD_1	1	2	0.0	{{(0, 0)}	{{(4)}	{0}
2	17	VMAP_LOBATTO_2	2	1	0.0	{{(-1, 1)}	{{(1, 1)}	{0}
3	100001	VMAP_GAUSS_QUAD_1xVMAP_LOBATTO_2	22	3	0.0	{{(0, 0, -1, ...	{{(4, 4)}	{{(71, 17)}
4	72	VMAP_GAUSS_QUAD_4	4	2	0.0	{{(-0.57735...{	{{(1, 1, 1, ...	{0}
5	2	VMAP_GAUSS_3	3	1	0.0	{{(-0.77459...{	{{(0.5555...{	{0}
6	100002	VMAP_GAUSS_QUAD_4xVMAP_GAUSS_3	12	3	0.0	{{(-0.57735...{	{{(0.5555...{	{{(72, 2)}

Figure 7.37: INTEGRATIONTYPES Dataset View - Metadata

7.11.4 METADATA Dataset

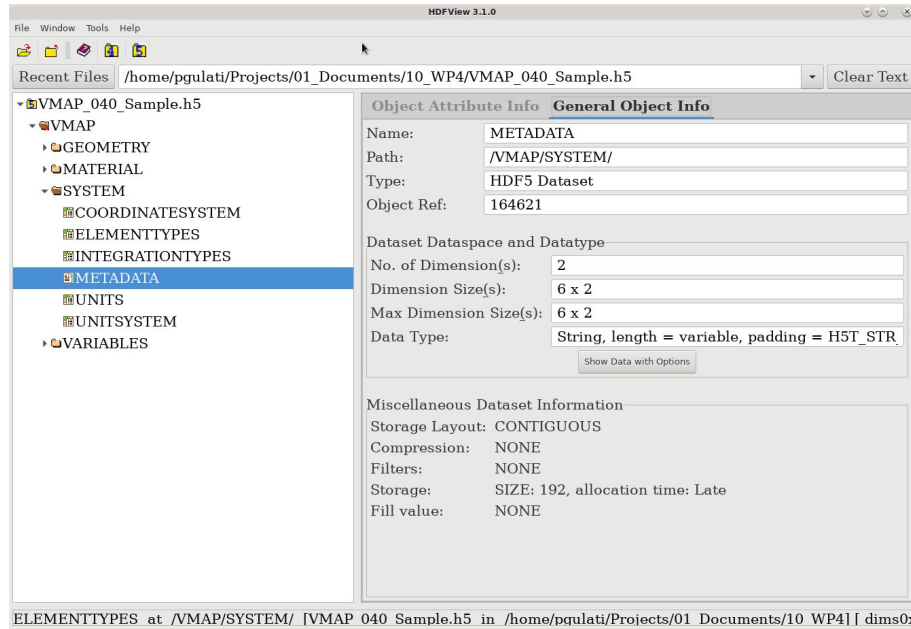
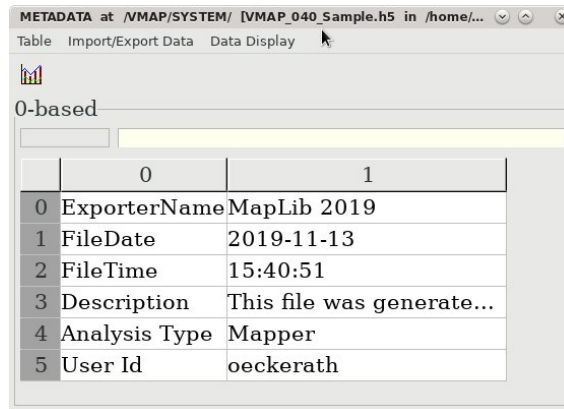


Figure 7.38: METADATA Dataset View - General Object Info



0-based

	0	1
0	ExporterName	MapLib 2019
1	FileDate	2019-11-13
2	FileTime	15:40:51
3	Description	This file was generate...
4	Analysis Type	Mapper
5	User Id	oeckerath

Figure 7.39: METADATA Dataset View - Metadata

7.11.5 UNITS Dataset

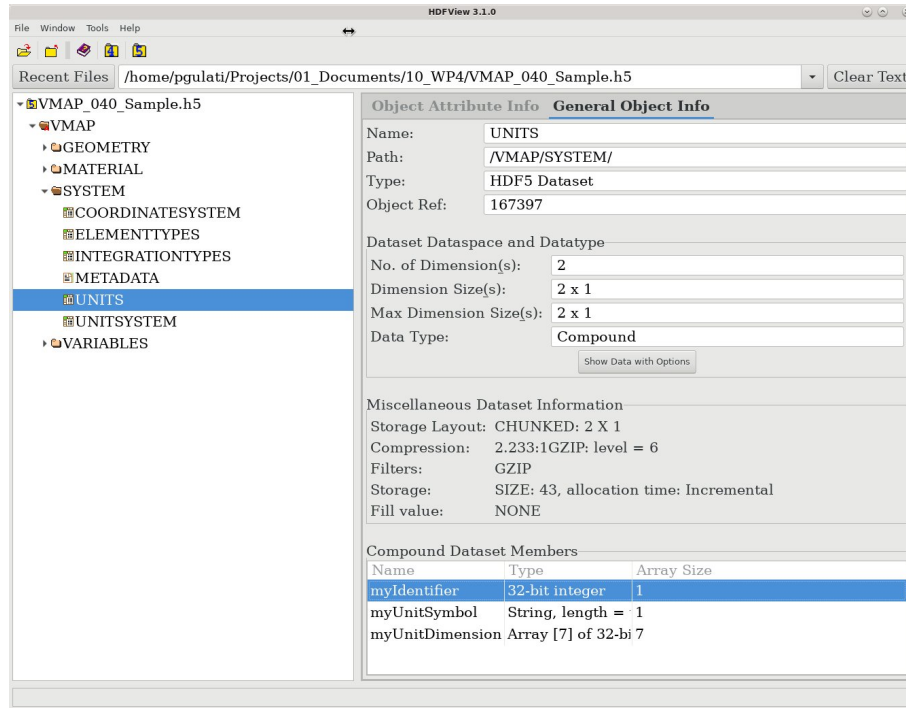


Figure 7.40: UNITS Dataset View - General Object Info

UNITS at /VMAP/SYSTEM/ [VMAP_040_Sample.h5 in /home/pgulati/Projects/01_Doc...]

Table Import/Export Data

0-based

	myIdentifier	myUnitSymbol	myUnitDimension
0	1	mm	[1, 0, 0, 0, 0, 0, 0]
1	2	MPa	[-1, 1, -2, 0, 0, 0, 0]

Figure 7.41: UNITS Dataset View - Metadata

7.11.6 UNITSYSTEM Dataset

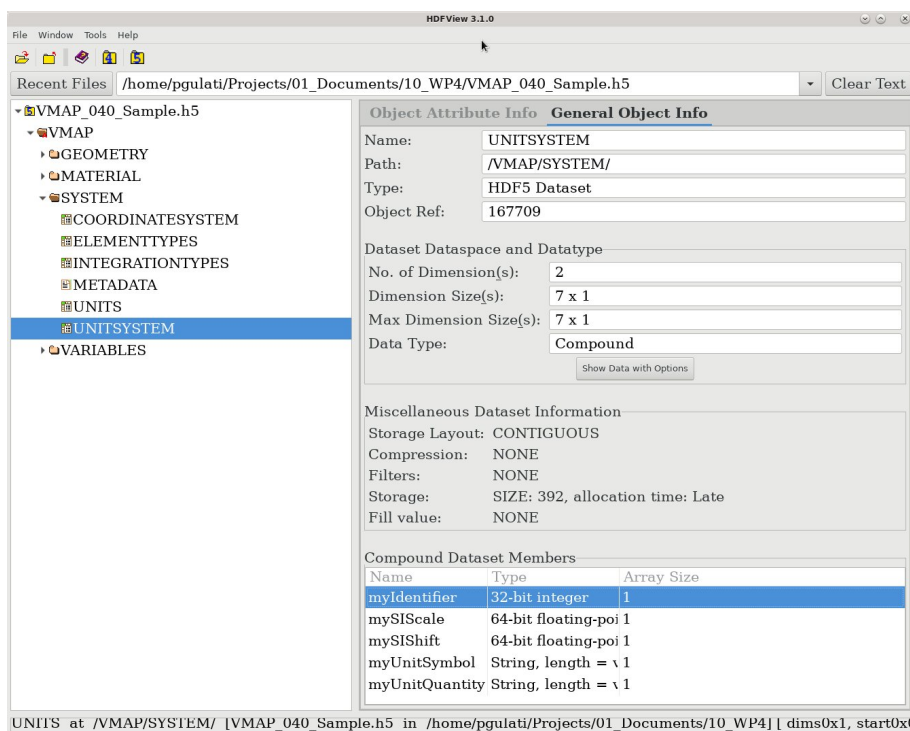



Figure 7.42: UNITSYSTEM Dataset View - General Object Info

UNITSYSTEM at /VMAP/SYSTEM/ [VMAP_040_Sample.h5 in /home/pgulati/Projects/01_Documents/10_WP4]

Table Import/Export Data



0-based

	0				
	myIdentifier	mySIScale	mySIShift	myUnitSymbol	myUnitQuantity
0	1	0.001	0.0	mm	LENGTH
1	2	1000.0	0.0	t	MASS
2	3	1.0	0.0	s	TIME
3	4	1.0	0.0	A	ELECTRIC CURRENT
4	5	1.0	0.0	K	TEMPERATURE
5	6	1.0	0.0	mol	AMOUNT OF SUBSTANCE
6	7	1.0	0.0	cd	LUMINOUS INTENSITY

Figure 7.43: UNITSYSTEM Attribute View - Metadata

Chapter 8

Element Definition Specifications

VMAP Element factory contains 31 different 1D, 2D and 3D element definitions. These are some of the basic elements which are largely used in the CAE Domain. This chapter provides the specifications of these elements. Additionally, it outlines a standard method to define your own elements, which follows the same specifications as VMAP Elements. All elements defined in the VMAP Element Library, `VMAPElementTypeFactory.cxx`, belong to domain VMAP e.g. `VMAP_ELEM_3D_QUAD_4`.

8.1 USER_DEFINED

A user-defined element is explained with the help of an example. First step is to defined the correct point order, in VMAP the point numbering is defined using the right hand rule with counter-clockwise direction and the vector pointing out-of plane for 2D elements. For 3D elements, right hand rule is used with the counter-clockwise direction and the vector pointing inwards (inside the volume). See figure 8.1.

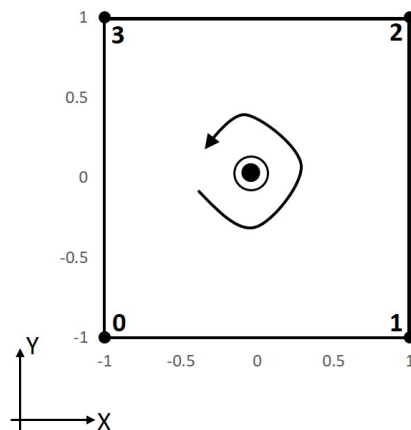


Figure 8.1: Point Ordering for a QUAD Element

Once the correct point numbering is in place, the following method is used by VMAP to define an element. See Figure 8.2

```

case sElementType::QUAD_4: {
    vmapType.setNumberOfNodes(4);

    std::vector<int> connectivity(4);
    connectivity[0] = 0;
    connectivity[1] = 1;
    connectivity[2] = 2;
    connectivity[3] = 3;
    vmapType.setConnectivity(connectivity);

    // Shell like element with two faces
    if(elemDimension == sElementType::ELEM_3D) {
        std::vector<int> faceconnectivity(2*4+2+1);
        faceconnectivity[0] = 2; ①
        // face 1
        faceconnectivity[1] = 4; ②
        faceconnectivity[2] = 0;
        faceconnectivity[3] = 1; ③
        faceconnectivity[4] = 2;
        faceconnectivity[5] = 3;
        // face 2
        faceconnectivity[6] = 4; ②
        faceconnectivity[7] = 0; ③
        faceconnectivity[8] = 3;
        faceconnectivity[9] = 2;
        faceconnectivity[10] = 1;
        vmapType.setFaceConnectivity(faceconnectivity);
    }
}

```

A switch case with the Element Type

Setting up the connectivity with the correct point ordering.

Length of faceconnectivity vector =
 No. of Faces x Points per Face
 + No. of Faces + 1

1. A 3D Quad has 2 faces.
2. Each Face has 4 Points.
3. The order in which the Points are connected

Figure 8.2: C++ Code for QUAD 3D Element Definition

8.2 POINT

POINT is a point element. Such an element can be used to define MASS (STATE VARIABLE). There will be no `connectivity` or `faceconnectivity` vector defined for this element; only parameter `myNumberOfNodes` set via function call `setNumberOfNodes`.

8.3 LINE_2

A line element with 2 Points can be used to define a BEAM, BAR etc. Figure 8.3 shows the LINE_2 element with points 0 and 1.

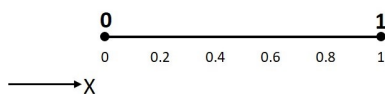


Figure 8.3: LINE_2 Element

8.4 LINE_3

LINE_3 is a line element with 2 corner points and 1 middle point. Figure 8.4 shows the LINE_3 element with points 0, 1 and 2.

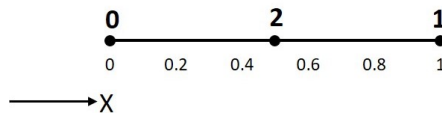


Figure 8.4: LINE_3 Element

8.5 LINE_4

LINE_4 is a line element with 2 corner points and 2 middle points. Figure 8.5 shows the LINE_4 element with points 0, 1, 2 and 3.

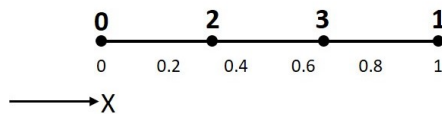


Figure 8.5: LINE_4 Element

8.6 TRIANGLE_3

TRIANGLE_3 element is a triangle with 3 points. This triangle could be used as a 2D element with 3 faces (3 lines) or as a 3D element with 2 faces. Figures 8.6 shows 2D and 3D TRIANGLE_3 elements.

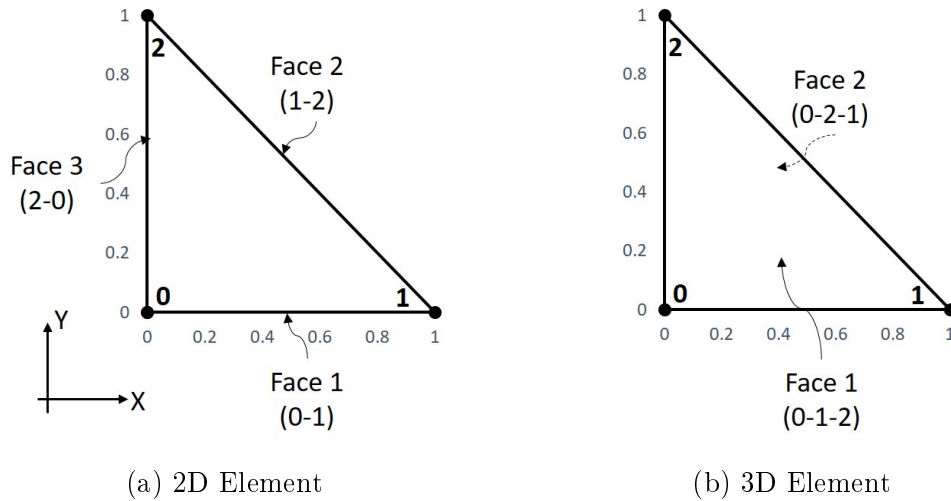


Figure 8.6: TRIANGLE_3

8.7 TRIANGLE_4

TRIANGLE_4 has the same parameterization as shown for TRIANGLE_3, with an additional center point. Figure 8.7 shows TRIANGLE_4 element.

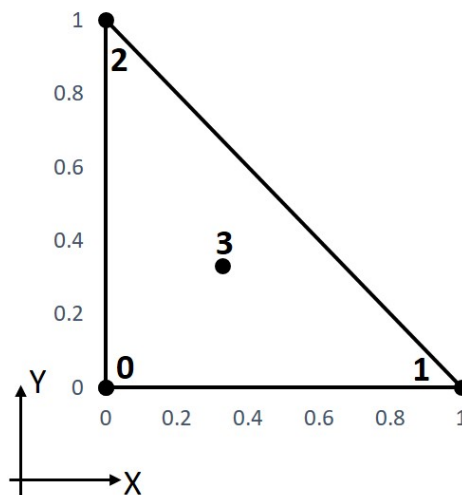


Figure 8.7: TRIANGLE_4 Element

8.8 TRIANGLE_6

TRIANGLE_6 element is a triangle with 3 corner points and 3 middle points - one for each face (line). This triangle could be used as a 2D element with 3 faces (3 lines) or as a 3D element with 2 faces. Figure 8.8 shows 2D and 3D TRIANGLE_6 elements.

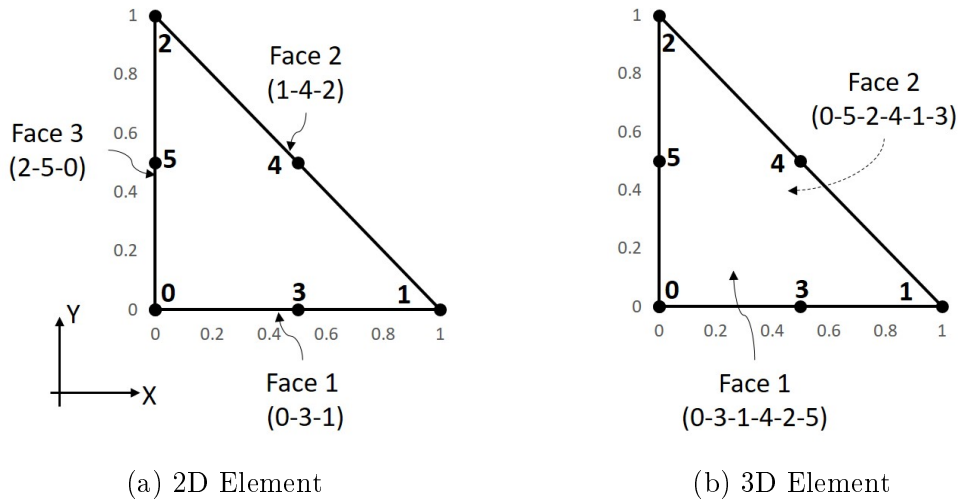


Figure 8.8: TRIANGLE_6

8.9 QUAD_4

QUAD_4 is a quadrilateral with 4 points. The quadrilateral could be used as a 2D element with 4 faces (4 lines) or a 3D element with 2 faces. Figures 8.9 shows 2D and 3D QUAD_4 elements.

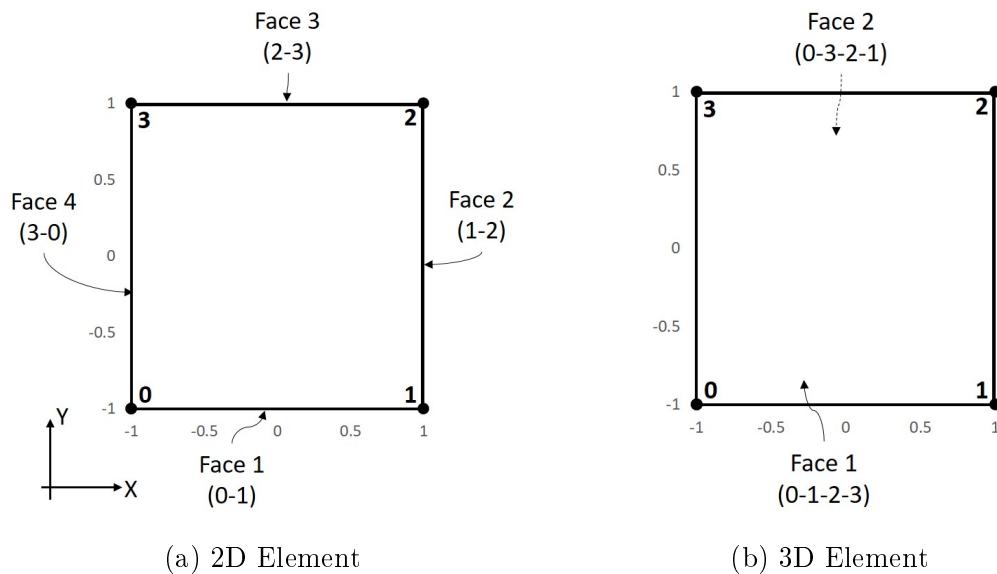


Figure 8.9: QUAD_4

8.10 QUAD_8

QUAD_8 is a quadrilateral with 4 corner points and 4 middle points - one for each face (line). The quadrilateral could be used as a 2D element with 4 faces (4 lines)

or a 3D element with 2 faces. Figures 8.10 shows 2D and 3D QUAD_8 elements.

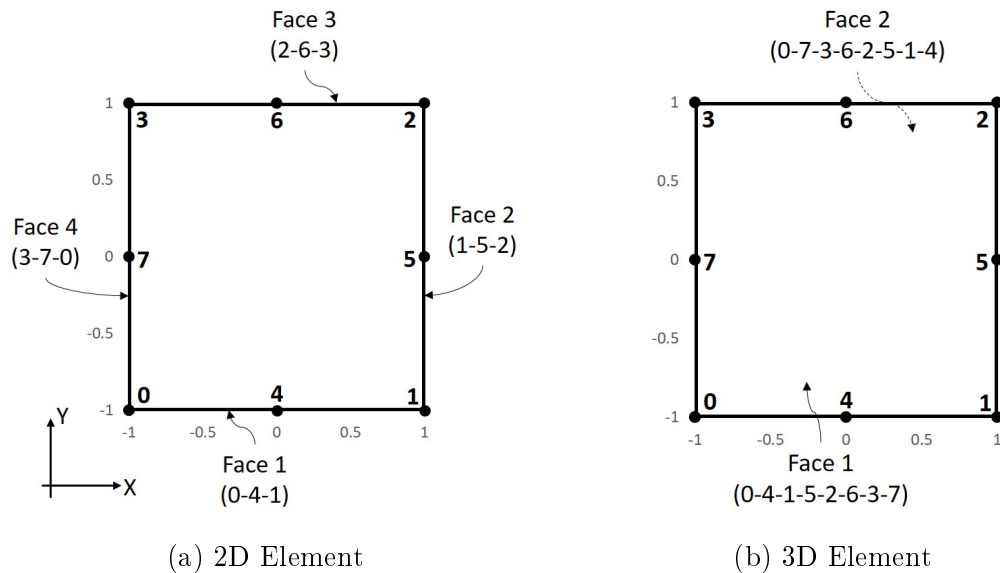


Figure 8.10: QUAD_8

8.11 QUAD_9

QUAD_9 has the same parameterization as shown for QUAD_8, with an additional center point. Figures 8.11 shows QUAD_9 element.

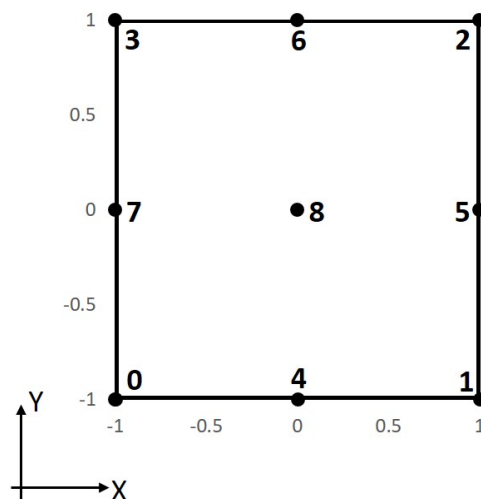


Figure 8.11: QUAD_9 Element

8.12 TETRAHEDRON_4

TETRAHEDRON_4 is a tetrahedral element with 4 points. Figure 8.12 shows a TETRAHEDRON_4 element.

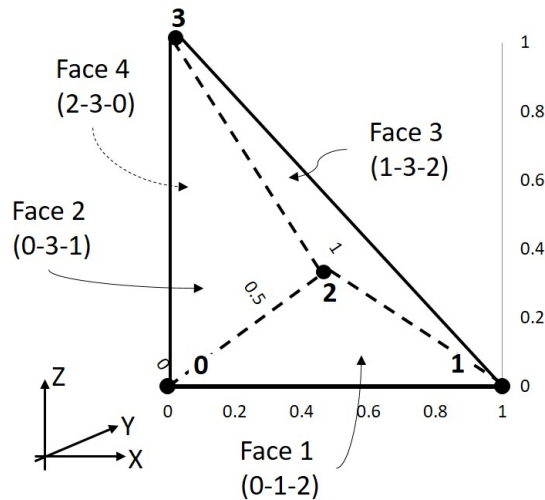


Figure 8.12: TETRAHEDRON_4 Element

8.13 TETRAHEDRON_5

TETRAHEDRON_5 has the same parameterization as TETRAHEDRON_4, with an additional center point. Figure 8.13 shows a TETRAHEDRON_5 element.

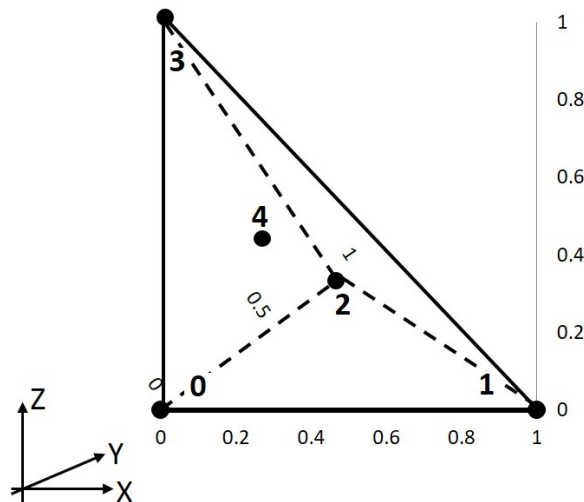


Figure 8.13: TETRAHEDRON_5 Element

8.14 TETRAHEDRON_10

TETRAHEDRON_10 is a tetrahedral element with 4 corner points and 6 middle points - one for each face (line). Figure 8.14 shows a TETRAHEDRON_10 element.

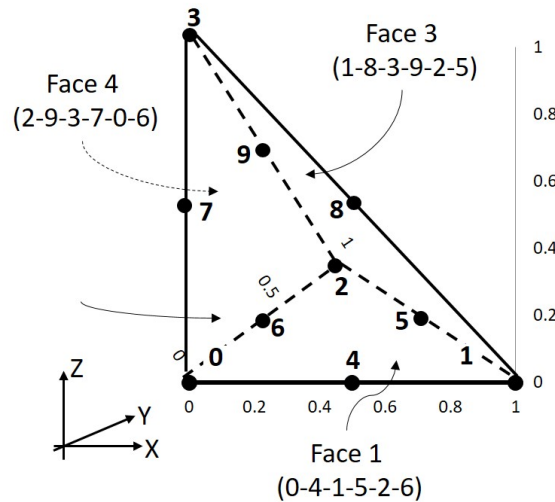


Figure 8.14: TETRAHEDRON_10 Element

8.15 TETRAHEDRON_11

TETRAHEDRON_11 has the same parameterization as TETRAHEDRON_10, with an additional center point. Figure 8.15 shows a TETRAHEDRON_11 element.

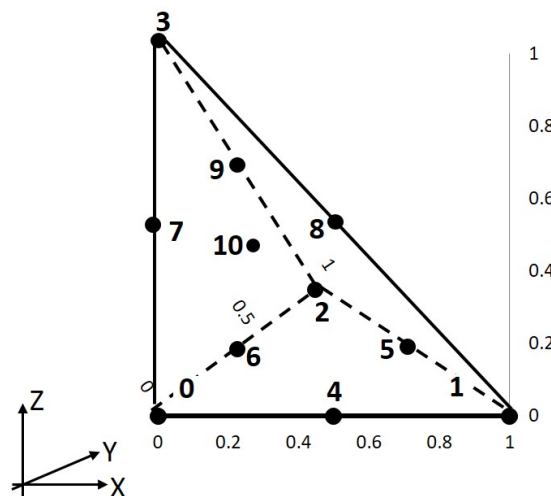


Figure 8.15: TETRAHEDRON_11 Element

8.16 PYRAMID_5

PYRAMID_5 is a pyramid element with 5 points. Figure 8.16 shows a PYRAMID_5 element.

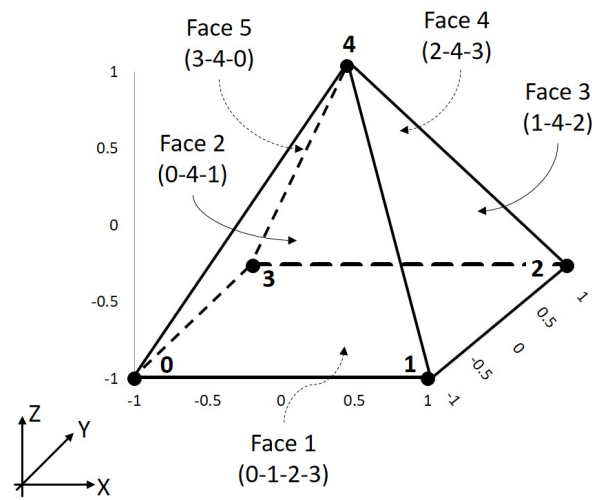


Figure 8.16: PYRAMID_5 Element

8.17 PYRAMID_6

PYRAMID_6 has the same parameterization as PYRAMID_5, with an additional center point. Figure 8.17 shows a PYRAMID_6 element.

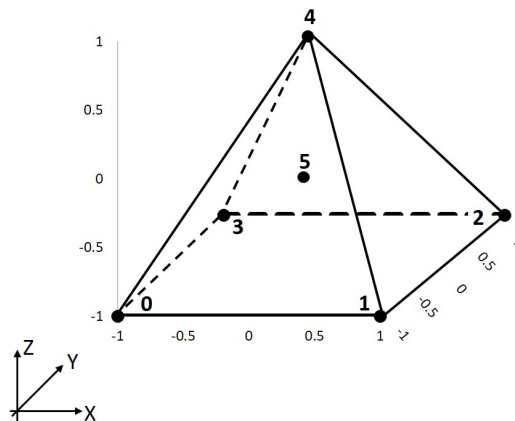


Figure 8.17: PYRAMID_6 Element

8.18 PYRAMID_13

PYRAMID_13 is a pyramid element with 5 corner points, additionally 8 middle points - one on each face (line). Figure 8.18 shows a PYRAMID_13 element.

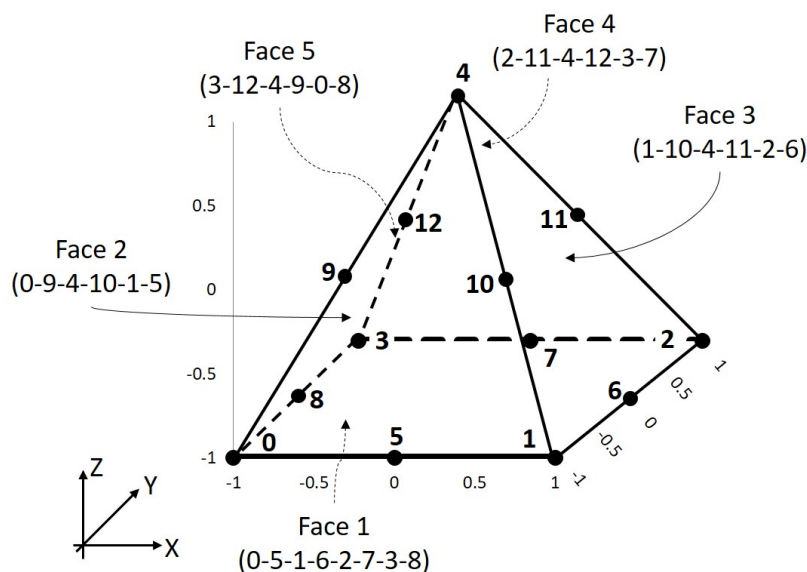


Figure 8.18: PYRAMID_13 Element

8.19 WEDGE_6

WEDGE_6 is a prism element with 6 points. Figure 8.19 shows a WEDGE_6 element.

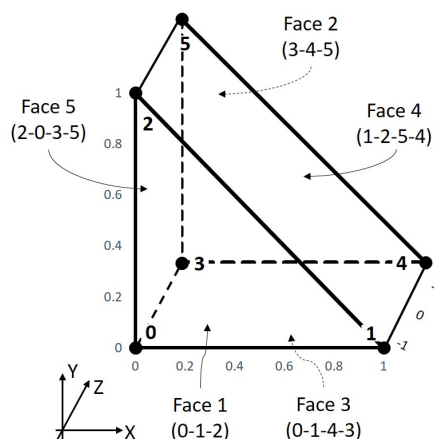


Figure 8.19: WEDGE_6 Element

8.20 WEDGE_15

WEDGE_15 is a prism element with 6 corner points and 9 middle points - one for each face (line). Figure 8.20 shows a WEDGE_15 element.

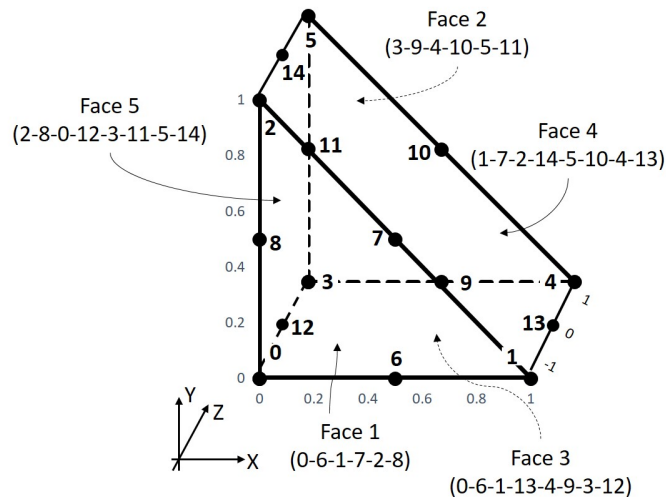


Figure 8.20: WEDGE_15 Element

8.21 HEXAHEDRON_8

HEXAHEDRON_8 is a brick element with 8 points. Figure 8.21 shows a HEXAHEDRON_8 element.

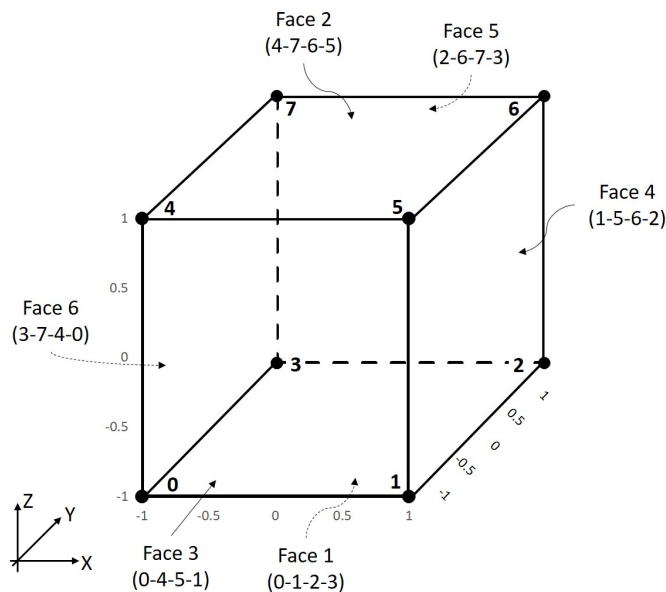


Figure 8.21: HEXAHEDRON_8 Element

8.22 HEXAHEDRON_9

HEXAHDREDON_9 has the same parameterization as HEXAHEDRON_8, with an additional center point. Figure 8.22 shows a HEXAHEDRON_9 element.

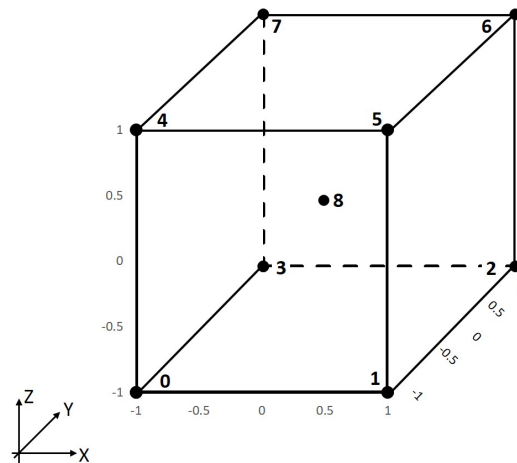


Figure 8.22: HEXAHEDRON_9 Element

8.23 HEXAHEDRON_20

HEXAHDREDON_20 is a brick element with 8 corner points and 12 middle points - one on each face (line) . Figure 8.23 shows a HEXAHEDRON_20 element.

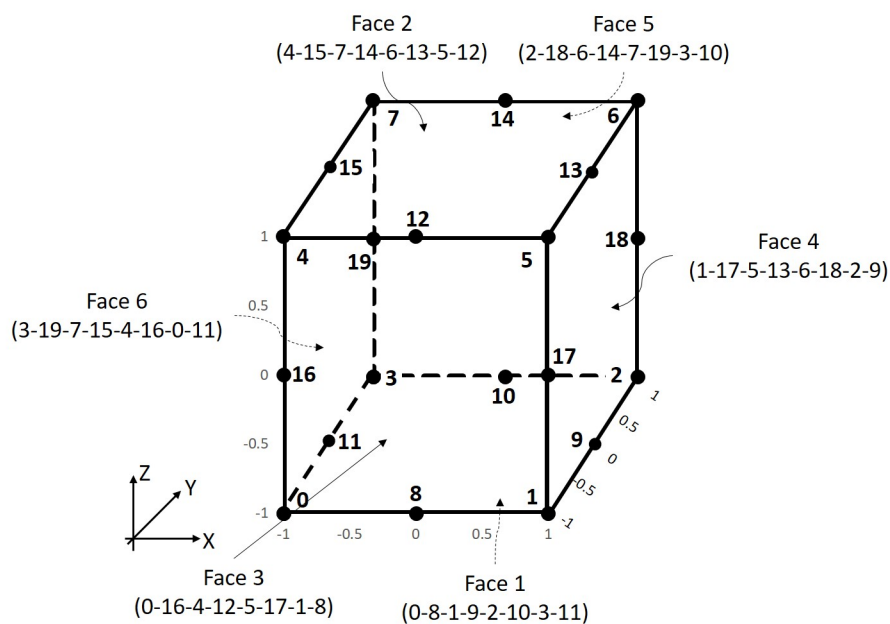


Figure 8.23: HEXAHEDRON_20 Element

8.24 HEXAHEDRON_21

HEXAHEDRON_21 has the same parameterization as HEXAHEDRON_20, with an additional middle point. Figure 8.24 shows a HEXAHEDRON_21 element.

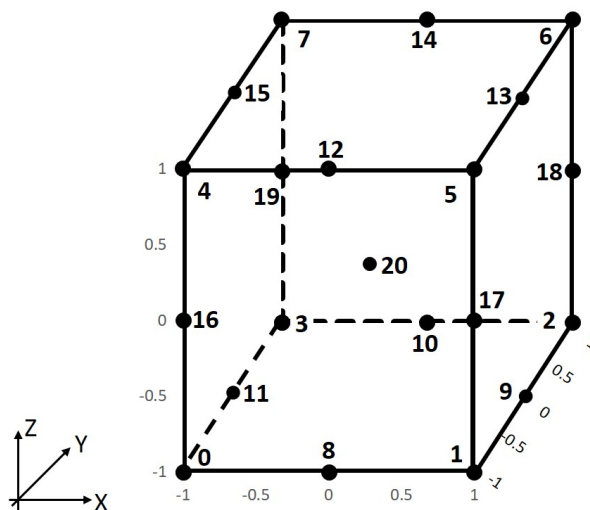


Figure 8.24: HEXAHEDRON_21 Element

8.25 HEXAHEDRON_27

HEXAHEDRON_27 is a brick element with 8 corner points, 12 middle points - one on each face (line), 6 middle points - one on each face and one center point. Figure 8.23 shows a HEXAHEDRON_27 element.

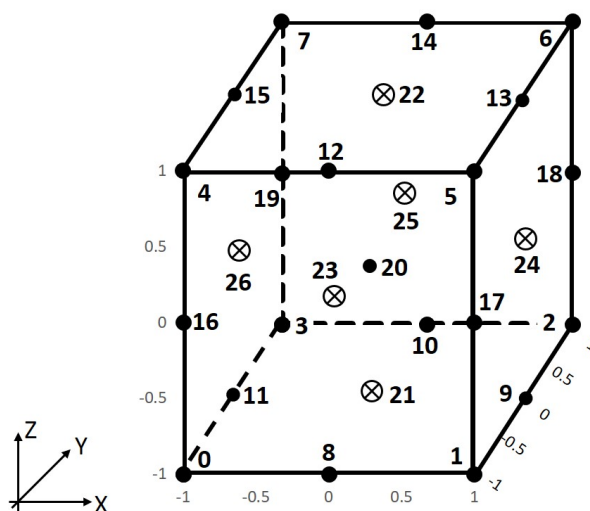


Figure 8.25: HEXAHEDRON_27 Element



8.26 POLYGON

....

8.27 POLYHEDRON

....

Chapter 9

Integration Type Definition Specifications

This chapter contains all the integration types provided by the `VMAPIntegrationTypeFactory.cxx` file, including user defined integration type. Additionally, it covers information about **Combined** integration types and **Composite** Integration Types. All integration types defined in VMAP Integration Types Library, belong to the domain VMAP e.g. `VMAP_GAUSS_4`. All Gauss quadrature rules are over interval $[-1, 1]$. For all 1-Dimensional integration types, the integration point numbering is from the lowest to the highest abscissa value.

9.1 USER_DEFINED

A user-defined integration rule is explained with the help of an example. The numbering for combined user defined type starts from

100000

. See figure 9.1.

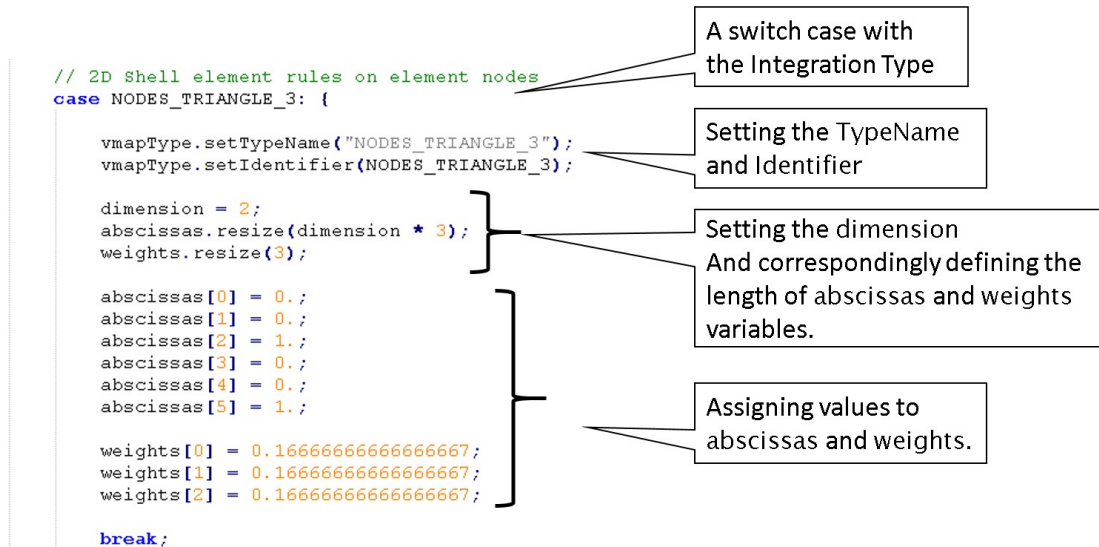


Figure 9.1: C++ Code for NODE_TRIANGLE_3 Integration Rule Definition

9.2 GAUSS_1

Number of Points - n	Weight - w_i	Abscissa - x_i
1	2.000000	0.000000

Table 9.1: GAUSS_1

9.3 GAUSS_2

Number of Points - n	Weight - w_i	Abscissa - x_i
2	1	± 0.5773502691896257

Table 9.2: GAUSS_2

9.4 GAUSS_3

Number of Points - n	Weight - w_i	Abscissa - x_i
3	0.888888	0.000000
	0.555555	± 0.7745966692414834

Table 9.3: GAUSS_3

9.5 GAUSS_4

Number of Points - n	Weight - w_i	Abscissa - x_i
4	0.6521451548625461	± 0.3399810435848563
	0.3478548451374538	± 0.8611363115940526

Table 9.4: GAUSS_4

9.6 GAUSS_5

Number of Points - n	Weight - w_i	Abscissa - x_i
5	0.568889	0.000000
	0.4786286704993665	± 0.5384693101056831
	0.2369268850561891	± 0.9061798459386640

Table 9.5: GAUSS_5

9.7 GAUSS_6

Number of Points - n	Weight - w_i	Abscissa - x_i
6	0.3607615730481386	± 0.6612093864662645
	0.4679139345726910	± 0.2386191860831969
	0.1713244923791704	± 0.9324695142031521

Table 9.6: GAUSS_6

9.8 GAUSS_7

Number of Points - n	Weight - w_i	Abscissa - x_i
7	0.4179591836734694	0.000000
	0.3818300505051189	± 0.4058451513773972
	0.2797053914892766	± 0.7415311855993945
	0.1294849661688697	± 0.9491079123427585

Table 9.7: GAUSS_7

9.9 GAUSS_8

Number of Points - n	Weight - w_i	Abscissa - x_i
8	0.3626837833783620	± 0.1834346424956498
	0.3137066458778873	± 0.5255324099163290
	0.2223810344533745	± 0.7966664774136267
	0.1012285362903763	± 0.9602898564975363

Table 9.8: GAUSS_8

9.10 GAUSS_9

Number of Points - n	Weight - w_i	Abscissa - x_i
9	0.3302393550012598	0.000000
	0.1806481606948574	± 0.8360311073266358
	0.0812743883615744	± 0.9681602395076261
	0.3123470770400029	± 0.3242534234038089
	0.2606106964029354	± 0.6133714327005904

Table 9.9: GAUSS_9

9.11 GAUSS_10

Number of Points - n	Weight - w_i	Abscissa - x_i
10	0.2955242247147529	± 0.1488743389816312
	0.2692667193099963	± 0.4333953941292472
	0.2190863625159820	± 0.6794095682990244
	0.1494513491505806	± 0.8650633666889845
	0.0666713443086881	± 0.9739065285171717

Table 9.10: GAUSS_10

9.12 GAUSS_11

Number of Points - n	Weight - w_i	Abscissa - x_i
11	0.2729250867779006	0.000000
	0.2628045445102467	± 0.2695431559523450
	0.2331937645919905	± 0.5190961292068118
	0.1862902109277343	± 0.7301520055740494
	0.1255803694649046	± 0.8870625997680953
	0.0556685671161737	± 0.9782286581460570

Table 9.11: GAUSS_11

9.13 GAUSS_12

Number of Points - n	Weight - w_i	Abscissa - x_i
12	0.2491470458134028	± 0.1252334085114689
	0.2334925365383548	± 0.3678314989981802
	0.2031674267230659	± 0.5873179542866175
	0.1600783285433462	± 0.7699026741943047
	0.1069393259953184	± 0.9041172563704749
	0.0471753363865118	± 0.9815606342467192

Table 9.12: GAUSS_12

9.14 GAUSS_13

Number of Points - n	Weight - w_i	Abscissa - x_i
13	0.2325515532308739	0.000000
	0.2262831802628972	± 0.2304583159551348
	0.2078160475368885	± 0.4484927510364469
	0.1781459807619457	± 0.6423493394403402
	0.1388735102197872	± 0.8015780907333099
	0.0921214998377285	± 0.9175983992229779
	0.0404840047653159	± 0.9841830547185881

Table 9.13: GAUSS_13

9.15 GAUSS_14

Number of Points - n	Weight - w_i	Abscissa - x_i
14	0.2152638534631578	± 0.1080549487073437
	0.2051984637212956	± 0.3191123689278897
	0.1855383974779378	± 0.5152486363581541
	0.1572031671581935	± 0.6872929048116855
	0.1215185706879032	± 0.8272013150697650
	0.0801580871597602	± 0.9284348836635735
	0.0351194603317519	± 0.9862838086968123

Table 9.14: GAUSS_14

9.16 GAUSS_15

Number of Points - n	Weight - w_i	Abscissa - x_i
15	0.2025782419255613	0.000000
	0.1984314853271116	± 0.2011940939974345
	0.1861610000155622	± 0.3941513470775634
	0.1662692058169939	± 0.5709721726085388
	0.1395706779261543	± 0.7244177313601701
	0.1071592204671719	± 0.8482065834104272
	0.0703660474881081	± 0.9372733924007060
	0.0307532419961173	± 0.9879925180204854

Table 9.15: GAUSS_15

9.17 GAUSS_16

Number of Points - n	Weight - w_i	Abscissa - x_i
16	0.1894506104550685	± 0.0950125098376374
	0.1826034150449236	± 0.2816035507792589
	0.1691565193950025	± 0.4580167776572274
	0.1495959888165767	± 0.6178762444026438
	0.1246289712555339	± 0.7554044083550030
	0.0951585116824928	± 0.8656312023878318
	0.0622535239386479	± 0.9445750230732326
	0.0271524594117541	± 0.9894009349916499

Table 9.16: GAUSS_16

9.18 LOBATTO_1

This integration type is same as GAUSS_1. See table 9.1

9.19 LOBATTO_2

Number of Points - n	Weight - w_i	Abscissa - x_i
2	1.000000	± 1.000000

Table 9.17: LOBATTO_2

9.20 LOBATTO_3

Number of Points - n	Weight - w_i	Abscissa - x_i
3	1.333333	0.000000
	0.333333	± 1.000000

Table 9.18: LOBATTO_3

9.21 LOBATTO_4

Number of Points - n	Weight - w_i	Abscissa - x_i
4	0.833333	± 0.4472135954
	0.166667	± 1.000000

Table 9.19: LOBATTO_4

9.22 LOBATTO_5

Number of Points - n	Weight - w_i	Abscissa - x_i
5	0.711111	0.000000
	0.544444	± 0.6546536707
	0.100000	± 1.000000

Table 9.20: LOBATTO_5

9.23 LOBATTO_6

Number of Points - n	Weight - w_i	Abscissa - x_i
6	0.5548583770	± 0.2852315164
	0.3784749562	± 0.7650553239
	0.066667	± 1.000000

Table 9.21: LOBATTO_6

9.24 LOBATTO_7

Number of Points - n	Weight - w_i	Abscissa - x_i
7	0.4876190476	0.000000
	0.4317453812	± 0.4688487934
	0.2768260473	± 0.8302238962
	0.0476190476	± 1.000000

Table 9.22: LOBATTO_7

9.25 LOBATTO_8

Number of Points - n	Weight - w_i	Abscissa - x_i
8	0.4124587946	± 0.2092992179
	0.3411226924	± 0.5917001814
	0.2107042271	± 0.8717401485
	0.0357142857	± 1.000000

Table 9.23: LOBATTO_8

9.26 LOBATTO_9

Number of Points - n	Weight - w_i	Abscissa - x_i
9	0.3715192743	0.000000
	0.3464285109	± 0.3631174638
	0.2745387125	± 0.6771862795
	0.1654953615	± 0.8997579954
	0.0277777778	± 1.000000

Table 9.24: LOBATTO_9

9.27 LOBATTO_10

Number of Points - n	Weight - w_i	Abscissa - x_i
10	0.3275397611	± 0.1652789576
	0.2920426836	± 0.4779249498
	0.2248893420	± 0.7387738651
	0.1333059908	± 0.9195339081
	0.022222	± 1.000000

Table 9.25: LOBATTO_10

9.28 LOBATTO_11

Number of Points - n	Weight - w_i	Abscissa - x_i
11	0.3002175954	0.000000
	0.2868791247	± 0.2957581355
	0.2480481042	± 0.5652353269
	0.1871698817	± 0.7844834736
	0.1096122732	± 0.9340014304
	0.0181818	± 1.000000

Table 9.26: LOBATTO_11

9.29 LOBATTO_12

Number of Points - n	Weight - w_i	Abscissa - x_i
12	0.271405241	± 0.136552933
	0.251275603	± 0.399530941
	0.212508418	± 0.632876153
	0.157974705	± 0.819279322
	0.09168452	± 0.944899272
	0.015151515	± 1.000000

Table 9.27: LOBATTO_12

9.30 LOBATTO_13

Number of Points - n	Weight - w_i	Abscissa - x_i
13	0.25193085	0.000000
	0.244015790	± 0.24928693
	0.220767793	± 0.48290982
	0.183646865	± 0.68618847
	0.134981926	± 0.84634756
	0.077801687	± 0.95330984
	0.012820512	± 1.000000

Table 9.28: LOBATTO_13

9.31 LOBATTO_14

Number of Points - n	Weight - w_i	Abscissa - x_i
14	0.2316128	± 0.11633187
	0.219126253	± 0.34272401
	0.194826149	± 0.55063940
	0.160021852	± 0.72886859
	0.116586656	± 0.86780105
	0.066837284	± 0.95993505
	0.010989011	± 1.000000

Table 9.29: LOBATTO_14

9.32 LOBATTO_15

Number of Points - n	Weight - w_i	Abscissa - x_i
15	0.2170481	0.000000
	0.211973586	± 0.21535395
	0.196987236	± 0.42063805
	0.172789647	± 0.60625321
	0.140511699	± 0.76351969
	0.101660070	± 0.88508204
	0.058029893	± 0.96524593
	0.009523809	± 1.000000

Table 9.30: LOBATTO_15

9.33 LOBATTO_16

Number of Points - n	Weight - w_i	Abscissa - x_i
16	0.2019583	± 0.10132627
	0.193690024	± 0.29983047
	0.177491913	± 0.48605942
	0.154026981	± 0.65238870
	0.124255382	± 0.79200829
	0.089393697	± 0.89920053
	0.050850361	± 0.96956805
	0.008333333	± 1.000000

Table 9.31: LOBATTO_16

9.34 SIMPSON_1

This integration type is same as GAUSS_1. See table 9.1

9.35 SIMPSON_3

Number of Points - n	Weight - w_i	Abscissa - x_i
3	1.333333	0.000000
	0.333333	± 1.000000

Table 9.32: SIMPSON_3

9.36 SIMPSON_5

Number of Points - n	Weight - w_i	Abscissa - x_i
5	0.333333	0.000000
	0.666667	± 0.500000
	0.166667	± 1.000000

Table 9.33: SIMPSON_5

9.37 SIMPSON_7

Number of Points - n	Weight - w_i	Abscissa - x_i
7	0.444444	0.000000
	0.222222	± 0.333337
	0.444444	± 0.666667
	0.111111	± 1.000000

Table 9.34: SIMPSON_7

9.38 SIMPSON_9

Number of Points - n	Weight - w_i	Abscissa - x_i
9	0.166667	0.00
	0.333333	± 0.25
	0.166667	± 0.50
	0.333333	± 0.75
	0.083333	± 1.00

Table 9.35: SIMPSON_9

9.39 SIMPSON_11

Number of Points - n	Weight - w_i	Abscissa - x_i
11	0.266667	0.00
	0.133333	± 0.20
	0.266667	± 0.40
	0.133333	± 0.60
	0.266667	± 0.80
	0.066667	± 1.00

Table 9.36: SIMPSON_11

9.40 SIMPSON_13

Number of Points - n	Weight - w_i	Abscissa - x_i
13	0.111111	0.00
	0.222222	± 0.166667
	0.111111	± 0.333333
	0.222222	± 0.50
	0.111111	± 0.666667
	0.222222	± 0.833333
	0.055555	± 1.00

Table 9.37: SIMPSON_13

9.41 SIMPSON_15

Number of Points - n	Weight - w_i	Abscissa - x_i
15	0.190476	0.00
	0.095238	± 0.142857
	0.190476	± 0.285714
	0.095238	± 0.428571
	0.190476	± 0.571428
	0.095238	± 0.714285
	0.190476	± 0.857142
	0.047619	± 1.00

Table 9.38: SIMPSON_15

9.42 TRAPEZOIDAL_1

This integration type is same as GAUSS_1. See table 9.1

9.43 TRAPEZOIDAL_2

Number of Points - n	Weight - w_i	Abscissa - x_i
2	1.0	± 1.0

Table 9.39: TRAPEZOIDAL_2

9.44 TRAPEZOIDAL_3

Number of Points - n	Weight - w_i	Abscissa - x_i
3	1.0	0.0
	0.5	± 1.0

Table 9.40: TRAPEZOIDAL_3

9.45 TRAPEZOIDAL_4

Number of Points - n	Weight - w_i	Abscissa - x_i
4	0.666667	± 0.333333
	0.333333	± 1.0

Table 9.41: TRAPEZOIDAL_4

9.46 TRAPEZOIDAL_5

Number of Points - n	Weight - w_i	Abscissa - x_i
5	0.5	0.0
	0.5	± 0.5
	0.25	± 1.0

Table 9.42: TRAPEZOIDAL_5

9.47 TRAPEZOIDAL_6

Number of Points - n	Weight - w_i	Abscissa - x_i
6	0.4	± 0.199999
	0.4	± 0.6
	0.2	± 1.0

Table 9.43: TRAPEZOIDAL_6

9.48 TRAPEZOIDAL_7

Number of Points - n	Weight - w_i	Abscissa - x_i
7	0.333333	0.0
	0.333333	± 0.333333
	0.333333	± 0.666667
	0.166667	± 1.0

Table 9.44: TRAPEZOIDAL_7

9.49 TRAPEZOIDAL_8

Number of Points - n	Weight - w_i	Abscissa - x_i
8	0.285714	± 0.142857
	0.285714	± 0.428571
	0.285714	± 0.714285
	0.142857	± 1.0

Table 9.45: TRAPEZOIDAL_8

9.50 TRAPEZOIDAL_9

Number of Points - n	Weight - w_i	Abscissa - x_i
9	0.25	0.0
	0.25	± 0.25
	0.25	± 0.50
	0.25	± 0.75
	0.125	± 1.0

Table 9.46: TRAPEZOIDAL_9

9.51 TRAPEZOIDAL_10

Number of Points - n	Weight - w_i	Abscissa - x_i
10	0.222222	± 0.111111
	0.222222	± 0.333333
	0.222222	± 0.555556
	0.222222	± 0.777778
	0.111111	± 1.0

Table 9.47: TRAPEZOIDAL_10

9.52 TRAPEZOIDAL_11

Number of Points - n	Weight - w_i	Abscissa - x_i
11	0.25	0.0
	0.2	± 0.20
	0.2	± 0.40
	0.2	± 0.60
	0.2	± 0.80
	0.1	± 1.0

Table 9.48: TRAPEZOIDAL_11

9.53 TRAPEZOIDAL_12

Number of Points - n	Weight - w_i	Abscissa - x_i
12	0.181818	± 0.09090909
	0.181818	± 0.272727273
	0.181818	± 0.454545455
	0.181818	± 0.636363636
	0.181818	± 0.818181818
	0.090909	± 1.0

Table 9.49: TRAPEZOIDAL_12

9.54 TRAPEZOIDAL_13

Number of Points - n	Weight - w_i	Abscissa - x_i
13	0.166667	0.0
	0.166667	± 0.16666667
	0.166667	± 0.33333333
	0.166667	± 0.5
	0.166667	± 0.66666667
	0.166667	± 0.83333333
	0.083333	± 1.0

Table 9.50: TRAPEZOIDAL_13

9.55 TRAPEZOIDAL_14

Number of Points - n	Weight - w_i	Abscissa - x_i
14	0.153846	± 0.076923
	0.153846	± 0.230769
	0.153846	± 0.384615
	0.153846	± 0.538461
	0.153846	± 0.692307
	0.153846	± 0.846153
	0.076923	± 1.0

Table 9.51: TRAPEZOIDAL_14

9.56 TRAPEZOIDAL_15

Number of Points - n	Weight - w_i	Abscissa - x_i
15	0.14285714285714285	0.0
	0.14285714285714285	± 0.142857
	0.14285714285714285	± 0.285714
	0.14285714285714285	± 0.428571
	0.14285714285714285	± 0.571428
	0.14285714285714285	± 0.714285
	0.14285714285714285	± 0.857142
	0.07142857142857142	± 1.0

Table 9.52: TRAPEZOIDAL_15

9.57 GAUSS_TRIANGLE_1

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
1	0.5	0.333333	0.333333

Table 9.53: GAUSS_TRIANGLE_1

9.58 GAUSS_TRIANGLE_3

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
3	0.333333	0.166667	0.166667
	0.333333	0.666667	0.166667
	0.333333	0.166667	0.666667

Table 9.54: GAUSS_TRIANGLE_3

9.59 GAUSS_TRIANGLE_4

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
4	0.520833	0.2	0.2
	0.520833	0.2	0.6
	0.520833	0.6	0.2
	-0.5625	0.333333	0.333333

Table 9.55: GAUSS_TRIANGLE_4

9.60 GAUSS_TRIANGLE_6

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
6	0.109951743655322	0.091576213509771	0.091576213509771
	0.109951743655322	0.816847572980459	0.091576213509771
	0.109951743655322	0.091576213509771	0.816847572980459
	0.223381589678011	0.445948490915965	0.108103018168070
	0.223381589678011	0.445948490915965	0.445948490915965
	0.223381589678011	0.108103018168070	0.445948490915965

Table 9.56: GAUSS_TRIANGLE_6

9.61 GAUSS_QUAD_1

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
1	4.0	0.0	0.0

Table 9.57: GAUSS_QUAD_1

9.62 GAUSS_QUAD_4

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
4	1.0	-0.5773502691896257	-0.5773502691896257
	1.0	0.5773502691896257	-0.5773502691896257
	1.0	-0.5773502691896257	0.5773502691896257
	1.0	0.5773502691896257	0.5773502691896257

Table 9.58: GAUSS_QUAD_4

9.63 GAUSS_QUAD_9

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
9	0.308641976	-0.7745966692414834	-0.7745966692414834
	0.493827161	0.0	-0.7745966692414834
	0.308641976	0.7745966692414834	-0.7745966692414834
	0.493827161	-0.7745966692414834	0.0
	0.790123457	0.0	0.0
	0.493827161	0.7745966692414834	0.0
	0.308641976	-0.7745966692414834	0.7745966692414834
	0.493827161	0.0	0.7745966692414834
	0.308641976	0.7745966692414834	0.7745966692414834

Table 9.59: GAUSS_QUAD_9

9.64 NODES_TRIANGLE_3

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
3	0.166667	0.000000	0.000000
	0.166667	1.000000	0.000000
	0.166667	0.000000	1.000000

Table 9.60: NODES_TRIANGLE_3

9.65 NODES_TRIANGLE_6

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
6	0.000000	0.000000	0.000000
	0.000000	1.000000	0.000000
	0.000000	0.000000	1.000000
	0.166667	0.500000	0.000000
	0.166667	0.500000	0.500000
	0.166667	0.000000	0.500000

Table 9.61: NODES_TRIANGLE_6

9.66 NODES_QUAD_4

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
4	1.000000	-1.000000	-1.000000
	1.000000	1.000000	-1.000000
	1.000000	1.000000	1.000000
	1.000000	-1.000000	1.000000

Table 9.62: NODES_QUAD_4

9.67 NODES_QUAD_8

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
8	1.000000	-1.000000	-1.000000
	1.000000	1.000000	-1.000000
	1.000000	1.000000	1.000000
	1.000000	-1.000000	1.000000
	1.000000	0.000000	-1.000000
	1.000000	1.000000	0.000000
	1.000000	0.000000	1.000000
	1.000000	-1.000000	0.000000

Table 9.63: NODES_QUAD_8

9.68 NODES_QUAD_9

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i
9	0.111111	-1.000000	-1.000000
	0.111111	1.000000	-1.000000
	0.111111	1.000000	1.000000
	0.111111	-1.000000	1.000000
	0.444444	0.000000	-1.000000
	0.444444	1.000000	0.000000
	0.444444	0.000000	1.000000
	0.444444	-1.000000	0.000000
	1.777778	0.000000	0.000000

Table 9.64: NODES_QUAD_9

9.69 GAUSS_LAYERED_HEXAHEDRON_4

to be defined.....

9.70 GAUSS_TETRAHEDRON_1

Please refer to Figure 8.12 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
1	0.166667	0.250000	0.250000	0.250000

Table 9.65: GAUSS_TETRAHEDRON_1

9.71 GAUSS_TETRAHEDRON_4

Please refer to Figure 8.12 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
4	0.0416667	0.1381966	0.1381966	0.1381966
	0.0416667	0.5854102	0.1381966	0.1381966
	0.0416667	0.1381966	0.5854102	0.1381966
	0.0416667	0.1381966	0.1381966	0.5854102

Table 9.66: GAUSS_TETRAHEDRON_4

9.72 GAUSS_TETRAHEDRON_8

Please refer to Figure 8.12 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
8	0.020833	0.131445	0.166666	0.211325
	0.020833	0.490562	0.166666	0.211325
	0.020833	0.035221	0.622008	0.211325
	0.020833	0.131446	0.622008	0.211325
	0.020833	0.035221	0.044658	0.788675
	0.020833	0.131446	0.044658	0.788675
	0.020833	0.009437	0.166667	0.788675
	0.020833	0.035221	0.166667	0.788675

Table 9.67: GAUSS_TETRAHEDRON_8

9.73 GAUSS_TETRAHEDRON_11

Please refer to Figure 8.12 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
11	-0.013156	0.250000	0.250000	0.250000
	0.007622	0.071429	0.071429	0.071429
	0.007622	0.785714	0.071429	0.071429
	0.007622	0.071429	0.785714	0.071429
	0.007622	0.071429	0.071429	0.785714
	0.024889	0.100596	0.399404	0.100596
	0.024889	0.399404	0.399404	0.100596
	0.024889	0.399404	0.100596	0.100596
	0.024889	0.100596	0.100596	0.399404
	0.024889	0.399404	0.100596	0.399404
	0.024889	0.100596	0.399404	0.399404

Table 9.68: GAUSS_TETRAHEDRON_11

9.74 GAUSS_TETRAHEDRON_15

Please refer to Figure 8.12 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
15	0.019753	0.250000	0.250000	0.250000
	0.011989	0.091971	0.091971	0.091971
	0.011989	0.724087	0.091971	0.091971
	0.011989	0.091971	0.724087	0.091971
	0.011989	0.091971	0.091971	0.724087
	0.011511	0.319794	0.319794	0.319794
	0.011511	0.040619	0.319794	0.319794
	0.011511	0.319794	0.040619	0.319794
	0.011511	0.319794	0.319794	0.040619
	0.008818	0.056351	0.056351	0.443649
	0.008818	0.443649	0.056351	0.056351
	0.008818	0.443649	0.443649	0.056351
	0.008818	0.056351	0.443649	0.443649
	0.008818	0.056351	0.443649	0.056351
	0.008818	0.443649	0.056351	0.443649

Table 9.69: GAUSS_TETRAHEDRON_15

9.75 GAUSS_PYRAMID_1

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
1	1.33	0.0	0.0	0.25

Table 9.70: GAUSS_PYRAMID_1

9.76 GAUSS_PYRAMID_5

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
5	0.28	∓ 0.487950036474	-0.487950036474	0.165484574527
	0.28	± 0.487950036474	0.487950036474	0.165484574527
	0.213333	0.0	0.0	0.693705983732

Table 9.71: GAUSS_PYRAMID_5

9.77 GAUSS_PYRAMID_8

to be defined.....

9.78 GAUSS_WEDGE_1

Please refer to Figure 8.19 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
1	1.000000	0.333333	0.333333	0.000000

Table 9.72: GAUSS_WEDGE_1

9.79 GAUSS_WEDGE_2

Please refer to Figure 8.19 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
2	0.500000	0.333333	0.333333	∓ 0.577350

Table 9.73: GAUSS_WEDGE_2

9.80 GAUSS_WEDGE_6

Please refer to Figure 8.19 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
6	0.166667	0.166667	0.166667	-0.577334
	0.166667	0.666667	0.166667	-0.577334
	0.166667	0.166667	0.666667	-0.577334
	0.166667	0.166667	0.166667	0.577334
	0.166667	0.666667	0.166667	0.577334
	0.166667	0.166667	0.666667	0.577334

Table 9.74: GAUSS_WEDGE_6

9.81 GAUSS_WEDGE_8

Please refer to Figure 8.19 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
8	0.125	0.166667	0.211325	-0.57735
	0.125	0.622008	0.211325	-0.57735
	0.125	0.044658	0.788675	-0.57735
	0.125	0.166667	0.788675	-0.57735
	0.125	0.166667	0.211325	0.57735
	0.125	0.622008	0.211325	0.57735
	0.125	0.044658	0.788675	0.57735
	0.125	0.166667	0.788675	0.57735

Table 9.75: GAUSS_WEDGE_8

9.82 GAUSS_WEDGE_9

Please refer to Figure 8.19 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
9	0.092593	0.166667	0.166667	-0.774597
	0.092593	0.666667	0.166667	-0.774597
	0.092593	0.166667	0.666667	-0.774597
	0.148148	0.166667	0.166667	0.0
	0.148148	0.666667	0.166667	0.0
	0.148148	0.166667	0.666667	0.0
	0.092593	0.166667	0.166667	0.774597
	0.092593	0.666667	0.166667	0.774597
	0.092593	0.166667	0.666667	0.774597

Table 9.76: GAUSS_WEDGE_9

9.83 GAUSS_WEDGE_18

Please refer to figure 8.19 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
18	0.030542	0.091576	0.091576	-0.774597
	0.030542	0.816848	0.091576	-0.774597
	0.030542	0.091576	0.816848	-0.774597
	0.062050	0.445948	0.108103	-0.774597
	0.062050	0.445948	0.445948	-0.774597
	0.062050	0.108103	0.445948	-0.774597
	0.048867	0.091576	0.091576	0.0
	0.048867	0.816848	0.091576	0.0
	0.048867	0.091576	0.816848	0.0
	0.099281	0.445948	0.108103	0.0
	0.099281	0.445948	0.445948	0.0
	0.099281	0.108103	0.445948	0.0
	0.030542	0.091576	0.091576	0.774597
	0.030542	0.816848	0.091576	0.774597
	0.030542	0.091576	0.816848	0.774597
	0.062050	0.445948	0.108103	0.774597
	0.062050	0.445948	0.445948	0.774597
	0.062050	0.108103	0.445948	0.774597

Table 9.77: GAUSS_WEDGE_18

9.84 GAUSS_HEXAHEDRON_1

Please refer to figure 8.21 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
1	8.0	0.0	0.0	0.0

Table 9.78: GAUSS_HEXAHEDRON_1

9.85 GAUSS_HEXAHEDRON_8

Please refer to figure 8.21 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
8	1.0	∓ 0.57735026918962	-0.57735026918962	0.57735026918962
	1.0	∓ 0.57735026918962	0.57735026918962	0.57735026918962
	1.0	∓ 0.57735026918962	-0.57735026918962	-0.57735026918962
	1.0	∓ 0.57735026918962	0.57735026918962	-0.57735026918962

Table 9.79: GAUSS_HEXAHEDRON_8

9.86 GAUSS_HEXAHEDRON_27

Please refer to figure 8.21 for node locations.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
27	0.17146776	-0.77459666924148	-0.77459666924148	0.77459666924148
	0.27434842	0.0	-0.77459666924148	0.77459666924148
	0.17146776	0.77459666924148	-0.77459666924148	0.77459666924148
	0.27434842	-0.77459666924148	0.0	0.77459666924148
	0.43895748	0.0	0.0	0.77459666924148
	0.27434842	0.77459666924148	0.0	0.77459666924148
	0.17146776	-0.77459666924148	0.77459666924148	0.77459666924148
	0.27434842	0.0	0.77459666924148	0.77459666924148
	0.17146776	0.77459666924148	0.77459666924148	0.77459666924148
	0.27434842	-0.77459666924148	-0.77459666924148	0.0
	0.43895748	0.0	-0.77459666924148	0.0
	0.27434842	0.77459666924148	-0.77459666924148	0.0
	0.43895748	-0.77459666924148	0.0	0.0
	0.70233196	0.0	0.0	0.0
	0.43895748	0.77459666924148	0.0	0.0
	0.27434842	-0.77459666924148	0.77459666924148	0.0
	0.43895748	0.0	0.77459666924148	0.0
	0.27434842	0.77459666924148	0.77459666924148	0.0
	0.17146776	-0.77459666924148	-0.77459666924148	-0.77459666924148
	0.27434842	0.0	-0.77459666924148	-0.77459666924148
	0.17146776	0.77459666924148	-0.77459666924148	-0.77459666924148
	0.27434842	-0.77459666924148	0.0	-0.77459666924148
	0.43895748	0.0	0.0	-0.77459666924148
	0.27434842	0.77459666924148	0.0	-0.77459666924148
	0.17146776	-0.77459666924148	0.77459666924148	-0.77459666924148
	0.27434842	0.0	0.77459666924148	-0.77459666924148
	0.17146776	0.77459666924148	0.77459666924148	-0.77459666924148

Table 9.80: GAUSS_HEXAHEDRON_8

9.87 NODES_TETRAHEDRON_4

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
4	0.041666	0.0	0.0	0.0
	0.041666	1.0	0.0	0.0
	0.041666	0.0	1.0	0.0
	0.041666	0.0	0.0	1.0

Table 9.81: NODES_TETRAHEDRON_4

9.88 NODES_TETRAHEDRON_10

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
10	-0.008333	0.0	0.0	0.0
	-0.008333	1.0	0.0	0.0
	-0.008333	0.0	1.0	0.0
	-0.008333	0.0	0.0	1.0
	0.033333	0.5	0.0	0.0
	0.033333	0.5	0.5	0.0
	0.033333	0.0	0.5	0.0
	0.033333	0.0	0.0	0.5
	0.033333	0.5	0.0	0.5
	0.033333	0.0	0.5	0.5

Table 9.82: NODES_TETRAHEDRON_10

9.89 NODES_WEDGE_6

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
6	0.166667	0.0	0.0	-1.0
	0.166667	1.0	0.0	-1.0
	0.166667	0.0	1.0	-1.0
	0.166667	0.0	0.0	1.0
	0.166667	1.0	0.0	1.0
	0.166667	0.0	1.0	1.0

Table 9.83: NODES_WEDGE_6

9.90 NODES_WEDGE_15

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
15	-0.111111	0.0	0.0	-1.0
	-0.111111	1.0	0.0	-1.0
	-0.111111	0.0	1.0	-1.0
	-0.111111	0.0	0.0	1.0
	-0.111111	1.0	0.0	1.0
	-0.111111	0.0	1.0	1.0
	0.166667	0.5	0.0	-1.0
	0.166667	0.5	0.5	-1.0
	0.166667	0.0	0.5	-1.0
	0.166667	0.5	0.0	1.0
	0.166667	0.5	0.5	1.0
	0.166667	0.0	0.5	1.0
	0.222222	0.0	0.0	0.0
	0.222222	1.0	0.0	0.0
	0.222222	0.0	1.0	0.0

Table 9.84: NODES_WEDGE_15

9.91 NODES_PYRAMID_5

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
5	0.250000	∓ 1.0	-1.0	0.0
	0.250000	± 1.0	1.0	0.0
	0.333333	0.0	0.0	1.0

Table 9.85: NODES_PYRAMID_5

9.92 NODES_HEXAHEDRON_8

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
8	1.0	∓ 1.0	-1.0	-1.0
	1.0	± 1.0	1.0	-1.0
	1.0	∓ 1.0	-1.0	1.0
	1.0	± 1.0	1.0	1.0

Table 9.86: NODES_HEXAHEDRON_8

9.93 NODES_HEXAHEDRON_20

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
20	-1.0	∓ 1.0	-1.0	-1.0
	-1.0	± 1.0	1.0	-1.0
	-1.0	∓ 1.0	-1.0	1.0
	-1.0	± 1.0	1.0	1.0
	1.333333	0.0	-1.0	-1.0
	1.333333	1.0	0.0	-1.0
	1.333333	0.0	1.0	-1.0
	1.333333	-1.0	0.0	-1.0
	1.333333	0.0	-1.0	1.0
	1.333333	1.0	0.0	1.0
	1.333333	0.0	1.0	1.0
	1.333333	-1.0	0.0	1.0
	1.333333	∓ 1.0	-1.0	0.0
	1.333333	± 1.0	1.0	0.0

Table 9.87: NODES_HEXAHEDRON_20

9.94 NODES_HEXAHEDRON_27

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
27	0.037037	∓ 1.0	-1.0	-1.0
	0.037037	± 1.0	1.0	-1.0
	0.037037	∓ 1.0	-1.0	1.0
	0.037037	± 1.0	1.0	1.0
	0.148148	0.0	-1.0	-1.0
	0.148148	1.0	0.0	-1.0
	0.148148	0.0	1.0	-1.0
	0.148148	-1.0	0.0	-1.0
	0.148148	0.0	-1.0	1.0
	0.148148	1.0	0.0	1.0
	0.148148	0.0	1.0	1.0
	0.148148	-1.0	0.0	1.0
	0.148148	∓ 1.0	-1.0	0.0
	0.148148	± 1.0	1.0	0.0
	2.370370	0.0	0.0	0.0
	0.592592	0.0	0.0	∓ 1.0
	0.592592	0.0	-1.0	0.0
	0.592592	1.0	0.0	0.0
	0.592592	0.0	1.0	0.0
	0.592592	-1.0	0.0	0.0

Table 9.88: NODES_HEXAHEDRON_27

9.95 Combined Integration Types

A Combined Integration Type is used for 3D shell elements, where 2D Integration Type is used for In-Plane & 1D Integration Type is used for Out-of-Plane. The definition is as follows:

$$\text{IN} - \text{PLANE} \times \text{OUT} - \text{OF} - \text{PLANE}$$

The numbering for combined type starts from 100000.

9.96 Composite Integration Types

A Composite Integration type can be built using the function `createVMAPCompositeIntegrationType`. It requires the following parameters:

- Number of Composite Layers
- list of integration rule per layer
- list of thicknesses per layer

Chapter 10

Tutorials

This chapter provides tutorials for VMAP I/O Lib in C++.

10.1 Creating or Opening a VMAP .h5 File

VMAPFile(const std::string & path, int mode = CREATEORREPLACE)

Function that creates or opens a file based on the mode supplied. The three available modes are:

- **CREATEORREPLACE = 0**, creates new file or overwrite existing. This is the default mode.
- **OPENREADWRITE = 1**, opens existing file with read/write access.
- **OPENREADONLY = 2**, opens existing file with read only access.

10.1.1 Parameters:

- **path** - Local path to file location.
- **mode** - eFileOpenMode to access the file.

10.1.2 Returns:

On success,

with mode **CREATEORREPLACE**:

A VMAP .h5 file, with four groups, VMAP Group, and three sub-groups within VMAP - GEOMETRY, SYSTEM & RESULT. In addition, the VERSION attribute is created to the VMAP Group and initialized with the current version of VMAP Standard I/O Library being used.

with mode **OPENREADWRITE**:

Opens the given file in read and write mode. In addition, provides

details of the VMAP version of the file and throws an error if the major, minor or patch version is older than the VMAP Standard I/O Library version.

with mode `OPENREADONLY`:

Opens the given file in read only mode. In addition, provides details of the VMAP version of the file and throws an error if the major, minor or patch version is older than the VMAP Standard I/O Library version.

On failure, doesn't open any file and provides information about possible errors.

The snippet below shows how to use the `VMAPFile` module to open or create VMAP .h5 file.

10.2 Create a Group in VMAP File

`VMAPFile::createGroup(const std::string & groupName)`

Function that creates a group in the VMAP file.

10.2.1 Parameters:

- `groupName` - Name of the HDF Group to create.

10.2.2 Returns:

On success,

- creates the group with the given group name.

On failure,

10.3 Get Sub-Groups from VMAP File

`std::vector<std::string> VMAPFile::getSubGroups(const std::string & groupName)`

.....

10.3.1 Parameters:

- `groupName` - Name of the HDF Group to create.

10.3.2 Returns:

On success,

- creates the group with the given group name.

On failure,

10.4 Check if a Group exists in VMAP File

`std::vector<std::string> VMAPFile::getSubGroups(const std::string & group-Name)`

.....

10.4.1 Parameters:

- groupName - Name of the HDF Group to create.

10.4.2 Returns:

On success,

creates the group with the given group name.

On failure,

10.5 Write the Version to VMAP File

`VMAPFile::writeVersion(const sVersion & version)`

Function that creates VERSION attribute in group '/VMAP/' and sets the value of VERSION parameters. This function is run automatically, when a new file is created. This function is called in the VMAPFile::openFile, which is called by the constructor VMAPFile (10.1).

10.5.1 Parameters:

- version - Data structure containing the sVersion.

10.5.2 Returns:

On success,

sets the version of VMAP Standard I/O Library to the VERSION attribute in '/VMAP/' group.

On failure,

prints error.

10.6 Read the Version of VMAP File

VMAPFile::readVersion(sVersion & version)

Function that reads VERSION attribute from group '/VMAP/' and stores content in sVersion. This function is run automatically, when an existing file is opened. This function is called in the VMAPFile::openFile, which is called by the constructor VMAPFile (10.1).

10.6.1 Parameters:

- **version** - Data structure read from VMAP file.

10.6.2 Returns:

On success,

returns the version stored in VERSION attribute from '/VMAP/' group.

On failure,

prints error.

more tutorials to be defined.....

Chapter 11

Simple Test Cases

11.1 Break Forming of a Metal Bracket

This test case shows interoperability among the following using VMAP Standard Specifications.

MSC Marc \longleftrightarrow Abaqus \longleftrightarrow LS-DYNA

11.1.1 General Description of the test case “Break Forming of a Metal Bracket”

A flat sheet is formed into an angled bracket by punching it through a hole in a table using a cylindrical punch. After the punch has reached its maximum stroke, the elastic spring-back is computed. The material is elastic-plastic with work hardening.

The analysis consists of two stages. The first stage, in which the punch drives the sheet down into the hole, is a large strain quasi-static contact analysis. The second stage, in which the spring-back is computed, is a large strain quasi-static analysis. In both stages, the sheet is subject to gravity. plane strain conditions are assumed. Furthermore, plane strain conditions are assumed and the additive decomposition of the incremental strain tensor into an elastic and plastic part is used in the plasticity calculation.

Model description

The model is depicted below in figure 11.1 . It consists of three bodies: a sheet, a punch and a table with a hole. The dimensions of the sheet are:

length : 120

thickness : 12

The material properties are given by:

Young’s modulus

Poisson’s ratio

Yield stress

Mass density

The radius of the punch is 0.1" and the size of the hole in the table is 0.6". Both the punch and the table are assumed much stiffer than the sheet, such that they can be modelled as rigid (contact) bodies.

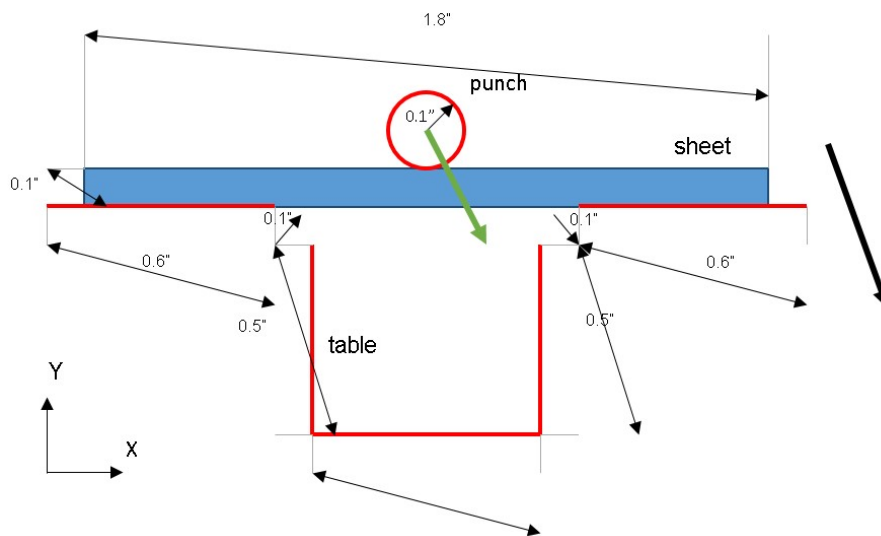


Figure 11.1: VMAP Group

Model Set-up for Step-1: Application of Force

Step-1 is carried out using MSC Marc. The sheet is modeled by 120 x 12 fully integrated bilinear plane strain elements. In the first stage of the analysis, the punch drives the sheet down into the hole (i.e. in the negative Y-direction), to a total stroke of 0.3" and at a constant pace (0.6"/s). Frictionless contact conditions are defined between the sheet and the punch and between the sheet and the table. The sheet is subject to gravitational forces acting in the negative Y-direction. The acceleration due to gravity is given by *to be defined*....

The displacements of the nodes in the center of the sheet, right above the center of the hole are suppressed in the horizontal direction (X-direction):

Constraints

to be defined....

Model Set-up for Step-2: Break forming

Step-2 is carried out using Abaqus. In the second stage, the contact conditions and the boundary condition on the center nodes are removed and are replaced by two boundary conditions that suppress the rigid body modes: at two nodes at the center of the sheet, the displacements in X-direction are suppressed and at one of these nodes, the displacement in Y-direction is suppressed as well. In addition, the sheet

is subject to gravitation acting in the negative Y-direction. Other than that, no other loads or boundary conditions are applied to the sheet in this stage.

Constraints

to be defined.....

Variables Transferred from Step-1 to Step-2

The following data must be transferred at a minimum from the first stage to the second stage in order to compute the spring-back in the latter:

- Nodal coordinates;
- Nodal displacements;
- Element connectivity;
- The four non-zero components of the stress tensor ($\sigma_{xx}, \sigma_{yy}, \sigma_{xy}, \sigma_{yz}$) at the integration points of the elements;
- The total equivalent plastic strain (ϵ_p) at the integration points of the elements;
- The out-of-plane thickness (t) at the integration points of the elements.

The following quantities are not required for a successful calculation of the spring-back, but may optionally be transferred for post-processing purposes in the second stage:

- The total strain tensor;
- The elastic strain tensor;
- The plastic strain tensor;
- The total strain energy;
- The elastic strain energy;
- The plastic strain energy.

11.1.2 Software Implementation

Methodology

The two stages are solved in two separate runs. At the end of the first stage, the data listed in Sec. 1.2.4 is saved into a VMAP file, which is then imported into the second stage to define the initial state for the spring-back analysis. For any given solver, the VMAP implementation can be validated by comparing the results of the multi-stage analysis with those of a run in which the two stages are solved in a single analysis.

Software Tool

Step-1 - Marc and Mentat 2018 are used (Marc is the Finite Element solver, Mentat is the associated pre and post processor).

Step-2 - *to be defined.....*

11.1.3 Results

Simulation Results

to be defined.....

Related analytical results / Reliable experimental results

There are no analytical or experimental results available for this test case.

Evaluation and Validation

MSC Marc

With Marc, the multi-stage analysis can also be solved using Model Sections. A Model Section is a self-contained model, that contains all relevant Finite Element data (i.e. depending on the analysis run, the associated nodal and elemental results are available). A Model Section written at the end of a Marc analysis can be used as the starting point of a subsequent Marc analysis. For this test case, a Model Section file written out at the end of the first stage contains all data listed in Sec. 11.1.1 and some additional quantities. The solution obtained via Model Sections in Marc can be used as a reference solution for the VMAP implementation in Marc Comparing results with MSC Marc as a reference solver.

Simulation Steps	Solver	Result Evaluation
Single Step	MSC Marc	-
Using MSC Marc for Step-1, changing solvers for Step-2.		
Step-2	MSC Marc	
Step-2	Abaqus	
Step-2	LS-DYNA	

Abaqus

In Abaqus the analysis is performed in one step or in two steps. The two-step case can be run in one solver run or in two solver runs. For the latter, the ‘*RESTART’ keyword is used in Abaqus. The files that are generated and read by the ‘*RESTART’ functionality cannot be created by end-users. Therefore, for the VMAP implementation some other way should be used. This could either be using user subroutines

(SIGINI, HARDINI, SDVINI) or using initial conditions (keyword *INITIAL CONDITIONS, TYPE=STRESS, etc). This is not implemented for the current case. Comparing results with Abaqus as a reference solver

Simulation Steps	Solver	Result Evaluation
Single Step	Abaqus	-
Using Abaqus for Step-1, changing solvers for Step-2.		
Step-2	Abaqus	
Step-2	MSC Marc	
Step-2	LS-DYNA	

LS-DYNA

to be defined.....

Comparing results with LS-DYNA as a reference solver

Simulation Steps	Solver	Result Evaluation
Single Step	LS-DYNA	-
Using LS-DYNA for Step-1, changing solvers for Step-2.		
Step-2	LS-DYNA	
Step-2	Abaqus	
Step-2	MSC Marc	

VMAP Result File

to be defined.....

11.1.4 Extension of the test case “Break Forming of a Metal Bracket”

MSC Marc

An obvious extension of this test case would be a full 3-D model. A second extension of this test case would be to use the multiplicative decomposition of the deformation gradient into elastic and a plastic part, instead of the additive decomposition of the incremental strain tensor, in the plasticity calculation.

Abaqus

to be defined.....

LS-DYNA

to be defined.....



more simple test cases to be defined.....

Bibliography

- [1] ISO 10303-209 "Multidisciplinary Analysis and Design"
<http://www.ap209.org/main-concepts>
- [2] LOTAR – Longer Term Archiving And Retrieval
<http://www.lotar-international.org/lotar-workgroups.html>
- [3] EMMC - The European Materials Modelling Council
<https://emmc.info/>
- [4] Thielen, M. ; Hartwig, K. ; Gust, P. *Blasformen von Kunststoffhohlkörpern*
München : Hanser, 2006.
- [5] SWIG - Simplified Wrapper and Interface Generator
<http://www.swig.org/>

Appendix A

Project Funding

The project is organised via the ITEA programme and funded by national regional agencies and companies over the period from October 2017 to September 2020. The total budget is about 16M€ for the 30 project partners from Austria, Belgium, Canada, Germany (including NAFEMS), Netherlands and Switzerland.

ITEA is the EUREKA Cluster programme supporting innovative, industry-driven, pre-competitive R& D projects in the area of Software-intensive Systems & Services (SiSS). ITEA stimulates projects in an open community of large industry, SMEs, universities, research institutes and user organisations.

As ITEA is a EUREKA Cluster, the community is founded in Europe based on the EUREKA principles and is open to participants worldwide.

The **Austrian part** of the joint project is funded by the Austrian Research Promotion Agency (FFG) (number: Projekt 864080 – EUREKA ITEA 3 2017 VMAP Moulding).

The **Belgian part** of the joint project is funded by the companies partaking.

The **Canadian part** of the joint project is funded by the Scientific Research and Development Tax Credit Program (SR& ED)

The **German part** of the joint project is funded by the German Federal Ministry of Education and Research (BMBF) with 3.5 million euros via the ITEA 3 cluster of the European research initiative EUREKA. (number: DLR-Projektträger, Softwaresysteme und Wissenstechnologien – Funding Sign 01|S17025 A – K).



The **Netherlands part** of the joint project is funded by the Netherlands Enterprise Agency

The **Swiss part** of the joint project is funded by the companies partaking.

Project Key Data

ACRONYM and full-length title

16010	VMAP
Program Call	ITEA 3 Call 3
Full-length Title	A new interface Standard for Integrated Virtual Material Modelling in Manufacturing Industry
Roadmap Challenge	Smart Industry

Project duration & size

Size	Effort:119.62 PY Costs: 14.9M€
Time frame	Start: 2019-09-01 End: 2020-09-30 (37 months)

Coordinator

Germany	Fraunhofer SCAI
Type	Research Institute
Contact Person	Mr. Klaus Wolf
Email Address	klaus.wolf@scai.fraunhofer.de

Consortium

Austria	4a engineering GmbH, Wittmann Battenfeld GmbH
Belgium	MSC Software Belgium S.A.
Canada	Convergent Manufacturing Technologies Inc.
Germany	Audi AG, Dr. Reinold Hagen Stiftung, DYNAmore GmbH, EDAG Engineering GmbH, ESI Software Germany GmbH, Fraunhofer SCAI, Hagen Engineering GmbH, inuTech GmbH, Karlsruhe Institute of Technology (KIT), Kautex Maschi- nenbau GmbH, NAFEMS Deutschland, Österreich, Schweiz GmbH, RIKUTEC Richter Kunststofftechnik GmbH & Co. KG, Robert Bosch GmbH, Simcon kunststofftechnische Soft- ware GmbH
Netherlands	Delft University of Technology, DevControl B.V., In Summa Innovation b.v., KE-works, Material innovation institute M2i, MSC Software Benelux, Philips, Reden BV, University of Groningen
Switzerland	BETA CAE Systems International AG, Sintratec