VMAP

A new Interface Standard for Integrated Virtual Material Modelling in Manufacturing Industry

STANDARD SPECIFICATIONS

Version no.: 1.2.0
Edited by: Fraunhofer SCAI - Priyanka Gulati

List of Authors

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

License of this document

If you would like to join the community or know more about the project, send an email to info@vmap-standard.org

To download the VMAP Standard IO Library, please agree and sign the license terms here https://vmap-standard.org/Specifications/Downloads/

How to Use This Booklet

The aim of this book is to aid understanding of the VMAP Standard Specifications. If you are new to 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, software 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:

- 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:

- 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 types.
- Chapter 10 shows the standardized VMAP variables. These will also be available in the API in the next versions.
- Chapter 11 provides some basic tutorials on how to use the VMAP Standard API.
- Chapter 12 defines simple test cases which could be used by a developer or an end user.
- Chapter 13 shows few of the additional features offered by VMAP.
- Appendix B shows the VMAP standard sensor data specifications.

Target Audience

To be able to use the VMAP Documentation efficiently, prior knowledge of modelling and simulation is required. The user should have hands-on experience of at least one CAE Tool, or at the very least, basic knowledge of Finite Element Analysis. Users and Developers may have different needs so in the table below we have categorised the documentation accordingly.

<table>
<tr><th colspan="2">VMAP Documentation Chapters of Interest</th></tr>
<tr><td colspan="2">CAE Tool End Users to understand the VMAP Standard background, format and testing.</td></tr>
<tr><td>VMAP General Information</td><td>VMAP Standards Document</td></tr>
<tr><td>1
2
3
4
5</td><td>12
10</td></tr>
<tr><td colspan="2">CAE Tool Developers to understand and implement VMAP Standard API within their own software tool.</td></tr>
<tr><td>VMAP General Information</td><td>VMAP Standards Document</td></tr>
<tr><td>1
4</td><td>2
4
7
8
9
11
12
13</td></tr>
<tr><td colspan="2">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.</td></tr>
<tr><td>VMAP General Information</td><td>VMAP Standards Document</td></tr>
<tr><td>1
4</td><td>6
7
8
9
11
12
13</td></tr>
</table>

Abbreviations

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

Contents

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.

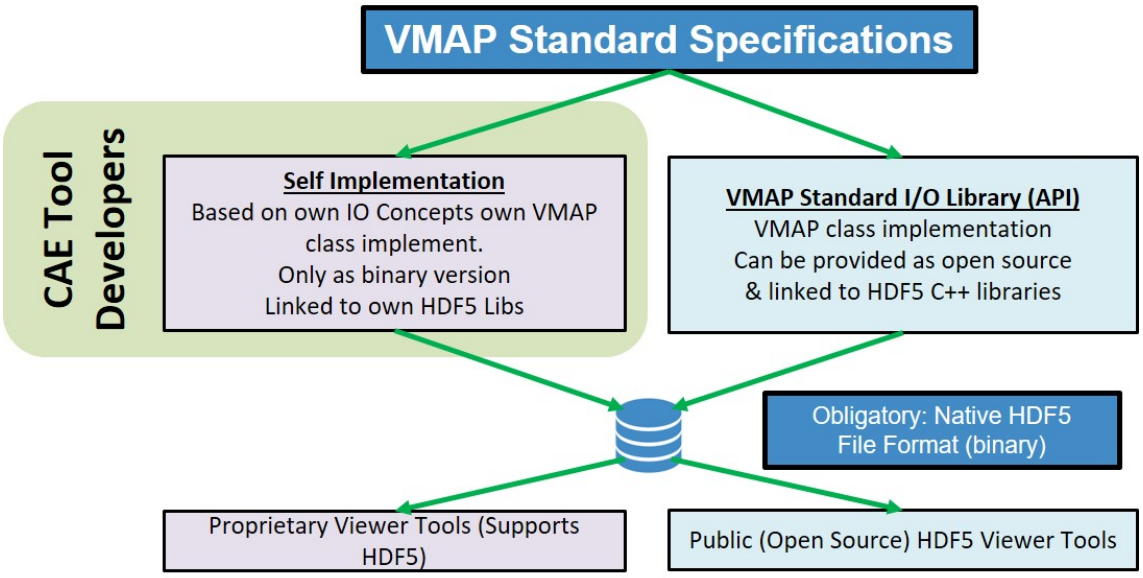


Figure 1.1: VMAP Software Architecture

VMAP Standard Specifications are at the core of the software architecture. VMAP offers two options for every user. The 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 the user's 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 the user's 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.

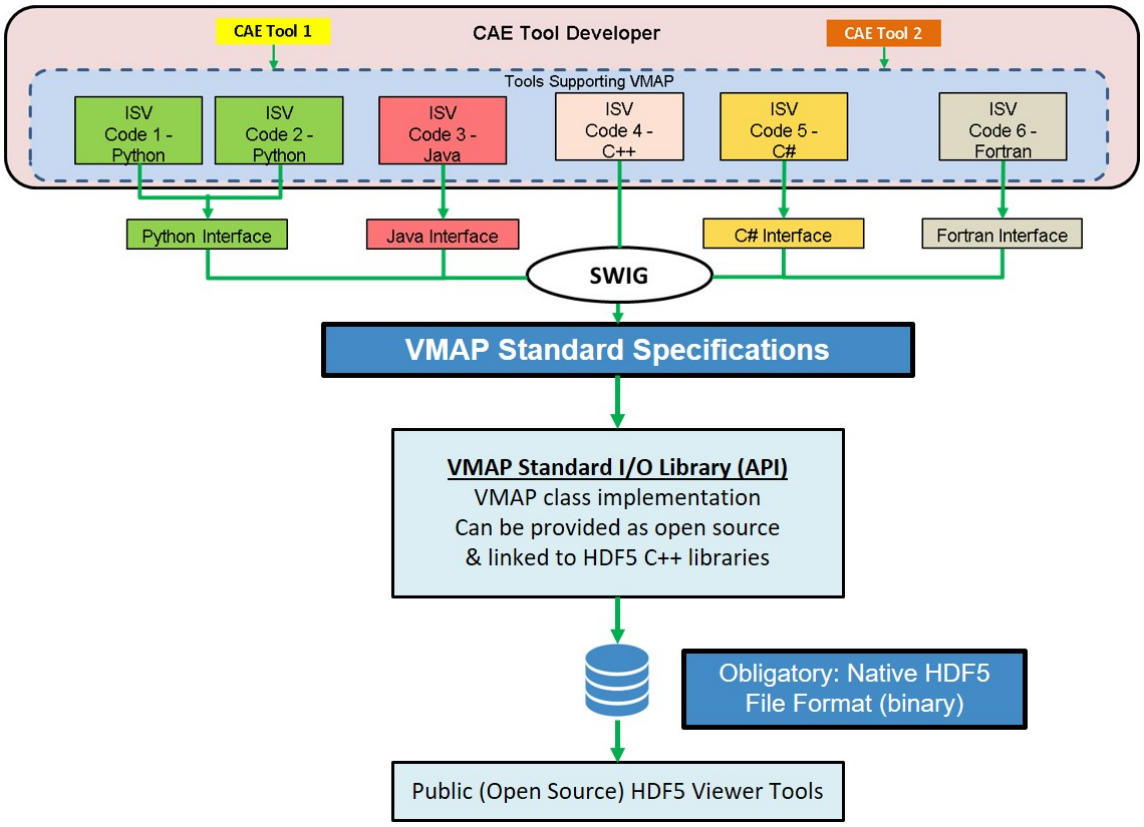


Figure 1.2: Extended VMAP Software Architecture

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 required to carry out another simulation. Since, there are multiple CAE tools (Codes) available on the market, each time a combination of tools is used a new specific converter needs to be developed. This is

where VMAP Standard comes in; with all CAE tools providing VMAP Standard format as one of the output options, the specific converters become unnecessary. VMAP Standard will facilitate reusability and thus will be time saving. Since VMAP Standard is currently in development phase, the converter is replaced by an external VMAP converter. As the standard is completely formalized, 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.

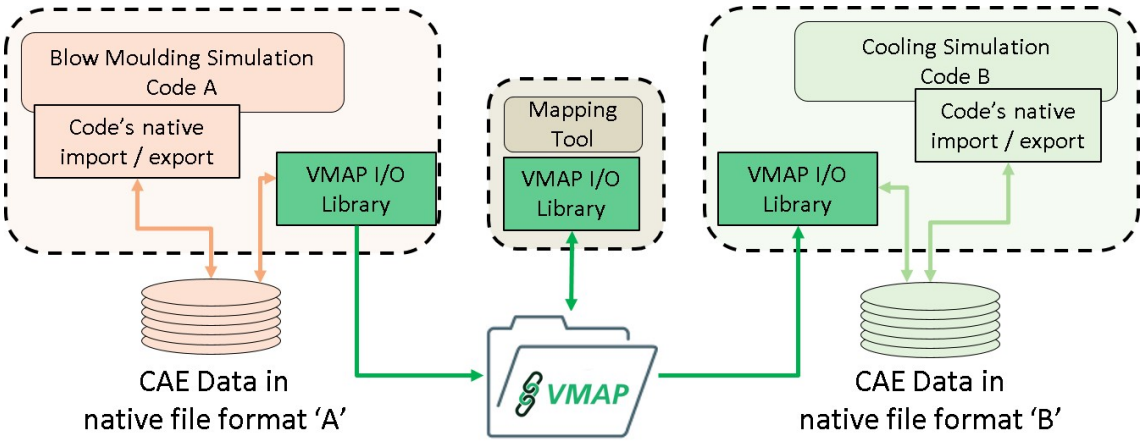


Figure 1.3: VMAP Standard API in CAE chain simulation process

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, an abstract storage model (the data format) and libraries to implement the abstract model and 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.

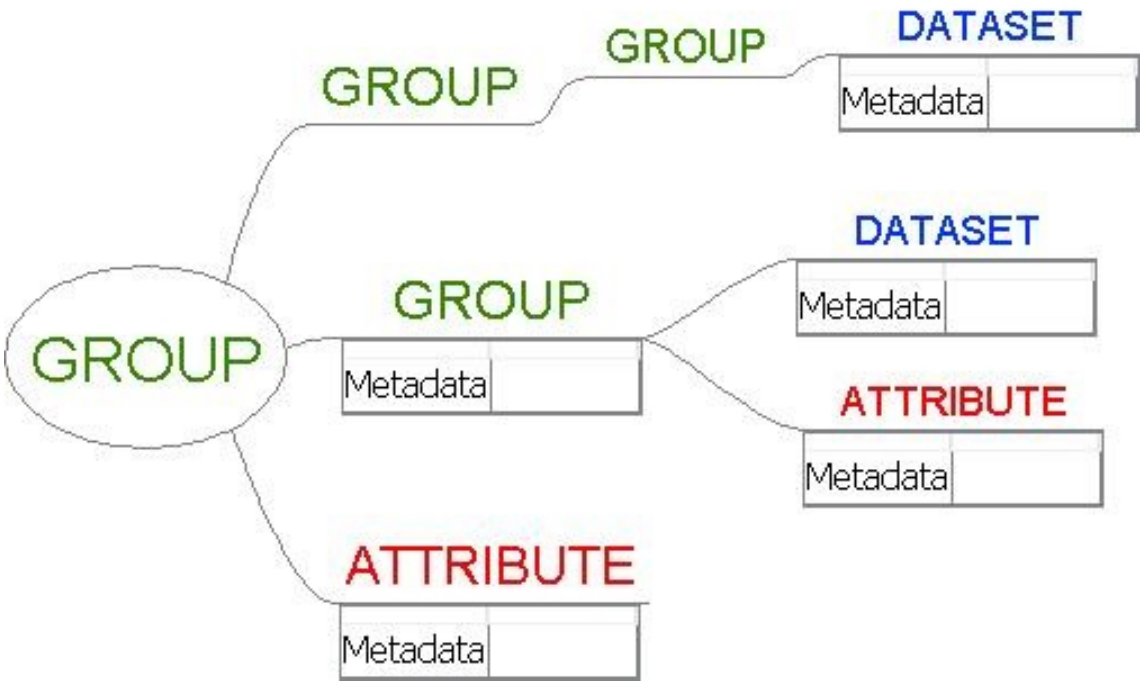


Figure 1.4: HDF5 file format

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 the 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 `VMAPFile` class which is defined in **VMAPFile.h**. Figure 2.1 shows a sample initiation code for VMAP API.

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

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

  Initialize();
  VMAPFile 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 VMAPFile.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.

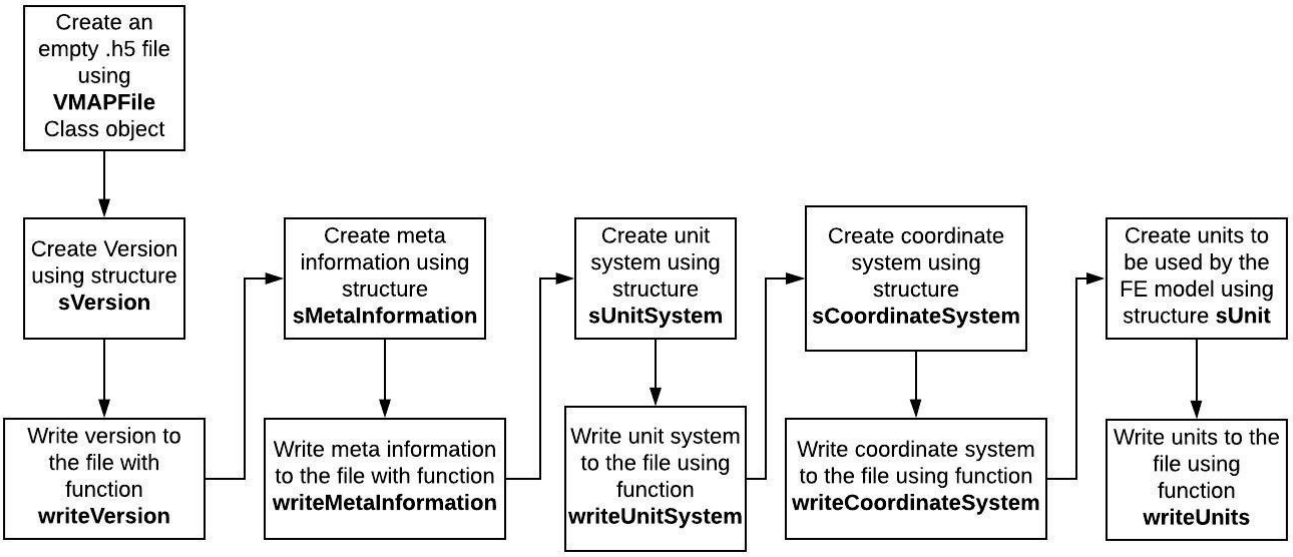


Figure 2.2: Write VMAP .h5 File

2.2.2 Writing an FE Mesh

The following flowchart (Figure 2.3) can be followed to create an 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 for writing your own element type can be found in Chapter 8. Additionally, VMAP offers the option to create an Integration Type or use one of the Integration Type defined in **VMAPIntegrationTypeFactory.cxx** (Chapter 9)

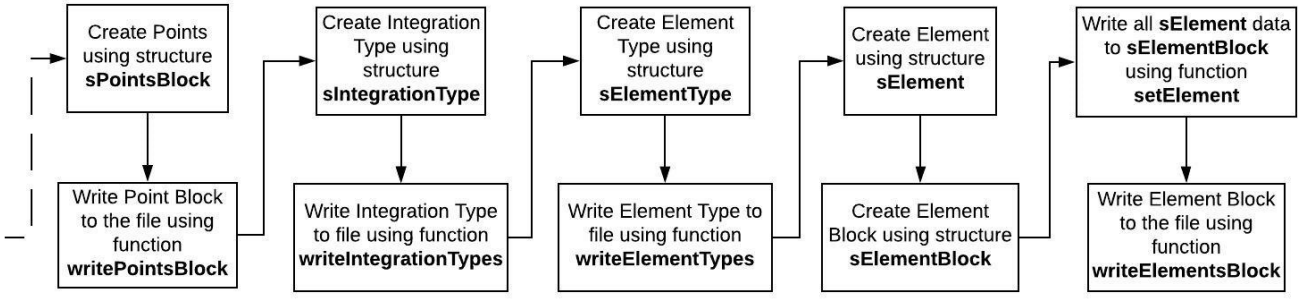


Figure 2.3: Write Mesh to file

In order to define elements and further build element blocks, it is necessary to define the points block, integration types and element types.

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.

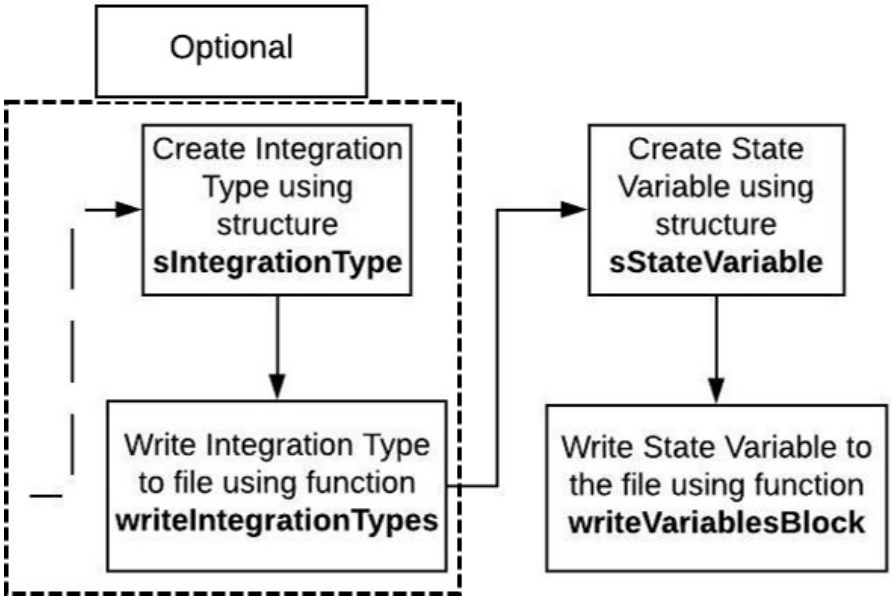


Figure 2.4: Write State Variable

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`.

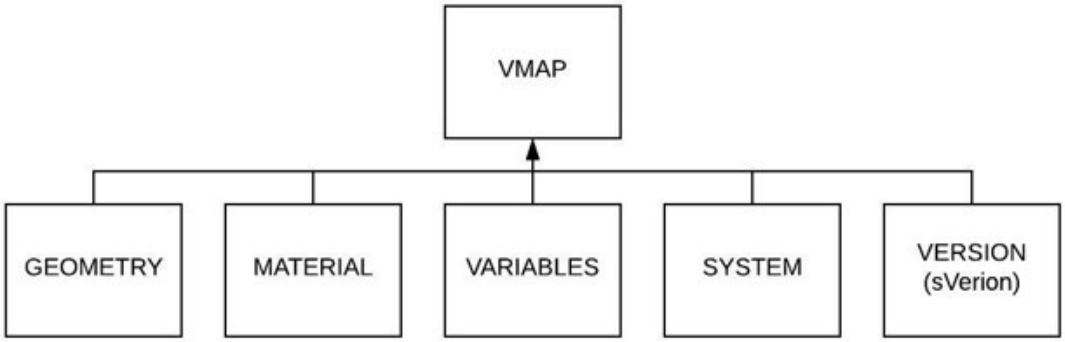


Figure 3.1: VMAP Group Dependency

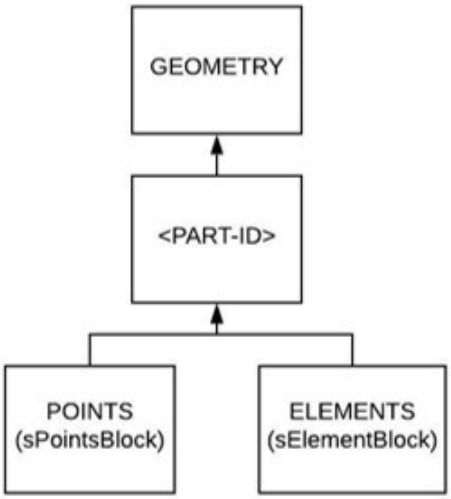


Figure 3.2: GEOMETRY Group Dependency

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`.

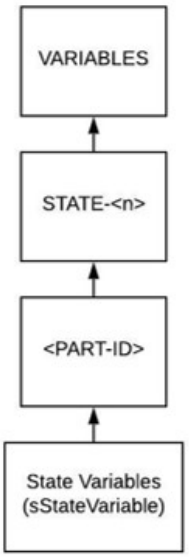


Figure 3.3: VARIABLES Group Dependency

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.

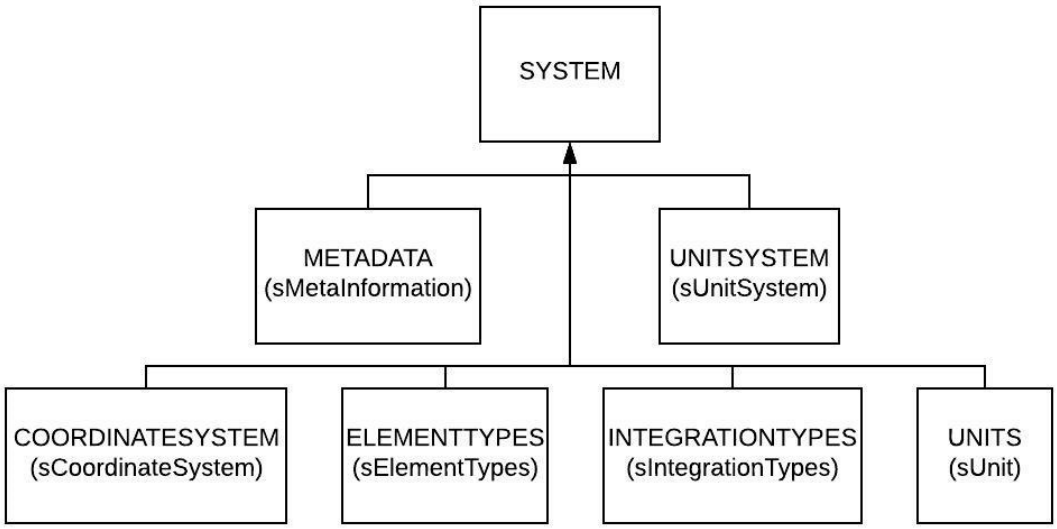


Figure 3.4: SYSTEM Group Dependency

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.

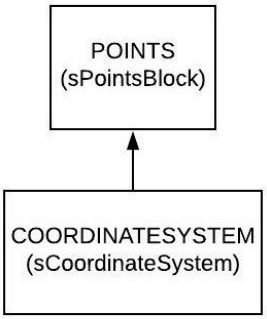


Figure 3.5: POINTS Group Dependency

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

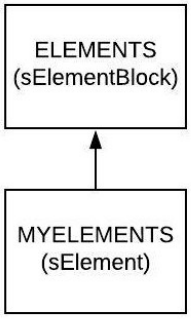


Figure 3.6: ELEMENTS Group Dependency

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.

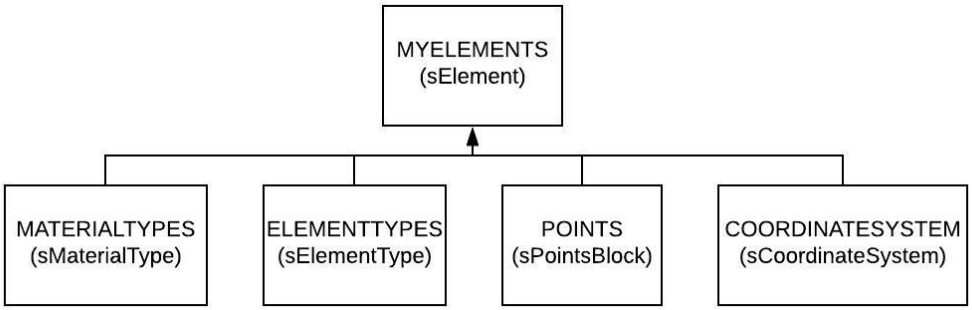


Figure 3.7: MYELEMENTS Data Set Dependency

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`.

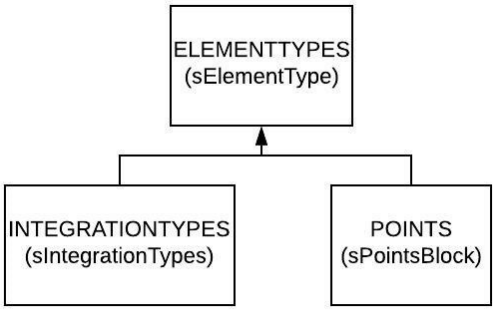


Figure 3.8: ELEMENTTYPES Data Set Dependency

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 global, a point, an integration point, an element or an element face. Figure 3.9 shows a schematic of a state variable group.

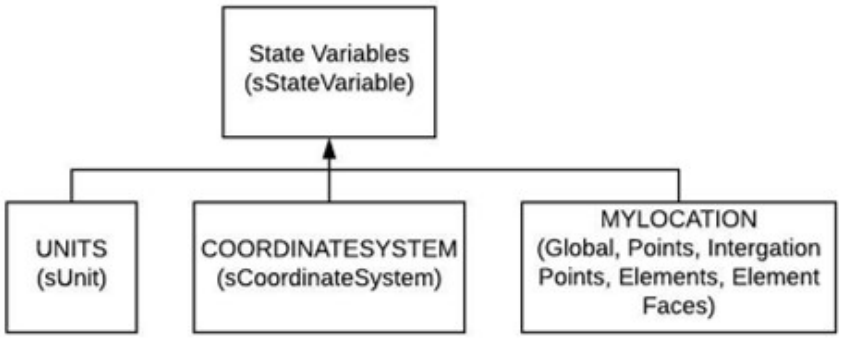


Figure 3.9: State Variable Dependency

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`.

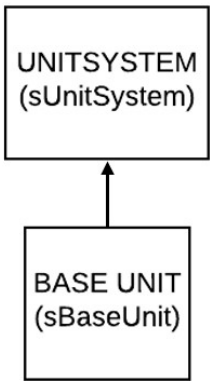


Figure 3.10: UNITSYSTEM Dependency

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 sSection
- struct sElementType
- struct sElement
- struct sElementBlock
- struct sGeometrySet
- struct sStateVariable
- struct sTable
- struct sParameter
- struct sMaterialCard

- struct sMaterial
- class VMAPElementTypeFactory
- class Exception
- class ErrOutOfRange
- class ErrSpaceMismatch
- class ErrSizeMismatch
- class ErrTypeMismatch
- class ErrUnsupported
- class ErrNotImplemented
- class ErrInvalid
- class ErrHDF5
- class VMAPFile
- class VMAPIntegrationTypeFactory

Functions

- template<class tException >
  void VMAP_DECLSPEC Assert (const bool condition)
- template<class tException >
  void VMAP_DECLSPEC Assert (const bool condition, const std::string &s)
- 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 H5SectionType (sizeof(sSection))
- H5::CompType H5ElementTypeType (sizeof(sElementType))
- H5::CompType H5ElementsType (sizeof(sElement))
- void Initialize ()
- herr_t collect_subgroup_names (hid_t loc_id, const char ∗name, const H5L_info_t ∗linfo, void ∗opdata)
- static void print_dataset (hid_t did)
- static void print_datatype (hid_t tid)
- static void print_link (hid_t gid, char ∗name)

- static void print_attribute (hid_t aid)
- static void print_propertylist (hid_t pid)
- static void scan_group (hid_t gid)
- static void scan_attrs (hid_t oid)
- std::string _getStringFromGroupAttribute (const H5::Group &group, const std-::string &attributeName)

Variables

- static const int VMAP_VERSION_MAJOR = 0
- static const int VMAP_VERSION_MINOR = 5
- static const int VMAP_VERSION_PATCH = 1
- static H5::VarLenType H5VariableIntType
- static H5::VarLenType H5VariableDoubleType
- static const char * GROUP_SYSTEM = "SYSTEM"
- static const char * GROUP_GEOMETRY = "GEOMETRY"
- static const char * GROUP_VARIABLE = "VARIABLES"
- static const char * GROUP_MATERIAL = "MATERIAL"
- 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 = "COORDINATESYS-TEM"
- static const char * DATASET_INTEGRATIONTYPES = "INTEGRATIONTYP-ES"
- static const char * DATASET_SECTION = "SECTION"
- 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"
```

Namespaces

- VMAP

4.5 src/VMAP.h File Reference

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

Classes

- struct VMAP::sVersion

  Basic data structure to define VMAP version information.

- struct VMAP::sMetaInformation

  sMetaInformation structure is required to collect metadata about the tool generating the file.

- struct VMAP::sCoordinateSystem

  sCoordinateSystem structure is used to define an arbitrary VMAP Coordinate System.

- struct VMAP::sBaseUnit

  sBaseUnit provides attributes to define a single unit.

- struct VMAP::sUnitSystem

  sUnitSystem is a data structure to define VMAP Unit System.

- struct VMAP::sUnit

  sUnit structure is used to define the derived VMAP units.

- struct VMAP::sPointsBlock

  sPointsBlock structure stores the three dimensional coordinates.

- struct VMAP::sIntegrationType

  sIntegrationType structure defines VMAP element integration type.

- struct VMAP::sSection

  Data structure to define a VMAP section information.
- struct VMAP::sElementType

  sElementType structure defines VMAP element type.
- struct VMAP::sElement

  sElement structure defines VMAP elements.
- struct VMAP::sElementBlock

  sElementBlock structure defines VMAP element block.
- struct VMAP::sGeometrySet

  Data structure to define a VMAP set based on geometric entities.
- struct VMAP::sStateVariable

  sStateVariable structure defines VMAP state variable.
- struct VMAP::sTable

  Structure to define two-dimensional tables in VMAP.
- struct VMAP::sParameter

  Structure to define parameters used in material card.
- struct VMAP::sMaterialCard

  Structure to define a material card.
- struct VMAP::sMaterial

  Structure to define materials in VMAP.

Namespaces

- VMAP

Variables

- static const int VMAP::VMAP_VERSION_MAJOR = 0

  Major version of VMAP IO library.
- static const int VMAP::VMAP_VERSION_MINOR = 5

  Minor version of VMAP IO library.
- static const int VMAP::VMAP_VERSION_PATCH = 1

  Patch version of VMAP IO library.

4.6 src/VMAPDeclspec.h File Reference

Macros

- #define VMAP_DECLSPEC

4.6.1 Macro Definition Documentation

#define VMAP_DECLSPEC

Definition at line 22 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

  Base class for all exceptions thrown by the VMAP.
- class VMAP::ErrOutOfRange

  Exception: Some index is out of expected range.
- class VMAP::ErrSpaceMismatch

  Exception: The space dimension does not fit.
- class VMAP::ErrSizeMismatch

  Exception: The size of an array does not fit.
- class VMAP::ErrTypeMismatch

  Exception: The type does not fit.
- class VMAP::ErrUnsupported

  Exception: Something is not supported.
- class VMAP::ErrNotImplemented

  Exception: Something is not implemented (yet)
- class VMAP::ErrInvalid

  Exception: Something is wrong with the object.
- class VMAP::ErrHDF5

  Exception: Something is wrong with HDF5.

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 "VMAPException.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::H5SectionType (sizeof(sSection))
- 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_SYSTEM = "SYSTEM"
- static const char ∗ VMAP::GROUP_GEOMETRY = "GEOMETRY"
- static const char ∗ VMAP::GROUP_VARIABLE = "VARIABLES"
- static const char ∗ VMAP::GROUP_MATERIAL = "MATERIAL"
- 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_SECTION = "SECTION"
- 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"
#include "H5Cpp.h"
```

Namespaces

- VMAP

Macros

- #define MAX_NAME 2048

  Maximal handled name in print utility.

Functions

- static void VMAP::print_dataset (hid_t did)

  : Print information about a dataset.
- static void VMAP::print_datatype (hid_t tid)

  : Print data type description
- static void VMAP::print_link (hid_t gid, char ∗name)

  : Print a symbolic link.
- static void VMAP::print_attribute (hid_t aid)

  : Print an attribute.
- static void VMAP::print_propertylist (hid_t pid)

: Print information of dataset property list.

- static void VMAP::scan_group (hid_t gid)

  : Process a group and all it's members

- static void VMAP::scan_attrs (hid_t oid)

  : Run through all the attributes of a dataset or group.

- std::string VMAP::_getStringFromGroupAttribute (const H5::Group &group, const std::string &attributeName)

4.13.1 Macro Definition Documentation

#define MAX_NAME 2048

Maximal handled name in print utility.

Definition at line 20 of file VMAPH5Tools.h.

Referenced by VMAP::print_attribute(), VMAP::print_dataset(), VMAP::print_link(), VMAP::print_propertylist(), 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.

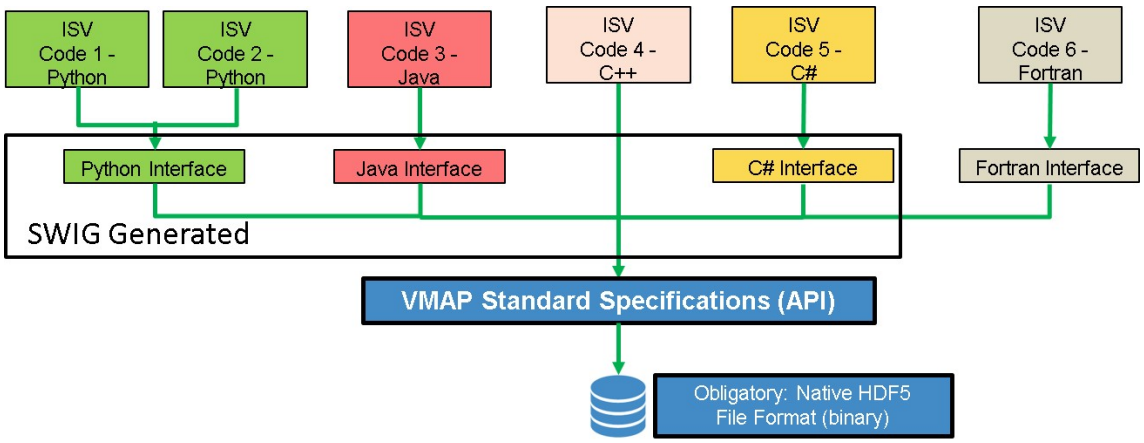


Figure 5.1: SWIG Interface for Linking VMAP Standard API

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
        - Enable HDF5_ENABLE_Z_LIB_SUPPORT in the cmake-gui to enable compression in HDF5.
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. Unicode characters in filename and file path are not allowed in HDF5 version lower than 1.12

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)

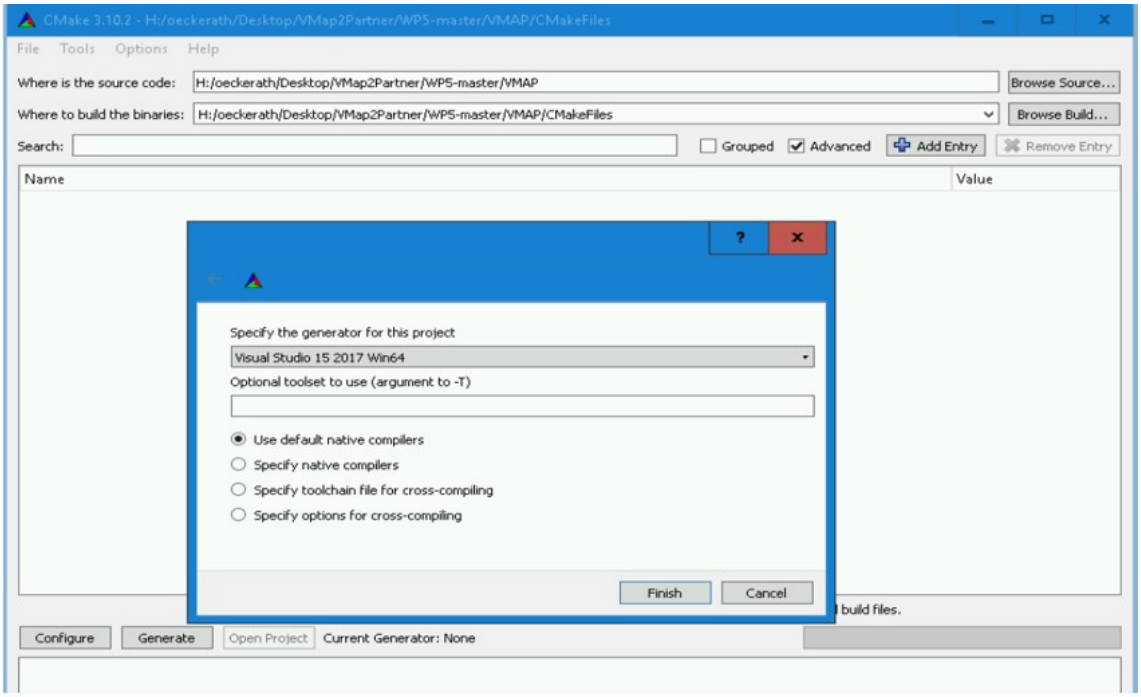


Figure 5.2: Selecting Generator

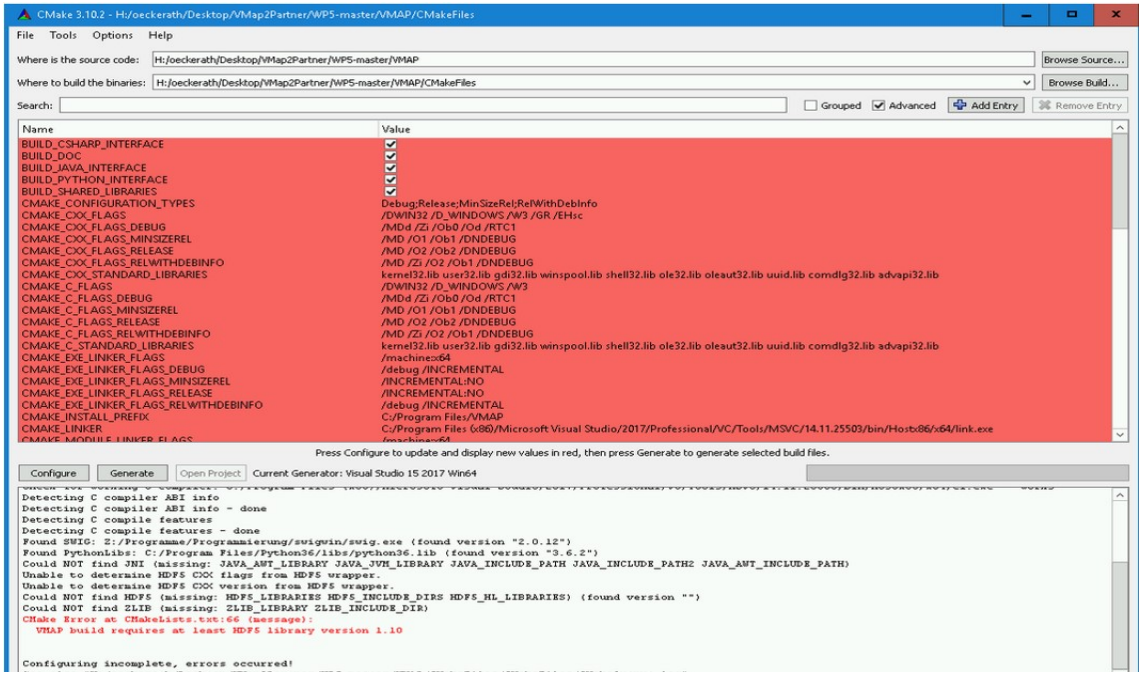


Figure 5.3: Creating Project File

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 for writing your own VMAP I/O Library which generates a standard VMAP .h5 output file. The aim 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 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 datasets respectively. Additionally, the letters 'g', 'a' and 'd' are used as sub-scripts denoting group, attribute and dataset respectively; this facilitates a black-and-white print option without losing the differentiation.

6.1 `VMAP` Group

`VMAP` Group has one attribute and four major groups, which clearly earmark the domains in an FE model. These four groups further consist of groups and datasets along with relevant metadata to comprehensively define an FE model. The hierarchical model (from left to right) in Figure 6.1 shows a schematic of the basic `VMAP` storage structure. 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

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

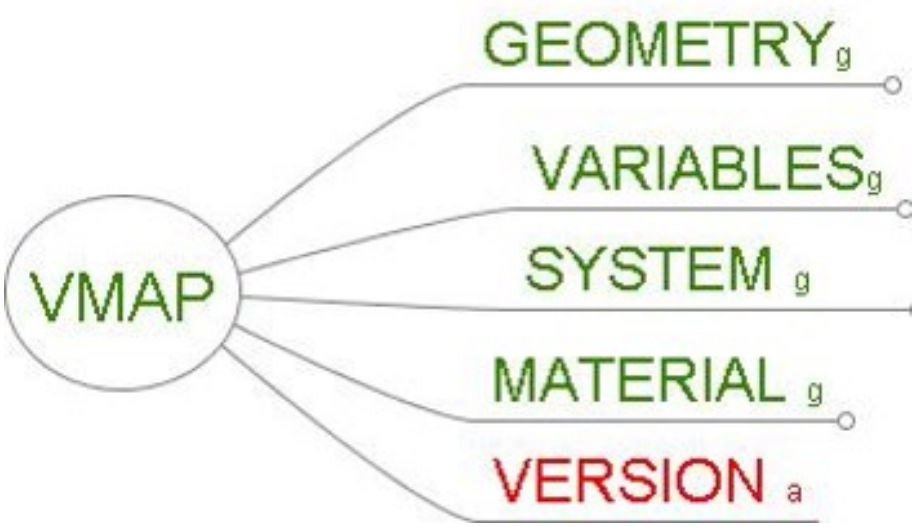


Figure 6.1: VMAP Group

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, which includes points and elements. This group has no attribute information. The groups <Part IDs> or <PROPERTY IDs> of an FE Model are associated with GEOMETRY. <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. Table 6.4 shows only General Object Info of GEOMETRY Group, these tables show information as seen in the HDF5 Viewer.

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

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

Table 6.4: General Object Info

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

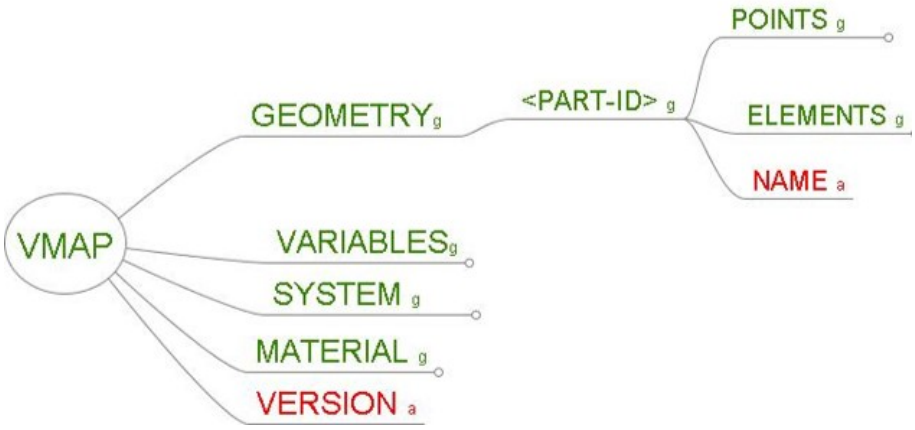


Figure 6.2: GEOMETRY Group

6.3 <PART-ID> Group

<PART-ID> Group stores the points and elements of a particular part/property-id. The attribute MYNAME and the groups POINTS & ELEMENTS are associated with <PART-ID>. Tables 6.5 & 6.6 show Object Attribute Info and General Object Info of <PART-ID> Group, these tables show information as seen in the HDF5 Viewer.. <PART-ID>.

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

Table 6.5: Object Attribute Info

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

Number of members:	2
Name	Type
POINTS	Group
ELEMENTS	Group

Table 6.6: General Object Info

Note: Within the VMAP Standard Library, there is a create function for generating 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 distinct part numbers and names for every part. Hence, the part number and name are taken directly from the solver file. Additionally, this offers the possibility of duplicate point(s) and/or element(s) numbering among various parts.

6.3.1 MYNAME Attribute

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

- Value: Defines the name of the part.

Table 6.7 shows the data types associated with MYNAME metadata

Metadata	Data Type
Value	String

Table 6.7: Data Types of MYNAME 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. The two attributes MYCOORDINATESYSTEM & MYSIZE and two datasets MYCOORDINATES & MYIDENTIFIERS are associated with POINTS, shown in Figure 6.4. Tables 6.8 & 6.9 show Object Attribute Info and General Object Info of POINTS Group, these tables show information as seen in the HDF5 Viewer.

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

Table 6.8: Object Attribute Info

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

Number of members:	2
Name	Type
MYCOORDINATES	Dataset
MYIDENTIFIERS	Dataset

Table 6.9: General Object Info

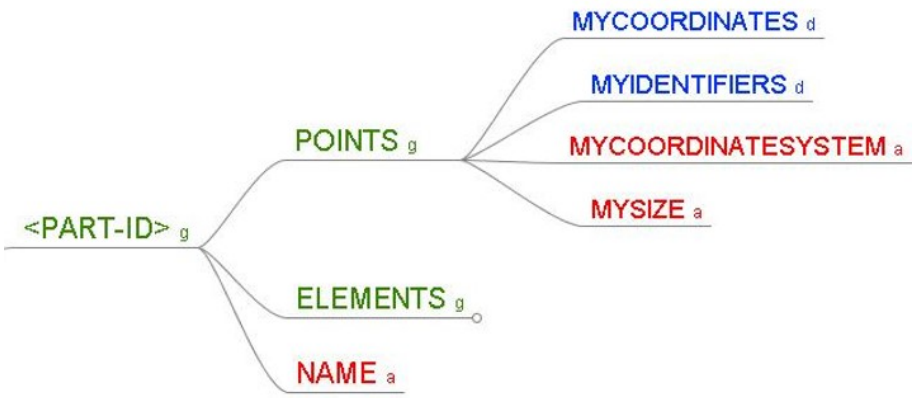


Figure 6.3: POINTS Group

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 sub-section 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 Dataset

The MYCOORDINATES dataset 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 dataset 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 Dataset

The MYIDENTIFIERS dataset 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 dataset 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 an FE model, this includes identifier, type, coordinate system, material type, element connectivity. The attribute `MYSIZE` and dataset `MYELEMENTS` are associated with `ELEMENTS`, shown in Figure 6.4. Tables 6.16 & 6.17 show `Object Attribute Info` and `General Object Info` of `ELEMENTS` Group, these tables show information as seen in the HDF5 Viewer.

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

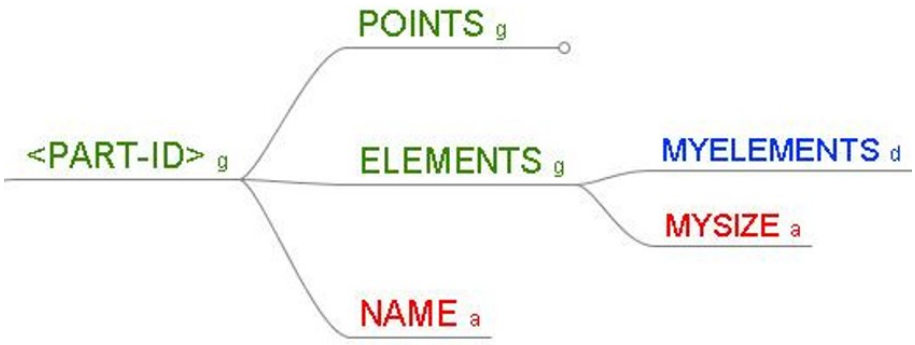


Figure 6.4: ELEMENTS Group

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 dataset `MYELEMENTS`.

Metadata	Data Type
Value	unsigned Integer

Table 6.18: Data Types of MYSIZE Metadata

6.5.2 MYELEMENTS Dataset

The `MYELEMENTS` dataset 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`, see section 6.11.
- `myConnectivity`: Stores the point number ordering which forms one element. For details about point number ordering see chapter 8.

This is a compound dataset.

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 dataset 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 all the variables of an FE model, this includes all input & output state variables. All the states STATE-<n> of the FE-Model are associated with VARIABLES. Figure 6.5 shows the complete storage structure of the VARIABLES Group. Table 6.21 shows the General Object Info of VARIABLES Group, the table shows information as seen in the HDF5 Viewer.

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

Number of members:	1
Name	Type
STATE-<n>	Group

Table 6.21: General Object Info

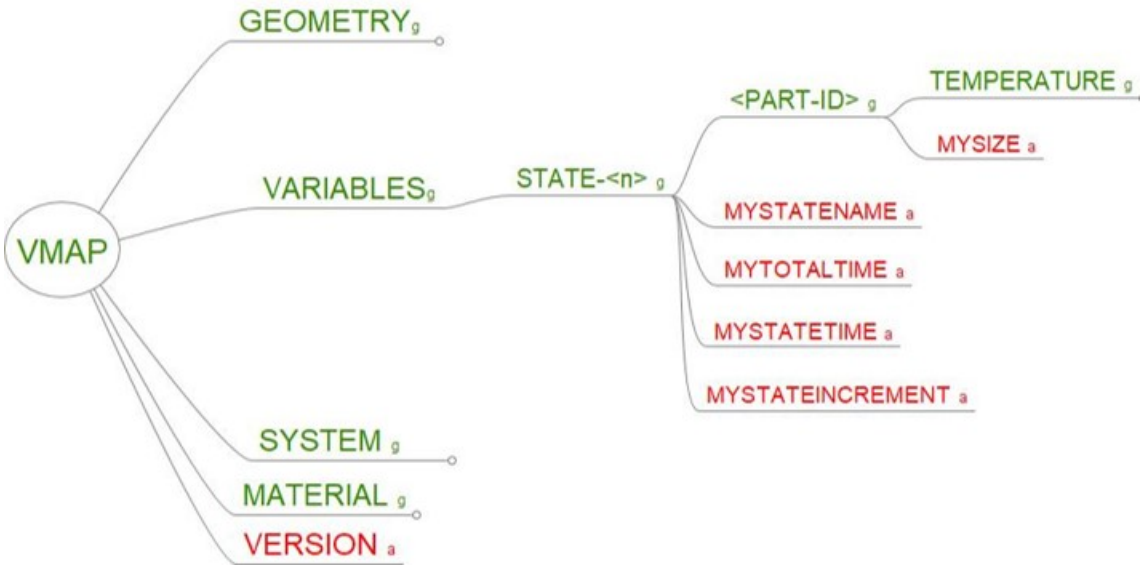


Figure 6.5: VARIABLES Group

6.7 STATE-<n> Group

STATE-<n> Group defines different states of an FE-Model. Here, a **time step**, a **solution step** or a **frame** could be referred as a state. A state is defined as any given time-step (e.g. transient simulations), load step (e.g. non-linear simulations) interval or a frame (e.g. different load case), where the FE-model has a certain result set, which may or may not change for the next state. This group contains all the part numbers which have state variables defined for the particular STATE-<n>. The four **optional** attributes, to store the state metadata, and group(s) <PART-ID>, with state variables for this state, are associated with STATE-<n>. Table 6.22 & Table 6.23 show `Object Attribute Info` and `General Object Info` of STATE-<n> Group, these tables show information as seen in the HDF5 Viewer.

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

Here is an example, which explains the difference between MYTOTALTIME & MYSTEPTIME.

MYSTATENAME	MYTOTALTIME	MYSTEPTIME
STATE-0 (1. step begin)	0	0
STATE-1 (1. step)	5	5
STATE-2 (1. step end)	10	10
STATE-3 (2. step)	11	1
STATE-4 (2. step end)	15	5

Table 6.24: Two Step Analysis with result export at each step

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 6.25: Data Types of MYSTATENAME Metadata

6.7.2 MYTOTALTIME Attribute

This attribute denotes the absolute physical simulation time.

- Value: Stores total time of the analysis.

Metadata	Data Type
Value	Double

Table 6.26: Data Types of MYTOTALTIME Metadata

6.7.3 MYSTEPTIME Attribute

This attribute is the relative physical time of a analysis step.

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

Metadata	Data Type
Value	Double

Table 6.27: Data Types of MYSTEPTIME 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 6.28: Data Types of MYSTATEINCREMENT Metadata

6.8 <PART-ID> Group

<PART-ID> Group corresponds to the <PART-ID> Group defined for in GEOMETRY Group. This means that for a given STATE-<n>, a particular <PART-ID> has state variable(s). The state variable(s) as group(s) are associated with <PART-ID>. Table 6.29 & 6.30 show Object Attribute Info and General Object Info of <PART-ID> Group, these tables show information as seen in the HDF5 Viewer.

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

Table 6.29: Object Attribute Info

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

Number of members:	MYSIZE
Name	Type
TEMPERATURE	Group

Table 6.30: 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 6.31: Data Types of MYSIZE Metadata

Section 6.9 explains an example as a State Variable group.

6.9 TEMPERATURE Group

TEMPERATURE is used as an example to show how any state variable can be stored in VMAP. TEMPERATURE Group stores data about the temperature variation over the FE model for a given time-stamp. There are eleven attributes and two datasets associated with TEMPERATURE or with any other state variable. The attributes are shown in table 6.32 and explained in the following sub-sections. The dataset MYVALUES & if the state variable is defined over a set then an additional dataset MYGEOMETRYIDS are associated with TEMPERATURE or with any other state variable, shown in Figure 6.6. Table 6.32 & 6.33 show Object Attribute Info and General Object Info of TEMPERATURE Group, these tables show information as seen in the HDF5 Viewer.

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 6.32: Object Attribute Info

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

Number of members:	2
Name	Type
MYVALUES	Dataset
MYGEOMETRYIDS*	Dataset

Table 6.33: General Object Info (*optional)

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 6.34: 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 6.35: Data Types of MYDIMENSION Metadata

Table 6.36 shows the available dimensions and their reference numbers in VMAP.

MYDIMENSION	Reference Number	Storage Format in VMAP
INVALID	0	
SCALAR	1	single value
VECTOR	3	v_X, v_Y, v_Z
2nd Order Plain Tensor Symmetric	4	t_XX, t_YY, t_ZZ, t_XY
2nd Order Tensor Symmetric	6	t_XX,t_YY,t_ZZ,t_XY, t_YZ,t_XZ
2nd Order Tensor	9	t_XX,t_YY,t_ZZ,t_XY, t_YZ,t_XZ,t_YX,t_ZY,t_ZX
STIFFNESS MATRIX	36	Voigt notation: t_11, t_22, t_33, . . . , t_66, t_12, t_23, . . . , t_56, t_13, t_24,. . . , t_16
4th Order Tensor Symmetric	45	[[t_1111,t_1122,t_1133,..,t_1121], [t_2222,t_2233,. . . ,t_2221] ,..,[t_2121]]
4th Order Tensor	81	[[t_1111,t_1122,t_1133,. . . ,t_1121], [t_2211,. . . ,t_2221] ,..,[t_2111,. . . ,t_2121]]

Table 6.36: 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 6.37: Data Types of MYENTITY Metadata

Table 6.38 shows the available entities and their reference numbers in VMAP

MYENTITY	Reference Number
REAL	1
COMPLEX	2
HAMILTONIAN	4

Table 6.38: 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 6.39: 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 6.40: 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 6.41: Data Types of MYLOCATION Metadata

Table 6.42 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 6.42: MYLOCATION Enumeration

6.9.7 MYMULTIPLICITY Attribute

Refers to the number of columns in the MYVALUES dataset. The majority of state variables have multiplicity 1.

- Value: Stores the multiplicity of the MYVALUES dataset.

Metadata	Data Type
Value	Integer

Table 6.43: Data Types of MYMULTIPLICITY Metadata

A majority of the state variables have MYMULTIPLICITY 1. Sometimes it might 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. A 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 6.44: 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 either SYSTEM/UNITSYSTEM or SYSTEM/UNITS, see sub-section 6.10.6 & 6.10.5 respectively. If a base unit is used then it can be directly referred from SYSTEM/UNITSYSTEM dataset

and if a derived or a scaled unit needs to be used, it should be defined in SYSTEM/UNITS dataset. If a state variable is unitless without any particular unit name, then use -1 here. However, if a unit needs to be defined, which is unitless e.g. radians, then define it in the SYSTEM/UNITS dataset and refer it here.

- Value: Stores the reference number of the unit.

Metadata	Data Type
Value	Integer

Table 6.45: Data Types of MYUNIT Metadata

6.9.10 MYVARIABLEDESCRIPTION Attribute

Stores detailed description of the variable.

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

Metadata	Data Type
Value	String

Table 6.46: Data Types of MYVARIABLEDESCRIPTION Metadata

6.9.11 MYVARIABLENAME Attribute

Stores name of the variable.

- Value: Stores name of the variable.

Metadata	Data Type
Value	String

Table 6.47: Data Types of MYVARIABLENAME Metadata

6.9.12 MYVALUES Dataset

The MYVALUES dataset 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 6.48).

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 dataset** 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 6.48: Data Types of MYVALUES Metadata

The General Object Info associated with this dataset is shown in Table 6.49.

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 6.49: General Object Info

m given in Table 6.49 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 = $\sum$ (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 MYGEOMETRYIDS Dataset - Optional

This dataset is optional and should be used only when a set of the points, elements, integration types or elements faces are used. The set is then defined in this dataset. MYLOCATION attribute at the state variable level is used to identify the set type (points, elements, integration types or elements faces). In the HDF5 Viewer, the column is named by default as 0 for myvalue (shown in Table 6.50).

Metadata	Data Type
myvalue	Integer

Table 6.50: Data Types of MYGEOMETRYIDS Metadata

The `General Object Info` associated with this dataset is shown in Table 6.51.

Name:	`MYGEOMETRYIDS`
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:	32-bit Integer

Table 6.51: General Object Info

m = number of values defined in the set.

6.9.14 `MYINTEGRATIONTYPES` Dataset

This dataset exists only when the `MYLOCATION` attribute defined at the state variable level (e.g. TEMPERATURE) = 4 (=Integration Point). The dataset 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 6.52).

Metadata	**Data Type**
myvalue	Floating-Point Array

Table 6.52: Data Types of MYINTEGRATIONTYPES Metadata

The `General Object Info` associated with this dataset is shown in Table 6.53.

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 x 1
Max Dimension Size(s):	m x 1
Data Type:	32-bit Integer

Table 6.53: General Object Info

m = MYSIZE of ELEMENTS or based on the number of values in MYGEOMETRYIDS, if defined.

6.10 SYSTEM Group

SYSTEM Group stores the system data related to the FE model. This includes six datasets - COORDINATESYSTEM, ELEMENTTYPES, INTEGRATIONTYPES, METADATA, UNITS & UNITSYSTEM, shown in Figure 6.7. Table 6.54 shows General Object Info of SYSTEM Group, the table shows information as seen in the HDF5 Viewer.

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

Number of members:	6
Name	Type
COORDINATESYSTEM	Dataset
ELEMENTTYPES	Dataset
INTEGRATIONTYPES	Dataset
METADATA	Dataset
UNITS	Dataset
UNITSYSTEM	Dataset

Table 6.54: General Object Info

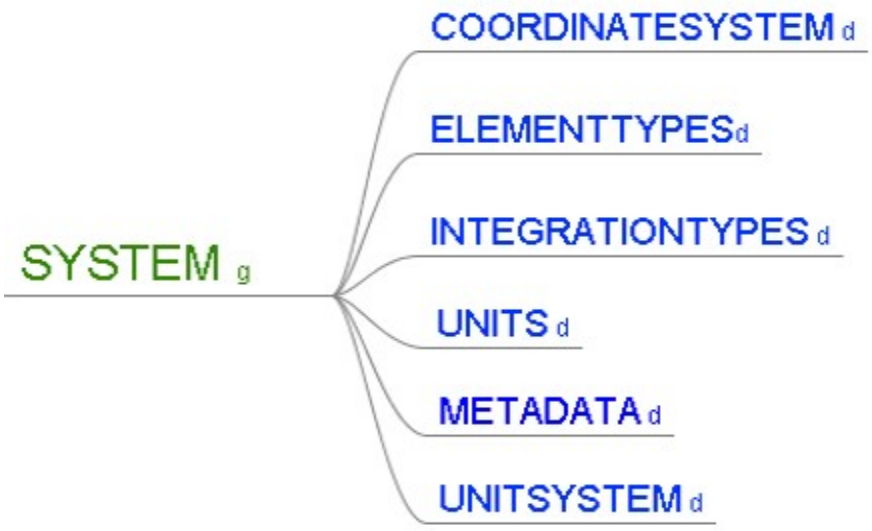


Figure 6.7: SYSTEM Group

6.10.1 `COORDINATESYSTEM` Dataset

The `COORDINATESYSTEM` dataset 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 6.56.
- `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 dataset.

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

Table 6.55: Data Types of COORDINATESYSTEM Metadata

Table 6.56 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 6.56: COORDINATESYSTEM Enumeration

The General Object Info associated with this dataset is shown in Table 6.57.

Name:	COORDINATESYSTEM
Path:	/VMAP/SYSTEM/
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:	Compound

Table 6.57: General Object Info

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

6.10.2 ELEMENTTYPES Dataset

The ELEMENTTYPES dataset 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.
- myTypeDescription: Stores a more generic element type description. Here are some of the type descriptions, THERMAL, STRUCTURAL, ELECTRICAL, ACOUSTIC, ELECTROMAGNETIC, PIEZOELECTRIC, FLUID, PARTICLE, HEAT-TRANSFER, CONVECTION, RIGID, CONTACT, MEMBRANE, SPRING, PLANESTRESS, PLANESTRAIN, USERDEFINED etc. These types can also be combined e.g. STRUCTURAL/THERMAL. This field is not fixed, user an assign any relevant value.
- 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 6.60 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 dataset.

Metadata	**Data Type**
`myIdentifier`	Integer
`myTypeName`	Character Pointer
`myTypeDescription`	Character Pointer
`myNumberofNodes`	Integer
`myDimension`	Integer
`myShapeType`	Integer
`myInterpolationType`	Integer
`myIntegrationType`	Integer
`myNumberofNormalComponents`	Integer
`myNumberofShearComponents`	Integer
`myConnectivity`	Integer Array
`myFaceConnectivity`	Integer Array

Table 6.58: Data Types of ELEMENTTYPES Metadata

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

Name:	`ELEMENTTYPES`
Path:	`/VMAP/SYSTEM/`
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:	Compound

Table 6.59: General Object Info

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

Table 6.60 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 6.60: myInterpolationType Enumeration

6.10.3 `INTEGRATIONTYPES` Dataset

The `INTEGRATIONTYPES` dataset 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. This value should be taken directly from the Integration Type library `VMAPIntegrationTypeFactory.cxx`. For user defined integration types, the integral identifier starts from 100000.
- `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, $Gauss\_Triangle\_1 \times Gauss\_3$. According to VMAP Standard, the order of compound integration type is,

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

This is a compound dataset.

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

Table 6.61: Data Types of INTEGRATIONTYPES Metadata

The General Object Info associated with this dataset is shown in Table 6.62.

Name:	INTEGRATIONTYPES
Path:	/VMAP/SYSTEM/
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:	Compound

Table 6.62: General Object Info

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

6.10.4 METADATA Dataset

The METADATA dataset 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 6.63: Data Types of METADATA Metadata

The General Object Info associated with this dataset is shown in Table 6.64.

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 6.64: General Object Info

6.10.5 UNITS Dataset

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

- myIdentifier: Integral identifier for each unit used in the FE Model. It is always > 7 because the first 7 identifiers belong to the UNITSYSTEM dataset.
- 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 dataset. Please note that to be able to store unitless values which have a unit symbol, myUnitDimension should be assigned all $0s$. Hence, if a percentage needs to be stored then, make sure to use the decimal value and not the percentage itself e.g. to store 30%, please store it as 0.3.

Metadata	Data Type
myIdentifier	Integer
myUnitSymbol	Character Pointer
myUnitDimension	Integer Array

Table 6.65: Data Types of UNITS Metadata

The General Object Info associated with this dataset is shown in Table 6.66.

Name:	UNITS
Path:	/VMAP/SYSTEM/
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:	Compound

Table 6.66: General Object Info

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

6.10.6 UNITSYSTEM Dataset

The UNITSYSTEM dataset 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 'm' - meter.
  - 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 'kg' - kilogram.
  - 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 dataset.

Note: The following format is used, when converting the unit system. Conversion factor to SI unit, $SI = \texttt{mySIScale} * value + \texttt{mySIShift}$,
`mySIScale` has a default value of 1 & `mySIShift` has a default value of 0.

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

Table 6.67: Data Types of UNITSYSTEM metadata

The `General Object Info` associated with this dataset is shown in Table 6.68.

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 6.68: General Object Info

6.11 MATERIAL Group

This group is only the very first approach to material storage. Please use the Material XML format as a reference document to the MATERIAL Group. MATERIAL Group stores the material cards used by the FE model. The group <MAT> is associated with MATERIAL. Figure 6.8 shows the complete storage structure of the MATERIAL group. Table 6.69 shows General Object Info of MATERIAL Group, the table shows information as seen in the HDF5 Viewer..

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

Number of members:	n
Name	Type
<MAT>	Group

Table 6.69: General Object Info

where n is the number of Materials used in the FE model.

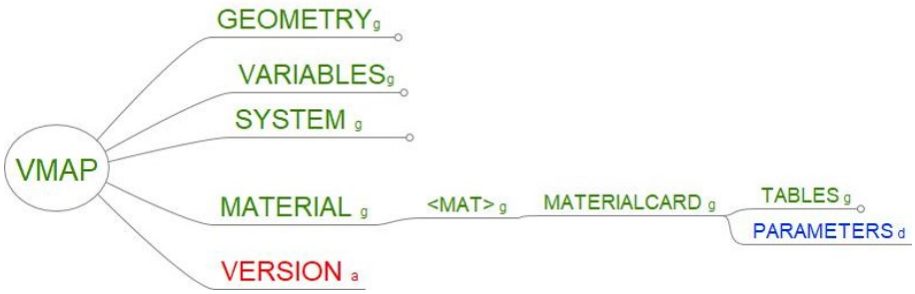


Figure 6.8: MATERIAL Group

6.12 <MAT> Group

<MAT> Group stores one material card. This group name is user-dependent e.g. MAT1, STEEL, MATUNKNOWN etc. Six attributes and one group MATERIALCARD are associated with <MAT>. Table 6.70 & 6.71 show Object Attribute Info and General Object Info of <MAT> Group, these tables show information as seen in the HDF5 Viewer.

Number of attributes:	6		
Name	Type	Array Size	Value
MYDESCRIPTION	String	Scalar	...
MYIDENTIFIER	Integer	Scalar	...
MYNAME	String	Scalar	...
MYSTATE	String	Scalar	...
MYSUPPLIER	String	Scalar	...
MYTYPE	String	Scalar	...

Table 6.70: Object Attribute Info

Name:	<MAT>
Path:	/VMAP/MATERIAL/
Type:	HDF5 Group
Object Ref:	...

Number of members:	1
Name	Type
MATERIALCARD	Group

Table 6.71: General Object Info

6.12.1 MYDESCRIPTION Attribute

Provides a short description of the material e.g. Steel.

- Value: Stores the description.

Metadata	Data Type
Value	String

Table 6.72: Data Types of MYDESCRIPTION Metadata

6.12.2 MYIDENTIFIER Attribute

Refers to the unique integral identifier for the <MAT> Group.

- Value: Stores the unique identifier.

Metadata	Data Type
Value	Integer

Table 6.73: Data Types of MYIDENTIFIER Metadata

6.12.3 `MYNAME` Attribute

Provides a short name of the material.

- Value: Stores name of material.

Metadata	Data Type
Value	String

Table 6.74: Data Types of MYNAME Metadata

6.12.4 `MYSTATE` Attribute

Provides the state of the material e.g. solid, liquid, gas.

- Value: Stores the state of material.

Metadata	Data Type
Value	String

Table 6.75: Data Types of MYSTATE Metadata

6.12.5 `MYSUPPLIER` Attribute

Provides the supplier name.

- Value: Stores the supplier information of the material.

Metadata	Data Type
Value	String

Table 6.76: Data Types of MYSUPPLIER Metadata

6.12.6 `MYTYPE` Attribute

Provides the material type e.g. composite, metal. elastomer etc..

- Value: Stores the type of material.

Metadata	Data Type
Value	String

Table 6.77: Data Types of MYTYPE Metadata

6.13 MATERIALCARD Group

MATERIALCARD Group stores the material card information. Eight attributes, one group TABLES and one dataset PARAMETERS are associated with MATERIALCARD. The TABLES Group associated with MATERIALCARD Group can store the material curves, which belong to the material card. Table 6.78 & 6.79 show Object Attribute Info and General Object Info of MATERIALCARD Group, these tables show information as seen in the HDF5 Viewer.

Number of attributes:	8		
Name	Type	Array Size	Value
MYIDEALIZATION	String	Scalar	...
MYIDENTIFIER	String	Scalar	...
MYMODELNAME	String	Scalar	...
MYPHYSICS	String	Scalar	...
MYSOLUTION	String	Scalar	...
MYSOLVER	String	Scalar	...
MYSOLVERVERSION	String	Scalar	...
MYUNITSYSTEM	String	Scalar	...

Table 6.78: Object Attribute Info

Name:	MATERIALCARD
Path:	/VMAP/MATERIAL/<MAT>/
Type:	HDF5 Group
Object Ref:	...

Number of members:	2
Name	Type
PARAMETERS	Dataset
TABLES	Group

Table 6.79: General Object Info

6.13.1 MYIDEALISATION Attribute

Provides the idealisation of the material e.g. shell, beam, solid etc.

- Value: Stores the idealisation.

Metadata	Data Type
Value	String

Table 6.80: Data Types of MYIDEALISATION Metadata

6.13.2 MYIDENTIFIER Attribute

Refers to the unique integral identifier for the MATERIALCARD Group. This value is generally taken from the FE Model.

- Value: Stores the unique identifier.

Metadata	Data Type
Value	String

Table 6.81: Data Types of MYIDENTIFIER Metadata

6.13.3 MYMODELNAME Attribute

Provides the solver specific material model.

- Value: Stores the material model.

Metadata	Data Type
Value	String

Table 6.82: Data Types of MYMODELNAME Metadata

6.13.4 MYPHYSICS Attribute

Explains the physics behind the material e.g. solid mechanics, fluid mechanics, heat transfer etc.

- Value: Stores the physics associated with the material card.

Metadata	Data Type
Value	String

Table 6.83: Data Types of MYPHYSICS Metadata

6.13.5 MYSOLUTION Attribute

Explains the type of solution e.g. implicit, explicit etc.

- Value: Stores the type of solution.

Metadata	Data Type
Value	String

Table 6.84: Data Types of MYSOLUTION Metadata

6.13.6 MYSOLVER Attribute

Provides the solver from which the material card has been taken.

- Value: Stores the solver name.

Metadata	Data Type
Value	String

Table 6.85: Data Types of MYSOLVER Metadata

6.13.7 MYSOLVERVERSION Attribute

Provides the version of the solver from which the material card has been taken.

- Value: Stores the solver version.

Metadata	Data Type
Value	String

Table 6.86: Data Types of MYSOLVERVERSION Metadata

6.13.8 MYUNITSYSTEM Attribute

Provides the unit system of the material card e.g. 1_SI_N_kg_m_s_N_Pa

- Value: Stores the unit system for the material card.

Metadata	Data Type
Value	String

Table 6.87: Data Types of MYUNITSYSTEM Metadata

6.13.9 PARAMETERS Dataset

The PARAMETERS dataset stores the material parameters of the material card. The following information is stored.

- NAME: Stores a short form of the material parameter.
- VALUE: Stores the value of the corresponding material parameter.

- DESCRIPTION: Stores the description/long name of the corresponding material parameter.

This is a compound dataset. In the HDF5 Viewer, the columns are named as in the list above, see Table 6.88.

Metadata	**Data Type**
NAME	String
VALUE	String
DESCRIPTION	String

Table 6.88: Data Types of PARAMETERS Metadata

The `General Object Info` associated with this dataset is shown in Table 6.89.

Name:	`PARAMETERS`
Path:	`/VMAP/MATERIAL/<MAT>/MATERIALCARD/`
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:	Compound

Table 6.89: General Object Info

m depends on the number of parameters associated with the material card.

6.14 TABLES Group

`TABLES` Group stores the tables. This group is highly user-defined with minimal standardization provided by VMAP Specifications. This group can be stored within any of the four major VMAP groups. Here is just an example of how it can be used to store a material curve. The group stores individual tables as datasets. Table 6.90 shows `General Object Info` of `TABLES` Group, this tables show information as seen in the HDF5 Viewer.

Name:	TABLES
Path:	*/VMAP/MATERIAL/<MAT>/MATERIALCARD/
Type:	HDF5 Group
Object Ref:	...

Number of members:	n
Name	Type
<TABLENAME>	Dataset

Table 6.90: General Object Info

where n refers to the number of datasets associated with the table.
***this path is dependent on the location of the table.**

6.14.1 <TABLENAME> Dataset

The <TABLENAME> dataset stores a table with multiple columns and rows. The dataset name <TABLENAME> is user-defined. This dataset has a few fixed attributes and it can further have user-defined attributes. The fixed attributes are listed in 6.91. Table 6.91 & 6.92 show `Object Attribute Info` and `General Object Info` of <TABLENAME> dataset, these tables show information as seen in the HDF5 Viewer.

Number of attributes:	2		
Name	Type	Array Size	Value
MYIDENTIFIER	Integer	Scalar	...
MYTABLENAME	String	Scalar	...

Table 6.91: Object Attribute Info

Name:	<TABLENAME>
Path:	*/VMAP/MATERIAL/<MAT>/MATERIALCARD/TABLES/
Type:	HDF5 Dataset
Object Ref:	...

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

Table 6.92: General Object Info

m depends on the number of table rows.
***this path is dependent on the location of the table.**

Chapter 7

Storage Format

This chapters shows the VMAP format 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

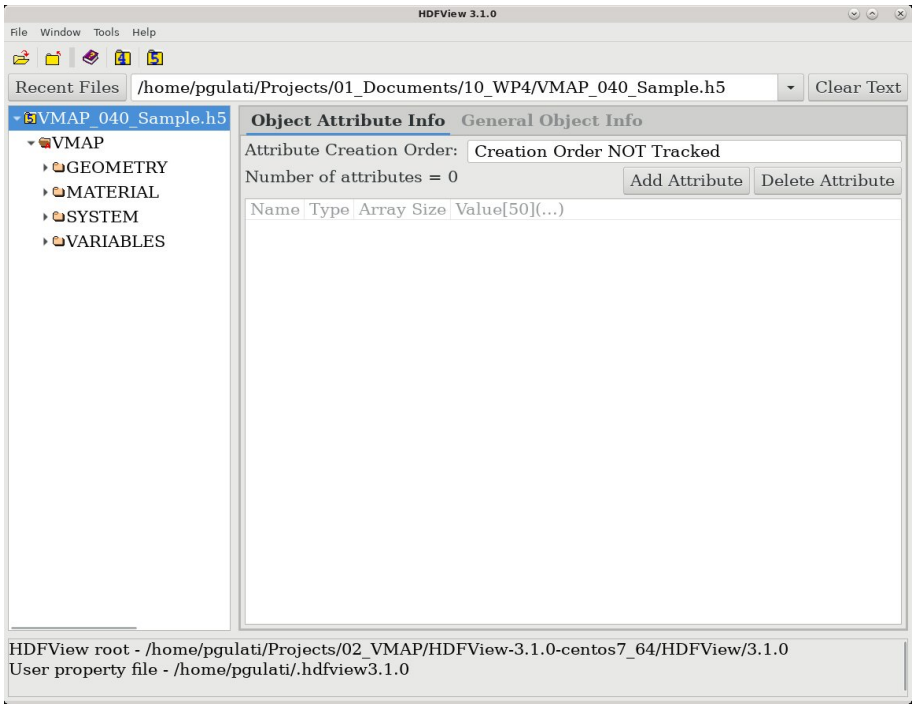


Figure 7.1: .h5 File View - Object Attribute Info

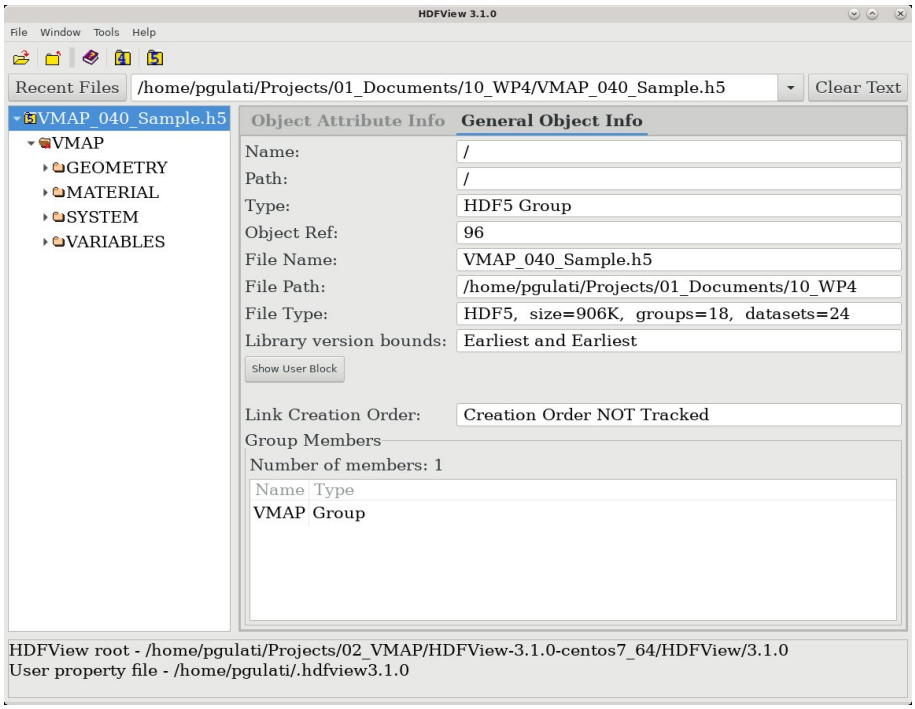


Figure 7.2: .h5 File View - General Object Info

7.2 VMAP Group View

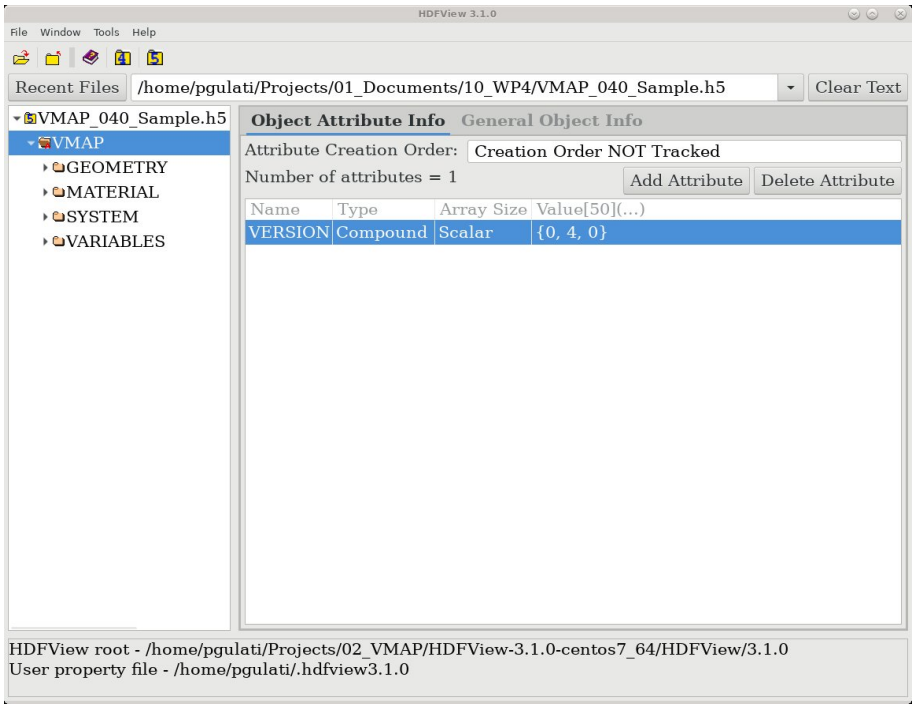


Figure 7.3: VMAP Group View - Object Attribute Info

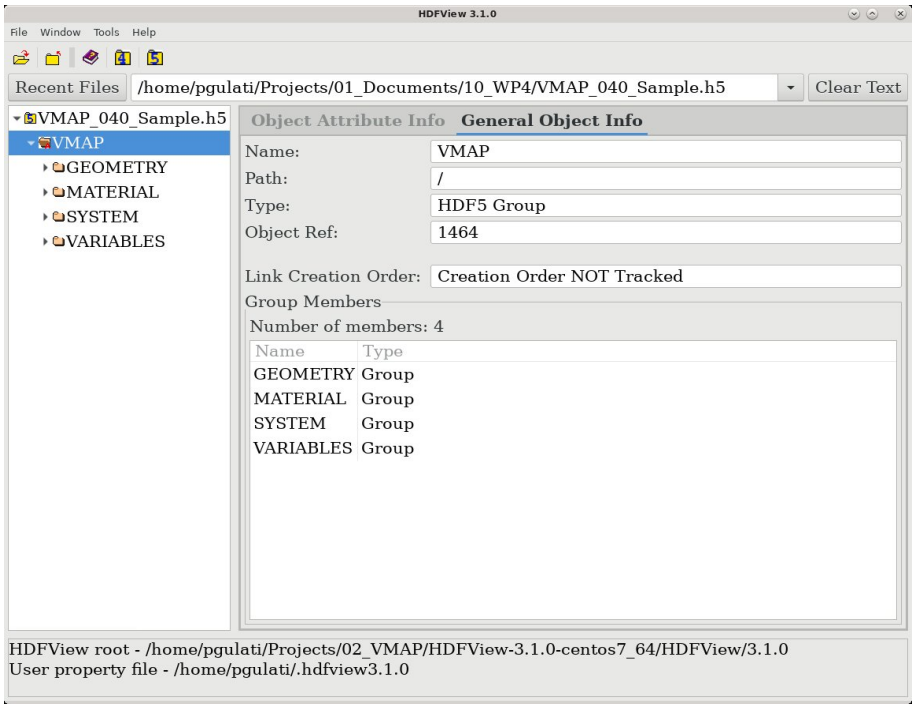


Figure 7.4: VMAP Group View - General Object Info

7.2.1 VERSION Attribute View

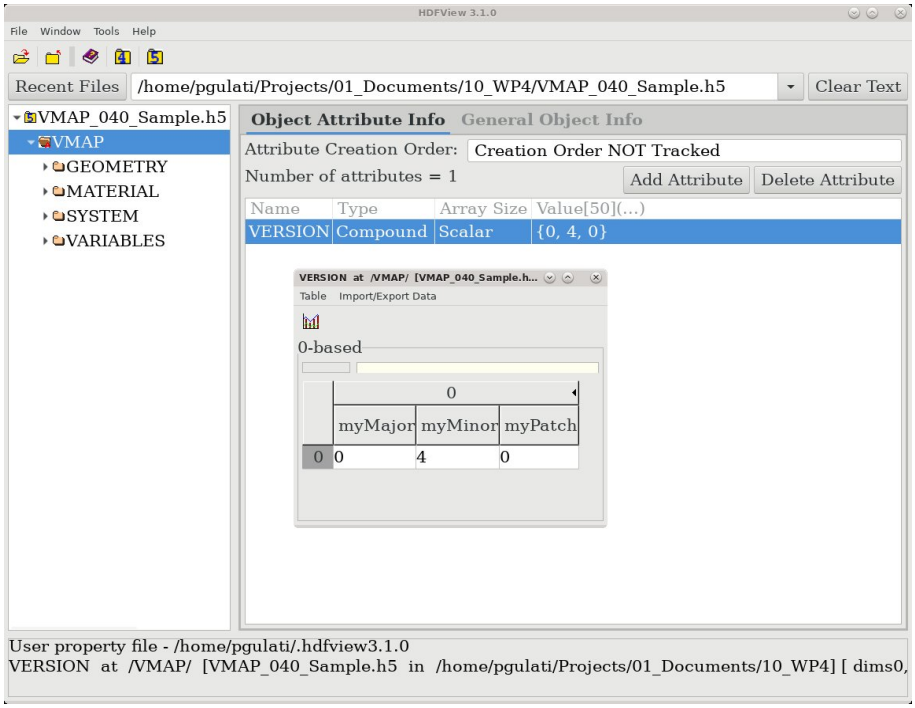


Figure 7.5: VERSION Attribute View - Metadata

7.3 GEOMETRY Group View

GEOMETRY Group has no attributes.

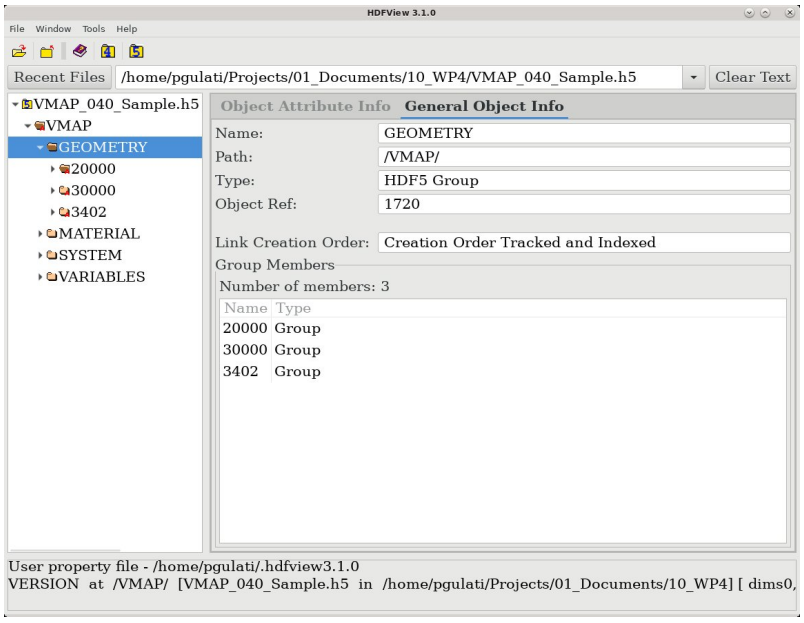


Figure 7.6: GEOMETRY Group View - General Object Info

7.4 <PART-ID> Group View

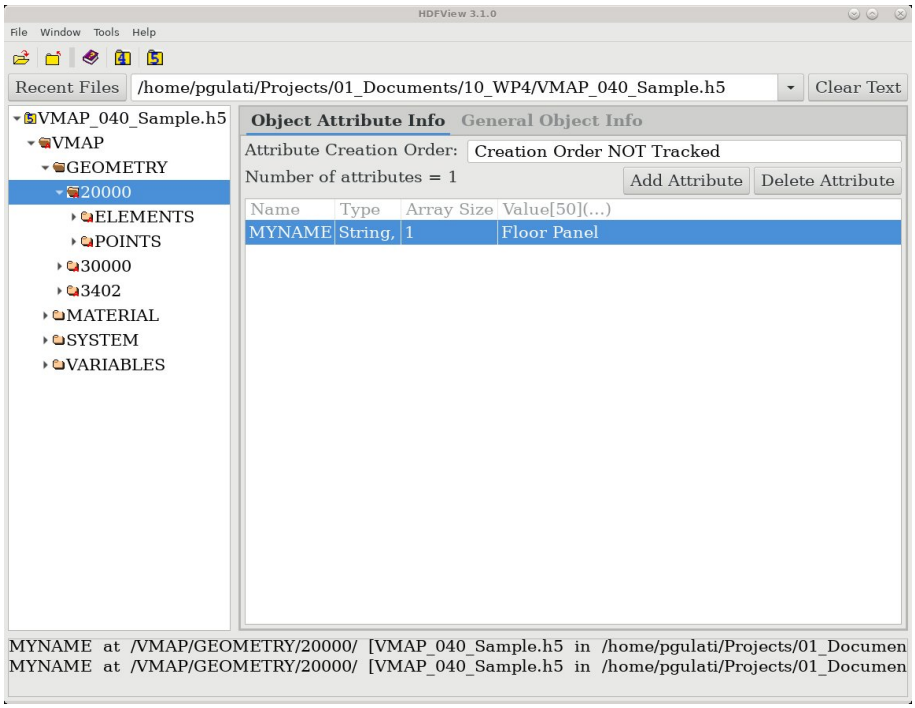


Figure 7.7: <PART-ID> Group View - Object Attribute Info

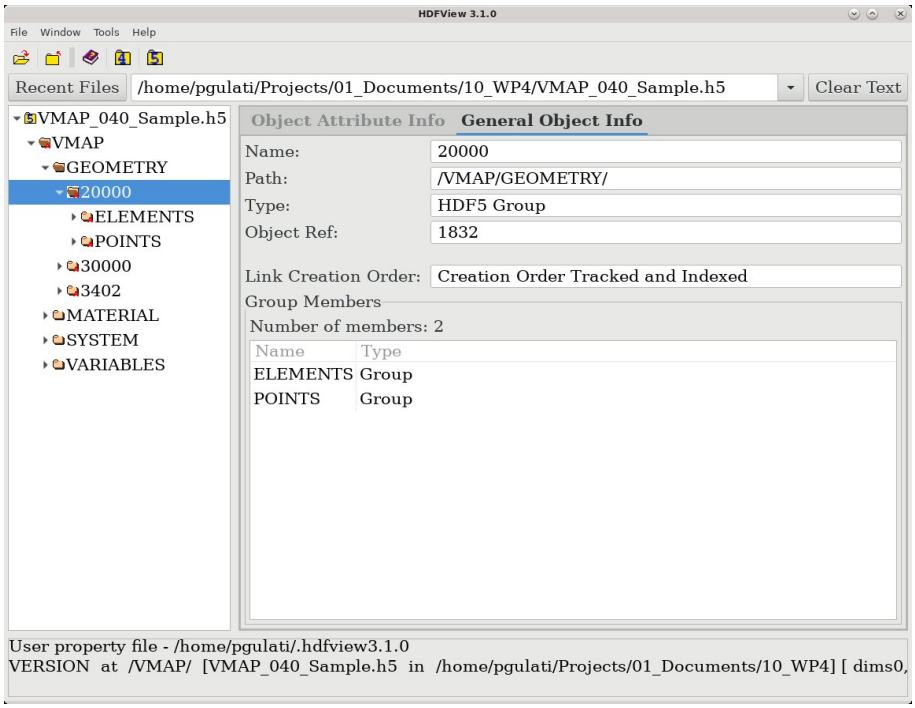


Figure 7.8: <PART-ID> Group View - General Object Info

The attribute MYNAME and its Value is shown in Figure 7.7

7.5 POINTS Group View

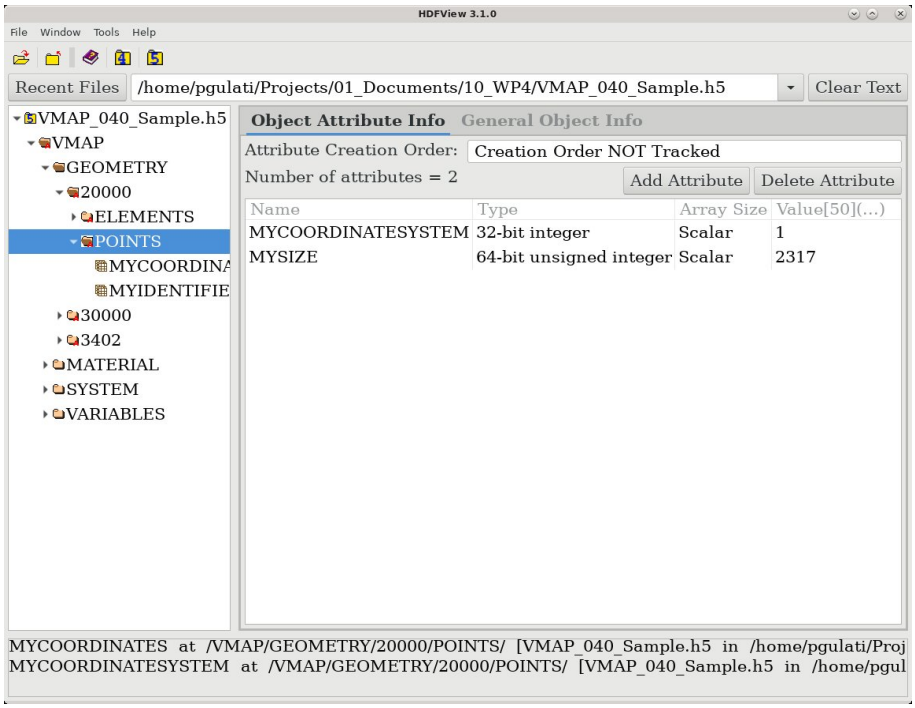


Figure 7.9: POINTS Group View - Object Attribute Info

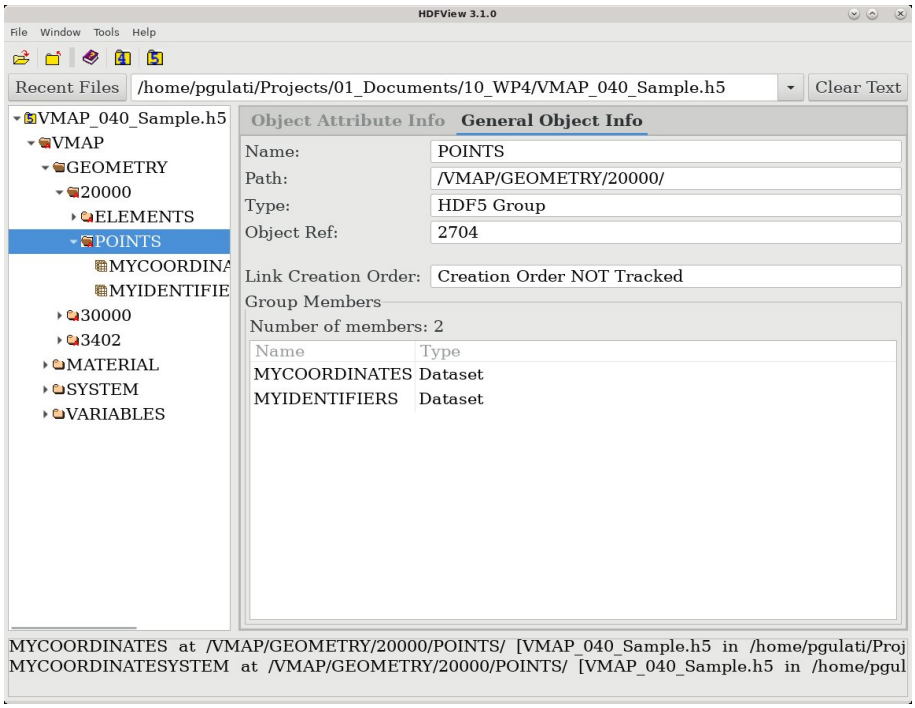


Figure 7.10: POINTS Group View - General Object Info

The attributes MYCOORDINATESYSTEM & MYSIZE are shown in Figure 7.9.

7.5.1 MYCOORDINATES Dataset

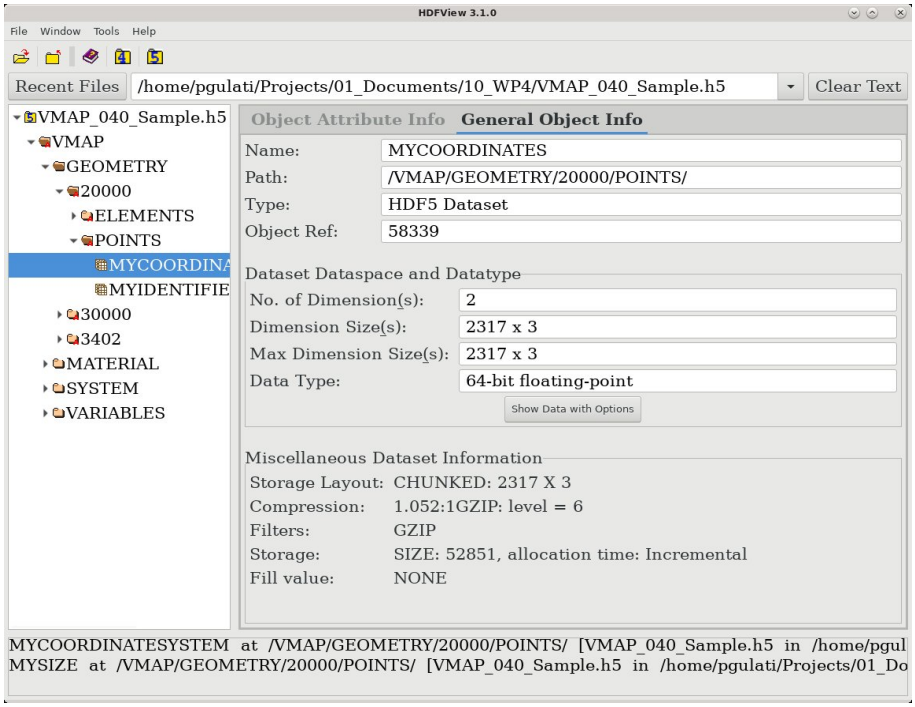


Figure 7.11: MYCOORDINATES Dataset View - General Object Info

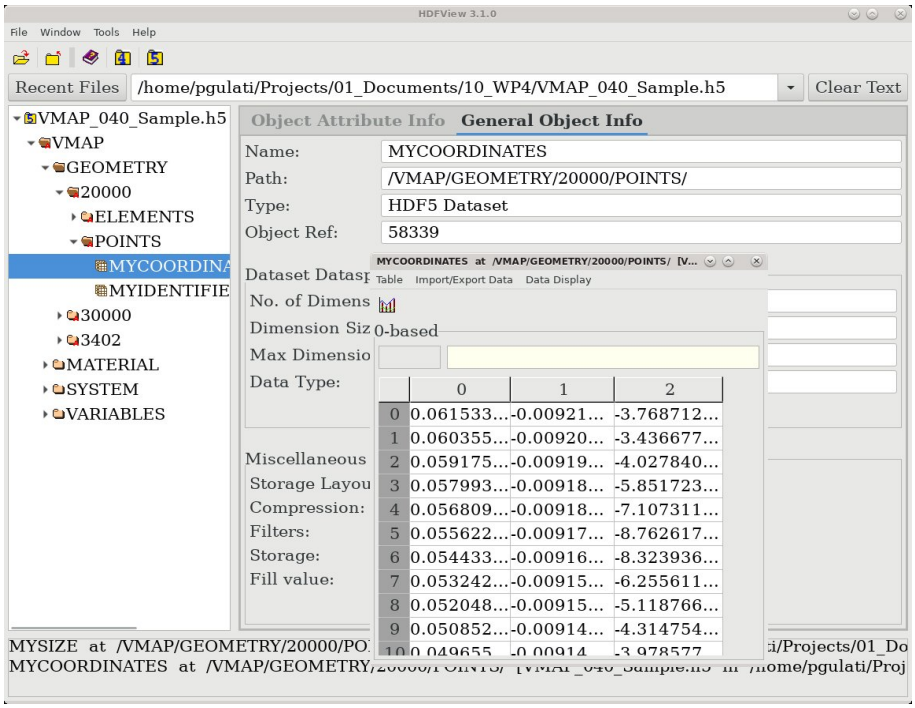


Figure 7.12: MYCOORDINATES Dataset View - Metadata

7.5.2 MYIDENTIFIERS Dataset

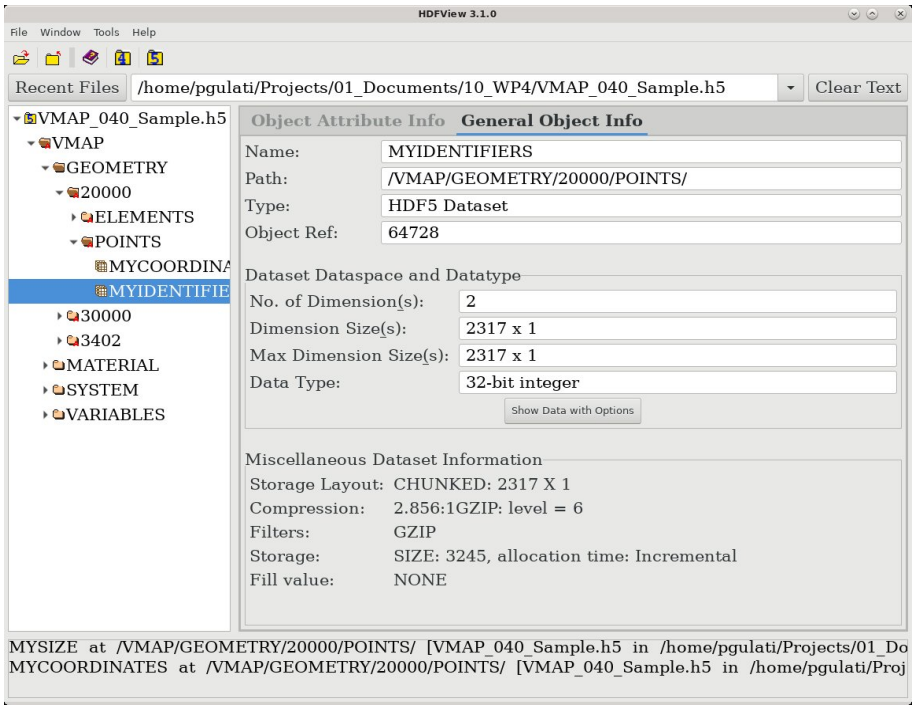


Figure 7.13: MYIDENTIFIERS Dataset View - General Object Info

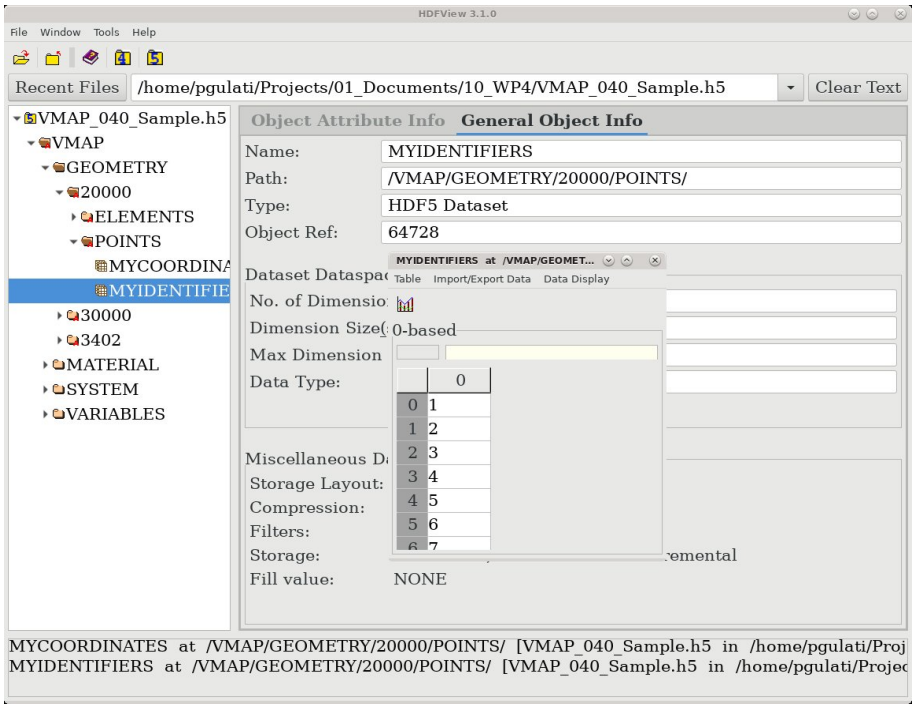


Figure 7.14: MYIDENTIFIERS Dataset View - Metadata

7.6 ELEMENTS Group View

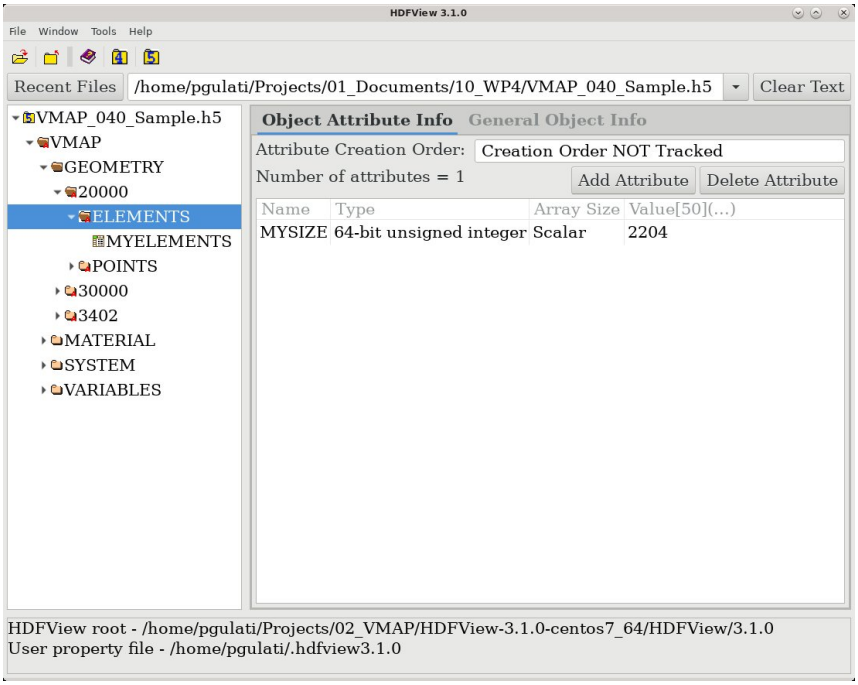


Figure 7.15: ELEMENTS Group View - Object Attribute Info

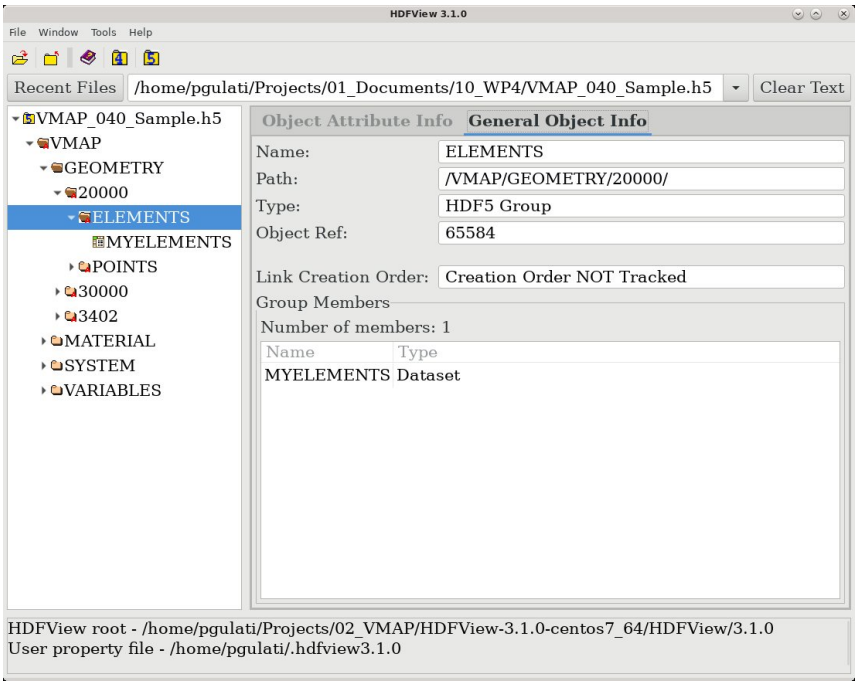


Figure 7.16: ELEMENTS Group View - General Object Info

The attribute MYSIZE is shown in Figure 7.15.

7.6.1 MYELEMENTS Dataset

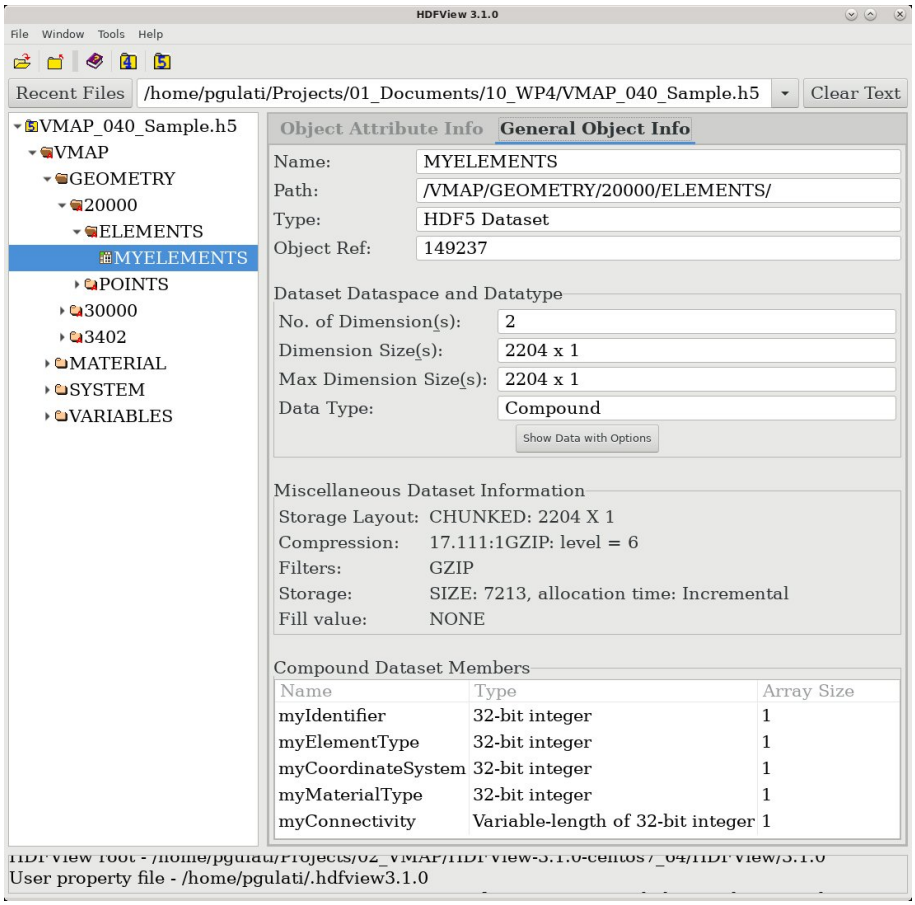


Figure 7.17: MYELEMENTS Dataset View - General Object Info

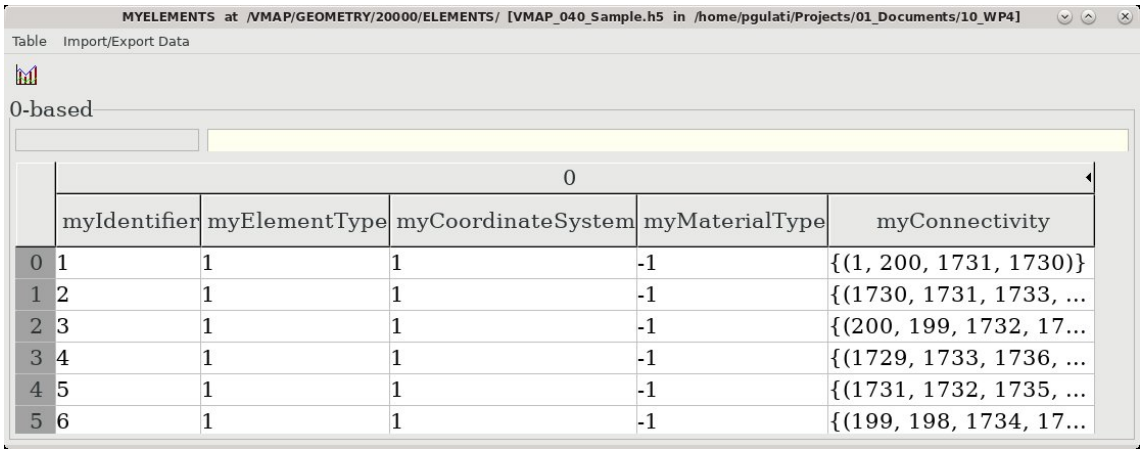


Figure 7.18: MYELEMENTS Dataset View - Metadata

7.7 VARIABLES Group View

VARIABLES Group has no attributes.

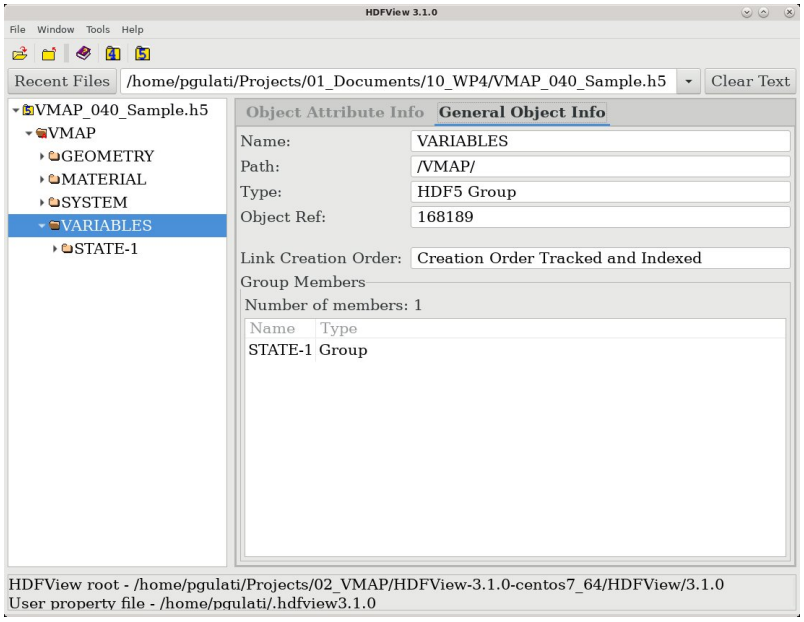


Figure 7.19: RESULT Group View - General Object Info

7.8 STATE-<n> Group View

STATE-<n> Group can have attributes. However, these are optional.

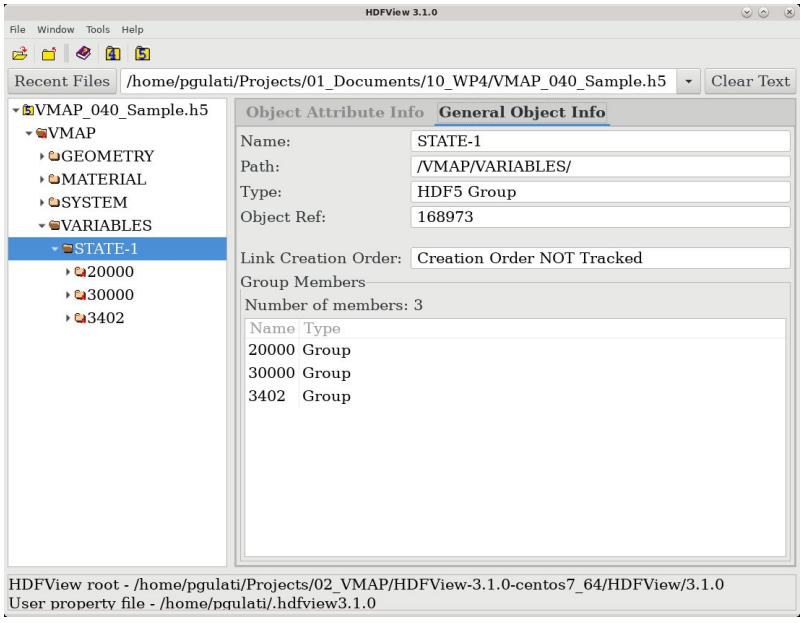


Figure 7.20: STATE-<n> Group View - General Object Info

7.9 <PART-ID> Group View

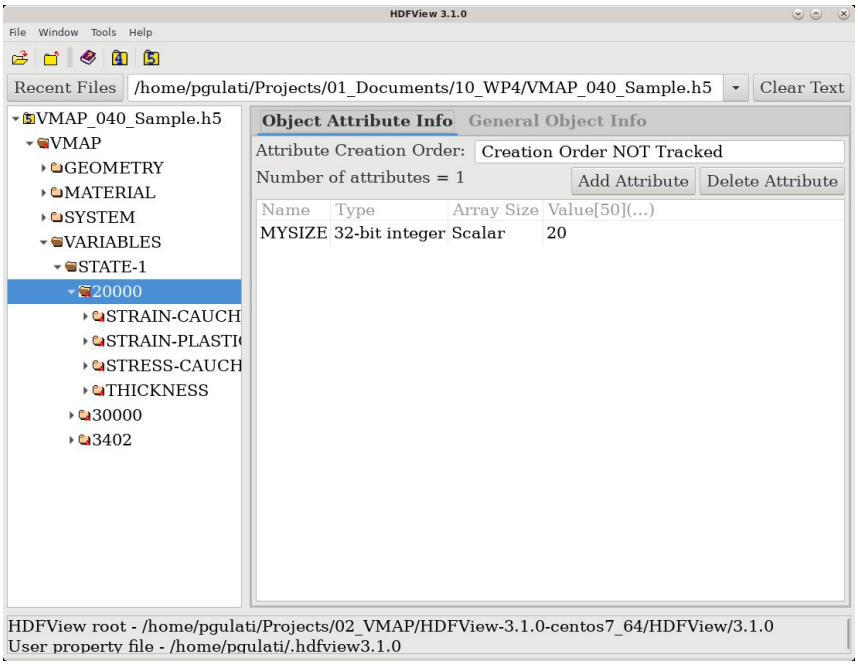


Figure 7.21: <PART-ID> Group View - Object Attribute Info

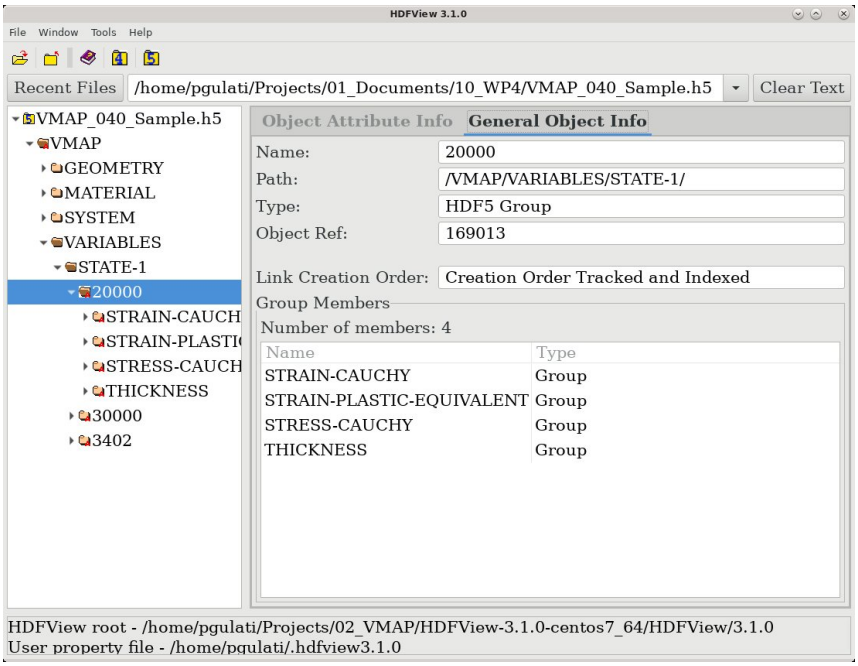


Figure 7.22: <PART-ID> Group View - General Object Info

The attribute MYSIZE is seen in Figure 7.21

7.10 STRAIN-CAUCHY Group View

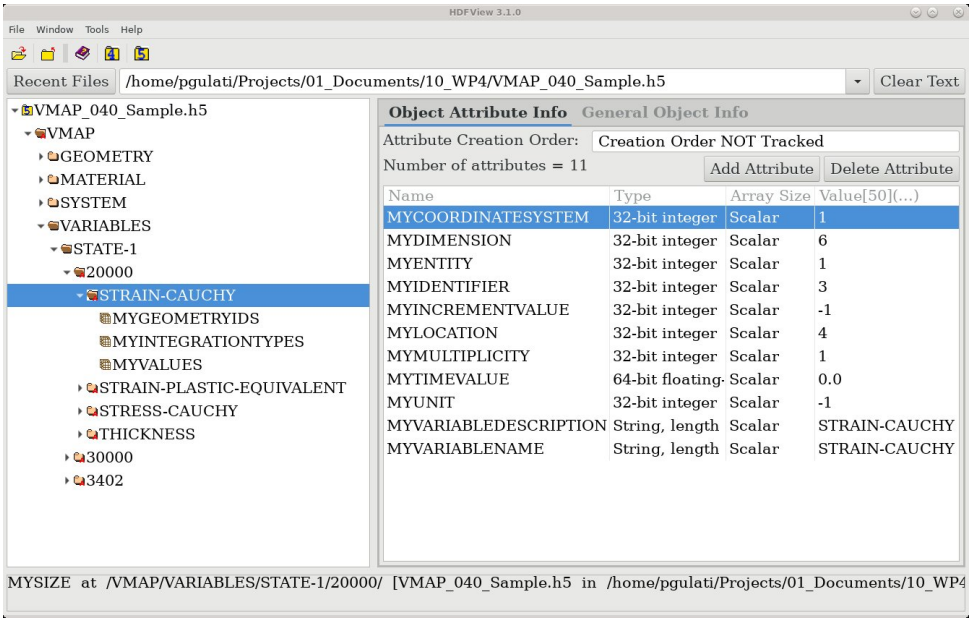


Figure 7.23: STRAIN-CAUCHY Group View - Object Attribute Info

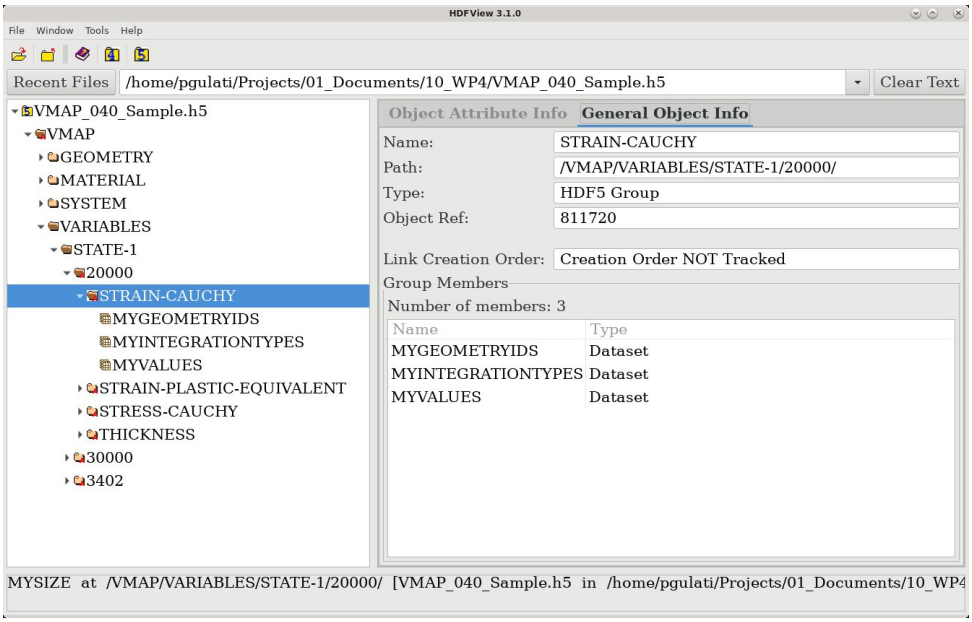


Figure 7.24: STRAIN-CAUCHY Group View - General Object Info

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

7.10.1 MYGEOMETRYIDS Dataset

This is an optional dataset.

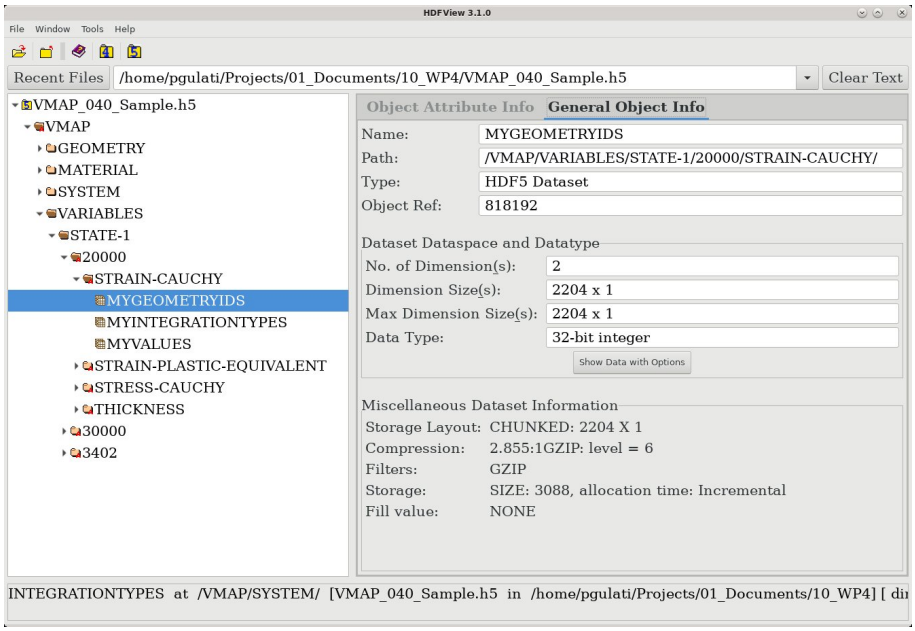


Figure 7.25: MYGEOMETRYIDS Dataset View - General Object Info

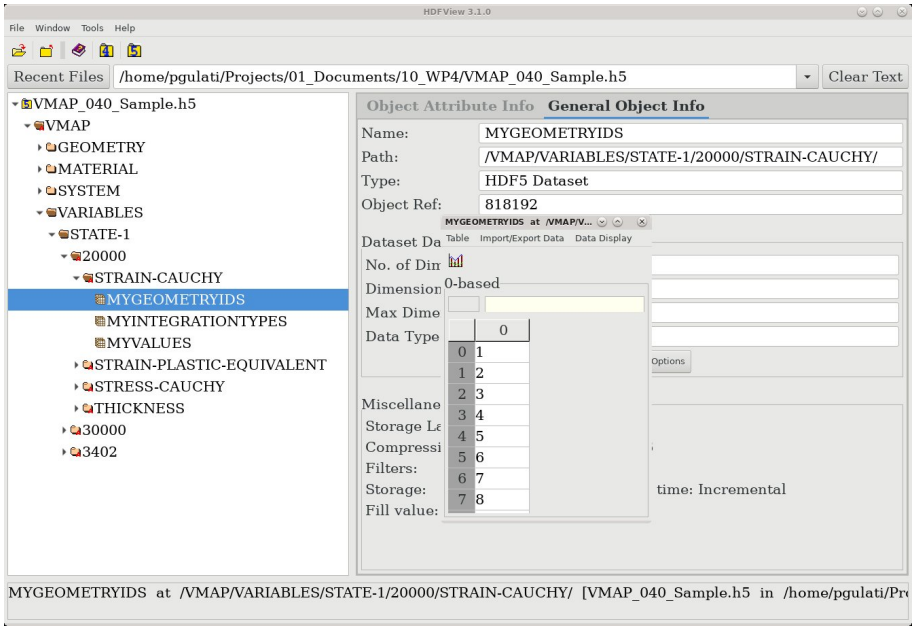


Figure 7.26: MYGEOMETRYIDS Dataset View - Metadata

7.10.2 MYVALUES Dataset

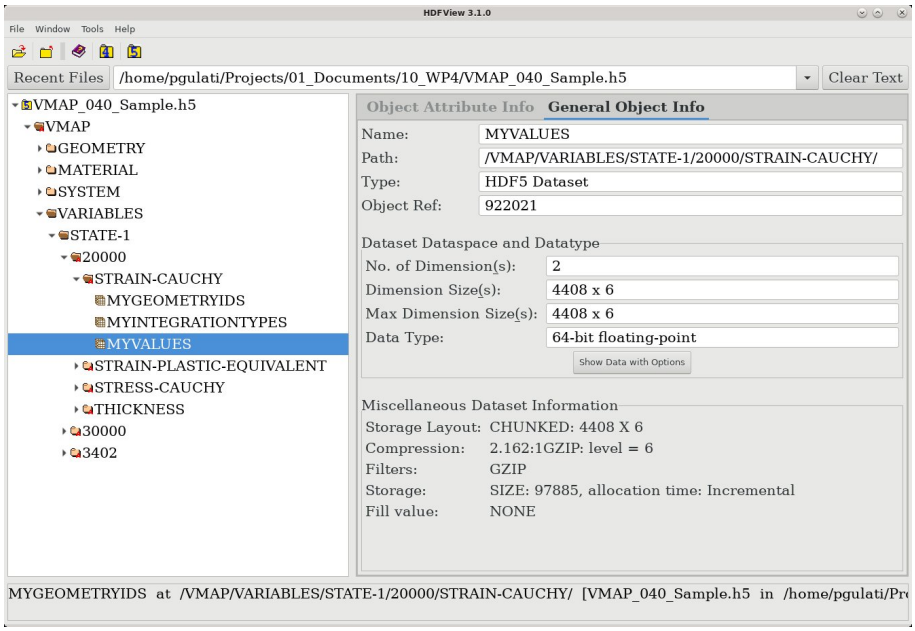


Figure 7.27: MYVALUES Dataset View - General Object Info

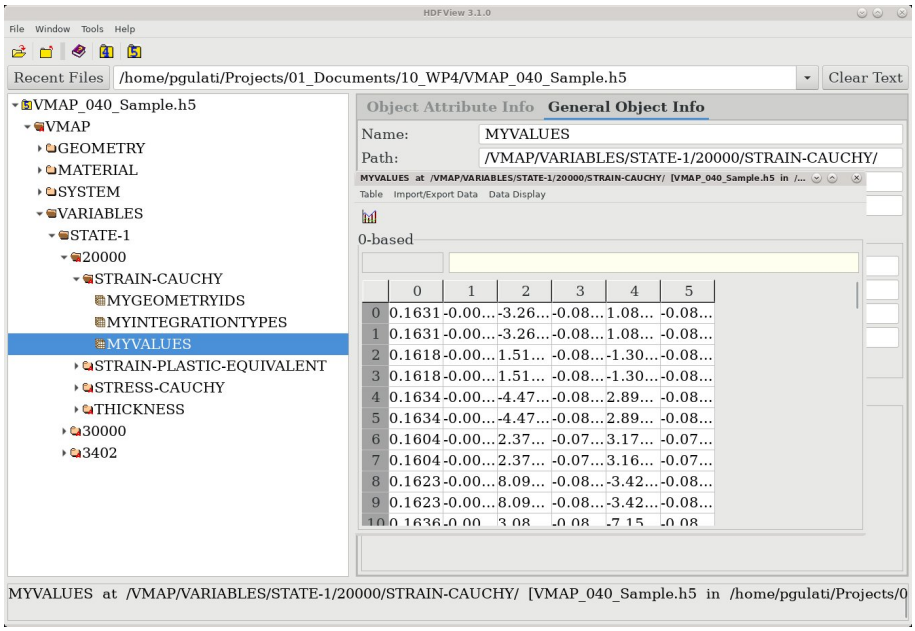


Figure 7.28: MYVALUES Dataset View - Metadata

7.10.3 MYINTEGRATIONTYPES Dataset

This dataset exists only when the MYLOCATION Attribute in Fig. 7.23 is 4.

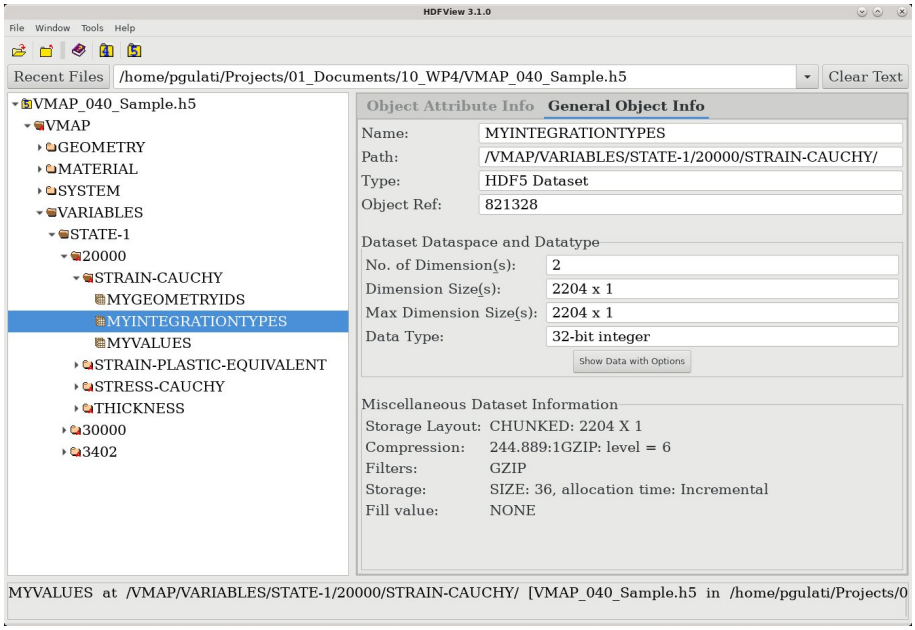


Figure 7.29: MYINTEGRATIONTYPES Dataset View - General Object Info

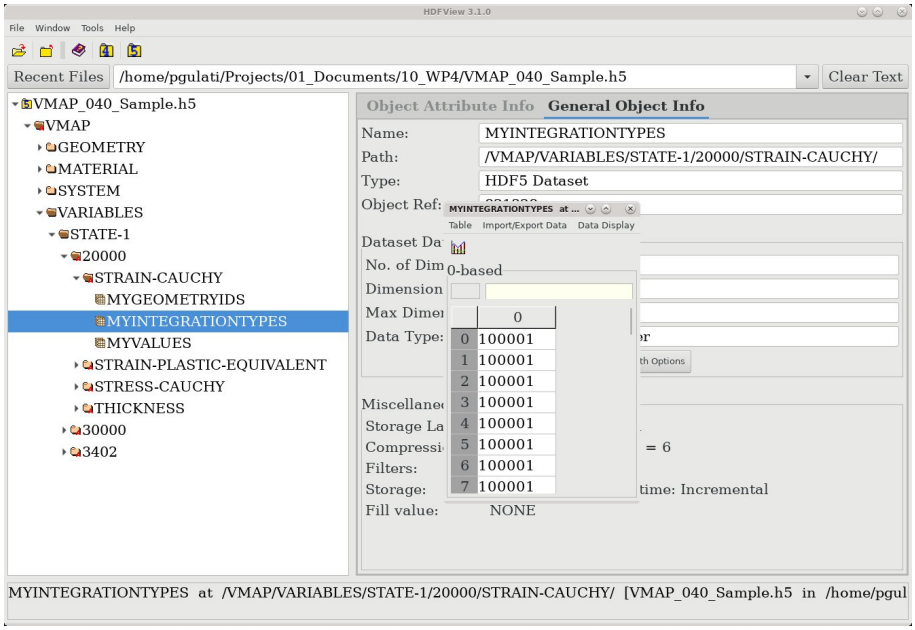


Figure 7.30: MYINTEGRATIONTYPES Dataset View - Metadata

7.11 SYSTEM Group View

SYSTEM Group has no attributes.

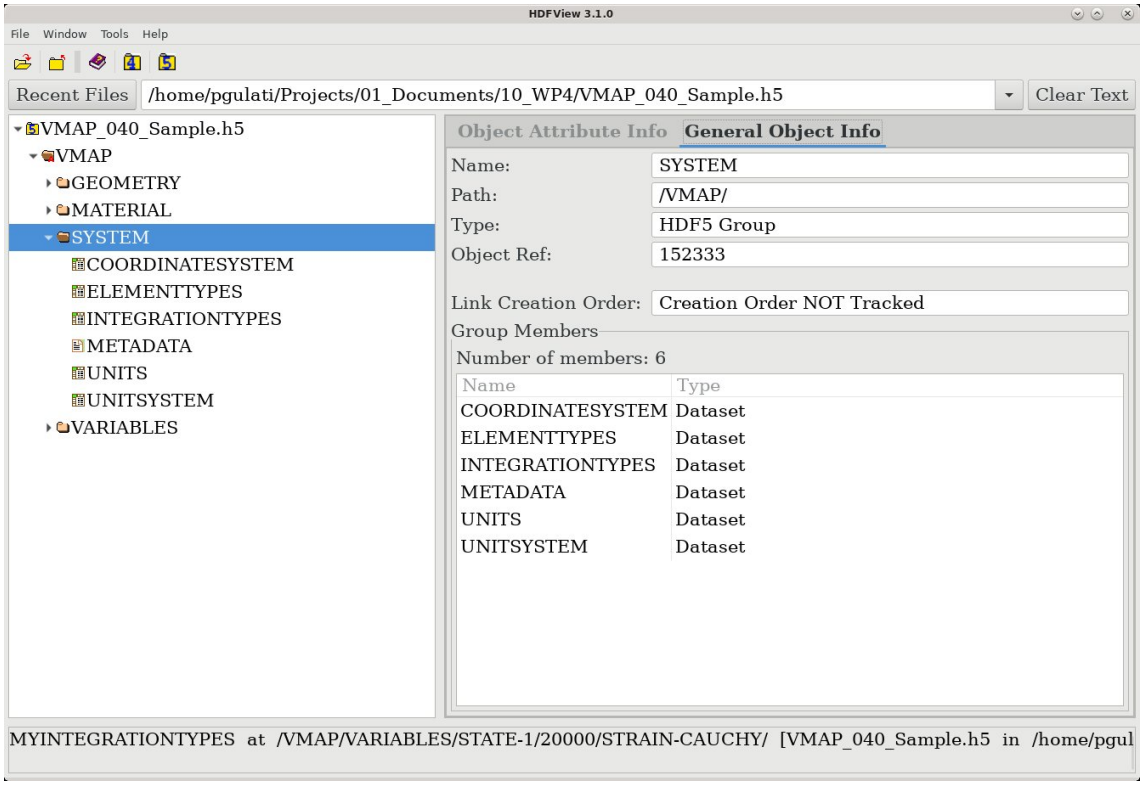


Figure 7.31: SYSTEM Group View - General Object Info

7.11.1 COORDINATESYSTEM Dataset

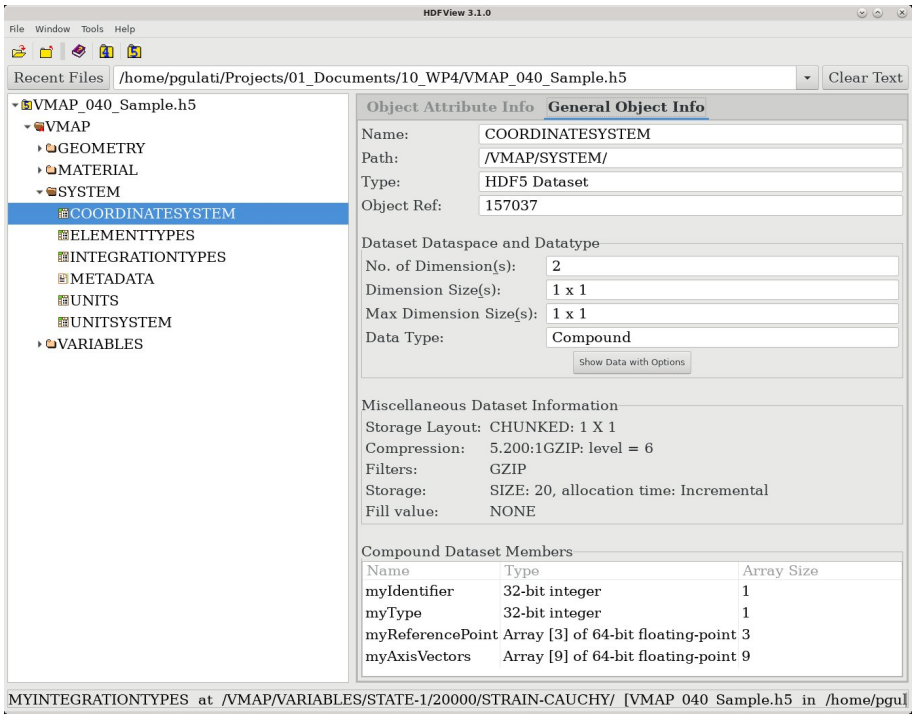


Figure 7.32: COORDINATESYSTEM Dataset View - General Object Info

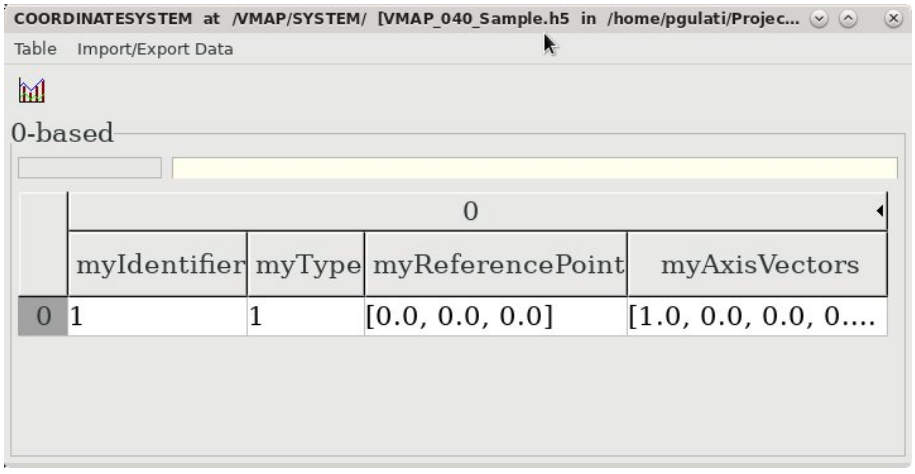


Figure 7.33: COORDINATESYSTEM Dataset View - Metadata

7.11.2 ELEMENTTYPES Dataset

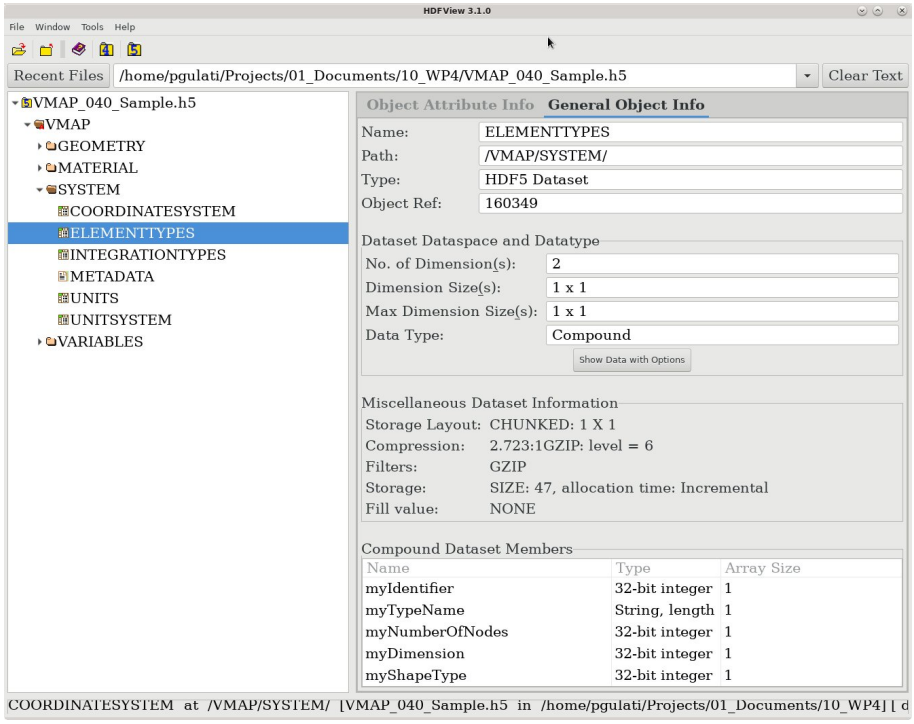


Figure 7.34: ELEMENTTYPES Dataset View - General Object Info

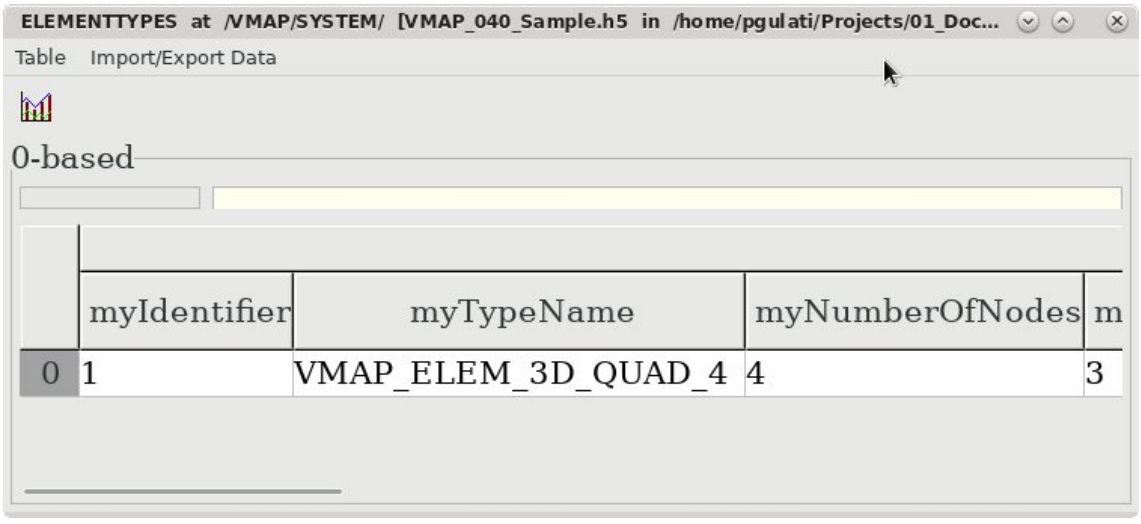


Figure 7.35: ELEMENTTYPES Dataset View - Metadata

7.11.3 INTEGRATIONTYPES Dataset

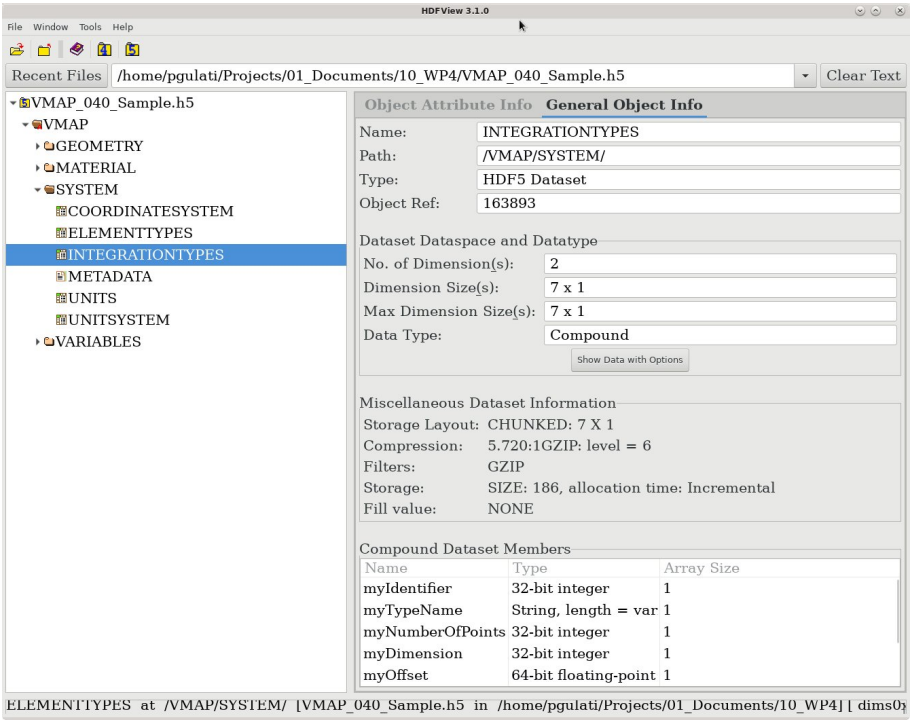


Figure 7.36: INTEGRATIONTYPES Dataset View - General Object Info

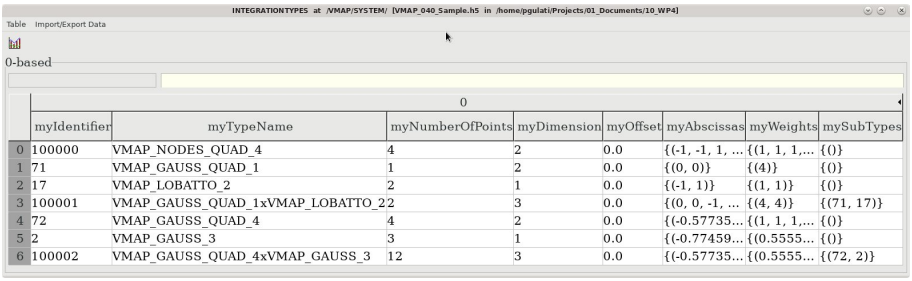


Figure 7.37: INTEGRATIONTYPES Dataset View - Metadata

7.11.4 METADATA Dataset

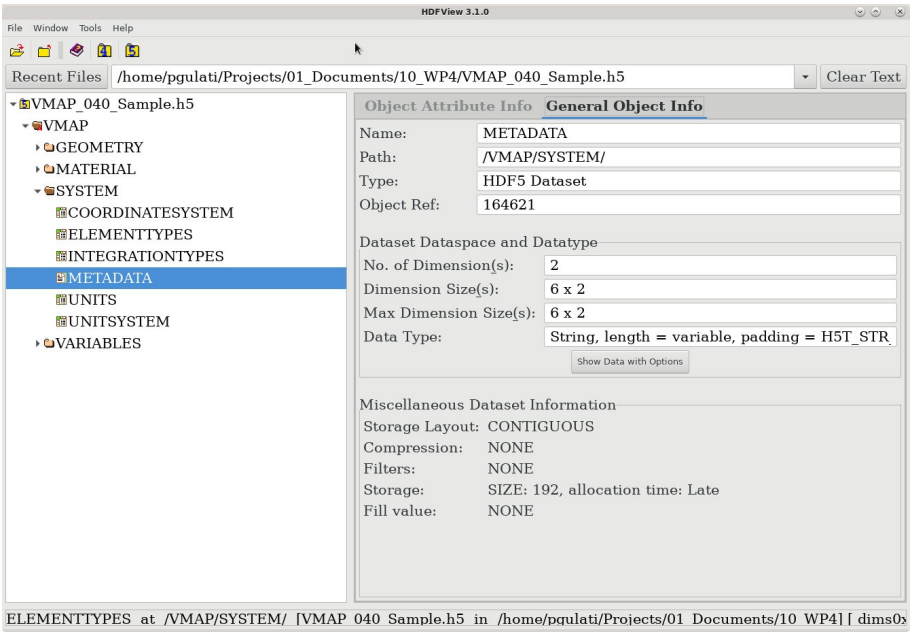
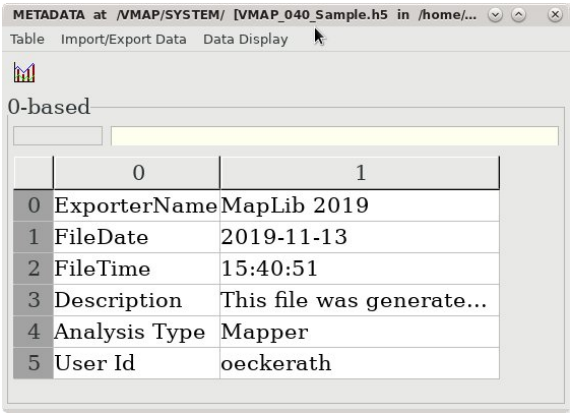


Figure 7.38: METADATA Dataset View - General Object Info



	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

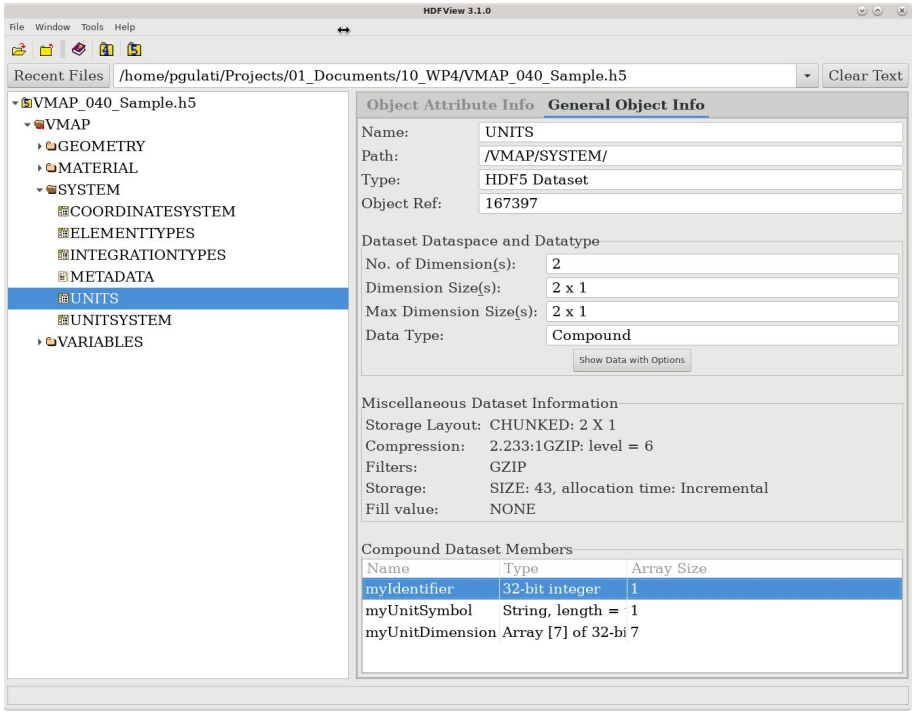


Figure 7.40: UNITS Dataset View - General Object Info

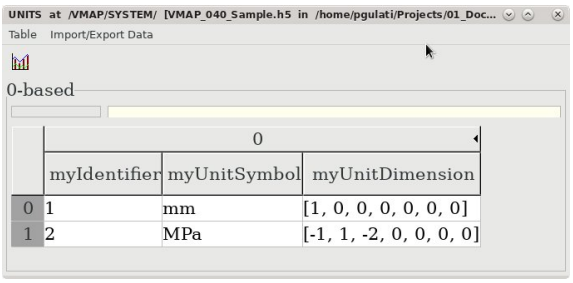


Figure 7.41: UNITS Dataset View - Metadata

7.11.6 UNITSYSTEM Dataset

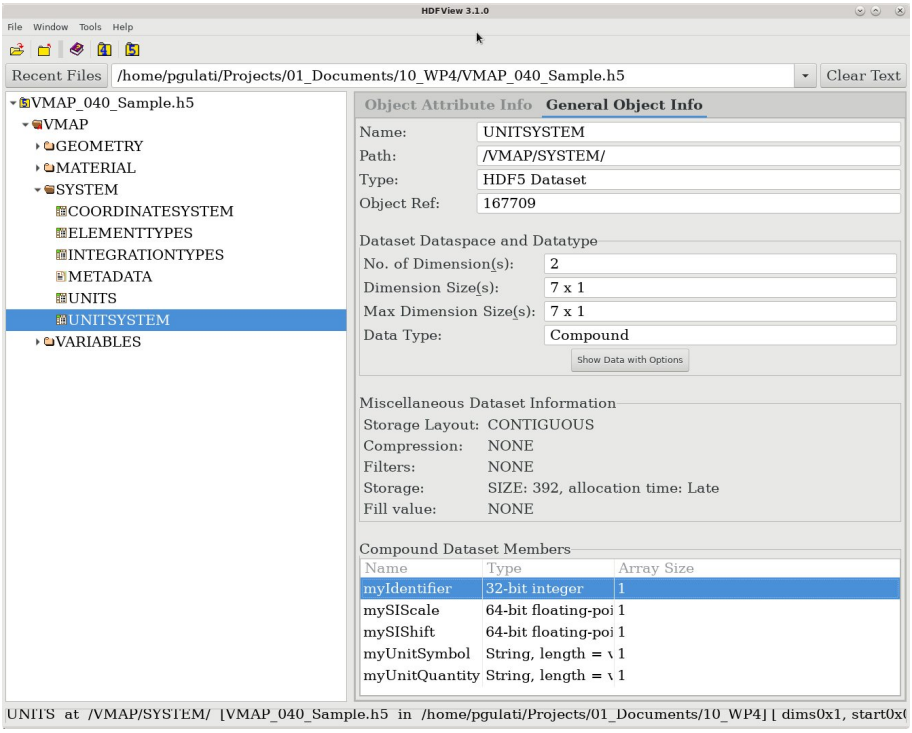
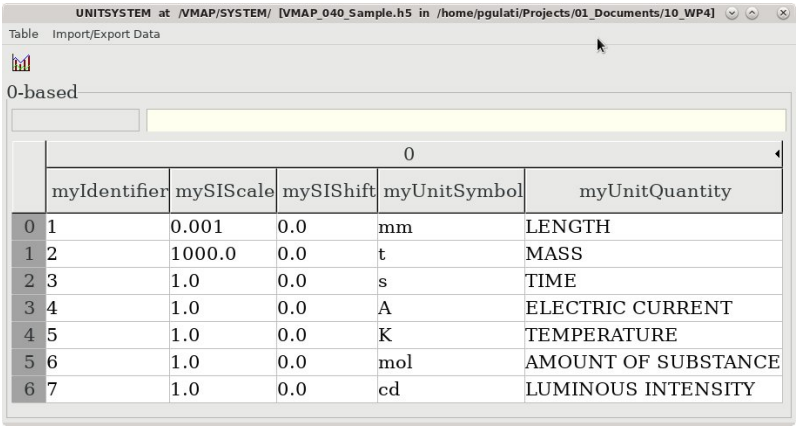


Figure 7.42: UNITSYSTEM Dataset View - General Object Info



	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

Using an example to demonstrate, a user-defined element is explained. First step is to define 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.

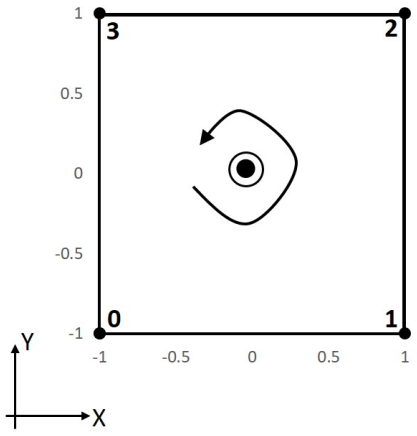


Figure 8.1: Point Ordering for a QUAD Element

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; ③
        faceconnectivity[3] = 1;
        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.

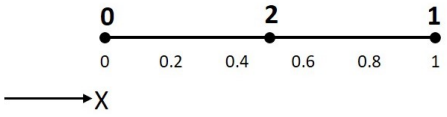


Figure 8.4: LINE_3 Element

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.

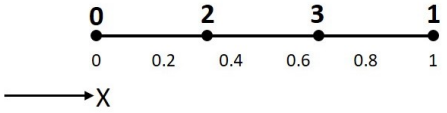
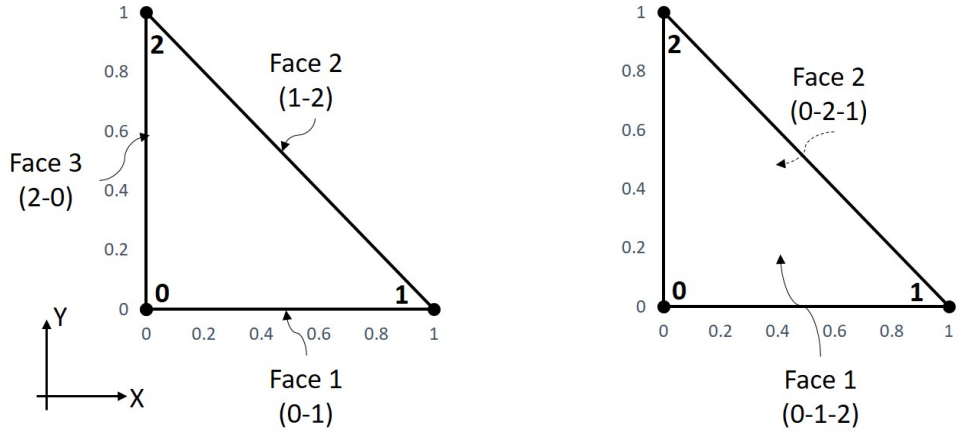


Figure 8.5: LINE_4 Element

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.



(a) 2D Element (b) 3D Element

Figure 8.6: TRIANGLE_3

8.7 TRIANGLE_4

TRIANGLE_4 2D has the same parameterization as shown for TRIANGLE_3 2D. The 3D element has an additional center point. The faces are given below.
Face 1: 0-1-2-3
Face 2: 0-2-1-3
Figures 8.7 shows TRIANGLE_4 element.

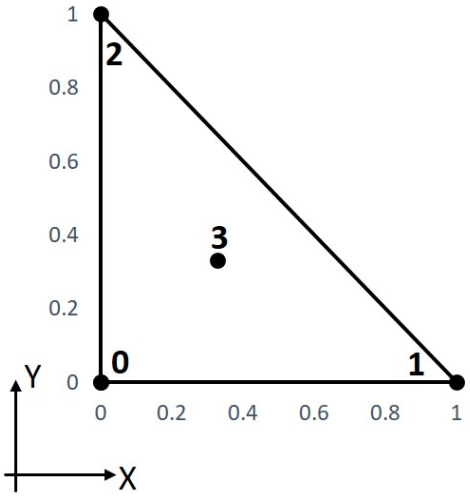
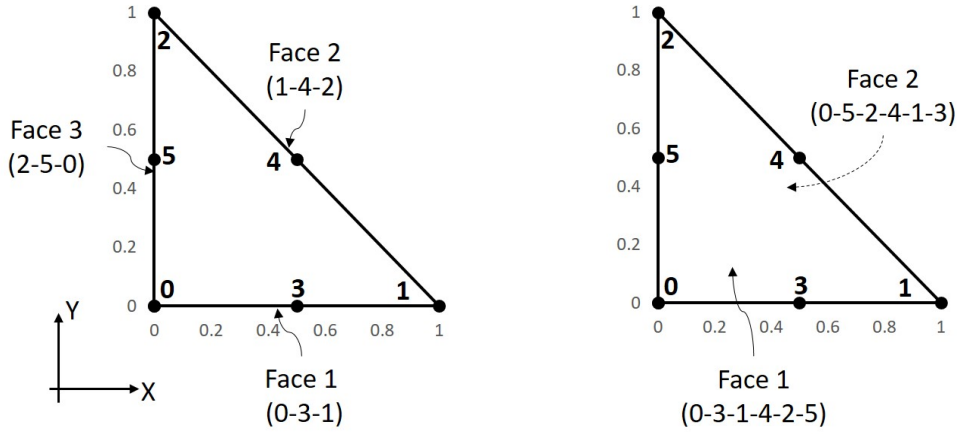


Figure 8.7: TRIANGLE_4 Element

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. Figures 8.8 shows 2D and 3D TRIANGLE_6 elements.

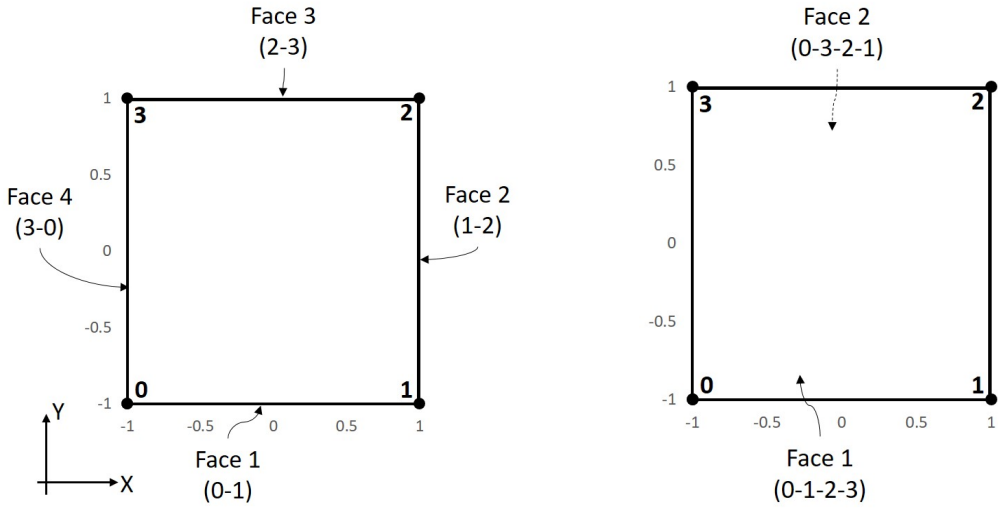


(a) 2D Element (b) 3D Element

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.

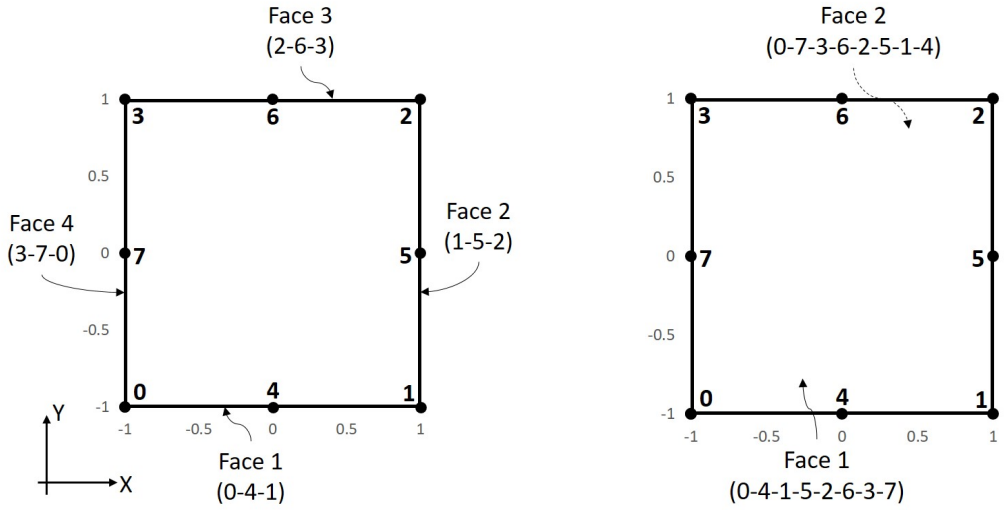


(a) 2D Element (b) 3D Element

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.



(a) 2D Element (b) 3D Element

Figure 8.10: QUAD_8

8.11 QUAD_9

QUAD_9 2D has the same parameterization as shown for QUAD_8 2D. The 3D element has an additional center point. The faces are given below.
Face 1: 0-4-1-5-2-6-3-7-8
Face 2: 0-7-3-6-2-5-1-4-8
Figures 8.11 shows QUAD_9 element.

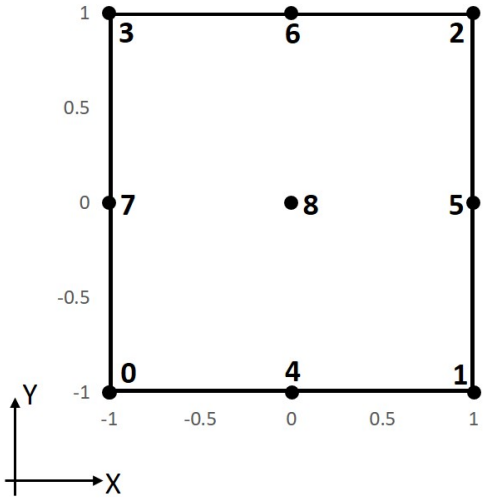


Figure 8.11: QUAD_9 Element

8.12 TETRAHEDRON_4

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

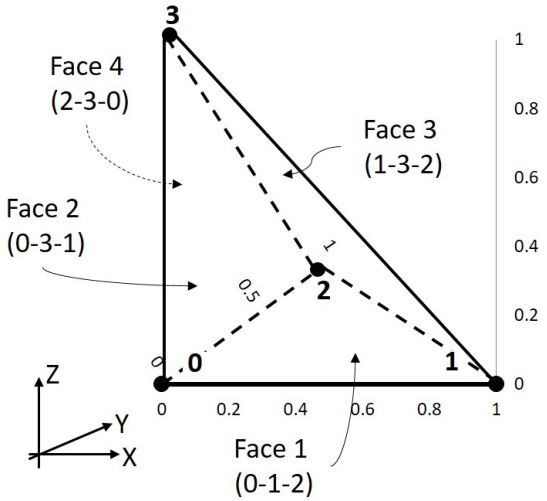


Figure 8.12: TETRAHEDRON_4 Element

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.

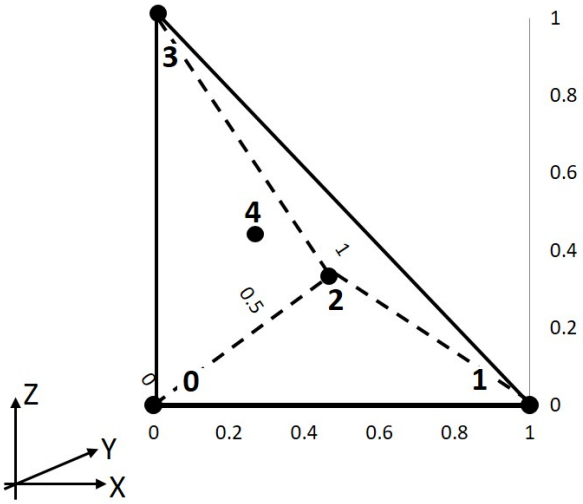


Figure 8.13: TETRAHEDRON_4 Element

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.

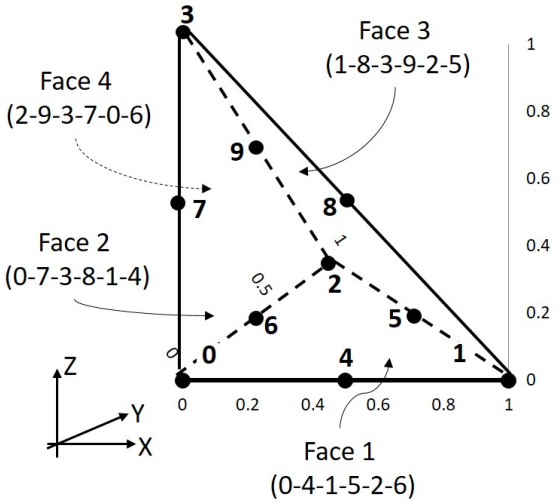


Figure 8.14: TETRAHEDRON_10 Element

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.

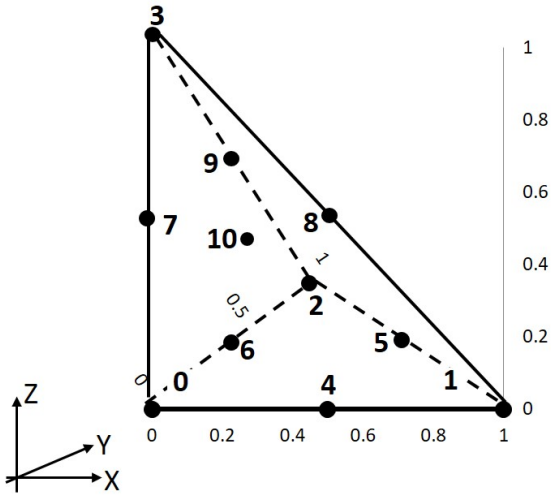


Figure 8.15: TETRAHEDRON_11 Element

8.16 PYRAMID_5

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

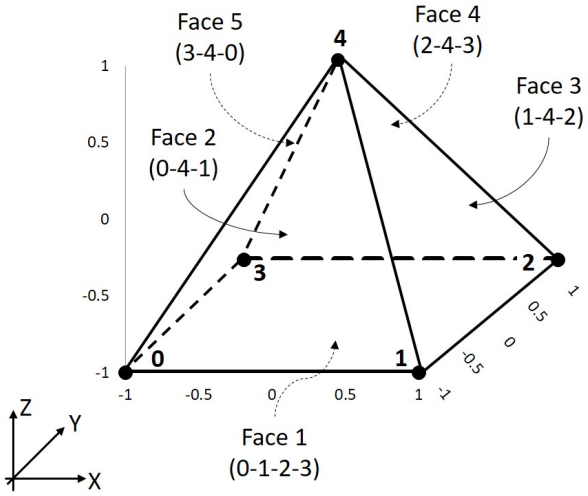


Figure 8.16: PYRAMID_5 Element

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.

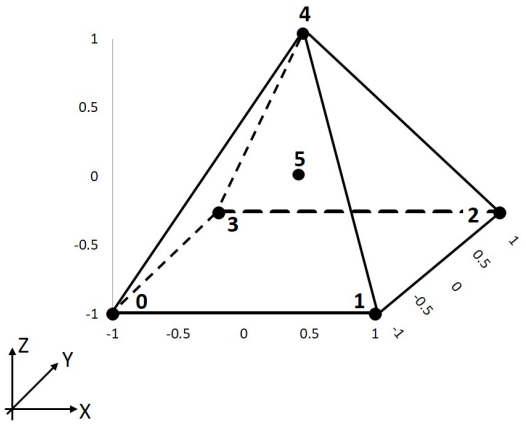


Figure 8.17: PYRAMID_6 Element

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.

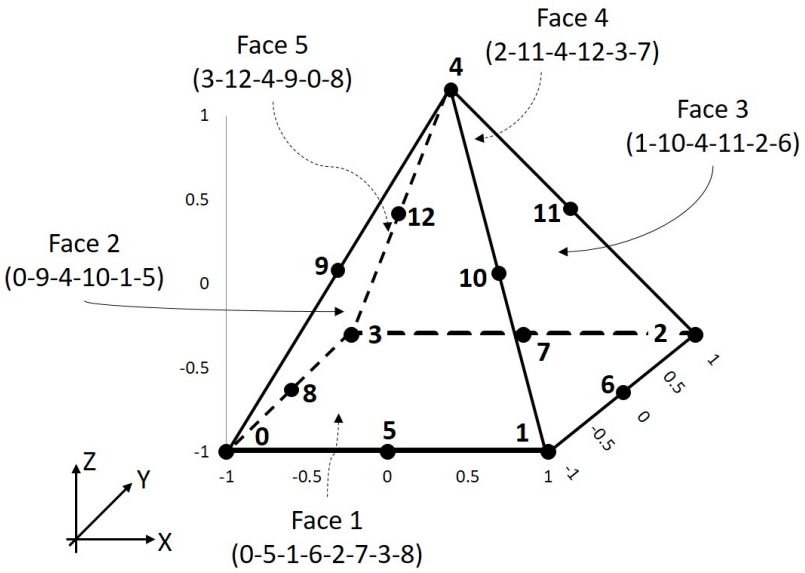


Figure 8.18: PYRAMID_13 Element

8.19 WEDGE_6

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

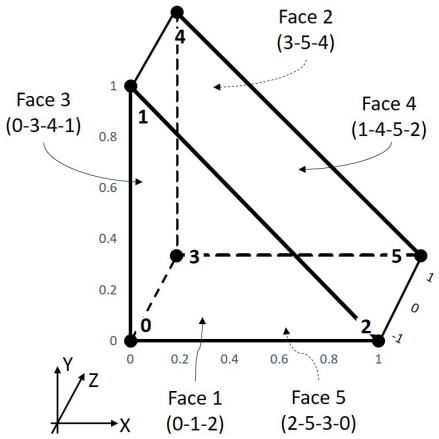


Figure 8.19: WEDGE_6 Element

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.

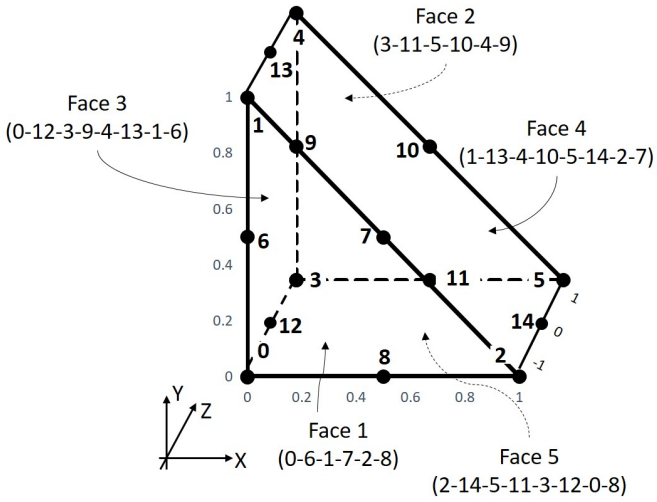


Figure 8.20: WEDGE_15 Element

8.21 HEXAHEDRON_8

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

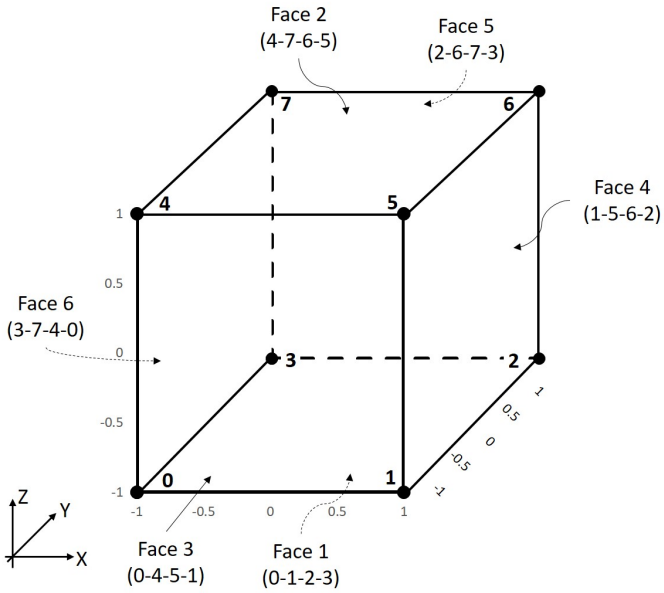


Figure 8.21: HEXAHEDRON_8 Element

8.22 HEXAHEDRON_9

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

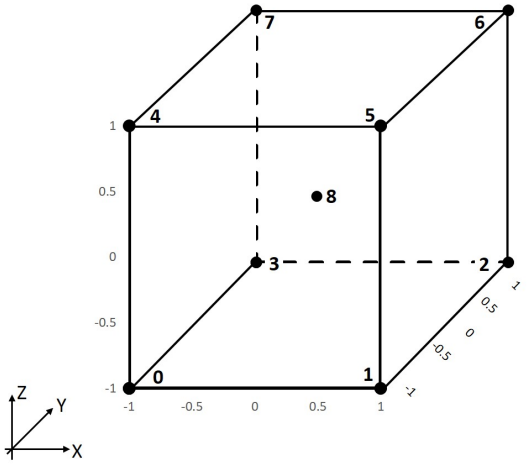


Figure 8.22: HEXAHEDRON_9 Element

8.23 HEXAHEDRON_20

HEXAHEDRON_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.

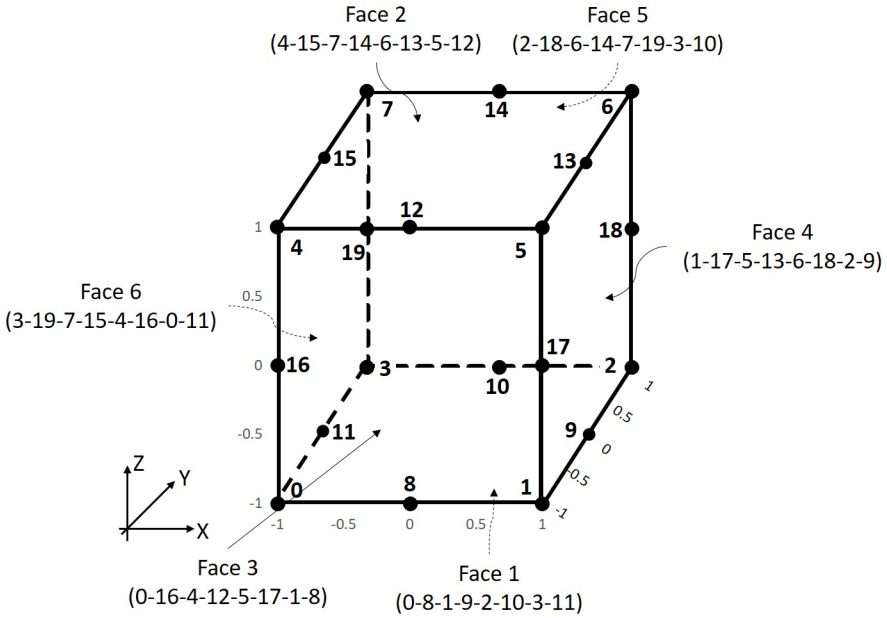


Figure 8.23: HEXAHEDRON_20 Element

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.

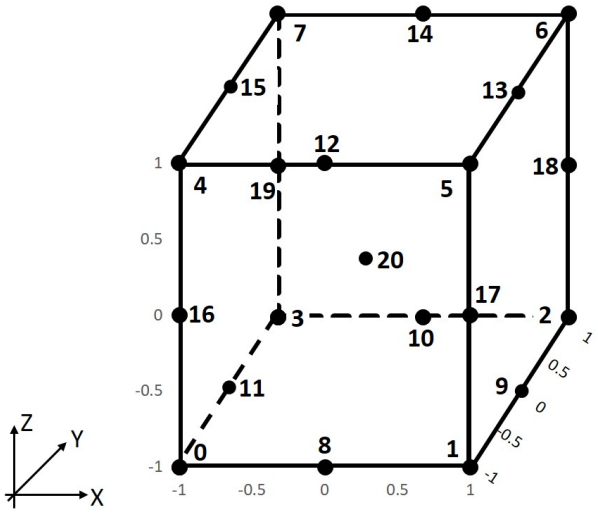


Figure 8.24: HEXAHEDRON_21 Element

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. The faces are listed below.
Face 1: 0-8-1-9-2-10-3-11-21
Face 2: 4-15-7-14-6-13-5-12-22
Face 3: 0-16-4-12-5-17-1-8-23
Face 4: 1-17-5-13-6-18-2-9-24
Face 5: 2-18-6-14-7-19-3-10-25
Face 6: 3-19-7-15-4-16-0-11-26
Figure 8.25 shows a HEXAHEDRON_27 element.

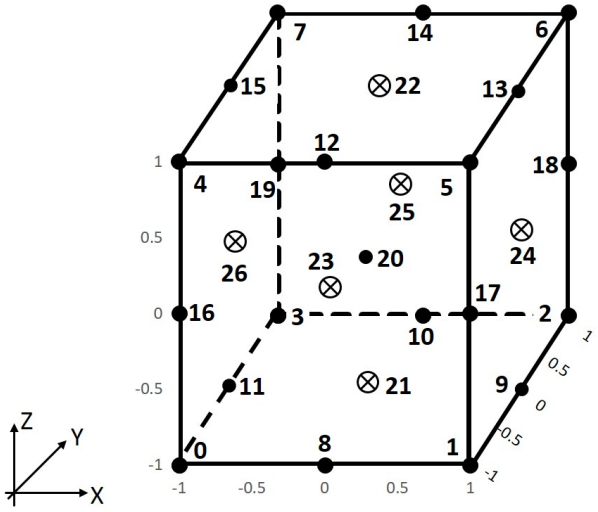


Figure 8.25: HEXAHEDRON_27 Element

8.26 POLYGON

Since polygons with an arbitrary number of vertices are supported, the number of nodes has to be provided by the user when constructing a polygon.
In VMAP there is only one polygon type for elements with different numbers of nodes, i.e. a polygon is only described by the length of the connectivity in the sElement structure.

The basic type is specified as a shape type using the parameter sElementType::POLYGON in the specification of the element type. For example, for a 3D element with constant interpolation (like a CFD face), this can be created using the ElementTypeFactory as follows:

$$VMAPElementTypeFactory :: createVMAPElementType(sElementType :: ELEM_3D, sElementType :: POLYGON, sElementType :: CONSTANT)$$

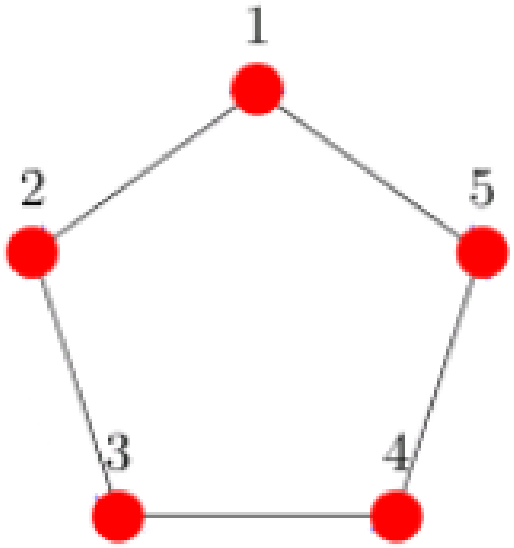


Figure 8.26: POLYGON Element

8.27 POLYHEDRON

Polyhedrons with an arbitrary number of polygonal shaped faces are the most complicated elements. A polyhedron is described in VMAP by a list of related polygons. The only basic type is the parameter sElement::POLYHEDRON as the shape type. If a polyhedron consists of N faces, each face requires a definition of the connectivity of the describing polygon. The connectivity of the polygon must be chosen so that the resulting polygon normal points out of the polyhedron volume.

To describe the connectivity of a polygon, VMAP uses so-called indexed face sets. First, the number of polyhedron faces is specified, followed by the number of nodes per face. For a hexahedron, this results in an indexed face set of the form:

[6 4 4 4 4 4 4]

So 6 faces, with each face described by 4 nodes.

```
// the face table "nodeIDs" of length 31
int nodeIDs[31];
```

```
  // first seven entries: information about the polyhedron's faces
  nodeIDs[0] = 7;    //  nodes of first face start with nodeIDS[7]
  nodeIDs[1] = 11;   //  nodes of second face start with nodeIDs[11],...
  nodeIDs[2] = 15;
  nodeIDs[3] = 19;
  nodeIDs[4] = 23;
  nodeIDs[5] = 27;
  nodeIDs[6] = 31;
//  face table ends at position 31 and has 6 = nodeIDs[0] - 1 faces

  // face 0          // face 1
  nodeIDs[7]  = 0;   nodeIDs[11] = 0;
  nodeIDs[8]  = 1;   nodeIDs[12] = 3;
  nodeIDs[9]  = 2;   nodeIDs[13] = 7;
  nodeIDs[10] = 3;   nodeIDs[14] = 4;

  // face 2          // face 3
  nodeIDs[15] = 3;   nodeIDs[19] = 5;
  nodeIDs[16] = 2;   nodeIDs[20] = 6;
  nodeIDs[17] = 6;   nodeIDs[21] = 2;
  nodeIDs[18] = 7;   nodeIDs[22] = 1;

  // face 4          // face 5
  nodeIDs[23] = 0;   nodeIDs[27] = 4;
  nodeIDs[24] = 4;   nodeIDs[28] = 5;
  nodeIDs[25] = 5;   nodeIDs[29] = 6;
  nodeIDs[26] = 1;   nodeIDs[30] = 7;
```

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, an example is provided to illustrate. The numbering for combined user defined type starts from 100000. See figure 9.1.

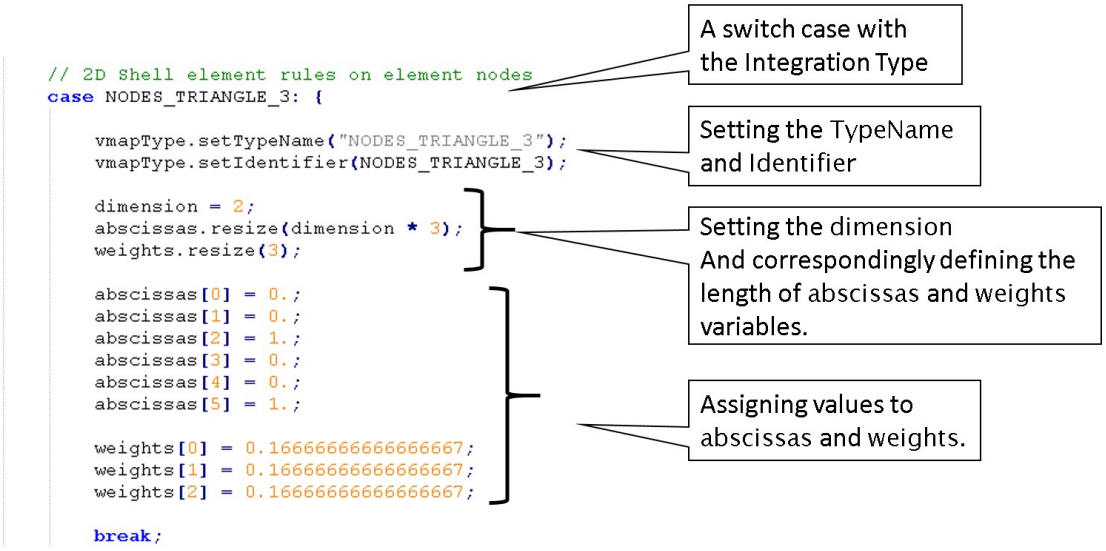


Figure 9.1: C++ Code for NODE_TRIANGLE_3 Integration Rule Definition

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	$\pm$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	$\pm$0.7745966692414834

Table 9.3: GAUSS_3

9.5 GAUSS_4

Number of Points - n	Weight - w_i	Abscissa - x_i
4	0.6521451548625461	$\pm$0.3399810435848563
	0.3478548451374538	$\pm$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	$\pm$0.5384693101056831
	0.2369268850561891	$\pm$0.9061798459386640

Table 9.5: GAUSS_5

9.7 GAUSS_6

Number of Points - n	Weight - w_i	Abscissa - x_i
6	0.3607615730481386	$\pm$0.6612093864662645
	0.4679139345726910	$\pm$0.2386191860831969
	0.1713244923791704	$\pm$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	$\pm$0.4058451513773972
	0.2797053914892766	$\pm$0.7415311855993945
	0.1294849661688697	$\pm$0.9491079123427585

Table 9.7: GAUSS_7

9.9 GAUSS_8

Number of Points - n	Weight - w_i	Abscissa - x_i
8	0.3626837833783620	$\pm$0.1834346424956498
	0.3137066458778873	$\pm$0.5255324099163290
	0.2223810344533745	$\pm$0.7966664774136267
	0.1012285362903763	$\pm$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	$\pm$0.8360311073266358
	0.0812743883615744	$\pm$0.9681602395076261
	0.3123470770400029	$\pm$0.3242534234038089
	0.2606106964029354	$\pm$0.6133714327005904

Table 9.9: GAUSS_9

9.11 GAUSS_10

Number of Points - n	Weight - w_i	Abscissa - x_i
10	0.2955242247147529	$\pm$0.1488743389816312
	0.2692667193099963	$\pm$0.4333953941292472
	0.2190863625159820	$\pm$0.6794095682990244
	0.1494513491505806	$\pm$0.8650633666889845
	0.0666713443086881	$\pm$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	$\pm$0.1080549487073437
	0.2051984637212956	$\pm$0.3191123689278897
	0.1855383974779378	$\pm$0.5152486363581541
	0.1572031671581935	$\pm$0.6872929048116855
	0.1215185706879032	$\pm$0.8272013150697650
	0.0801580871597602	$\pm$0.9284348836635735
	0.0351194603317519	$\pm$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	$\pm$0.2011940939974345
	0.1861610000155622	$\pm$0.3941513470775634
	0.1662692058169939	$\pm$0.5709721726085388
	0.1395706779261543	$\pm$0.7244177313601701
	0.1071592204671719	$\pm$0.8482065834104272
	0.0703660474881081	$\pm$0.9372733924007060
	0.0307532419961173	$\pm$0.9879925180204854

Table 9.15: GAUSS_15

9.17 GAUSS_16

Number of Points - n	Weight - w_i	Abscissa - x_i
16	0.1894506104550685	$\pm$0.0950125098376374
	0.1826034150449236	$\pm$0.2816035507792589
	0.1691565193950025	$\pm$0.4580167776572274
	0.1495959888165767	$\pm$0.6178762444026438
	0.1246289712555339	$\pm$0.7554044083550030
	0.0951585116824928	$\pm$0.8656312023878318
	0.0622535239386479	$\pm$0.9445750230732326
	0.0271524594117541	$\pm$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	$\pm$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	$\pm$1.000000

Table 9.18: LOBATTO_3

9.21 LOBATTO_4

Number of Points - n	Weight - w_i	Abscissa - x_i
4	0.833333	$\pm$0.4472135954
	0.166667	$\pm$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	$\pm$0.6546536707
	0.100000	$\pm$1.000000

Table 9.20: LOBATTO_5

9.23 LOBATTO_6

Number of Points - n	Weight - w_i	Abscissa - x_i
6	0.5548583770	$\pm$0.2852315164
	0.3784749562	$\pm$0.7650553239
	0.066667	$\pm$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	$\pm$0.4688487934
	0.2768260473	$\pm$0.8302238962
	0.0476190476	$\pm$1.000000

Table 9.22: LOBATTO_7

9.25 LOBATTO_8

Number of Points - n	Weight - w_i	Abscissa - x_i
8	0.4124587946	$\pm$0.2092992179
	0.3411226924	$\pm$0.5917001814
	0.2107042271	$\pm$0.8717401485
	0.0357142857	$\pm$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	$\pm$0.3631174638
	0.2745387125	$\pm$0.6771862795
	0.1654953615	$\pm$0.8997579954
	0.0277777778	$\pm$1.000000

Table 9.24: LOBATTO_9

9.27 LOBATTO_10

Number of Points - n	Weight - w_i	Abscissa - x_i
10	0.3275397611	$\pm$0.1652789576
	0.2920426836	$\pm$0.4779249498
	0.2248893420	$\pm$0.7387738651
	0.1333059908	$\pm$0.9195339081
	0.022222	$\pm$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	$\pm$0.2957581355
	0.2480481042	$\pm$0.5652353269
	0.1871698817	$\pm$0.7844834736
	0.1096122732	$\pm$0.9340014304
	0.0181818	$\pm$1.000000

Table 9.26: LOBATTO_11

9.29 LOBATTO_12

Number of Points - n	Weight - w_i	Abscissa - x_i
12	0.271405241	$\pm$0.136552933
	0.251275603	$\pm$0.399530941
	0.212508418	$\pm$0.632876153
	0.157974705	$\pm$0.819279322
	0.09168452	$\pm$0.944899272
	0.015151515	$\pm$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	$\pm$0.10132627
	0.193690024	$\pm$0.29983047
	0.177491913	$\pm$0.48605942
	0.154026981	$\pm$0.65238870
	0.124255382	$\pm$0.79200829
	0.089393697	$\pm$0.89920053
	0.050850361	$\pm$0.96956805
	0.008333333	$\pm$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	$\pm$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	$\pm$0.500000
	0.166667	$\pm$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	$\pm$0.333337
	0.444444	$\pm$0.666667
	0.111111	$\pm$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	$\pm$0.25
	0.166667	$\pm$0.50
	0.333333	$\pm$0.75
	0.083333	$\pm$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	$\pm$0.20
	0.266667	$\pm$0.40
	0.133333	$\pm$0.60
	0.266667	$\pm$0.80
	0.066667	$\pm$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	$\pm$0.166667
	0.111111	$\pm$0.333333
	0.222222	$\pm$0.50
	0.111111	$\pm$0.666667
	0.222222	$\pm$0.833333
	0.055555	$\pm$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	$\pm$0.142857
	0.190476	$\pm$0.285714
	0.095238	$\pm$0.428571
	0.190476	$\pm$0.571428
	0.095238	$\pm$0.714285
	0.190476	$\pm$0.857142
	0.047619	$\pm$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	$\pm$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	$\pm$1.0

Table 9.40: TRAPEZOIDAL_3

9.45 TRAPEZOIDAL_4

Number of Points - n	Weight - w_i	Abscissa - x_i
4	0.666667	$\pm$0.333333
	0.333333	$\pm$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	$\pm$0.5
	0.25	$\pm$1.0

Table 9.42: TRAPEZOIDAL_5

9.47 TRAPEZOIDAL_6

Number of Points - n	Weight - w_i	Abscissa - x_i
6	0.4	$\pm$0.199999
	0.4	$\pm$0.6
	0.2	$\pm$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	$\pm$0.333333
	0.333333	$\pm$0.666667
	0.166667	$\pm$1.0

Table 9.44: TRAPEZOIDAL_7

9.49 TRAPEZOIDAL_8

Number of Points - n	Weight - w_i	Abscissa - x_i
8	0.285714	$\pm$0.142857
	0.285714	$\pm$0.428571
	0.285714	$\pm$0.714285
	0.142857	$\pm$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	$\pm$0.25
	0.25	$\pm$0.50
	0.25	$\pm$0.75
	0.125	$\pm$1.0

Table 9.46: TRAPEZOIDAL_9

9.51 TRAPEZOIDAL_10

Number of Points - n	Weight - w_i	Abscissa - x_i
10	0.222222	$\pm$0.111111
	0.222222	$\pm$0.333333
	0.222222	$\pm$0.555556
	0.222222	$\pm$0.777778
	0.111111	$\pm$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	$\pm$0.20
	0.2	$\pm$0.40
	0.2	$\pm$0.60
	0.2	$\pm$0.80
	0.1	$\pm$1.0

Table 9.48: TRAPEZOIDAL_11

9.53 TRAPEZOIDAL_12

Number of Points - n	Weight - w_i	Abscissa - x_i
12	0.181818	$\pm$0.09090909
	0.181818	$\pm$0.272727273
	0.181818	$\pm$0.454545455
	0.181818	$\pm$0.636363636
	0.181818	$\pm$0.818181818
	0.090909	$\pm$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	$\pm$0.16666667
	0.166667	$\pm$0.33333333
	0.166667	$\pm$0.5
	0.166667	$\pm$0.66666667
	0.166667	$\pm$0.83333333
	0.083333	$\pm$1.0

Table 9.50: TRAPEZOIDAL_13

9.55 TRAPEZOIDAL_14

Number of Points - n	Weight - w_i	Abscissa - x_i
14	0.153846	$\pm$0.076923
	0.153846	$\pm$0.230769
	0.153846	$\pm$0.384615
	0.153846	$\pm$0.538461
	0.153846	$\pm$0.692307
	0.153846	$\pm$0.846153
	0.076923	$\pm$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	$\pm$0.142857
	0.14285714285714285	$\pm$0.285714
	0.14285714285714285	$\pm$0.428571
	0.14285714285714285	$\pm$0.571428
	0.14285714285714285	$\pm$0.714285
	0.14285714285714285	$\pm$0.857142
	0.07142857142857142	$\pm$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	-0.333333	-1.000000	-1.000000
	-0.333333	1.000000	-1.000000
	-0.333333	1.000000	1.000000
	-0.333333	-1.000000	1.000000
	1.333333	0.000000	-1.000000
	1.333333	1.000000	0.000000
	1.333333	0.000000	1.000000
	1.333333	-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_1_LAYERED_HEXAHEDRON_1

This is a 3D integration for layered solid element with GAUSS_QUAD_1 in-plane and stacking in x-direction.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
1	4	0.0	0.0	0.0

Table 9.65: GAUSS_1_LAYERED_HEXAHEDRON_1

9.70 GAUSS_1_LAYERED_HEXAHEDRON_2

This is a 3D integration for layered solid element with GAUSS_QUAD_1 in-plane and stacking in y-direction.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
1	4	0.0	0.0	0.0

Table 9.66: GAUSS_1_LAYERED_HEXAHEDRON_2

9.71 GAUSS_1_LAYERED_HEXAHEDRON_3

This is a 3D integration for layered solid element with GAUSS_QUAD_1 in-plane and stacking in z-direction.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
1	4	0.0	0.0	0.0

Table 9.67: GAUSS_1_LAYERED_HEXAHEDRON_3

9.72 GAUSS_4_LAYERED_HEXAHEDRON_1

This is a 3D integration for layered solid element with GAUSS_QUAD_4 in-plane and stacking in x-direction.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
4	1	0.0	-0.5773502691896257	-0.5773502691896257
	1	0.0	0.5773502691896257	-0.5773502691896257
	1	0.0	-0.5773502691896257	0.5773502691896257
	1	0.0	0.5773502691896257	0.5773502691896257

Table 9.68: GAUSS_4_LAYERED_HEXAHEDRON_1

9.73 GAUSS_4_LAYERED_HEXAHEDRON_2

This is a 3D integration for layered solid element with GAUSS_QUAD_4 in-plane and stacking in y-direction.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
4	1	-0.5773502691896257	0.0	-0.5773502691896257
	1	0.5773502691896257	0.0	-0.5773502691896257
	1	-0.5773502691896257	0.0	0.5773502691896257
	1	0.5773502691896257	0.0	0.5773502691896257

Table 9.69: GAUSS_4_LAYERED_HEXAHEDRON_2

9.74 GAUSS_4_LAYERED_HEXAHEDRON_3

This is a 3D integration for layered solid element with GAUSS_QUAD_4 in-plane and stacking in z-direction.

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
4	1	-0.5773502691896257	-0.5773502691896257	0.0
	1	0.5773502691896257	-0.5773502691896257	0.0
	1	-0.5773502691896257	0.5773502691896257	0.0
	1	0.5773502691896257	0.5773502691896257	0.0

Table 9.70: GAUSS_4_LAYERED_HEXAHEDRON_3

9.75 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 9.71: GAUSS_TETRAHEDRON_1

9.76 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 9.72: GAUSS_TETRAHEDRON_4

9.77 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 9.73: GAUSS_TETRAHEDRON_8

9.78 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 9.74: GAUSS_TETRAHEDRON_11

9.79 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 9.75: GAUSS_TETRAHEDRON_15

9.80 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 9.76: GAUSS_PYRAMID_1

9.81 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 9.77: GAUSS_PYRAMID_5

9.82 GAUSS_PYRAMID_9

Number of Points - n	Weight - w_i	Abscissa - x_i	Abscissa - y_i	Abscissa - z_i
9	0.183429925248	-0.526421704396	∓ 0.526421704396	0.087476609247
	0.140354060819	-0.335885351395	∓ 0.335885351395	0.420881747524
	0.140354060819	0.335885351395	± 0.335885351395	0.420881747524
	0.183429925248	0.526421704396	± 0.526421704396	0.087476609247
	0.038197389067	0.0	0.0	0.860272730596

Table 9.78: GAUSS_PYRAMID_9

9.83 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 9.79: GAUSS_WEDGE_1

9.84 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 9.80: GAUSS_WEDGE_2

9.85 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 9.81: GAUSS_WEDGE_6

9.86 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 9.82: GAUSS_WEDGE_8

9.87 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 9.83: GAUSS_WEDGE_9

9.88 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 9.84: GAUSS_WEDGE_18

9.89 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 9.85: GAUSS_HEXAHEDRON_1

9.90 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
	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 9.86: GAUSS_HEXAHEDRON_8

9.91 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 9.87: GAUSS_HEXAHEDRON_8

9.92 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 9.88: NODES_TETRAHEDRON_4

9.93 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 9.89: NODES_TETRAHEDRON_10

9.94 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 9.90: NODES_WEDGE_6

9.95 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 9.91: NODES_WEDGE_15

9.96 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 9.92: NODES_PYRAMID_5

9.97 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 9.93: NODES_HEXAHEDRON_8

9.98 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 9.94: NODES_HEXAHEDRON_20

9.99 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 9.95: NODES_HEXAHEDRON_27

9.100 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:

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

The numbering for combined type starts from 100000.

9.101 Composite Integration Types

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

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

Chapter 10

Variable Specifications

This chapter will present the syntaxes of some of the commonly used variables from the current use cases. Each of the variable defined here should be represented as is in the VMAP Files. The sub-types for the variables are optional and may be used if needed.

10.1 A

AREAL-DENSITY	Definition: Areal density of a 2D object is defined as mass per unit area. Dimension: ML^{-2} , SI Unit: $kg \cdot m^{-2}$ Subtypes defined in VMAP: FIBRE Nomenclature: AREAL-DENSITY, AREAL-DENSITY_FIBRE
ABSOLUTE-HUMIDITY	Definition: Measures the weight of water vapour per unit volume of air. Dimension: ML^{-3} , SI Unit: $g \cdot m^{-3}$ Subtypes defined in VMAP: – Nomenclature: ABSOLUTE-HUMIDITY
AXIAL-STIFFNESS	Definition: It is the extent to which an object resists deformation in response to an applied force (tension or compression). Dimension: MT^{-2} , SI Unit: $N \cdot m^{-1}$ Subtypes defined in VMAP: – Nomenclature: AXIAL-STIFFNESS

10.2 B

10.3 C

COEFFICIENT-OF-THERMAL-EXPANSION	Definition: Measures the fraction change in size of a material per degree change in temperature at a constant pressure. Dimension: Θ^{-1} , SI Unit: K^{-1} Subtypes defined in VMAP: PLY Nomenclature: COEFFICIENT-OF-THERMAL-EXPANSION, COEFFICIENT-OF-THERMAL-EXPANSION_PLY
COORDINATESYSTEM-LOCAL	Definition: COORDINATESYSTEM-LOCAL can be used to define element specific local coordinate systems. Dimension: – , SI Unit: – Subtypes defined in VMAP: – Nomenclature: COORDINATESYSTEM-LOCAL,
CRYSTALLINITY	Definition: Refers to degree of structural order in a solid. Same as degree of crystallinity. Measured as a ratio of melting enthalpy to the literature value of completely crystalline material. Dimension: – , SI Unit: $J \cdot g^{-1}/J \cdot g^{-1}$ Subtypes defined in VMAP: – Nomenclature: CRYSTALLINITY

10.4 D

DAMAGE	In fatigue analysis, when failure occurs damage is equal to 1. Dimension: – , SI Unit: $Nr.ofCycles/Nr.ofCycles$ Subtypes defined in VMAP: – Nomenclature: DAMAGE
DEGREE-OF-CURE	Definition: It describes the conversion rate achieved during crosslinking reactions (curing). Ranges from 0 - no crosslinking to 1 - fully crosslinked. In case of different convention, please specify in MYVARIABLEDESCRIPTION attribute 6.9.10

	Dimension: – , SI Unit: J/J Subtypes defined in VMAP: – Nomenclature: DEGREE-OF-CURE
DEGREE-OF-SHRINKAGE	Definition: It is defined as the decrease in the volume of material. It is expressed as percentage of original volume of material to the volume at the SHRINKAGE-LIMIT. Dimension: – , SI Unit: – Subtypes defined in VMAP: LINEAR, VOLUMETRIC Nomenclature: DEGREE-OF-SHRINKAGE_LINEAR, DEGREE-OF-SHRINKAGE_VOLUMETRIC
DENSITY	Definition: Defined as the ratio between mass and volume, where volume is the function of temperature & pressure. Dimension: ML^{-3} , SI Unit: $kg \cdot m^{-3}$ Subtypes defined in VMAP: FIBRE, PLY, RESIN Nomenclature: DENSITY, DENSITY_FIBRE, DENSITY_PLY, DENSITY_RESIN
DIAMETER	Definition: Defined as the length of a straight line that passes through the center of the circle and whose end points lie on the circle. Bubble diameters are measured in injection moulding to calculate the defects. Dimension: L , SI Unit: m Subtypes defined in VMAP: BUBBLE Nomenclature: DIAMETER, DIAMETER_BUBBLE
DIFFUSION-PERMEABILITY	Definition: It is the ability of a porous material to allow fluids to pass through it. Same as flow permeability. Dimension: L^2 ; SI Unit: m^2 Subtypes defined in VMAP: PLY Nomenclature: DIFFUSION-PERMEABILITY DIFFUSION-PERMEABILITY_PLY
DIRECTION	Definition: It is the direction of a vector. Dimension: – , SI Unit: - Subtypes defined in VMAP: FIBRE

	Nomenclature: DIRECTION, DIRECTION_FIBRE
DISPLACEMENT	Definition: Distance moved by an entity. Same as distortion. Dimension: L , SI Unit: m Subtypes defined in VMAP: – Nomenclature: DISPLACEMENT
DYNAMIC-VISCOSITY	Definition: It is the tangential force per unit area required to move one horizontal plane with respect to another plane - at a unit velocity - when maintaining a unit distance apart in the fluid. Dimension:$ML^{-1}T^{-1}$, SI Unit:$kg \cdot (m \cdot s)^{-1}$ Subtypes defined in VMAP: PLY, RESIN Nomenclature: DYNAMIC-VISCOSITY, DYNAMIC-VISCOSITY_PLY, DYNAMIC-VISCOSITY_RESIN

10.5 E

EQ-PLASTIC-STRAIN	Definition: It describes degree of work hardening in a material. Dimension: – , SI Unit: $m \cdot m^{-1}$ Subtypes defined in VMAP: LN Nomenclature: EQ-PLASTIC-STRAIN EQ-PLASTIC-STRAIN_LN

10.6 F

10.7 G

GRAVITY	Definition: Also called gravitational acceleration. It is the free fall acceleration of an object in vacuum. Dimension: LT^{-2} , SI Unit: $m \cdot s^{-2}$ Subtypes defined in VMAP: – Nomenclature:

GRAVITY

10.8 H

HEAT-CAPACITY	Definition: It is the amount of heat supplied to a given mass of material to produce a unit change in temperature. Dimension: $L^2MT^{-2}\,\Theta^{-1}$, SI Unit: $J \cdot K^{-1}$ Subtypes defined in VMAP: FIBRE, PLY, RESIN Nomenclature: HEAT-CAPACITY, HEAT-CAPACITY_FIBRE, HEAT-CAPACITY_PLY, HEAT-CAPACITY_RESIN
HEAT-FLUX	Definition: It is the flow of energy per unit time. Based on the subtype SURFACE - flow of energy per unit area per unit time. Dimension: MT^{-3} , SI Unit: $W \cdot m^{-2}$ (for HEAT-FLUX_SURFACE) Subtypes defined in VMAP: SURFACE, VOLUME Nomenclature: HEAT-FLUX_SURFACE, HEAT-FLUX_VOLUME (SI Unit:$W \cdot m^{-3}$
HEAT-TRANSFER-COEFFICIENT	Definition: It is the proportionality constant between the heat flux and the temperature difference. Dimension: $MT^{-3}\Theta^{-1}$, SI Unit: $W \cdot (m^2 \cdot K)^{-1}$ Subtypes defined in VMAP: – Nomenclature: HEAT-TRANSFER-COEFFICIENT
HEIGHT	Definition: It is a measure of vertical distance. e.g. HEIGHT_LAYER parameter is generally used in additive manufacturing processes. Dimension:L , SI Unit:m Subtypes defined in VMAP: LAYER Nomenclature: HEIGHT, HEIGHT_LAYER

HILL-COEFFICIENT	Definition: It provides a measure for the cooperativity between the binding sites. Hill Coefficient, n>1 Positively cooperative binding, n<1 Negatively cooperative binding, n=1 Noncooperative (completely independent) binding. Dimension: – Subtypes defined in VMAP: – Nomenclature: HILL-COEFFICIENT

10.9 I

ID	Definition: Describes an identifier. Can be used with any user-defined subtypes as well. Dimension: – Subtypes defined in VMAP: PROPERTY Nomenclature: ID, ID_PROPERTY
IMIDIZATION	Definition: Percentage of imide present in a material. Same as Imidization degree. Dimension: – , SI Unit: $m^2 \cdot m^{-2}$ Subtypes defined in VMAP: – Nomenclature: IMIDIZATION
INFRARED-HEATING-ENERGY	Definition: It is a form of energy which is transferred because of a difference in temperature and is generated by an infrared heater. Infrared heating works by providing electromagnetic radiation with wavelengths between 780 nm and 1 mm. Dimension: L^2MT^{-2} , SI Unit: J Subtypes defined in VMAP: – Nomenclature: INFRARED-HEATING-ENERGY

10.10 J

10.11 K

KINEMATIC-VISCOSITY	Definition: It is the ratio of dynamic viscosity to density of the fluid. Dimension:L^2T^{-1} , SI Unit:$m^2 \cdot s^{-1}$ Subtypes defined in VMAP: PLY, RESIN Nomenclature: KINEMATIC-VISCOSITY, KINEMATIC-VISCOSITY_PLY, KINEMATIC-VISCOSITY_RESIN

10.12 L

LENGTH	Definition: Measure of distance between 2 points of an entity. Dimension:L , SI Unit:m Subtypes defined in VMAP: FIBRE Nomenclature: LENGTH_FIBRE
LINEAR-DENSITY	Definition: It is defined as mass per unit length. Dimension:ML^{-1} , SI Unit:$kg \cdot m^{-1}$ Subtypes defined in VMAP: – Nomenclature: LINEAR-DENSITY

10.13 M

MODULUS-HARDENING	Definition: It is the slope of the stress versus strain curve after the point of yield of a material. Dimension: $ML^{-1}T^{-2}$, SI Unit:Pa Subtypes defined in VMAP: - Nomenclature: MODULUS-HARDENING
MODULUS-SHEAR	Definition: Also called modulus of rigidity, it is defined as the ratio of shear stress to shear strain. Dimension: $ML^{-1}T^{-2}$, SI Unit:Pa Subtypes defined in VMAP: - Nomenclature: MODULUS-SHEAR
MODULUS-YOUNGS	Definition: Also called the modulus of elasticity, it defines the stiffness of a solid material in the linear-elasticity domain.

Dimension:$ML^{-1}T^{-2}$, SI Unit:Pa
Subtypes defined in VMAP: -
Nomenclature:
MODULUS-YOUNGS

10.14 N

10.15 O

ORIENTATION

Definition: It describes a probability of the fiber direction and is a symmetric tensor.
Dimension: – , SI Unit: -
Subtypes defined in VMAP: FIBRE
Nomenclature:
ORIENTATION,
ORIENTATION_FIBRE

10.16 P

POISSONS-RATIO

Definition: It is the ratio of the relative contraction strain (transverse, lateral or radial strain) normal to the applied load - to the relative extension strain (or axial strain) in the direction of the applied load.
Dimension: – , SI Unit: $(m \cdot m^{-1}) \cdot (m \cdot m^{-1})^{-1}$
Subtypes defined in VMAP: FIBRE, PLY, RESIN
Nomenclature:
POISSONS-RATIO,
POISSONS-RATIO_FIBRE,
POISSONS-RATIO_PLY,
POISSONS-RATIO_RESIN

PRESSURE

Definition: It is the normal force per unit area exerted on a imaginary or real plane surface in a fluid or a gas. This is the Absolute pressure.
Dimension: $ML^{-1}T^{-2}$, SI Unit:Pa
Subtypes defined in VMAP: ABSOLUTE, HYDRO-STATIC, ATMOSPHERIC
Nomenclature:

PRESSURE_HYDROSTATIC - Also gauge pressure, HP on an object = DENSITY * GRAVITY * HEIGHT of Fluid above the object,
PRESSURE_ATMOSPHERIC - pressure in the surrounding air

10.17 Q

10.18 R

RELATIVE-HUMIDITY

Definition: It is the ratio of the amount of moisture in the air at a certain temperature to the maximum amount of moisture that the air can retain at the same temperature.
Dimension: –, SI Unit: –
Subtypes defined in VMAP: –
Nomenclature:
RELATIVE-HUMIDITY

ROTATIONAL-STIFFNESS

Definition: It is the extent to which an object resists rotational deformation in response to an applied moment.
Dimension: ML^2T^{-2} , SI Unit:$N \cdot m \cdot rad^{-1}$
Subtypes defined in VMAP: –
Nomenclature:
ROTATIONAL-STIFFNESS

10.19 S

SHRINKAGE-LIMIT

Definition: It is the water content where further loss of moisture will not result in any volume reduction of the material.
Dimension: –, SI Unit: –
Subtypes defined in VMAP: –
Nomenclature:
SHRINKAGE-LIMIT

SPECIFIC-HUMIDITY

Definition: It measures the weight of water vapour per unit of air.
Dimension: –, SI Unit: $g \cdot kg^{-1}$
Subtypes defined in VMAP: –
Nomenclature:
SPECIFIC-HUMIDITY

STIFFNESS	Definition: It is a 6 x 6 matrix. Dimension: –, SI Unit: – Subtypes defined in VMAP: – Nomenclature: STIFFNESS
STRAIN-TRUE	Definition: Also known as logarithmic strain. It is natural log of the engineering strain. Dimension: –, SI Unit: $m \cdot m^{-1}$ Subtypes defined in VMAP: – Nomenclature: STRAIN-TRUE
STRAIN-CAUCHY	Definition: For small strain theory, engineering strain and cauchy strain are synonymous. Dimension: –, SI Unit: $m \cdot m^{-1}$ Subtypes defined in VMAP: PRINCIPAL-MAX, PRINCIPAL-MID, PRINCIPAL-MIN Nomenclature: STRAIN-CAUCHY, STRAIN-CAUCHY_PRINCIPAL-MAX, STRAIN-CAUCHY_PRINCIPAL-MID, STRAIN-CAUCHY_PRINCIPAL-MIN
STRENGTH	Definition: It is the ability of a material to withstand an applied load without failure or plastic deformation. Dimension: $ML^{-1}T^{-2}$, SI Unit: MPa Subtypes defined in VMAP: TENSILE, YIELD Nomenclature: STRENGTH, STRENGTH_TENSILE - It is the maximum stress a material can withstand during stretching before breaking, STRENGTH_YIELD - It indicates teh limit of elastic behaviour and the beginning of plastic behaviour.
STRETCH	Definition: It is the ratio of final length to the original length. Dimension: –, SI Unit: $m \cdot m^{-1}$ Subtypes defined in VMAP: PRINCIPAL-1, PRINCIPAL-2, PRINCIPAL-DIR1, PRINCIPAL-DIR2 Nomenclature: STRETCH_PRINCIPAL-1 - calculated based on global coordinates, STRETCH_PRINCIPAL-2 - calculated based on global coordinates,

	STRETCH_PRINCIPAL-DIR1 - calculated based on local coordinates, STRETCH_PRINCIPAL-DIR2 - calculated based on local coordinates
STRESS-CAUCHY	Definition: It is a second order stress tensor, widely used for stress analysis in infinitesimal strain theory. Dimension: $ML^{-1}T^{-2}$, SI Unit: $N \cdot m^{-2}$ Subtypes defined in VMAP: PRINCIPAL-MAX, PRINCIPAL-MID, PRINCIPAL-MIN Nomenclature: STRESS-CAUCHY, STRESS-CAUCHY_PRINCIPAL-MAX, STRESS-CAUCHY_PRINCIPAL-MID, STRESS-CAUCHY_PRINCIPAL-MIN

10.20 T

TACK	Definition: Tack of a resin is defined as the property to form a bond immediately when in contact with another surface, and is usually referred to as cold tack because this occurs without the addition of temperature. Dimension: –, SI Unit: – Subtypes defined in VMAP: – Nomenclature: TACK
TEMPERATURE	Definition: It is a standard units of measurement of the kinetic energy of the atoms & molecules of the system. Dimension: Θ , SI Unit: K Subtypes defined in VMAP: – Nomenclature: TEMPERATURE
THERMAL-CONDUCTIVITY	It is the ability of the material to conduct heat. Dimension: $MLT^{-3}\Theta^{-1}$, SI Unit: $W \cdot (m \cdot K)^{-1}$ Subtypes defined in VMAP: FIBRE, PLY, RESIN Nomenclature: THERMAL-CONDUCTIVITY, THERMAL-CONDUCTIVITY_FIBRE, THERMAL-CONDUCTIVITY_PLY,

	THERMAL-CONDUCTIVITY_RESIN
THICKNESS	Definition: It is defined as distance through an object. Dimension: L , SI Unit: m Subtypes defined in VMAP: NODE, ELEMENT, SECTION Nomenclature: THICKNESS, THICKNESS_NODE, THICKNESS_ELEMENT, THICKNESS_SECTION
TIME	Definition: It is a scalar fundamental quantity. This should always be defined with a MYVARIABLEDESCRIPTION & MYVARIABLENAME attribute to provide the actual reference based on the solver. Dimension: T , SI Unit: s Subtypes defined in VMAP: – Nomenclature: TIME

10.21 U

10.22 V

VELOCITY-GRADIENT	Definition: It is the difference in velocity between two adjacent layers of fluid. Dimension: T^{-1} , SI Unit: s^{-1} Subtypes defined in VMAP: SECTION Nomenclature: VELOCITY-GRADIENT
VOLUME-FRACTION	Definition: It is a ratio of part volume to total volume for a ply unit cell. Here part could be Bubble, Fibre, Resin or Void Dimension: –, SI Unit: – Subtypes defined in VMAP: BUBBLE, FIBRE, RESIN, VOID Nomenclature: VOLUME-FRACTION_BUBBLE, VOLUME-FRACTION_FIBRE, VOLUME-FRACTION_RESIN, VOLUME-FRACTION_VOID

10.23 W

WELDLINE-ANGLE	Definition: The angle at which the molten plastic flows together, commonly used in injection moulding. Dimension: –, SI Unit: deg Subtypes defined in VMAP: – Nomenclature: WELDLINE-ANGLE
WELDLINE-POSITION	Definition: Commonly used in injection moulding, it is defined as the location of the weldline. It is given by the coordinates. Dimension: –, SI Unit: – Subtypes defined in VMAP: – Nomenclature: WELDLINE-POSITION
WELDLINE-QUALITY	Definition: commonly used in injection moulding, it defines the quality of the weldline. It is generally measured between 0 & 1. Dimension: –, SI Unit: – Subtypes defined in VMAP: – Nomenclature: WELDLINE-QUALITY

10.24 X

10.25 Y

10.26 Z

Chapter 11

Tutorials

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

11.1 Creating or Opening a VMAP .h5 File

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

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.

11.1.1 Parameters:

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

11.1.2 Returns:

On success,

with mode CREATEORREPLACE:
A VMAP .h5 file, with four groups, VMAP Group, and three sub-groups within VMAP - GEOMETRY, MATERIAL, VARIABLE & SYSTEM. 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.

Refer to the test case create_file.cxx.

11.2 Closing a VMAP .h5 File

VMAPFile::closeFile()

Function tries to close a VMAP file.

11.2.1 Returns:

On success,

closes the file.

On failure,

throws an exception.

11.3 Create a Group in VMAP File

void VMAPFile::createGroup(const std::string & groupName, bool trackCreationOrder)

Function that creates a group in the VMAP file.

11.3.1 Parameters:

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

11.3.2 Returns:

On success,

creates the group with the given group name.

On failure,

checks if the Group name already exists then opens that group in the VMAP file.

11.4 Get Sub-Groups from VMAP File

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

Collects all the sub-groups for the given `groupName`

11.4.1 Parameters:

- `groupName` - Name of the HDF Group to get sub-groups from.

11.4.2 Returns:

On success,

get the list of sub groups.

On failure,

catches failures caused by H5 operations.

11.5 Check if a Group exists in VMAP File

bool VMAPFile::existsGroup(const std::string & groupName)

Check is a group exists in a given `groupName`.

11.5.1 Parameters:

- `groupName` - Name of the HDF Group to check in.

11.5.2 Returns:

On success,

returns `True` if the group exists.

On failure,

catches failures caused by H5 operations.

11.6 Write the Version to VMAP File

void 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 (11.1).

11.6.1 Parameters:

- `version` - Data structure containing the sVersion.

11.6.2 Returns:

On success,

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

On failure,

prints error.

11.7 Read the Version of VMAP File

void 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 (11.1).

11.7.1 Parameters:

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

11.7.2 Returns:

On success,

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

On failure,

prints error.

11.8 Write the Meta Information to VMAP File

void VMAPFile::writeMetaInformation(const sMetaInformation & metaInfo)

Function that creates METADATA dataset in group '/VMAP/SYSTEM/' and sets the values in the METADATA dataset.

11.8.1 Parameters:

- `metaInfo` [IN] - Data structure containing the sMetaInformation.

11.8.2 Returns:

On success,

sets the meta information from the original file to the METADATA dataset in '/VMAP/SYSTEM/' group.

On failure,

prints error.

Refer to the test case write_read_metainforamtion.cxx.

11.9 Read the Meta Information from VMAP File

void VMAPFile::readMetaInformation(sMetaInformation &metaInfo)

Function that reads METADATA dataset from group '/VMAP/SYSTEM/' prints the data.

11.9.1 Parameters:

- `metaInfo` [OUT] - Data structure read from the VMAP file.

11.9.2 Returns:

On success,

prints the meta information.

On failure,

prints error.

Refer to the test case write_read_metainforamtion.cxx.

11.10 Read Points Block from VMAP File

void readPointsBlock(const std::string & groupName, sPointsBlock & points);

Function that reads a point block from a VMAP File.

11.10.1 Parameters:

- `groupName` [IN] - Name of the HDF Group to read information from.
- `points` [OUT] - Data structure containing a sPointsBlock

11.10.2 Returns:

On success,

returns the the data structure sPointBlock.

On failure,

prints an error message based on missing information. E.g. “sPointsBlock coordinate system does not match´´

Refer to the test case write_read_pointsblock.cxx .

more tutorials to be defined.....

Chapter 12

Simple Test Cases

12.1 Break Forming of a Metal Bracket

This test case shows interoperability between Abaqus $\longleftrightarrow$ LS-DYNA using VMAP Standard Specifications.

12.1.1 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 12.1 . It consists of three bodies: a sheet, a punch and a table with a hole. The dimensions of the sheet are:

length : $1.8''$
thickness : $0.1''$

The material properties are given by:

Young's modulus $E = 30 \times 10^6 Psi$
Poisson's ratio $\upsilon = 0.3$
Yield stress $\sigma_y = 5 \times 10^4 (1 + \overline{\epsilon_{\mathrm{p}}}^{0.6}) Psi$

Mass density $\rho = 7.36 \times 10^{-4}(lbf \cdot s^2)/in^4$

Here, $\overline{\epsilon_p}$ is the total equivalent plastic strain. 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.

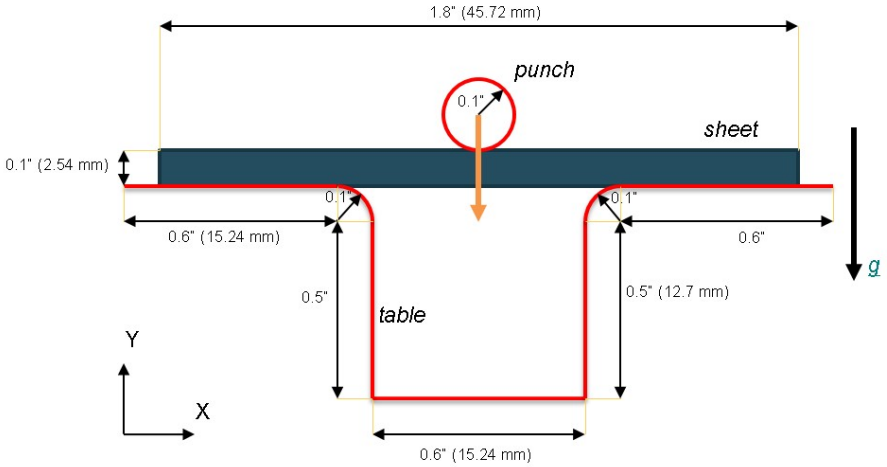


Figure 12.1: Model setup overview

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

Step-1 is carried out using Abaqus 2016 The sheet is modeled by $1.8'' \times 0.1''$ 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 subjected to gravitational forces acting in the negative Y-direction. The acceleration due to gravity is given by $386.1in/s^2$ 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

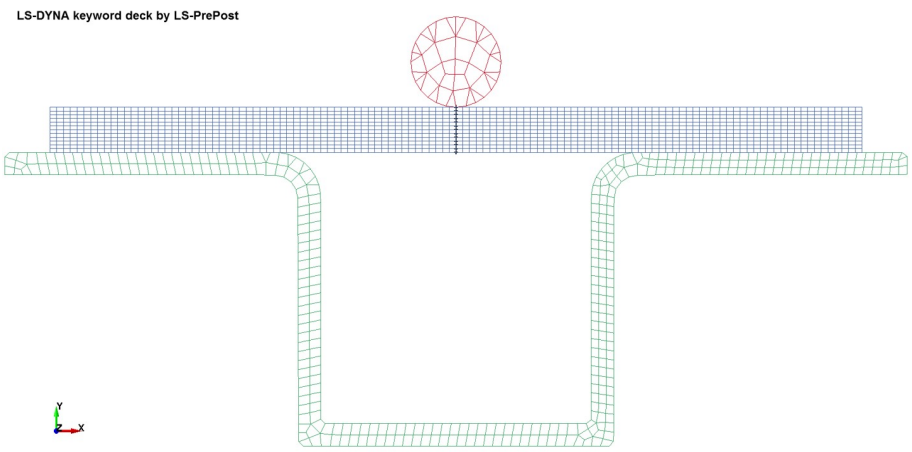


Figure 12.2: Forming simulation constraints. x-translation of the middle nodes are constrained.

Model Set-up for Step-2: Break forming

Step-2 is carried out using LS-DYNA r11.1 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

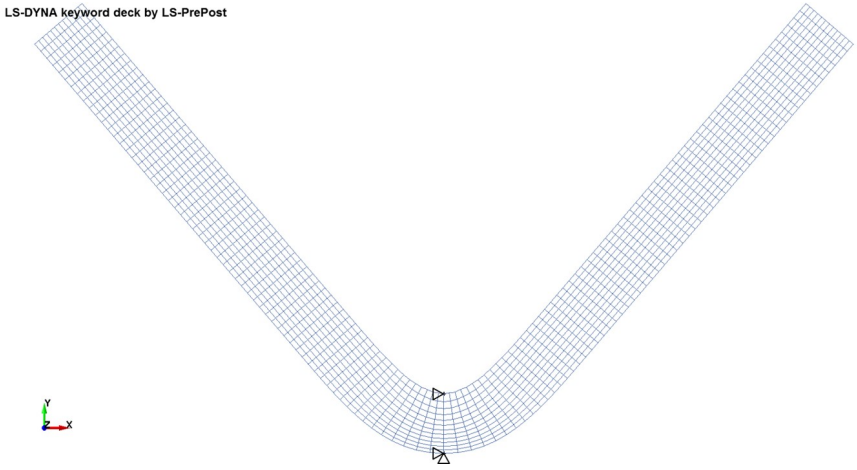


Figure 12.3: Spring-back analysis boundary conditions

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 ($\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}$) at the integration points of the elements;
- The total equivalent plastic strain ($\overline{\epsilon_{\mathrm{p}}}$) 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:

- Material model dependent history variables.

12.1.2 Software Implementation

Methodology

The two stages are solved in two separate runs. At the end of the first stage, the data listed in Sec. 12.1.1 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 - Abaqus Standard

Step-2 - LS-DYNA and LSPP were used (LS-DYNA is the Finite Element solver, LS-PrePost (LSPP) is the associated pre and post processor).

12.1.3 Results

Simulation Results

Step-1: Result of forming simulation

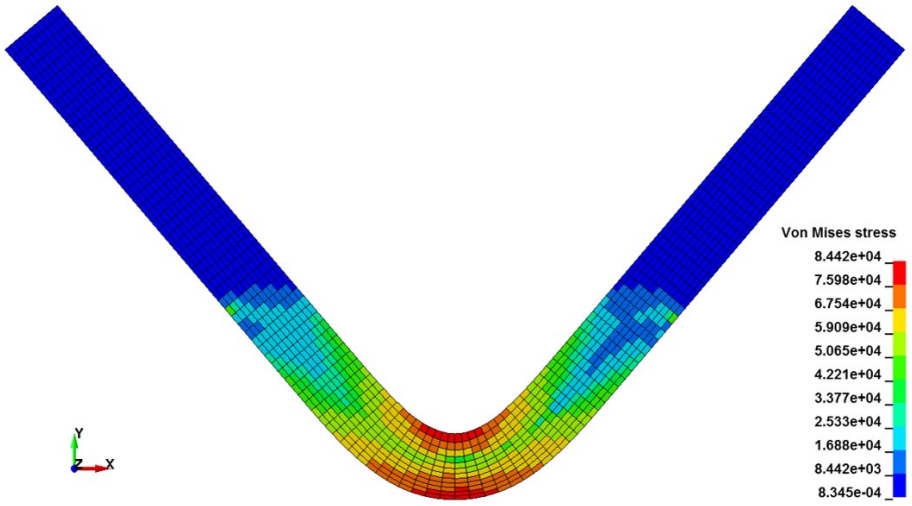


Figure 12.4: Abaqus solver von Mises stress

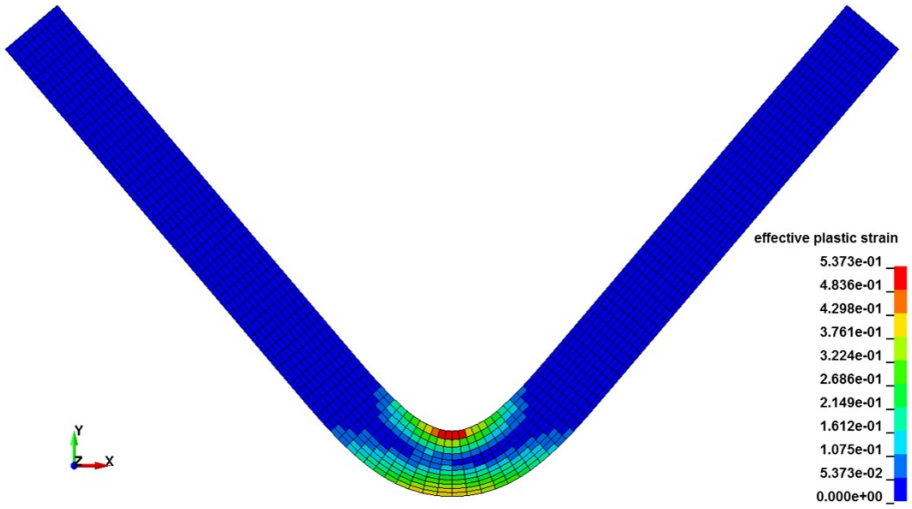


Figure 12.5: Abaqus solver effective plastic strain

Step-2: Spring-back simulation in LS-DYNA after reading Abaqus results in VMAP format

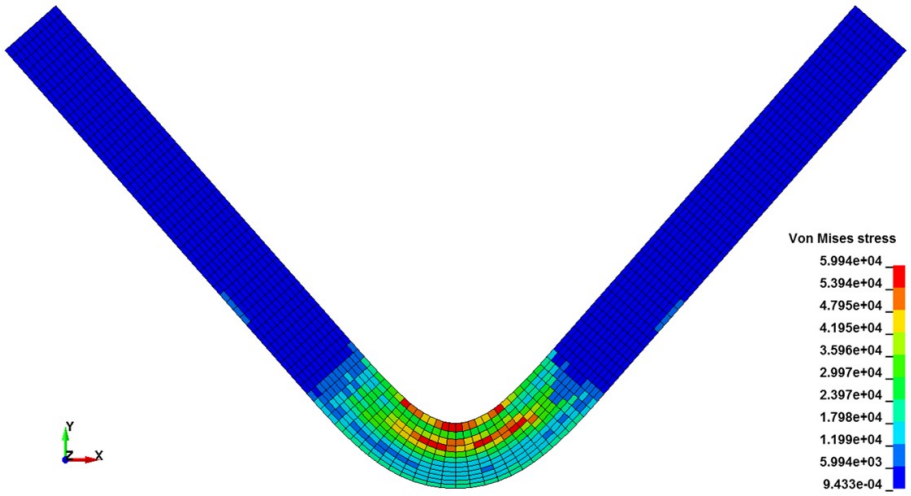


Figure 12.6: LS-DYNA solver spring-back simulation using Abaqus wrapper VMAP file – von Mises stress

The residual von Mises stress distribution at the end of the second stage after the spring-back is shown in Figure 12.6

Figure 12.7: To measure the spring-back, the displacement in X-direction of the node in the top-left corner of the sheet is plotted as a function of time in the figure below. The red curve shows the displacement of the node after Step-2, when Step-1 is carried out using LS-DYNA as well, while the green curve shows the displacement after Step-2 when Step-1 is carried out using Abaqus. The effect of the spring-back on the displacement of this node is apparent and amounts to $-0.0114''$.

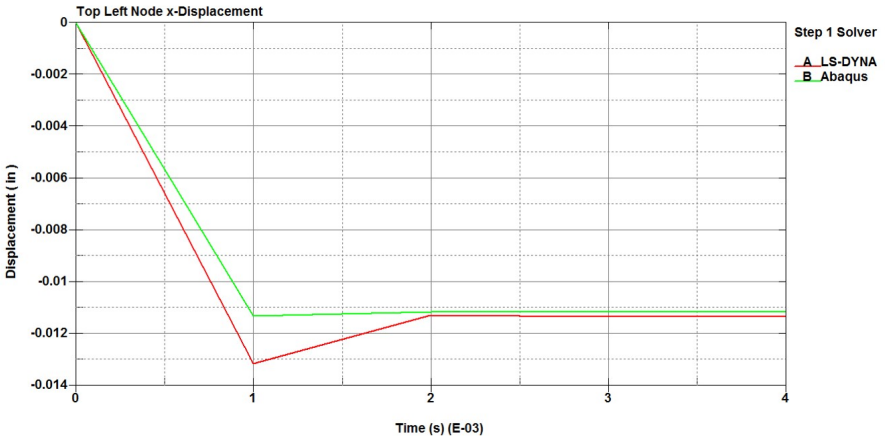


Figure 12.7: Comparison of top left node x-displacement during spring-back simulation

Related analytical results / Reliable experimental results

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

Evaluation and Validation

Splitting the analysis in two parts and evaluating the second part using the Model Section of the first part gives the same results as when running the analysis as a single model.

Abaqus
In Abaqus the analysis is performed in one step or in two steps. For a two-step run where LS-DYNA is used for Step-1, read the initial stresses and initial plastic strains from the VMAP file and prescribe them as initial conditions in the analysis using SIGINI and HARDINI. The solution obtained via single run in Abaqus can be used as a reference solution for the VMAP implementation in Abaqus.
Comparison of top left node x-displacement using Abaqus as a reference solver.

Simulation Steps	Solver	Result (% error)
Single Step	Abaqus	-0.011311 in
UsingAbaqus for Step-1, changing solvers for Step-2.		
Step-2	Abaqus	-0.0113148 in (0.03%)
Step-2	LS-DYNA	-0.011097 in (1.89%)

LS-DYNA

Comparison of top left node x-displacement using LS-DYNA as a reference solver.

Simulation Steps	Solver	Result (% error)
Single Step	LS-DYNA	-0.0113 in
Using LS-DYNA for Step-1, changing solvers for Step-2.		
Step-2	LS-DYNA	-0.0113 in (0)
Step-2	Abaqus	-0.0112 inch (1.55%)

VMAP Result File

The image below shows a sample VMAP file used to transfer data from Abaqus to LS-DYNA

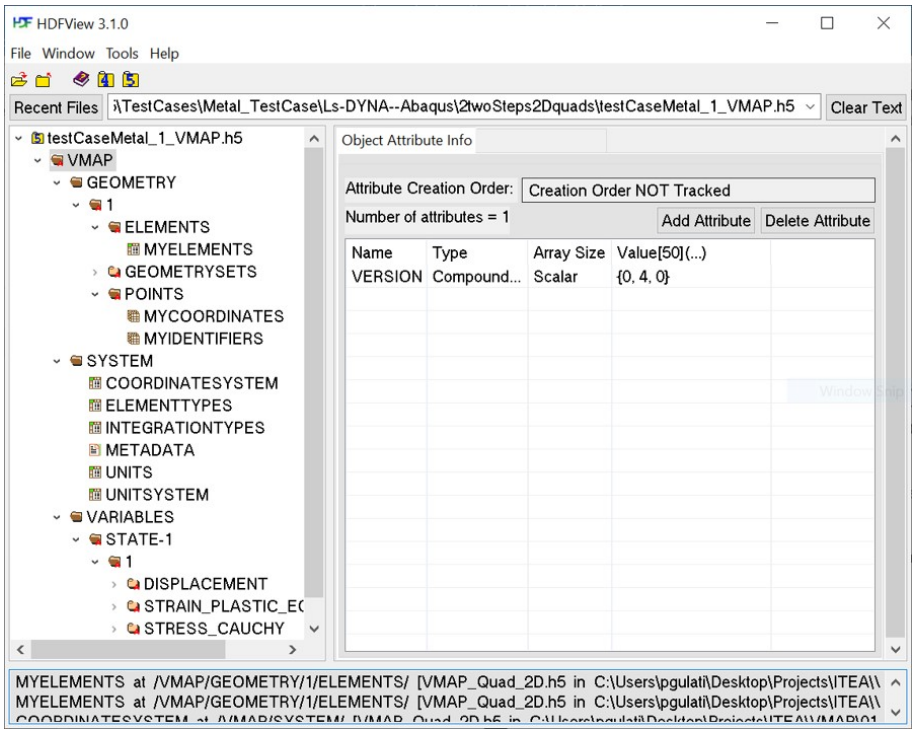


Figure 12.8: VMAP file generated from Abaqus ODB

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

Mesh Based Extensions

A few mesh based extension of this test case are also available for Abaqus.

- A mixed Quad & Tria- element mesh.
- 3D Hexahedron mesh
- 3D Quadratic Tetrahedral mesh

12.2 Fill Analysis of a Mouse Cover

This test case shows data transfer from Autodesk Moldflow to the VMAP Standard. The same model can then be used for structural or thermal analysis in other softwares supporting the VMAP Standard like LS-DYNA, MSC Marc etc. The test case performs one of the standard Autodesk Moldflow simulation tutorials and exports the results towards the VMAP format. Additionally, we illustrate the ability to export the simulations towards Paraview, an open source visualization toolbox.

12.2.1 General Description of the test case "Fill Analysis of a Mouse Cover"

The test case considers the filling phase of a injection molding analysis, without considering the complete fill system. The filling phase is the most important phase of the injection molding cycle, as any problems with the filling will result in suboptimal part quality. The specific analysis we perform, is also described in the tutorial manuals provided by Autodesk Moldflow by following this link. The tutorial considers a plastic shell of a computer mouse. The analysis has a single injection location, i.e. the location where the filling starts. During the simulation, the complete filling of the part is simulated, providing insight into, for example, the filling time, temperatures, and pressures. Additionally, when a fiber-reinforced material is chosen for the filling, the simulation also provides details on the resulting fiber orientations at the end of filling.

Model description

The test case considers the part geometry as given in Figure 12.9. This shell represents the outer part of the computer mouse. This part is filled from a single location, located at the back side of the mouse cover. However, the tutorial allows the user to decide their injection location, or even select multiple injection locations.

After providing the injection locations and material models, the simplest analysis in Autodesk Moldflow is performed, i.e. the filling analysis. For means of this test case, the default settings of the filling analysis are considered appropriate. However, more advanced users can modify these settings to their desires. The filling simulation will then run through the full analysis and virtually perform the filling of the shell.

Figure 12.9: This geometry represents the outer shell of a computer mouse. For the test case, this part will be analyzed during a filling analysis using Autodesk Moldflow. The molten plastic is injected at a single injection location at the back of the mouse

Exported state variables

Although this simplified test case considers just a single analysis step, it is of importance to be able to transfer variables from Autodesk Moldflow towards the VMAP Standards Specification. In principle, the ability to transfer this information towards the VMAP format, allows to couple the filling analysis to a wide variety of alternative simulation tools. For illustration purposes, we will consider the following variables to be of interest:

- Temperature over time
- Pressure at V/P switch-over
- Fill time (i.e. the time it took for the injected material to reach a location)
- Fiber orientations (if fiber reinforced material is used)

Many additional quantities in Autodesk Moldflow are available after the filling analysis, and might be required for successful calculation and could be exported towards VMAP format as well. However, for the current discussion, we limit ourselves to the mentioned variables.

12.2.2 Software Implementation

The software implementation requires Autodesk Moldflow and the Moldflow-VMAP wrapper as developed by M2i.

Methodology

The test case can be separated into two phases. First, the user can perform the Autodesk Moldflow tutorial of the filling analysis. In this tutorial, the user is free to modify any variables or process settings of interest. This first step should be performed with the graphical user interface (GUI) of Autodesk Moldflow. Together with the Moldflow tutorial, this interface allows for straightforward problem definitions. Once the model is defined, the user is able to evaluate the simulation in two approaches:

- The simulation is evaluated from the GUI.
- The simulation is evaluated through command-line using the *runstudy* command.

After the simulation is finalized, Autodesk Moldflow will present the results. To export these results towards the VMAP specification, the user is required to output the following components:

- The simulation mesh in Patran format
- The simulation variables in xml format

We suggest the user to export these variables using the Autodesk Moldflow GUI, which also allows to export the analysis log for future reference on simulation details. However, it should be noted that this information can also be extracted through the *studyrlt* command-line tool. With the simulation mesh and variables of interest as output, the user can run the Moldflow to VMAP wrapper in the specific directory as:

```
call python \src\moldflow2vmap.py %dir%mesh.pat %dir%*.xml
```

Where the variable %dir% should be replace with the directory of the output files, or can be removed if the output files are located in the current directory. The wrapper will now process the mesh, and provided analysis variables of interest. These are converted towards the VMAP Standard Specifications. Additional settings of the wrapper can be explored by running the command:

```
call python \src\moldflow2vmap.py –h
```

Software Tool

To transfer the Autodesk Moldflow results towards the VMAP Standard Specifications, the user requires access to the Moldflow – VMAP wrapper, as developed by M2i. This wrapper is evaluated through a command-line environment, e.g. *call python .\moldflow2vmap.py –h* to list all options of the package.

12.2.3 Results

Simulation Results

The Moldflow – VMAP wrapper allows to export data to Paraview, a powerful open source visualization toolbox. Thus, when converting the Autodesk Moldflow data towards

VMAP, also files suited for visualization with Paraview are generated. This functionality is enabled by simply calling the M2i wrapper with the additional argument *"–paraview"*. All images in the documentation of this test case were generated with Paraview. The filling analysis provides various fields to be analyzed. First of all, we can investigate the simulation mesh inside Moldflow. For the current analysis, this analysis was performed using shell elements. The mesh is shown in Figure 12.10. The wrapper allows the mesh (given in Patran format) to be exported towards the VMAP format. Additionally, if the mesh consists of multiple parts, the Moldflow – VMAP wrapper allows to export these parts separately.

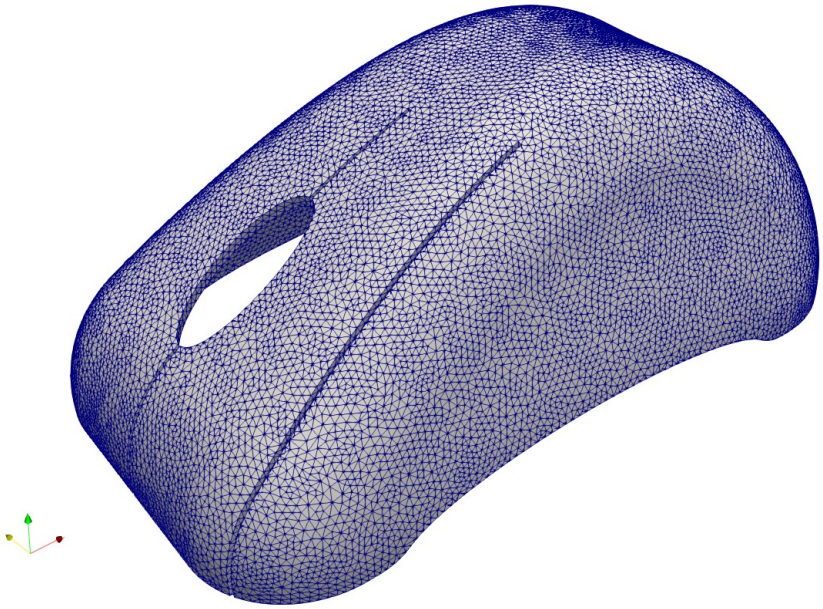


Figure 12.10: Simulation mesh for the filling analysis of the mouse cover. The analysis mesh consists of shell elements representing the outer surface of the domain.

Various results can be visualized on top of the analysis mesh. For instance, the fill time can be projected onto the analysis mesh. This is shown in Figure 12.11. Here, we can observe the time it took for the injected material to reach a certain location of the model. The blue color indicates a short fill time, i.e. the material reaches these locations early in the filling process, while the red colors indicate a large fill time. Especially the left-most part of the model has a high fill time.

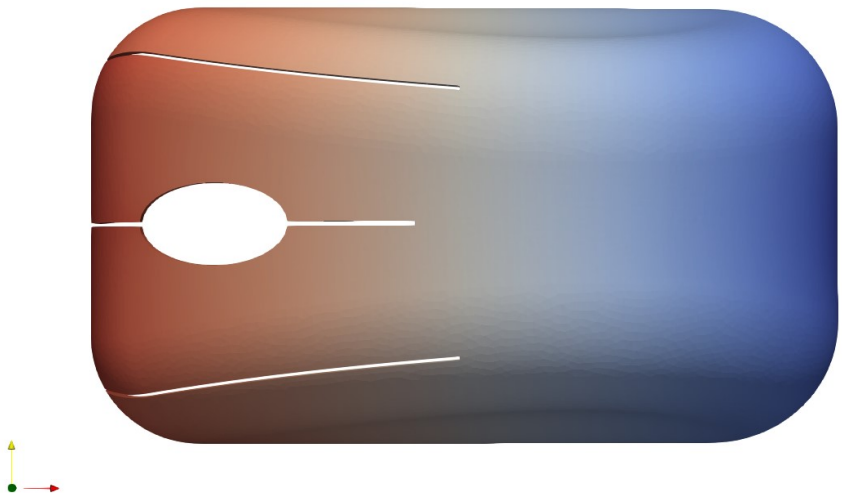


Figure 12.11: Fill time (blue start, red end). The fill time indicates a time stamp when the injected material reached a certain location of the mold. For this analysis, the injection location was located at the right. This is reflected by the short fill time (blue), whereas the left part of the domain was only filled near the end of the simulation, resulting in a large fill time (red).

In addition to data field obtained at the end of filling, such as the previously discussed fill time, we can also visualize transient data. For illustration, the pressure distribution at an intermediate time step is shown in Figure 12.12. In this visualization, one of the state variables at a desired time was selected from Moldflow and visualized. It should be noted, that the Moldflow – VMAP wrapper contains all provided time steps.

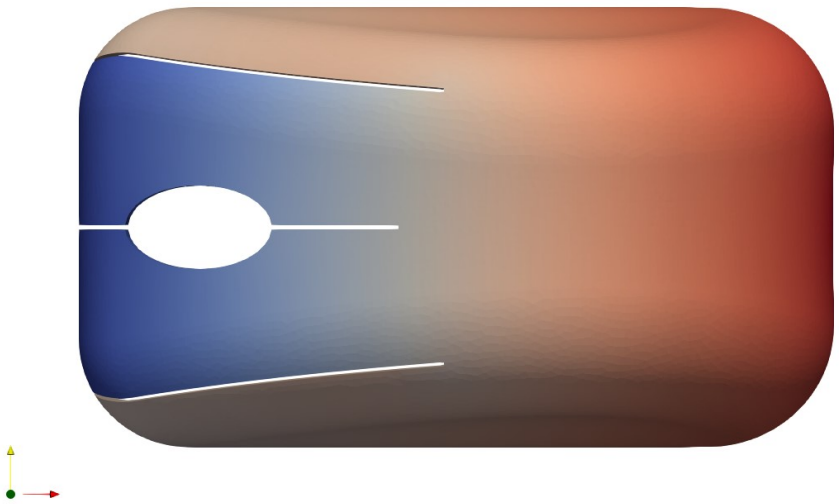


Figure 12.12: Pressure distribution (blue low, red high). The pressure field is extracted for one of the time steps during the filling analysis. As the filling is not completed yet, we can observe different pressures throughout the model. For instance, at this time step, the two buttons of the mouse, are of substantially lower pressure compared to main body of the mouse.

In case a fiber reinforced material is used, the orientation tensor filed is computed by Moldflow. The Moldflow-VMAP wrapper allows also to visualize fiber-orientations after filling, such as shown in Figure 12.13.

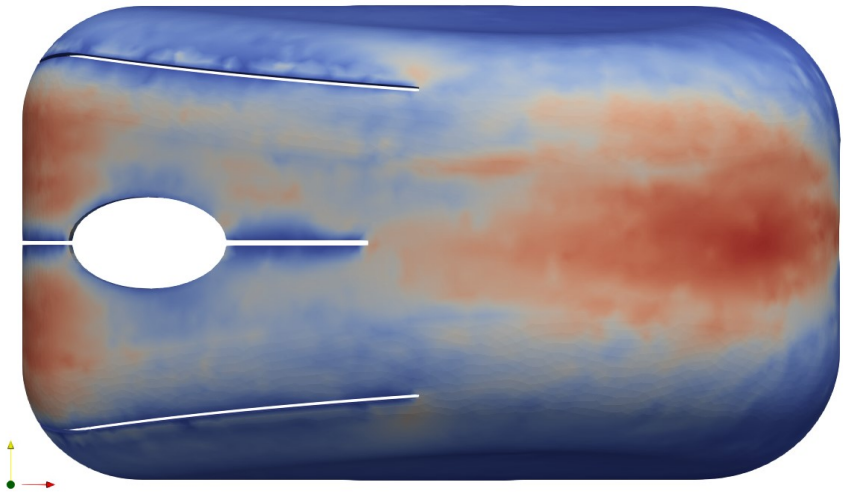


Figure 12.13: Fiber orientation yy-direction. Illustrative visualization of the fiber orientation of the test case. The colors illustrate the tensor component (yy) of the fiber orientation at 0 offset from the center.

VMAP Result File

The VMAP result file contains all necessary information for other software to perform additional analysis. At the moment, the VMAP format contains the geometry (with possible multiple, separable parts), the system (or meta-data) information, and the analysis variables. For the presented test case, the geometry consists of a single part, with a single element type, a single integration rule, a single unit system and a single coordinate system. For the variables, a field is exported for each part for each time step of the analysis. Thus, as can be seen in Figure 12.14, each state contains the relevant variables for each part at those instances. Data that is only available at initial time steps, is stored in “STATE-0”. Data that only becomes available at the end of the simulation, is appended to the last available state, “STATE-n”.

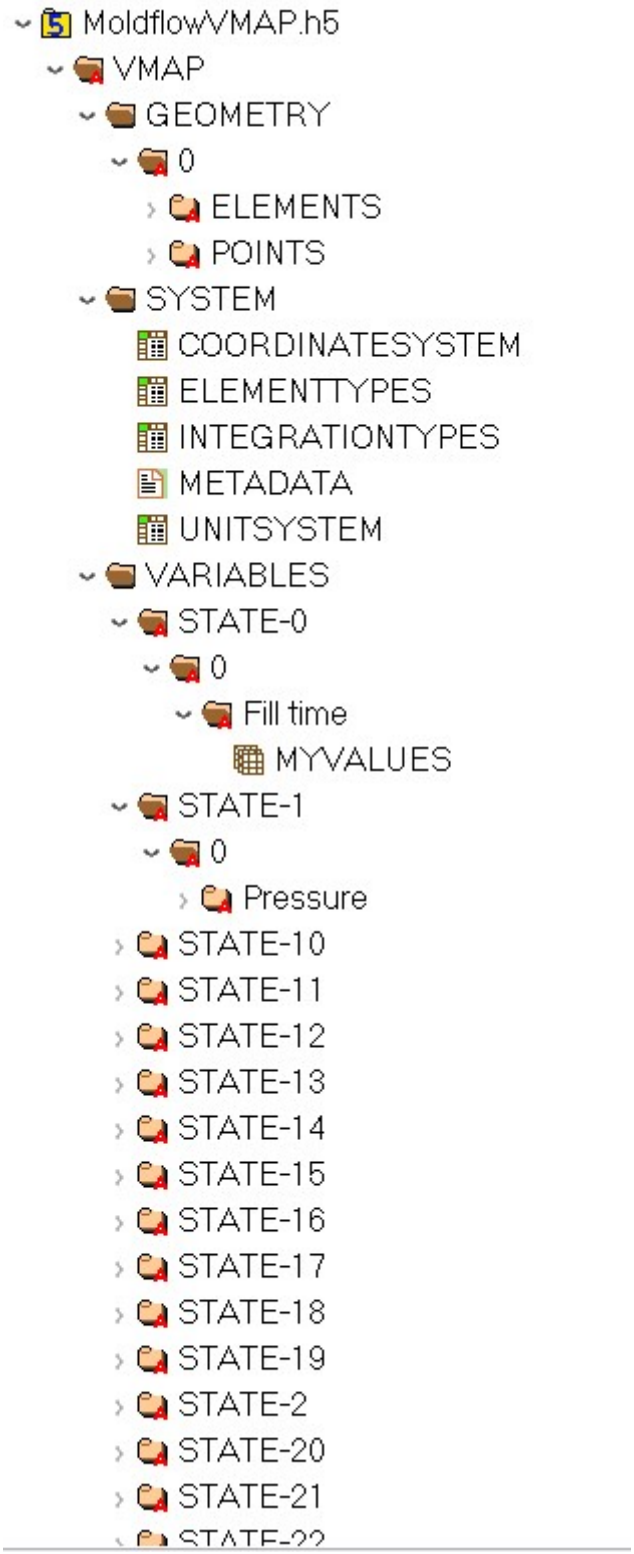


Figure 12.14: VMAP file generated from Moldflow-Wrapper

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

Chapter 13

Additional Features

VMAP I/O Library offers the following additional features.

13.1 Read & Write Images

VMAP offers the possibility to store images as HDF5 indexed images. The following functions are provided in the VMAP API, however, the functionality itself is supported by HDF5.

- **void VMAPFile::readImage(const std::string & groupName, const std::string & dataSet, int & width, int & height, std::vector<unsigned char> & rgb)**
- **void VMAPFile::writeImage(const std::string & groupName, const std::string & dataSet, const int & width, const int & height, const std::vector<unsigned char> & rgb)**

13.2 Read & Write Tables

VMAP also allows the user to store tables e.g. material curves etc. the following functions are associated with reading and writing tables in VMAP.

- **void VMAPFile::readTable(const std::string & groupName, const std::string & tableName, sTable & table)**
- **void VMAPFile::writeTable(const std::string & groupName, const sTable & table)**

Refer to Chapter 6 Sec. 6.14 for more details on table structure in VMAP.

13.3 Storing Binary Files

Since VMAP is an open format, some concerns from VMAP Partners have been about privacy of data. This is specially necessary for storing material data which is propriety to the firm. It might also be the case that the the material data is in a certain format which should be transferred to the following step but it is not necessary to view it. In this case, a binary file can come in handy and data can be easily stored The following function is offered by VMAP to save binary files:

void VMAPFile::saveExternalFile(const std::string & groupName, const std::string & dataSetName, const std::string & fileName)

There are a few rules to adhere to:

1. Binary file should be stored within these groups - Geometry, Variables, Material.
2. This function allows you to add a ‘fileName‘ as attribute, provide a relatable/understandable file name.

13.4 Storing Composite Layers - SECTION

VMAP offers the possibility to store composite layers in the form of section. The structure use to store the multiple layers is called Section. In VMAP, `SECTION` is a dataset under `SYSTEM` Group. The metadata in `SECTION` is stored in the following order:

- `myIdentifier`: Integral identifier for each section.
- `myName`: Stores the name of the section.
- `myType`: Stores Section reference number. For more details about reference number, see Table 13.2.
- `myMaterial`: Stores the material type reference number.
- `myCoordinateSystem`: Stores the reference coordinate system of the section.
- `myIntegrationType`: Stores the integration type of the section
- `myThicknessType`: Stores the thickness type of the Section. For more details, see Table 13.3.

Metadata	Data Type
myIdentifier	Integer
myName	const char*
myType	Integer
myMaterial	Integer
myCoordinateSystem	Integer
myIntegrationType	Integer
myThicknessType	Integer

Table 13.1: Data Types of Section Metadata

Table 13.2 shows the available section types and their reference numbers in VMAP.

SECTION Types	Reference Number
INVALID_SECTION	-1
SHELL_SECTION	0
SOLID_SECTION	1
BEAM_SECTION	2
MEMBRANE_SECTION	3

Table 13.2: SECTION TYPE Enumeration

Table 13.3 shows the available thickness types and their reference numbers in VMAP.

Thickness Types	Reference Number
INVALID_THICKNESS	-1
NODAL_THICKNESS	0
ELEMENT_THICKNESS	1

Table 13.3: Thickness Types Enumeration

The General Object Info associated with this data set is shown in Table 13.4.

Name:	`SECTION`
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 13.4: General Object Info

n refers to number of section defined for the model.
The read and write functions associated with the Section are:

- **void VMAPFile::readSections(std::vector<VMAP::sSection> & sections)**
- **void VMAPFile::writeSections(const std::vector<VMAP::sSection> & sections)**

13.5 Storing Multiple Result Files

VMAP allows you to store multiple simulation files in one VMAP file. Although, this is not recommended as a best practice, it is still available as an option. The following constructor supports the addition of a prefix to the '/VMAP/' storage hierarchy.
VMAPFile::VMAPFile(const std::string & path, int mode, const std::string & prefix)

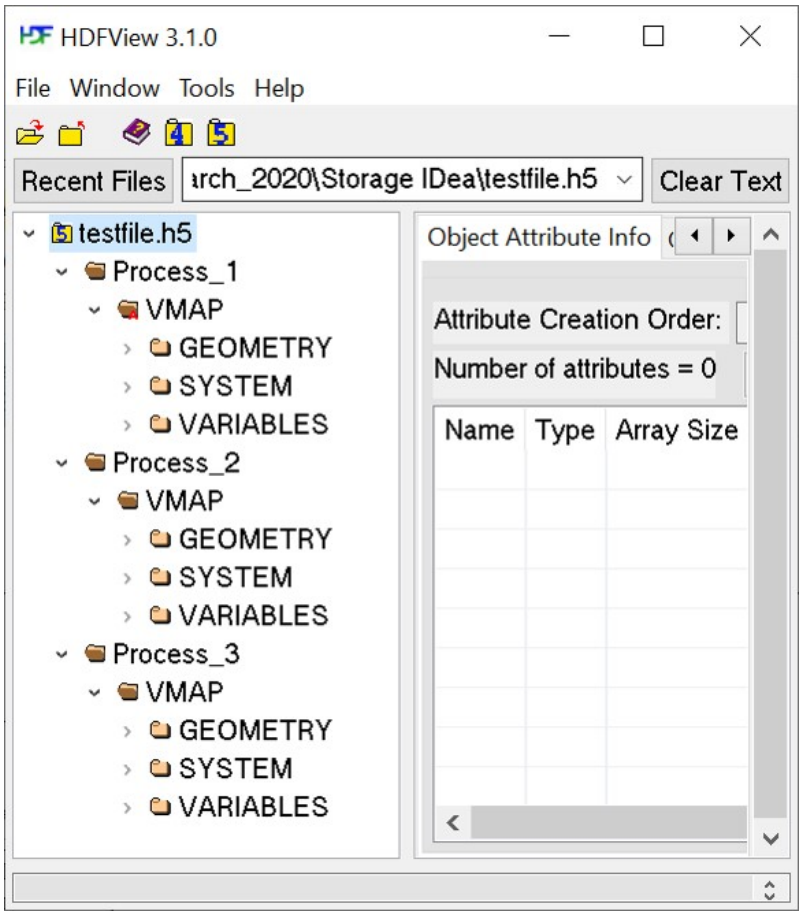


Figure 13.1: Multiple File Storage

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

ITEA VMAP Project Funding (2017-2020)

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

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

The **Canadian part** of the joint project was funded by the National Research Council of Canada Industrial Research Assistance Program (NRC IRAP)

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

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

ITEA VMAP 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 Maschinenbau GmbH, NAFEMS Deutschland, Österreich, Schweiz GmbH, RIKUTEC Richter Kunststofftechnik GmbH & Co. KG, Robert Bosch GmbH, Simcon kunststofftechnische Software 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

Appendix B

VMAP Standard Sensor Data Specifications

The following document provides a detailed overview to store test/measurement data in VMAP Standard format. This work has been exclusively carried out by VMAP SC e.V. and the Sensor Data Working Group, and the work has been supported by the EU projects listed at the end of the document.

VMAP

A new Interface Standard for Integrated Virtual Material Modelling in Manufacturing Industry

VMAP Standard Sensor Data Specifications

Version no.: 1.2.0

Edited by: Fraunhofer SCAI - Victor Lueddemann

Abbreviations

API	Application Programming Interface
VMAP SC	VMAP Standards Community e.V.
DIN	Deutsches Institut für Normung e. V.

Contents

List of Figures

List of Tables

List of Authors

DLR (Institute of Lightweight Systems)	Jean Lefèvre
Fraunhofer SCAI	Andre Oeckerath
	Klaus Wolf
	Priyanka Gulati
	Victor Lueddemann
Fraunhofer IZFP	Dietmar Weber
	Frank Leinenbach
Dr. Reinold Hagen Stiftung/ Hagen Engineering GmbH	Alexander Busch[1]
	Olaf Bruch[1]
Hochschule Bonn-Rhein-Sieg[1]	Patrick Michels
SICK AG	Morten Meyer
SWERIM	Sivaprasad Palla
	Krister Ekstrom
University Patras	Konstantinos Tzimanis

[1] *belongs to more than one institution/organization*

License of this document

If you would like to join the community or know more about the standard, send an email to info@vmap-standard.org
To download the VMAP IO Lib, please agree and sign the license terms here https://vmap-standard.org/Specifications/Downloads/

Chapter 1

Terms in Metrology

1.1 The Measurement Process

This section deals with the measurement process and aspects that are taken into account by VMAP.

The goal of a measurement is to determine the magnitude associated with a measured quantity [**DIN_1319-1**]. Typically, several steps are involved in this process (see Figure 1.1). Therefore, a measurement consists of the following fundamental elements: the desired measured quantity, a sensor that responds to the measurand, and a conversion of the signal so it can be interpreted by a human or a machine.

In essence, many measurements can be traced back to the representation in Figure 1.1. However, it does not satisfy the complexity of real measurement processes. Some additional elements are missing that are directly or indirectly involved in measurements. The disruptive influence of ambient conditions or the calibration of the measuring system might affect the result. Therefore, additional information about the measured object, influencing factors, the setup, or post-processing methods are decisive. All of these metadata support the interpretability of the actual result but are often inadequately mentioned in the data.

In order to compare measurement results with simulations, discrete location points are required for the measured values. A trustworthy validation of simulation models through comparative measurements also requires information regarding uncertainties.

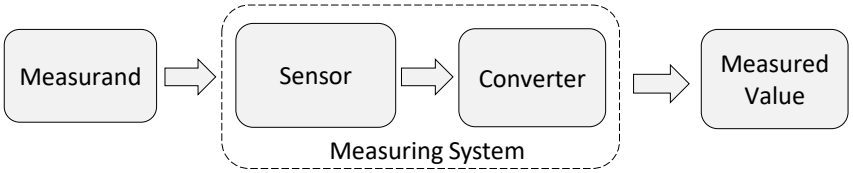


Figure 1.1: Working principle of measuring systems.

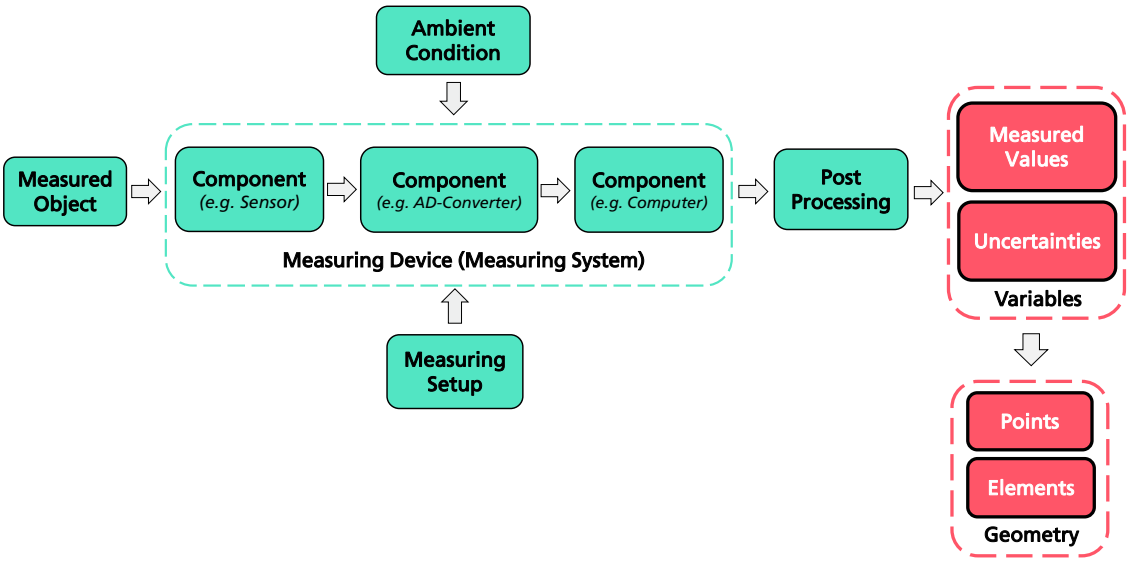


Figure 1.2: Elements of a comprehensive measurement process

Figure 1.2 shows a representation of the measurement process with the aforementioned, comprehensive aspects of measurement data. The data is displayed in red and green colors, which formally classify it into one of two categories. On the one hand, the ones depicted in red include the full measurement result, consisting of the measured value and the associated uncertainty, and on the other hand, measurement points, to which the measurement result is assigned.

Additional information about the measurement, such as descriptions of the measurement setup, the measurement object, influencing factors, and post-processing methods, is shown in green. Since this metadata highly depends on the use case, these datasets are not strictly standardised yet and therefore handled in a more user-open manner.

1.2 Definitions

Measuring System

Measuring systems are a key element of the VMAP structure, as measurement results, measured objects, the measuring setup, sensors (components), calibrations and ambient conditions are linked to them. They consist of one or more sensors (components) and convert the input variable into the output variable of the measured quantity. To avoid confusion with the VMAP Group `SYSTEM`, the term measuring device is used synonymously for measuring systems.

Measured Object

The measured object (or measuring object) is the carrier of the measured quantity [**DIN_1319-1**]. The measurand is therefore assigned to the measured object. Objects can be bodies, processes, or states. Examples for measurement objects are:

1. The measured quantity 'temperature' is assigned to the measured object 'surface of the plastic tank'.
2. The measured quantity 'radiant power' is assigned to the measured object 'infrared radiation'.

Components

A component can be a measuring instrument or any other part of a measuring system. Together, they form a measuring system. Examples of components are:

1. Sensors are the first component of a measuring system.
2. Components like analog-to-digital converters transform the analog signal from the sensor into a digital signal.
3. A micro processor or connected computer processes digital signals and performs computations on them.

Measuring Setup

The measuring setup refers to the characteristics of the whole measuring system and thus to the assembly or interaction of components it consists of. Examples of properties of the measurement Setup include

1. distance between sensors,
2. angle between two cameras in triangulation,
3. calibration processes.

Ambient Conditions

Ambient Conditions refer to parameters that are not the actual target of the measurement trial but might be relevant due to their potential influence on the measurement result. These are neither properties of the measurement object nor calibration parameters of the measurement setup. Examples of ambient conditions include

1. room temperature,
2. air humidity,
3. presence of dust,
4. magnetic interference.

Post Processing

Post-processing describes activities that may have been carried out to adjust the signal after it has exited a measuring system. Examples of post-processing include

1. retroactive calculation of temperature using a factor for emissivity,
2. interpolation of measurement values to specified time points,
3. mapping of measurement values to the existing geometry.

Measurement Result

Measurement results are the primary purpose of a measurement trial. The result can be obtained indirectly, through physical relationships, and does not have to come directly from the sensor or the measuring system. This might is the case when the measured resistance of a thermocouple and material properties are used to calculate temperature. Therefore, the result is not the measured resistance itself but the calculated temperature.

The result always includes the unit of the measurand and the measured value [**DIN_1301-1**]. The result of a temperature measurement can be expressed as follows

$$T = \{T\}[T] \tag{1.1}$$

with $\{T\}$ being the value and $[T]$ being the unit, e. g. $T = 100\ \mathrm{K}$.

Measuring Points

Measurement points and elements together are referred to as geometry. Measurement points are discrete coordinates to which the measured value is assigned. When measurement points collectively form geometric elements, a measured value can also be assigned to these elements instead.

Full Measurement Result

In repeated measurements, there is a variation in results, despite unchanged experimental conditions. Therefore, a full measurement result should always include, in addition to the measured value, information about uncertainties. The measurement uncertainty is determined either through statistical analyses of multiple independent measurements or given by accuracy specifications provided by the manufacturer.

The full Measurement Result X with the relation

$$X = x \pm u \tag{1.2}$$

consists of an estimate x and the standard measurement uncertainty u [**DIN_1319-3**]. For the estimated value x, the following applies

$$x = \bar{x} + K \tag{1.3}$$

with $\bar{x}$ commonly representing the arithmetic mean

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{1.4}$$

gained from n measured values x_i and a correction K corresponding to the known systematic measurement deviation. It is assumed that the true value is likely to lie within the interval $[x - u, x + u]$.

Uncertainties

The standard measurement uncertainty u consists of the uncertainty for random deviations u_1 and the uncertainty for systematic deviations u_2 [**DIN_1319-3**]. The relationship is given by

$$u = \sqrt{u_1^2 + u_2^2}. \tag{1.5}$$

For u_1, the empirical standard deviation s is used according to the equation

$$u_1 = \frac{s}{\sqrt{n}} = \sqrt{\frac{1}{n(n-1)} \sum_{i=1}^{n} (x_i - \overline{x})^2}. \tag{1.6}$$

Alternatively, if a standard deviation s_0 from previous measurements or experiences is available

$$u_1 = \frac{s_0}{\sqrt{n}} \tag{1.7}$$

can be used.

The uncertainty for systematic measurement deviations u_2 might be neglected depending on the individual case. It should be assessed accordingly, whether this is possible.

Expanded Measurement Uncertainty

Instead of the standard measurement uncertainty u, the expanded measurement uncertainty can be used:

$$U = k \cdot u \tag{1.8}$$

with the expansion factor $k \in [2, 3]$. The true value is then very likely to lie within the value range

$$X = x \pm U. \tag{1.9}$$

Confidence Interval

If the distribution of measurement values follows a probability distribution, in addition to the full measurement result, information regarding the confidence interval can be provided. In the following, a normal distribution is assumed.

The confidence interval indicates, with a certain probability, that the true value lies within the interval

$$X = x \pm t \cdot u. \tag{1.10}$$

The Student's t-factor t can be obtained from tables and depends on the number of trials n and the chosen confidence level.

For $n = 200$ measurements, the confidence interval corresponds to the standard uncertainty u. In this case, the true value lies within the chosen confidence interval with a probability of 68.26%. It is recommended to use a confidence interval of 95%, but not higher. Since then, the confidence interval is reasonably independent of the underlying probability distribution, which cannot always be assumed to be normally distributed [**DIN_1319-3**].

If a one-sided confidence interval is chosen, the following applies depending on the side

$$X \leq x + t \cdot u \text{ or } X \geq x - t \cdot u. \tag{1.11}$$

Error Limits

Error limits are limits set by the manufacturer that must not be exceeded by the measuring device. It is guaranteed that under proper use, random variations will fall within the error limits. Therefore, the true value can be found in the interval

$$X = x_i \pm E, \tag{1.12}$$

with the upper and lower error limit being the value E. The interval enclosed by error limits is considerably wider than the measurement uncertainty.

Chapter 2

Sensor Data Structure Relationship

2.1 Data Structure Dependency

The sensor 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).

2.1.1 `VMAP` Group

`VMAP` group in VMAP Standard I/O library contains data from groups - `MEASUREMENT` & `SIMULATION`. Figure 2.1 shows a schematic of the `VMAP` group and the data it contains.

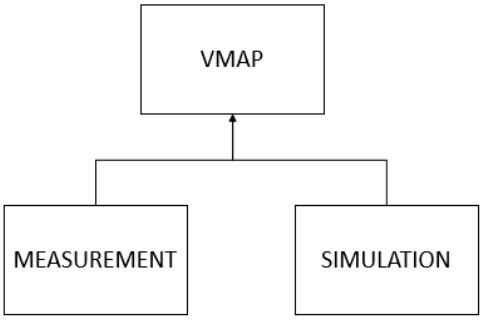


Figure 2.1: VMAP Group Dependency

2.1.2 `MEASUREMENT` Group

`MEASUREMENT` group in VMAP Standard I/O library contains data from groups - `DEVICES`, `GEOMETRY`, `SYSTEM` & `VARIABLES`. Figure 2.2 shows a schematic of the `MEASUREMENT` group and the data it contains.

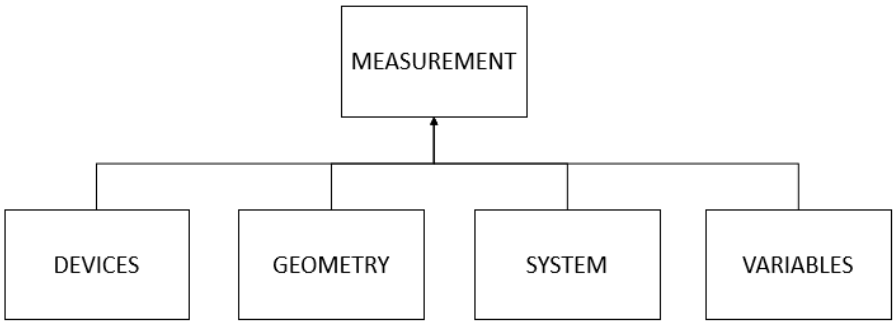


Figure 2.2: MEASUREMENT Group Dependency

2.1.3 DEVICES Group

DEVICES group in VMAP Standard I/O library contains sub-group <DEVICE-ID>, within this lies groups AMBIENTCONDITIONS with datasets <AMBIENTCONDITION-n>, MEASUREDOBJECTS with datasets <MEASUREDOBJECT-n>, COMPONENTS with datasets <COMPONENT-n>, MEASURINGSETUP with datasets <MEASURINGSETUP-n> & POSTPROCESSING with datasets <POSTPROCESSING-n>. Figure 2.3 shows a schematic of group DEVICES.

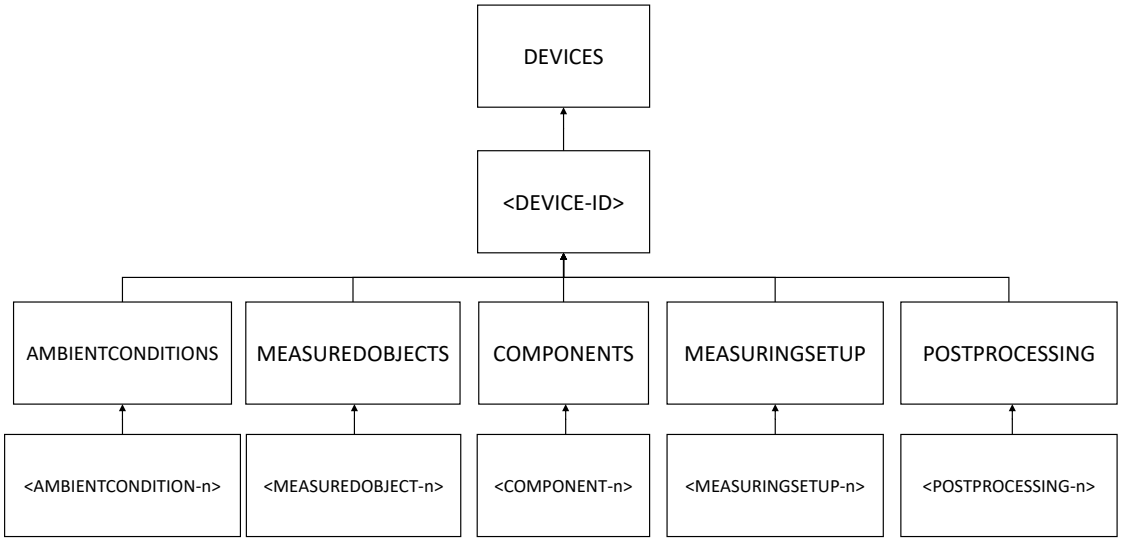


Figure 2.3: DEVICES Group Dependency

2.1.4 GEOMETRY Group

The GEOMETRY group in VMAP Standard I/O library contains sub-group <DEVICE-ID>, within this lies groups POINTS (sPointsBlock) and ELEMENTS (sElementBlock). Figure 2.4 shows a schematic of group GEOMETRY.

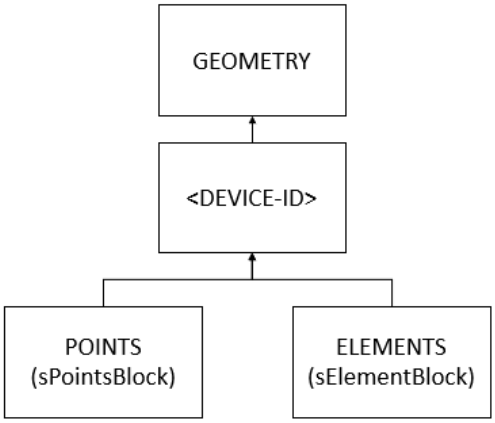


Figure 2.4: GEOMETRY Group Dependency

2.1.5 SYSTEM Group

The SYSTEM group in VMAP Standard I/O library contains datasets - COORDINATESYSTEM (sCoordinateSystem), ELEMENTTYPES (sElementTypes), UNITSYSTEM (sUnitSystem) & UNITS (sUnit). Figure 2.5 shows a schematic of group SYSTEM.

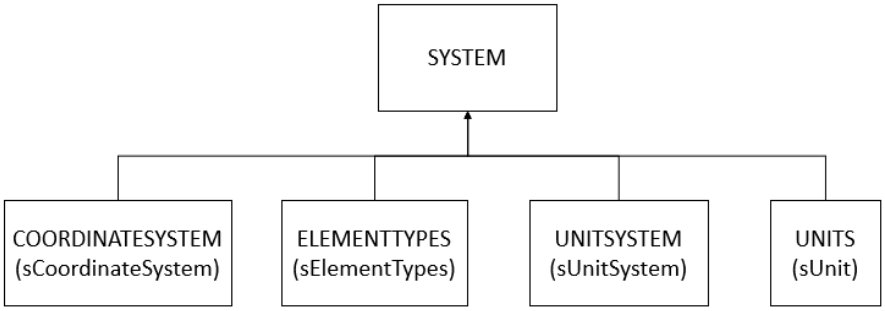


Figure 2.5: SYSTEM Group Dependency

2.1.6 VARIABLES Group

The VARIABLES group in VMAP Standard I/O library contains sub-group <DEVICE-ID> and further STATE-<n>, within this lies the measurand (sMeasurandValues). Figure 2.6 shows a schematic of group VARIABLES.

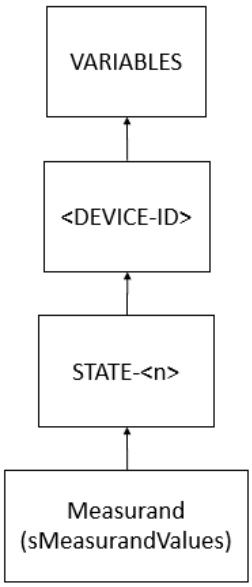


Figure 2.6: VARIABLES Group Dependency

2.1.7 POINTS Group

The POINTS group (sPointsBlock) in VMAP Standard I/O library requires data from the data set COORDINATESYSTEM (sCoordinateSystem). Figure 2.7 shows a schematic of group POINTS.

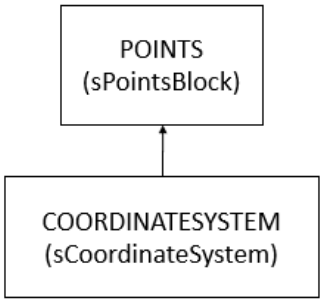


Figure 2.7: POINTS Group Dependency

2.1.8 ELEMENTS Group

The ELEMENTS group (sElementBlock) in VMAP Standard I/O library requires data from data set MYELEMENTS (sElement). Figure 2.8 shows a schematic of group ELEMENTS

2.1.9 MYELEMENTS Data Set

The MYELEMENTS data set (sElement) in VMAP Standard I/O library requires data from data sets ELEMENTTYPES (sElementType), POINTS ->MYIDENTIFIERS (sPointsBlock)

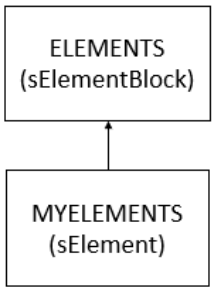


Figure 2.8: ELEMENTS Group Dependency

& `COORDINATESYSTEM` (`sCoordinateSystem`). Figure 2.9 shows a schematic of data set `MYELEMENTS`.

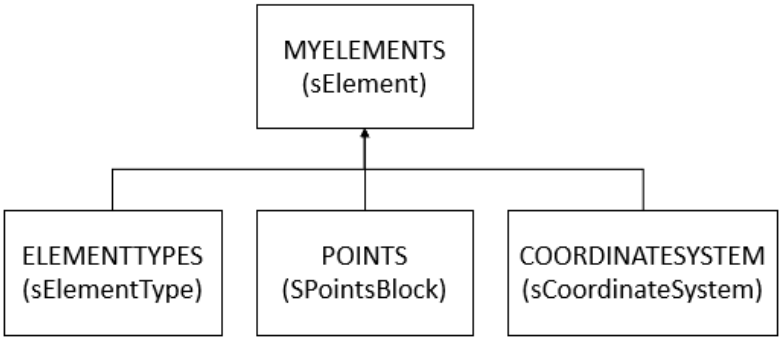


Figure 2.9: MYELEMENTS Dataset Dependency

2.1.10 `ELEMENTTYPES` Data Set

The `ELEMENTTYPES` data set (`sElementType`) in VMAP Standard I/O library requires data from `POINTS` (`sPointsBlock`). Figure 2.10 shows a schematic of data set `ELEMENTTYPES`.

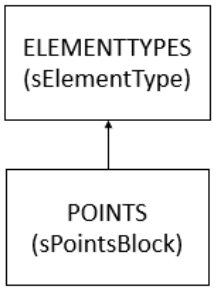


Figure 2.10: ELEMENTTYPES Dataset Dependency

2.1.11 `Measurand` Group

The Measurand group in VMAP Standard I/O library refers to all the measurands. Each measurand (sMeasurandValues) inherits data from `COORDINATESYSTEM` (`sCoordinateSystem`), `UNITS` (`sUnit`), and attribute `MYLOCATION` which could be global, a point, an element or an element face. Figure 2.11 shows a schematic of the Measurand group.

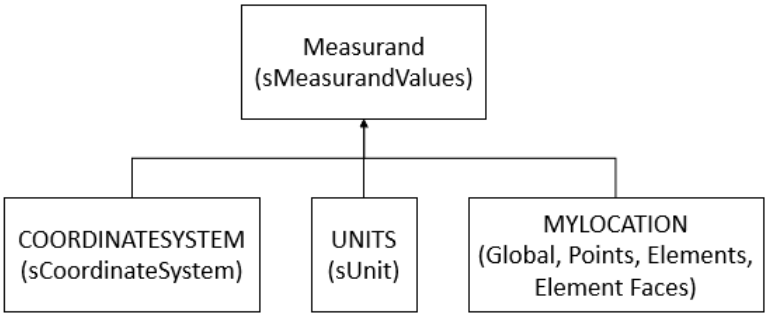


Figure 2.11: Measurand Group Dependency

2.1.12 `UNITSYSTEM` Attribute

The `UNITSYSTEM` attribute (`sUnitSystem`) in VMAP Standard I/O library requires data from Base Unit (`sBaseUnit`). Figure 2.12 shows a schematic of attribute `UNITSYSTEM`.

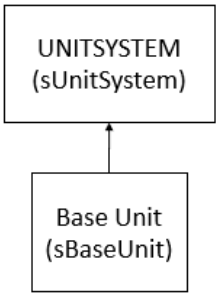


Figure 2.12: UNITSYSTEM Dependency

Chapter 3

Sensor Data Standard API

This chapter contains additional references that come with the sensor data integration.

3.1 VMAP Namespace Reference

3.1.1 Classes

- class sMultiParameterObject
- class sMeasurandValues
- class sMeasurementDevice

3.1.2 Variables

- static const char* GROUP_MEASUREMENT = "MEASUREMENT";
- static const char* GROUP_MEASUREMENT_SYSTEM = "MEASUREMENT/SYSTEM";
- static const char* GROUP_MEASUREMENT_GEOMETRY = "MEASUREMENT/GEOMETRY";
- static const char* GROUP_MEASUREMENT_VARIABLES = "MEASUREMENT/VARIABLES";

3.2 src/VMAP.h File Reference

3.2.1 Classes

- struct VMAP::sMultiParameterObject
- struct VMAP::sMeasurandValues
- struct VMAP::sMeasurementDevice

3.3 src/VMAPFile.h File Reference

3.3.1 Functions

- `void createMeasurementGroups();`

  Creates base groups for a measurement output

- `std::string createMeasurementGeometryGroup(int measurementDeviceId);`

  Creates a sub folder in /VMAP/MEASUREMENT/GEOMETRY/

- `std::string createMeasurementVariablesGroup(int measurementDeviceId, int stateId, int measurandValuesFormat);`

  Creates a sub folder in /VMAP/MEASUREMENT/VARIABLE/

- `void setMeasurementVariablesInformation(int measurementDeviceId, const std::string & timeStamp);`

  Sets time stamp information in folder /VMAP/MEASUREMENT/VARIABLE/<DEVICE-ID>

- `std::string getMeasurementVariablesInformation(int measurementDeviceId);`

  Gets time stamp information in folder /VMAP/MEASUREMENT/VARIABLE/<DEVICE-ID>

- `void writeMeasurementUnitSystem(const sUnitSystem & unitSystem);`

  Writes sUnitSystem to file; Creates dataset "UNITSYSTEM" in group "/VMAP/MEASUREMENT/SYSTEM"

- `void readMeasurementUnitSystem(sUnitSystem & unitSystem);`

  Reads dataset "UNITSYSTEM" from group "/VMAP/MEASUREMENT/SYSTEM"

- `void writeMeasurementUnits(const std::vector<VMAP::sUnit> & unitSystem);`

  Writes sUnitSystem to file; Creates dataset "UNITSYSTEM" in group "/VMAP/MEASUREMENT/SYSTEM"

- `void readMeasurementUnits(std::vector<VMAP::sUnit> & unitSystem);`

  Reads dataset "UNITSYSTEM" from group "/VMAP/MEASUREMENT/SYSTEM"

- `void writeMeasurementElementTypes(const std::vector<VMAP::sElementType> & types);`

Writes element types as DataSet "ELEMENTTYPES" to group "/VMAP/MEASUREMENT/SYSTEM"

- `void readMeasurementElementTypes(std::vector<VMAP::sElementType> & types);`

  Reads element types from group "/VMAP/MEASUREMENT/SYSTEM"

- `void writeMeasurementDevice(const std::string& groupName, const VMAP::sMeasurementDevice& device);`

  Writes measurement device as a group to path "groupName"

- `void readMeasurementDevice(const std::string& groupName, VMAP::sMeasurementDevice& device);`

  Reads measurement device from "groupName"

- `void writeMeasurandValues(const std::string& groupName, const VMAP::sMeasurandValues& variable);`

  Writes measurand values as a group to path "groupName"

- `void writeMeasurandValuesBlock(const std::string& groupName, const std::vector<VMAP::sMeasurandValues>& variables);`

  Writes a vector measurand values to path "groupName"

- `void readMeasurandValues(const std::string& groupName, VMAP::sMeasurandValues& variable);`

  Reads measurand values from group "groupName"

- `void readMeasurandValuesBlock(const std::string& groupName, std::vector<VMAP::sMeasurandValues>& variables);`

  Reads all measurand values from group "groupName"

- `void setMeasurementStateInformation(const std::string& groupName, int stateId, double runTime);`

  Creates state attributes for the group "groupName"

- `void getMeasurementStateInformation(const std::string& groupName, int& stateId, double& runTime);`

  Gets attributes of the state from group "groupName"

Chapter 4

Sensor Data Implementation Specification

This chapter can be seen as an addition to the VMAP structure. The Figures explain the hierarchy of sensor data storage in VMAP format. These Figures are colour coded with green, red and blue to differentiate among groups, attributes and datasets respectively. Additionally, the letters 'g', 'a' and 'd' are used as sub-scripts denoting group, attribute and dataset respectively; this facilitates a black-and-white print option without losing the differentiation.

4.1 `VMAP` Group

With the integration of sensor data, the hierarchy of the `VMAP` group changes. The `VMAP` group has one attribute and now contains two groups `MEASUREMENT` and `SIMULATION`, whereby `SIMULATION` corresponds to the former `VMAP` group from `v1.0.0`. The hierarchical model (from left to right) in Figure 4.1 shows a schematic of the basic `VMAP` storage structure. Tables 4.1 & 4.2 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 4.1: Object Attribute Info

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

Number of members:	2
Name	Type
SIMULATION	Group
MEASUREMENT	Group

Table 4.2: General Object Info

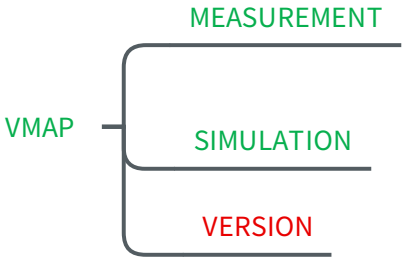


Figure 4.1: VMAP Group

4.1.1 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 4.3 shows the data types associated with VERSION metadata

Metadata	Data Type
myMajor	Integer
myMinor	Integer
myPatch	Integer

Table 4.3: Data Types of VERSION Metadata

4.2 MEASUREMENT Group

The MEASUREMENT group is divided into four groups (Figure 4.2). Each of these groups contains different aspects of the measurement data. Tables 4.4 show the Object Attribute Info and General Object Info of VMAP Group, these tables show information as seen in the HDF5 Viewer.

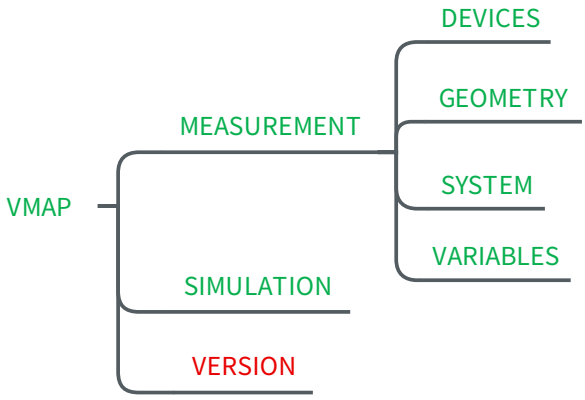


Figure 4.2: MEASUREMENT Group

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

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

Table 4.4: General Object Info

4.3 DEVICES Group

IN VMAP measurement data is saved according to the associated measuring device (measuring system). The DEVICES Group contains all of the measuring devices as a different sub-group DEVICE-<ID>. The groups associated with GEOMETRY are shown in Figure 4.3. Tables 4.5 show General Object Info of DEVICES Group, these tables show information as seen in the HDF5 Viewer.

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

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

Table 4.5: General Object Info

where n is the number of devices contributing to the measurement.

4.4 <DEVICE-ID> Group

<DEVICE-ID> Group stores the comprehensive metadata that provides additional information about a particular measuring device (measuring system). The metadata is then stored in one of the following sub-groups MEASUREDOBJECTS, COMPONENTS, SETUP,

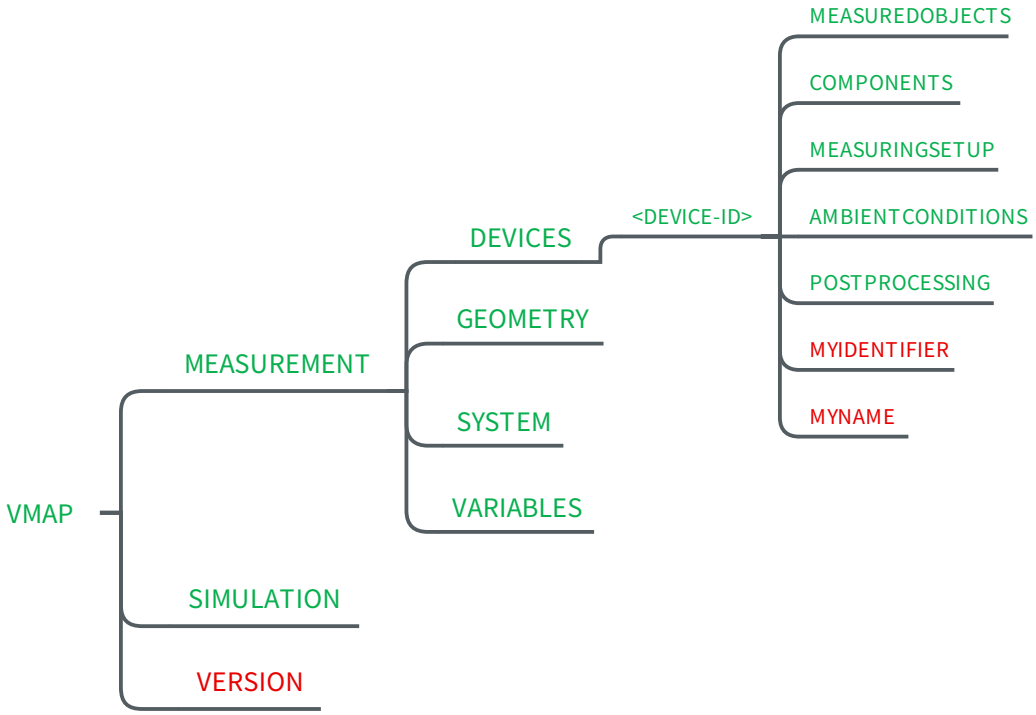


Figure 4.3: DEVICES Group

AMBIENTCONDITIONS, POSTPROCESSING. A detailed description of these terms is given in chapter 1.2. Tables 4.6 & 4.7 show the Object Attribute Info and General Object Info of <DEVICE-ID> Group, these tables show information as seen in the HDF5 Viewer.

Number of attributes:	2		
Name	Type	Array Size	Value
MYIDENTIFIER	Integer	Scalar	...
MYNAME	String	Scalar	...

Table 4.6: Object Attribute Info

Name:	<DEVICE-ID>
Path:	/VMAP/MEASUREMENT/DEVICES/
Type:	HDF5 Group
Object Ref:	...

Number of members:	5
Name	Type
MEASUREDOBJECTS	Group
COMPONENTS	Group
MEASURINGSETUP	Group
AMBIENTCONDITIONS	Group
POSTPROCESSING	Group

Table 4.7: General Object Info

4.4.1 MYIDENTIFIER Attribute

The MYIDENTIFIER attribute stores the unique identifier of the device. The following metadata is associated with MYIDENTIFIER:

- Value: Defines the identifier of the device.

Table 4.8 shows the data types associated with MYIDENTIFIER metadata

Metadata	Data Type
Value	Integer

Table 4.8: Data Types of MYIDENTIFIER Metadata

4.4.2 MYNAME Attribute

The MYNAME attribute stores the name of the device. The following metadata is associated with MYNAME:

- Value: Defines the name of the device.

Table 4.9 shows the data types associated with MYNAME metadata

Metadata	Data Type
Value	Integer

Table 4.9: Data Types of MYNAME Metadata

4.5 COMPONENTS Group

The <DEVICE-ID> sub-groups displayed in Figure 4.4 refer to different aspects of metadata associated to the particular measuring device (measuring system). Since all five sub-groups MEASUREDOBJECTS, COMPONENTS, SETUP, AMBIENTCONDITIONS, POSTPROCESSING share the same format, the COMPONENTS group is presented on behalf of all five sub-groups.

Components are parts of a measuring device. For a detailed description of terms refer to chapter 1.2. Tables 4.10 show the General Object Info of <DEVICE-ID> group, these tables show information as seen in the HDF5 Viewer.

Name:	COMPONENTS
Path:	/VMAP/MEASUREMENT/DEVICES/<DEVICE-ID>/
Type:	HDF5 Group
Object Ref:	...

Number of members:	n
Name	Type
<COMPONENT-n>	Dataset

Table 4.10: General Object Info

where n is the number of components in the meausering device (measuring system).

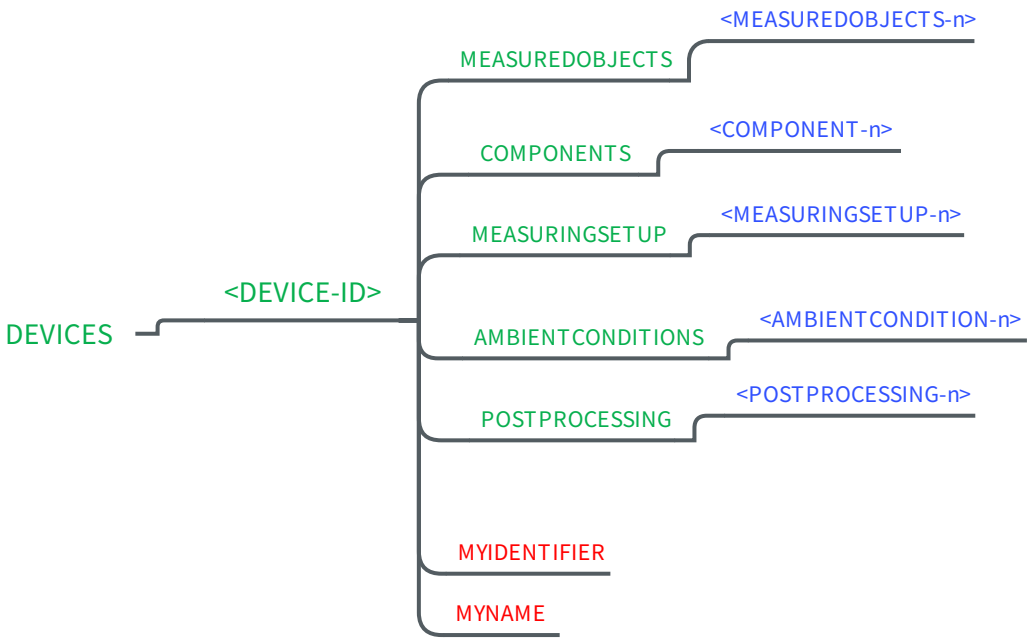


Figure 4.4: <DEVICE-ID> sub-groups

4.5.1 <COMPONENT-n> Dataset

The <COMPONENT-n> dataset stores information of a particular component. The datasets of the other <DEVICE-ID> sub-groups - <MEASUREDOBJECTS-n>, <MEASURINGSETUP-n>, <AMBIENTCONDITIONS-n> & <POSTPROCESSING-n> follow the same principles. The dataset contains parameters that are completely open to users. A parameter can be added to the dataset and is defined by the following items

- NAME: Stores a short form of the component parameter.
- VALUE: Stores the value of the corresponding component parameter.
- DESCRIPTION: Stores a description/long name of the corresponding component parameter.

This is a compound dataset. In the HDF5 Viewer, the columns are named as in the list above, see Table 4.11.

Metadata	Data Type
NAME	String
VALUE	String
DESCRIPTION	String

Table 4.11: Data Types of <COMPONENT-n> parameters metadata

The General Object Info associated with this dataset is shown in Table 4.12.

Name:	<COMPONENT-n>
Path:	/VMAP/MEASUREMENT/DEVICES/<DEVICE-ID>/COMPONENTS/
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:	Compound

Table 4.12: General Object Info

m depends on the number of parameters associated with the specific component.

4.6 GEOMETRY Group

GEOMETRY Group stores the geometrical data associated with the measurement trial, which includes points and elements. This group has no attribute information. The group <DEVICE-ID> represents one measuring device (measuring system) of the measurement trial. Thus, the points and elements belong to the respective <DEVICE-ID>. The groups associated with GEOMETRY are shown in Figure 4.5. Table 4.13 shows only General Object Info of GEOMETRY Group, these tables show information as seen in the HDF5 Viewer.

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

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

Table 4.13: General Object Info

where n is the number of measuring devices (measuring systems) contributing to the measurement.

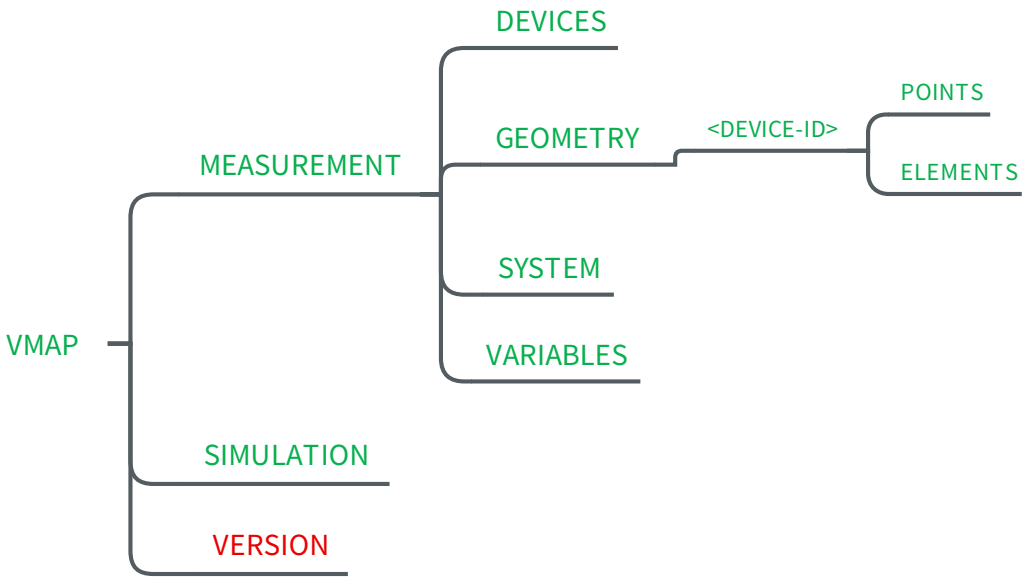


Figure 4.5: GEOMETRY Group

4.7 <DEVICE-ID> Group

<DEVICE-ID> Group stores the points and elements of a particular measuring device (measuring system). The groups POINTS & ELEMENTS are associated with <DEVICE-ID>. Tables 4.14 show Object Attribute Info and General Object Info of <DEVICE-ID> Group, these tables show information as seen in the HDF5 Viewer.

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

Number of members:	2
Name	Type
POINTS	Group
ELEMENTS	Group

Table 4.14: General Object Info

4.7.1 MYIDENTIFIER Attribute

The MYIDENTIFIER attribute stores the unique identifier of the device. The following metadata is associated with MYIDENTIFIER:

- Value: Defines the identifier of the device.

Table 4.15 shows the data types associated with MYIDENTIFIER metadata

Metadata	Data Type
Value	Integer

Table 4.15: Data Types of MYIDENTIFIER Metadata

4.7.2 MYNAME Attribute

The MYNAME attribute stores the name of the device. The following metadata is associated with MYNAME:

- Value: Defines the name of the device.

Table 4.16 shows the data types associated with MYNAME metadata

Metadata	Data Type
Value	Integer

Table 4.16: Data Types of MYNAME Metadata

4.8 POINTS Group

POINTS Group stores the points that are associated with the result of a particular device (measuring system), this includes X,Y,Z coordinates and the unique integral identifier for each point. The two attributes MYCOORDINATESYSTEM & MYSIZE and two datasets MYCOORDINATES & MYIDENTIFIERS are associated with POINTS, shown in Figure 4.6. Tables 4.17 & 4.18 show Object Attribute Info and General Object Info of POINTS Group, these tables show information as seen in the HDF5 Viewer.

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

Table 4.17: Object Attribute Info

Name:	POINTS
Path:	/VMAP/MEASUREMENT/GEOMETRY/<DEVICE-ID>/
Type:	HDF5 Group
Object Ref:	...

Number of members:	2
Name	Type
MYCOORDINATES	Dataset
MYIDENTIFIERS	Dataset

Table 4.18: General Object Info

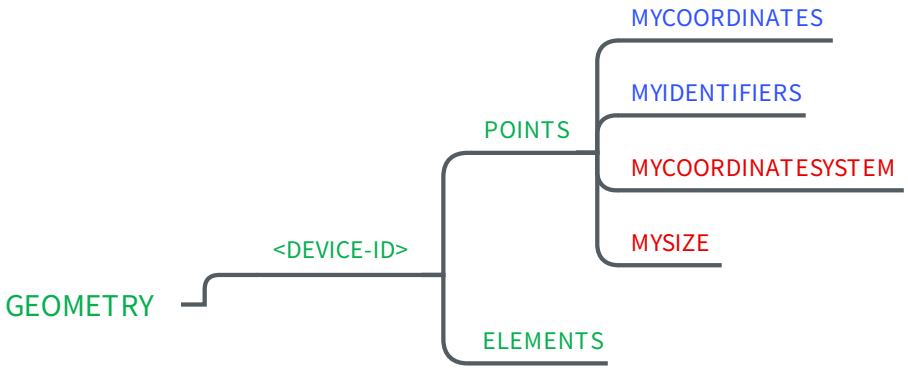


Figure 4.6: POINTS Group

4.8.1 `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 sub-section 4.14.1 for more details.

Metadata	Data Type
Value	Integer

Table 4.19: Data Types of MYCOORDINATESYSTEM Metadata

4.8.2 `MYSIZE` Attribute

This attribute contains the number of points associated with a particular measuring device (measuring system).

- Value: Stores the number of points.

Metadata	Data Type
Value	unsigned Integer

Table 4.20: Data Types of MYSIZE Metadata

4.8.3 `MYCOORDINATES` Dataset

The `MYCOORDINATES` dataset 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 4.21: Data Types of MYCOORDINATES Metadata

The `General Object Info` associated with this dataset is shown in Table 4.22.

Name:	`MYCOORDINATES`
Path:	`/VMAP/MEASUREMENT/GEOMETRY/<DEVICE-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 4.22: General Object Info

4.8.4 `MYIDENTIFIERS` Dataset

The `MYIDENTIFIERS` dataset 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 4.23: Data Types of MYIDENTIFIERS Metadata

The `General Object Info` associated with this dataset is shown in Table 4.24.

Name:	MYIDENTIFIERS
Path:	/VMAP/MEASUREMENT/GEOMETRY/<DEVICE-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 4.24: General Object Info

4.9 ELEMENTS Group

ELEMENTS Group stores the elements associated with a particular measuring device (measuring system), this includes identifier, type, coordinate system and element connectivity. The attribute MYSIZE and dataset MYELEMENTS are associated with ELEMENTS, shown in Figure 4.7. Tables 4.25 & 4.26 show Object Attribute Info and General Object Info of ELEMENTS Group, these tables show information as seen in the HDF5 Viewer.

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

Table 4.25: Object Attribute Info

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

Number of members:	1
Name	Type
MYELEMENTS	Dataset

Table 4.26: General Object Info

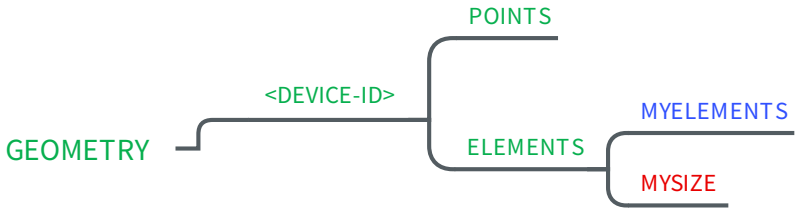


Figure 4.7: ELEMENTS Group

4.9.1 `MYSIZE` Attribute

This attribute contains the number of elements associated with a particular measuring device (measuring system).

- Value: Stores the number of elements/the size of the dataset `MYELEMENTS`.

Metadata	Data Type
Value	unsigned Integer

Table 4.27: Data Types of MYSIZE Metadata

4.9.2 `MYELEMENTS` Dataset

The `MYELEMENTS` dataset 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 4.14.2.
- `myCoordinateSystem`: Stores the coordinate system reference number. For more details about `/SYSTEM/COORDINATESYSTEM`, see sub-section 4.14.1.
- `myConnectivity`: Stores the point number ordering which forms one element.

This is a compound dataset.

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

Table 4.28: Data Types of MYELEMENTS Metadata

The `General Object Info` associated with this dataset is shown in Table 4.29.

Name:	`MYELEMENTS`
Path:	`/VMAP/MEASUREMENT/GEOMETRY/<COMPONENT-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 4.29: General Object Info

4.10 `VARIABLES` Group

`VARIABLES` Group stores all results of a measurement trial. Figure 4.8 shows the complete storage structure of the `VARIABLES` Group. Table 4.30 shows the `General Object Info` of `VARIABLES` Group, the table shows information as seen in the HDF5 Viewer.

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

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

Table 4.30: General Object Info

where n is the number of measuring devices (measuring systems) contributing to the measurement.

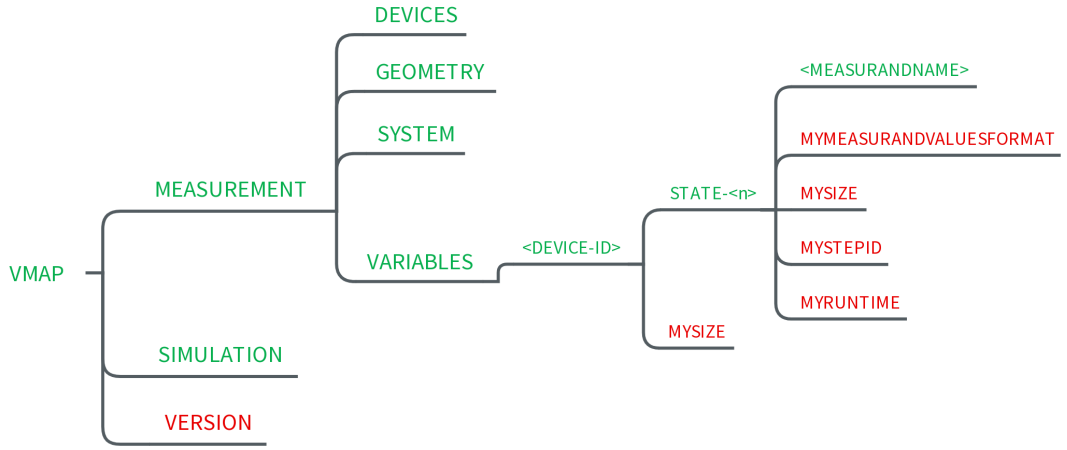


Figure 4.8: VARIABLES Group

4.11 <DEVICE-ID> Group

<DEVICE-ID> Group corresponds to the <DEVICE-ID> Group defined in the DEVICE & GEOMETRY Groups. Thus, all datasets located in Groups <DEVICE-ID> refer to the same measuring device (measuring system), whether they are stored inside the DEVICES, GEOMETRY or VARIABLES group. Table 4.31 & 4.32 show Object Attribute Info and General Object Info of <PART-ID> Group, these tables show information as seen in the HDF5 Viewer.

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

Table 4.31: Object Attribute Info

Name:	<DEVICE-ID>
Path:	/VMAP/MEASUREMENT/VARIABLES/
Type:	HDF5 Group
Object Ref:	...

Number of members:	n
Name	Type
STATE-<n>	Group

Table 4.32: General Object Info

where n is the number of measuring time steps from that particular measuring device (measuring system).

4.11.1 MYTIMESTAMP Attribute

This attribute contains the absolute timestamp that marks the beginning of the measuring process from that particular measuring device (measuring system). The format of the timestamp is given by ISO 8601, e. g. `YYYY-MM-DDThh:mm:ss.ms`.

- Value: Stores the measuring begin of a device.

Metadata	Data Type
Value	string

Table 4.33: Data Types of MYTIMESTAMP Metadata

4.12 STATE-<n> Group

`STATE-<n>` Group hold the results at a certain time step from a particular measuring device (measuring system). The four attributes, to store the state metadata, and group(s) `<MEASURANDNAME>`, with results for this time step, are associated with `STATE-<n>`. Table 4.34 & Table 4.35 show `Object Attribute Info` and `General Object Info` of `STATE-<n>` Group, these tables show information as seen in the HDF5 Viewer.

Number of attributes:	4		
Name	Type	Array Size	Value
MYMEASURANDVALUESFORMAT	Integer	Scalar	...
MYSIZE	Integer	Scalar	...
MYSTEPID	Integer	Scalar	...
MYRUNTIME	Floating-Point	Scalar	...

Table 4.34: Object Attribute Info

Name:	STATE-<n>
Path:	/VMAP/MEASUREMENT/VARIABLES/<DEVICE-ID>/
Type:	HDF5 Group
Object Ref:	...

Number of members:	MYSIZE
Name	Type
TEMPERATURE	Group

Table 4.35: General Object Info

4.12.1 MYMEASURANDVALUESFORMAT Attribute

For each measuring device it has to be stated, whether the device suits format 1 or 2.

- 1: For each time step the result consists of many measuring points (e. g. three-dimensional optical measuring systems, ...): Values are stored as MYVALUES Dataset.
- 2: For each time step there is only one or a few measuring points (e. g. strain gauge, thermocouple, ...): Values are stored as TABLES Dataset.

This attribute stores the measurand values format.

- Value: Stores the measurand values format of the STATE-<n>.

Metadata	Data Type
Value	Integer

Table 4.36: Data Types of MYMEASURANDVALUESFORMAT Metadata

4.12.2 MYSIZE Attribute

This attribute refers to the number of different measurands from a particular measuring device at a specific time step.

- Value: Storesthe number of measurands.

Metadata	Data Type
Value	Integer

Table 4.37: Data Types of MYSIZE Metadata

4.12.3 MYSTEPID Attribute

This attribute stores the unique identifier of the specific time step.

- Value: Stores the unique identifier of the STATE-<n>.

Metadata	Data Type
Value	Integer

Table 4.38: Data Types of MYSTEPID Metadata

4.12.4 MYRUNTIME Attribute

This attribute contains the relative time of the time steps in relation to the absolute timestamp from given by attribute MYTIMESTAMP in group /VARIABLES/<DEVICE-ID>. The time unit is given by the definition from /SYSTEM/UNITSYSTEM.

- Value: Stores the time at STATE-<n> in relation to the begin of the measurement.

Metadata	Data Type
Value	Floating-Point

Table 4.39: Data Types of MYRUNTIME Metadata

4.13 TEMPERATURE Group

TEMPERATURE is used as an example to show how any measurement result can be stored in VMAP. The TEMPERATURE Group stores data about the temperature measured by a measuring device (measuring system) at a given timestamp. There are 11 attributes and two datasets associated with TEMPERATURE or with any other measuring result. The attributes are shown in table 4.40 and explained in the following sub-sections. The datasets MYVALUES[1], TABLES[2] & MYGEOMETRYIDS* are associated with TEMPERATURE or with any other measurand, shown in Figure 4.9. Table 4.40 & 4.41 show Object Attribute Info and General Object Info of TEMPERATURE Group, these tables show information as seen in the HDF5 Viewer.

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	...
MYMEASURANDDESCRIPTION	String	Scalar	...
MYMEASURANDNAME	String	Scalar	...
MYMULTIPLICITY	Integer	Scalar	...
MYTIMEVALUE	Floating-Point	Scalar	...
MYUNIT	Integer	Scalar	...

Table 4.40: Object Attribute Info

Name:	TEMPERATURE
Path:	/VMAP/MEASUREMENT/VARIABLES/<DEVICE-ID>/<STATE-n>/
Type:	HDF5 Group
Object Ref:	...

Number of members:	2
Name	Type
MYVALUES[1]	Dataset
TABLE[2]	Dataset
MYGEOMETRYIDS*	Dataset

Table 4.41: General Object Info ([1,2] depends on whether attribute MYMEASURANDVALUESFORMAT is 1 or 2, * is optional)

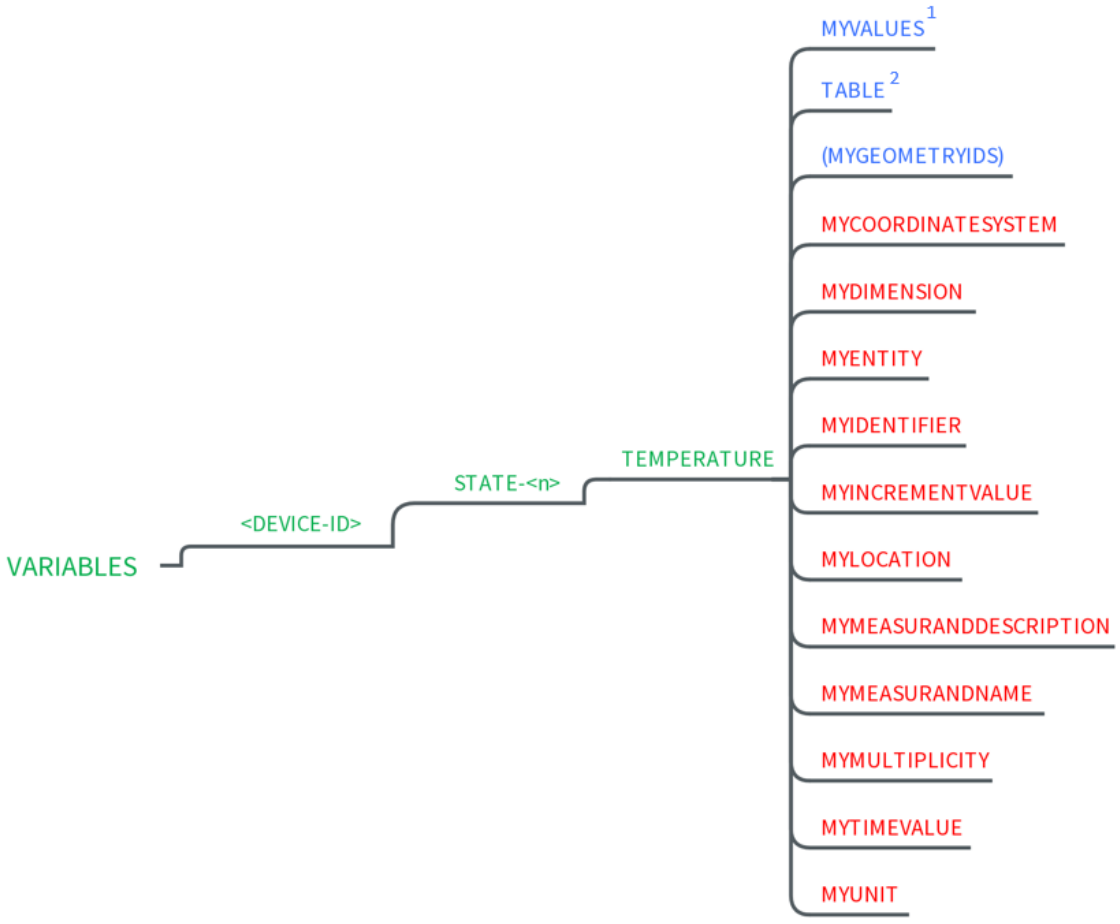


Figure 4.9: TEMPERATURE Group

4.13.1 MYCOORDINATESYSTEM Attribute

Refers to the coordinate system used for the measurand. The details of this coordinate system can be found in /SYSTEM/COORDINATESYSTEM, see sub-section 4.14.1 for more details.

- Value: Stores the coordinate system reference number.

Metadata	Data Type
Value	Integer

Table 4.42: Data Types of MYCOORDINATESYSTEM Metadata

4.13.2 MYDIMENSION Attribute

Refers to the dimension of the measurand TEMPERATURE.

- Value: Stores the reference number of MYDIMENSION.

Metadata	Data Type
Value	Integer

Table 4.43: Data Types of MYDIMENSION Metadata

Table 4.44 shows the available dimensions and their reference numbers in VMAP.

MYDIMENSION	Reference Number	Storage Format in VMAP
INVALID	0	
SCALAR	1	single value
VECTOR	3	v_X, v_Y, v_Z
2nd Order Plain Tensor Symmetric	4	t_XX, t_YY, t_ZZ, t_XY
2nd Order Tensor Symmetric	6	t_XX,t_YY,t_ZZ,t_XY, t_YZ,t_XZ
2nd Order Tensor	9	t_XX,t_YY,t_ZZ,t_XY, t_YZ,t_XZ,t_YX,t_ZY,t_ZX
STIFFNESS MATRIX	36	Voigt notation: t_11, t_22, t_33, . . . , t_66, t_12, t_23, . . . , t_56, t_13, t_24,. . . , t_16
4th Order Tensor Symmetric	45	[[t_1111,t_1122,t_1133,..,t_1121], [t_2222,t_2233,. . . ,t_2221] ,..,[t_2121]]
4th Order Tensor	81	[[t_1111,t_1122,t_1133,. . . ,t_1121], [t_2211,. . . ,t_2221] ,..,[t_2111,. . . ,t_2121]]

Table 4.44: MYDIMENSION Enumeration

4.13.3 `MYENTITY` Attribute

Refers to the entity of the measurand `TEMPERATURE`.

- Value: Stores the reference number of `MYENTITY`

Metadata	Data Type
Value	Integer

Table 4.45: Data Types of MYENTITY Metadata

Table 4.46 shows the available entities and their reference numbers in VMAP

MYENTITY	Reference Number
REAL	1
COMPLEX	2
HAMILTONIAN	4

Table 4.46: MYENTITY Enumeration

4.13.4 `MYIDENTIFIER` Attribute

Refers to the unique integral identifier for the measurand `TEMPERATURE`.

- Value: Stores the unique identifier.

Metadata	Data Type
Value	Integer

Table 4.47: Data Types of MYIDENTIFIER Metadata

4.13.5 `MYINCREMENTVALUE` Attribute

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

- Value: Stores the step value for the measurand.

Metadata	Data Type
Value	Integer

Table 4.48: Data Types of MYINCREMENTVALUE Metadata

4.13.6 `MYLOCATION` Attribute

Refers to the location where the measurand `TEMPERATURE` is stored.

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

Metadata	Data Type
Value	Integer

Table 4.49: Data Types of MYLOCATION Metadata

Table 4.50 shows the available locations and their reference numbers in VMAP

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

Table 4.50: MYLOCATION Enumeration

4.13.7 `MYMEASURANDDESCRIPTION` Attribute

If result is stored as `MYVALUES`[1] Dataset attribute `MYMEASURANDDESCRIPTION` gives a detailed description of the measurand.
If result is stored as `TABLES`[2] Dataset attribute `MYMEASURANDDESCRIPTION` defines the name of the `TABLES`[2] Dataset columns. If the measurand is `TEMPERATURE`, the attribute should be defined as `TIME:TEMPERATURE`.

- Value: Stores detailed description of the measurand or defines the name of the column based on the measurand values format.

Metadata	Data Type
Value	String

Table 4.51: Data Types of MYMEASURANDDESCRIPTION Metadata

4.13.8 `MYMEASURANDNAME` Attribute

Stores name of the measurand.

- Value: Stores name of the measurand.

Metadata	Data Type
Value	String

Table 4.52: Data Types of MYMEASURANDNAME Metadata

4.13.9 MYMULTIPLICITY Attribute

Refers to the number of columns in the MYVALUES[1] & TABLES[2] dataset. The majority of measurands have multiplicity 1.

- Value: Stores the multiplicity of the MYVALUES[1] & TABLES[2] dataset.

Metadata	Data Type
Value	Integer

Table 4.53: Data Types of MYMULTIPLICITY Metadata

4.13.10 MYTIMEVALUE Attribute

This attribute contains the relative time of the measured value in relation to the absolute timestamp from the measuring device given in attribute MYTIMESTAMP. The time unit is given by the definition from /SYSTEM/UNITSYSTEM.

- Value: Stores the time value of the measurand TEMPERATURE.

Metadata	Data Type
Value	Floating-Point

Table 4.54: Data Types of MYTIMEVALUE Metadata

4.13.11 MYUNIT Attribute

Refers to the unit used by the measurand. The details of this unit can be found in either /SYSTEM/UNITSYSTEM or /SYSTEM/UNITS, see sub-section 4.14.4 & 4.14.3 respectively. If a base unit is used then it can be directly referred from SYSTEM/UNITSYSTEM dataset and if a derived or a scaled unit needs to be used, it should be defined in /SYSTEM/UNITS dataset. If a measurand is unitless without any particular unit name, then use -1 here. However, if a unit needs to be defined, which is unitless e.g. radians, then define it in the MEASUREMENT/SYSTEM/UNITS dataset and refer it here.

- Value: Stores the reference number of the unit.

Metadata	Data Type
Value	Integer

Table 4.55: Data Types of MYUNIT Metadata

4.13.12 MYVALUES[1] Dataset

The values of the measurement result is stored as MYVALUES[1] Dataset if the measurand values format is of type 1 (refer to subsection 4.12.1). 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 4.56).

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

Metadata	Data Type
myvalue	Floating-Point

Table 4.56: Data Types of MYVALUES Metadata

The General Object Info associated with this dataset is shown in Table 4.57.

Name:	MYVALUES
Path:	/VMAP/MEASUREMENT/VARIABLES/<DEVICE-ID>/STATE-<n>/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 4.57: General Object Info

m given in Table 4.57 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 = 5 (=Element Face) then m = SUM of all ELEMENT FACES

4.13.13 TABLE^2 Dataset

The values of the measurement result is stored as TABLE^2 Dataset if the measurand values format is of type 2 (refer to subsection 4.12.1). The number of columns can be determined based on MYDIMENSION, Number of measuring points and MYMULTIPLICITY.

In the HDF5 Viewer, the columns are named according to the column names defined in attribute MYMEASURANDDESCRIPTION. If MYMEASURANDDESCRIPTION is TIME:TEMPERATURE and the number of points is 2, the columns are named as shown in Table 4.58.

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

Metadata	Data Type
TIME	Floating-Point
TEMPERATURE-<1>	Floating-Point
TEMPERATURE-<2>	Floating-Point

Table 4.58: Data Types of TABLES Metadata

The `General Object Info` associated with this dataset is shown in Table 4.59.

Name:	`TABLE`
Path:	`/VMAP/MEASUREMENT/VARIABLES/<DEVICE-ID>/STATE-<n>/TEMPERATURE/`
Type:	HDF5 Dataset
Object Ref:	...

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

Table 4.59: General Object Info

n refers to the number of time steps defined for the particular measuring device (measuring system). m is the number of columns. The first columns is allways the relative time of the measured value in relation to the absolute timestamp. The time unit is given by the definition from `/SYSTEM/UNITSYSTEM`.

m given in Table 4.59 is the number of the TABLES columns. When,
`MYLOCATION` = 2 then m = `MYSIZE`(points) × `MYDIMENSION` + 1
`MYLOCATION` = 3 then m = `MYSIZE`(elements) × `MYDIMENSION` + 1
`MYLOCATION` = 5 then m = SUM of all `ELEMENT FACES` × `MYDIMENSION` + 1

4.13.14 `MYGEOMETRYIDS` Dataset - Optional

This dataset is optional and should be used only when a set of the points, elements or elements faces are used. The set is then defined in this dataset. `MYLOCATION` attribute at the measurand level is used to identify the set type (points, element or elements faces). In the HDF5 Viewer, the column is named by default as `0` for myvalue (shown in Table 4.60).

Metadata	Data Type
myvalue	Integer

Table 4.60: Data Types of MYGEOMETRYIDS Metadata

The `General Object Info` associated with this dataset is shown in Table 4.61.

Name:	MYGEOMETRYIDS
Path:	/VMAP/MEASUREMENT/VARIABLES/<DEVICE-ID>/STATE-<n>/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:	32-bit Integer

Table 4.61: General Object Info

m = number of values defined in the set.

4.14 SYSTEM Group

SYSTEM Group stores the system data related to the measurement trial. This includes four datasets - COORDINATESYSTEM, ELEMENTTYPES, UNITS & UNITSYSTEM, shown in Figure 4.10. Table 4.62 shows General Object Info of SYSTEM Group, the table shows information as seen in the HDF5 Viewer.

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

Number of members:	4
Name	Type
COORDINATESYSTEM	Dataset
ELEMENTTYPES	Dataset
UNITS	Dataset
UNITSYSTEM	Dataset

Table 4.62: General Object Info

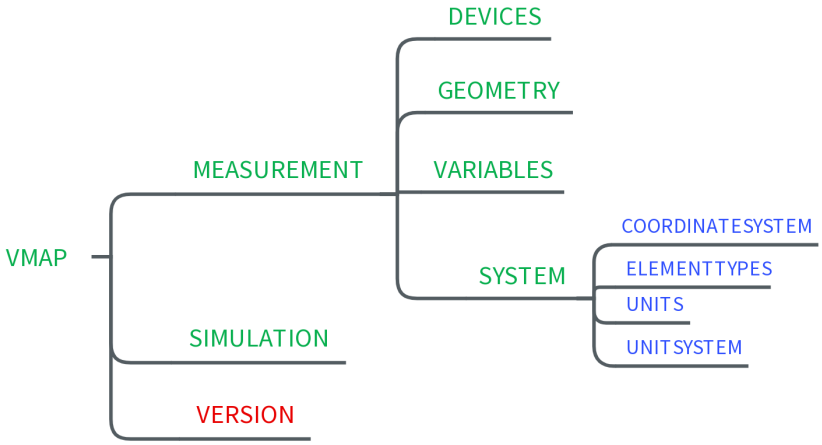


Figure 4.10: SYSTEM Group

4.14.1 `COORDINATESYSTEM` Dataset

The `COORDINATESYSTEM` dataset stores the global coordinate systems used by the different measuring devices. 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 4.63.
- `myReferencePoint`: Stores the 3D reference point for system definition, $[0, 0, 0]$ is default.
- `myAxisVector`: Defines up to three vectors describing the coordinate system $[u_1, u_2, u_3, v_1, v_2, v_3, w_1, w_2, w_3]$.

This is a compound dataset.

Metadata	**Data Type**
myIdentifier	Integer
myType	Integer
myReferencePoint	Floating-Point Array
myAxisVector	Floating-Point Array

Table 4.63: Data Types of COORDINATESYSTEM Metadata

Table 4.64 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 4.64: COORDINATESYSTEM Enumeration

The General Object Info associated with this dataset is shown in Table 4.65.

Name:	`COORDINATESYSTEM`
Path:	`/VMAP/MEASUREMENT/SYSTEM/`
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:	Compound

Table 4.65: General Object Info

m refers to number of coordinate systems defined for the measurement trial.

4.14.2 `ELEMENTTYPES` Dataset

The `ELEMENTTYPES` dataset stores the various types of elements used in the measurement. The metadata is stored in the following order:

- `myIdentifier`: Integral identifier for each element type used in the measurement.
- `myTypeName`: Stores the type of the element. VMAP offers a factory of various element types, which can be directly incorporated in the code.
- `myTypeDescription`: Stores a more generic element type description. Here are some of the type descriptions, THERMAL, STRUCTURAL, ELECTRICAL, ACOUSTIC, ELECTROMAGNETIC, PIEZOELECTRIC, FLUID, PARTICLE, HEAT-TRANSFER, CONVECTION, RIGID, CONTACT, MEMBRANE, SPRING, PLANESTRESS, PLANESTRAIN, USERDEFINED etc. These types can also be combined e.g. STRUCTURAL/THERMAL. This field is not fixed, users can assign any relevant value.
- `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 4.68 for more details on Interpolation types.
- `myIntegrationType`: Stores the reference number of the integration type used by the element type. For measurements this value is `-1` (invalid).
- `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 dataset.

Metadata	**Data Type**
`myIdentifier`	Integer
`myTypeName`	Character Pointer
`myTypeDescription`	Character Pointer
`myNumberofNodes`	Integer
`myDimension`	Integer
`myShapeType`	Integer
`myInterpolationType`	Integer
`myIntegrationType`	Integer
`myNumberofNormalComponents`	Integer
`myNumberofShearComponents`	Integer
`myConnectivity`	Integer Array
`myFaceConnectivity`	Integer Array

Table 4.66: Data Types of ELEMENTTYPES Metadata

The General Object Info associated with this dataset is shown in Table 4.67.

Name:	ELEMENTTYPES
Path:	/VMAP/MEASUREMENT/SYSTEM/
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:	Compound

Table 4.67: General Object Info

m refers to number of element types defined for the measurement.

Table 4.68 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 4.68: myInterpolationType Enumeration

4.14.3 UNITS Dataset

The UNITS dataset stores the various types of units used in the measurement trial. The metadata is stored in the following order:

- myIdentifier: Integral identifier for each unit used in the measurement. It is always > 7 because the first 7 identifiers belong to the UNITSYSTEM dataset.
- 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 4.14.4. It is an array with length 7.

This is a compound dataset. Please note that to be able to store unitless values which have a unit symbol, myUnitDimension should be assigned all $0s$. Hence, if a percentage needs

to be stored then, make sure to use the decimal value and not the percentage itself e.g. to store 30%, please store it as 0.3.

Metadata	Data Type
`myIdentifier`	Integer
`myUnitSymbol`	Character Pointer
`myUnitDimension`	Integer Array

Table 4.69: Data Types of UNITS Metadata

The General Object Info associated with this dataset is shown in Table 4.70.

Name:	`UNITS`
Path:	`/VMAP/MEASUREMENT/SYSTEM/`
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:	Compound

Table 4.70: General Object Info

m refers to number of units defined for the measurement.

4.14.4 `UNITSYSTEM` Dataset

The `UNITSYSTEM` dataset 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 '`m`' - meter.
  - `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 'kg' - kilogram.
  - 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 dataset.

Note: The following format is used, when converting the unit system. Conversion factor to SI unit, $SI = \texttt{mySIScale} \times value + \texttt{mySIShift}$, `mySIScale` has a default value of 1 & `mySIShift` has a default value of 0.

Table 4.71 shows the data type associated with `UNITSYSTEM` sub-fields.

Metadata	**Data Type**
`myIdentifier`	Integer
`mySIScale`	Floating-Point
`mySIShift`	Floating-Point
`myUnitSymbol`	Character Pointer
`myUnitQuantity`	Character Pointer

Table 4.71: Data Types of UNITSYSTEM metadata

The `General Object Info` associated with this dataset is shown in Table 4.72.

Name:	UNITSYSTEM
Path:	/VMAP/MEASUREMENT/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 4.72: General Object Info

Chapter 5

VMAP Sensor Data Example

5.1 Description

The following chapter intends to illustrate the application of the concept using measurement data from the blow moulded plastic pipe. The aim of this measurement trial is to gain insights into the behavior of blow moulded parts during cooling. For this purpose, the plastic pipe is first heated up in a chamber, which is then removed. After removing the part from the chamber, an stereoscopic measuring system (measuring device) is used to measure the surface temperature and shrinkage until the pipe has cooled down to room temperature. Some details of the measuring process are given in Figure 5.1). Temperature sensors are added to the measurement as a second measuring system at two points inside the pipe. Figure 5.2 shows how the measuring processes of both measurement systems can be transferred into the concept presented in chapter 1.1.

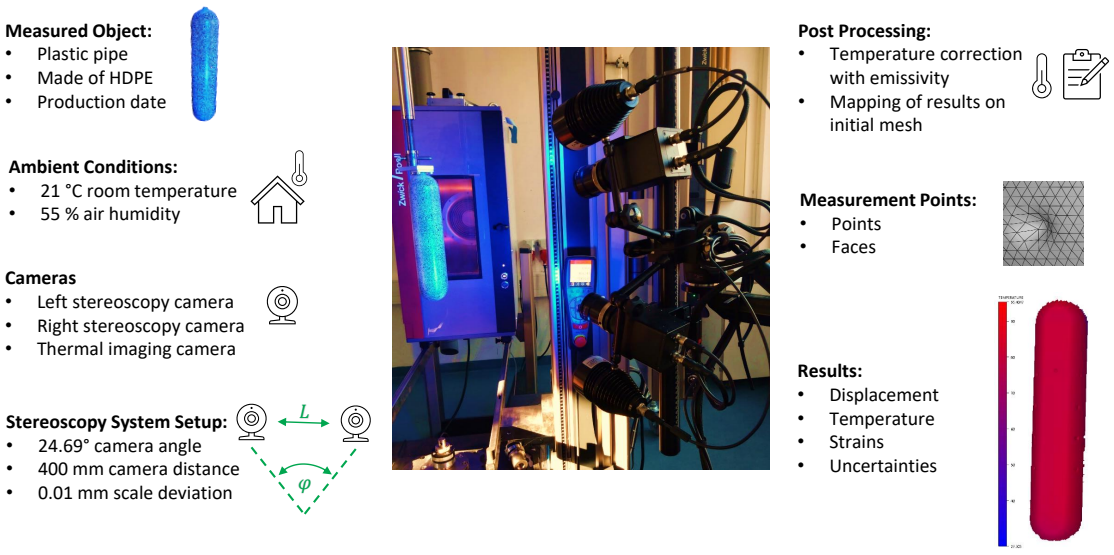


Figure 5.1: Stereoscopic measuring device from the blow moulding example

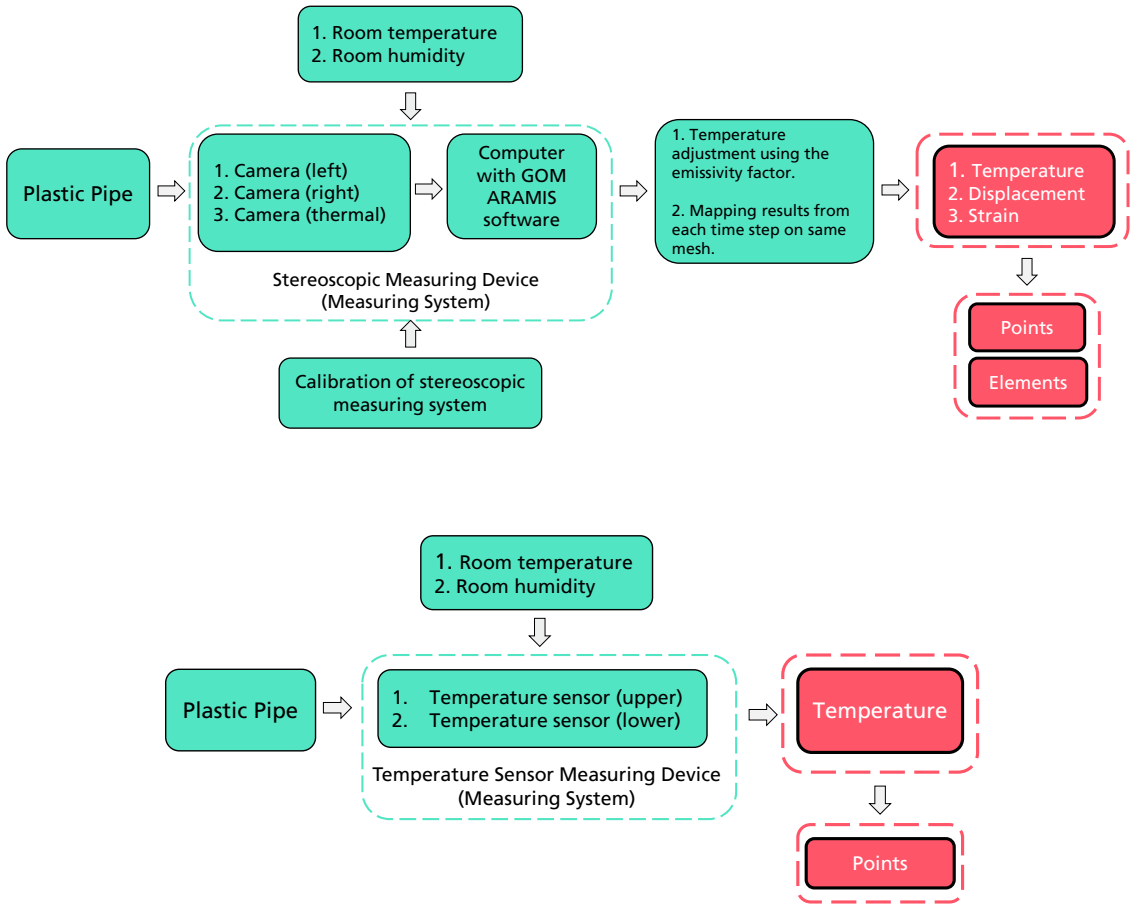


Figure 5.2: Measuring process of both devices (systems)

5.2 Measurement Data

The measurement data from the blow moulding example consists of several files that are to be transferred to a common VMAP file (see Figure 5.3). The 3D measurement results of the stereoscopic measuring system include one file in `ply` format for each measurand (temperature, strain-x, strain-y) and each timestamp (100 time steps), which makes a total of 300 files. The results of the temperature sensors inside the pipe are available as a `csv` table. The datasets of both measuring systems have an additional file `config.json` included that contains all of the metadata.

5.3 VMAP Structure

This chapter illustrates how the data from the blow moulding example is stored within the VMAP structure. In this example, data from the optical measurement system is stored as device with reference index 1. Data from the temperature sensors is stored as device with reference index 2.

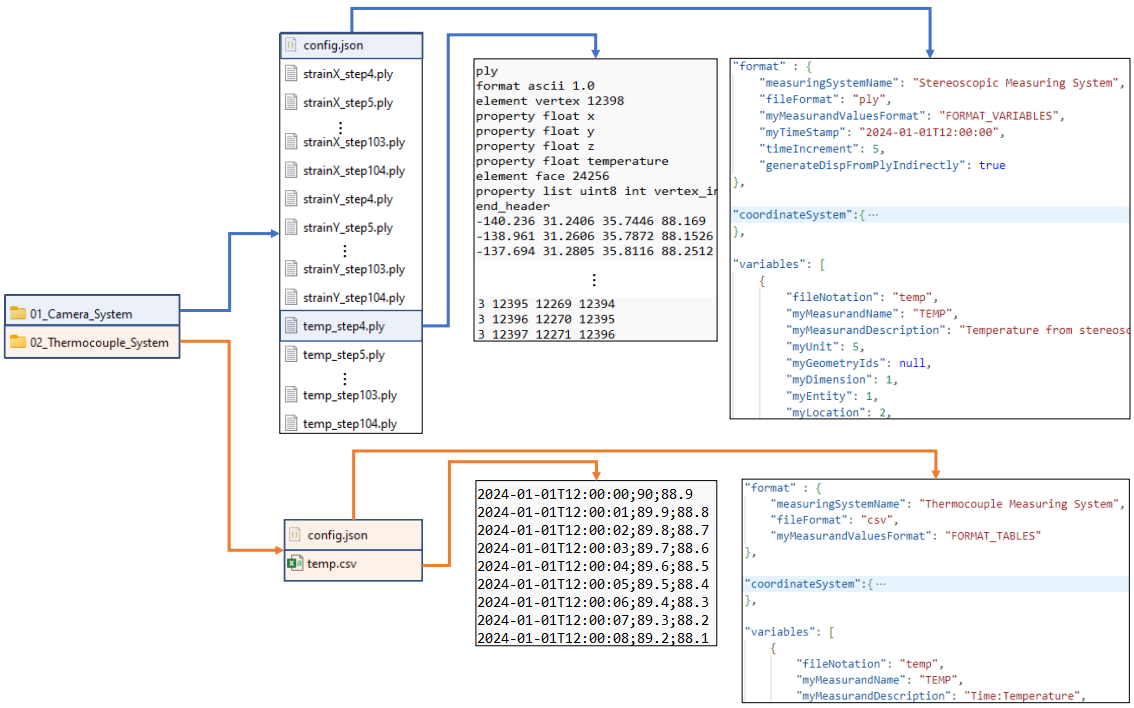


Figure 5.3: Overview of the measurement data from the example

5.3.1 `DEVICE` Group

Figure 5.4 illustrates how metadata data from the optical measurement is stored within the `DEVICE` group. For each subgroup a representative dataset is shown. These datasets provide additional descriptions of the measurement to support users with understanding the available results.

5.3.2 `GEOMETRY` Group

Figure 5.5 shows how the geometry associated with the stereoscopic measuring system can be stored inside the `GEOMETRY` group.

5.3.3 `VARIABLES` Group

Figure 5.6 and 5.7 show how the measurement results of both measuring devices can be stored by using the VMAP structure. The stereoscopic measurement system and the temperature sensor system have different formats. The first measurement system has the measurement format 1, which is why the results are saved in the `MYVALUES` dataset. The second measurement system has measurement format 2, which is why the results are saved in the `TABLE` dataset.

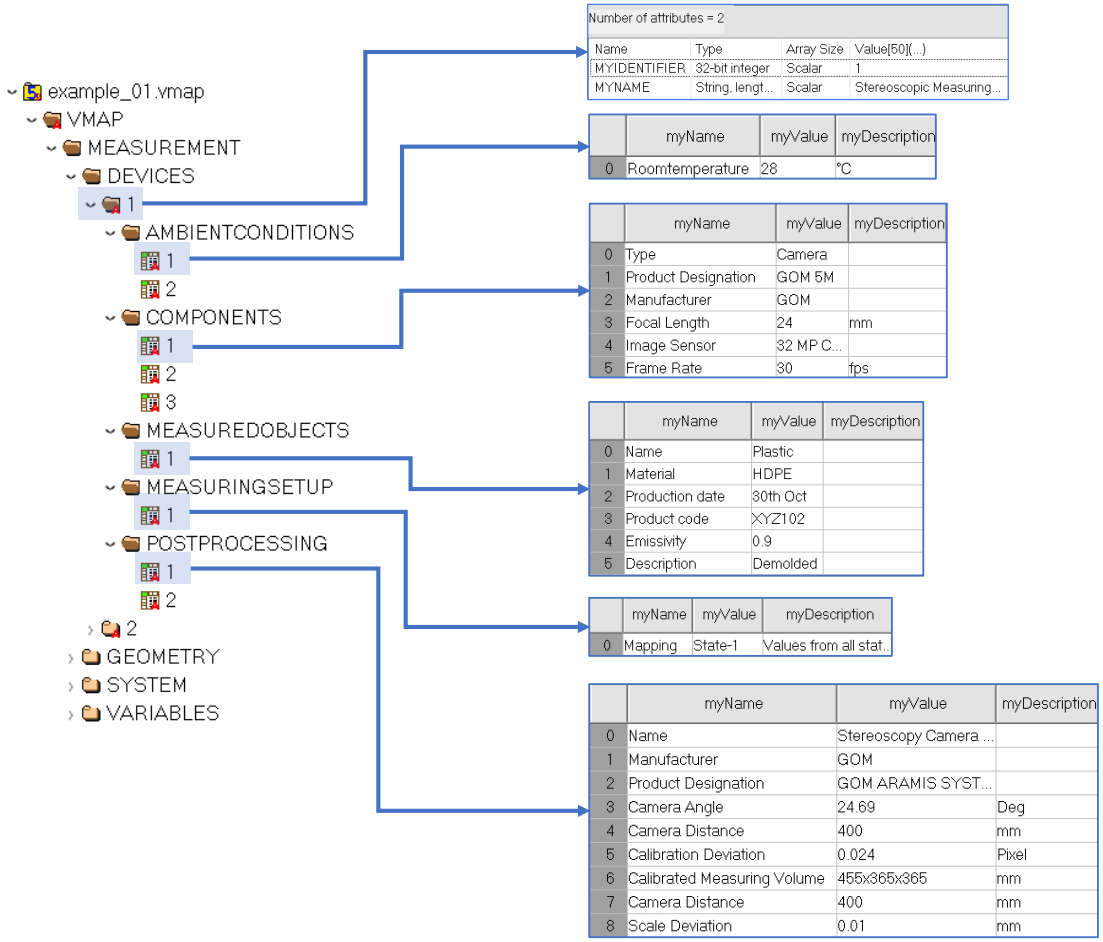


Figure 5.4: Store metadata from the measurement example inside the `DEVICE` group

5.3.4 `SYSTEM` Group

Figure 5.8 shows how general information of the measurement trial can be stored inside the `SYSTEM` group.

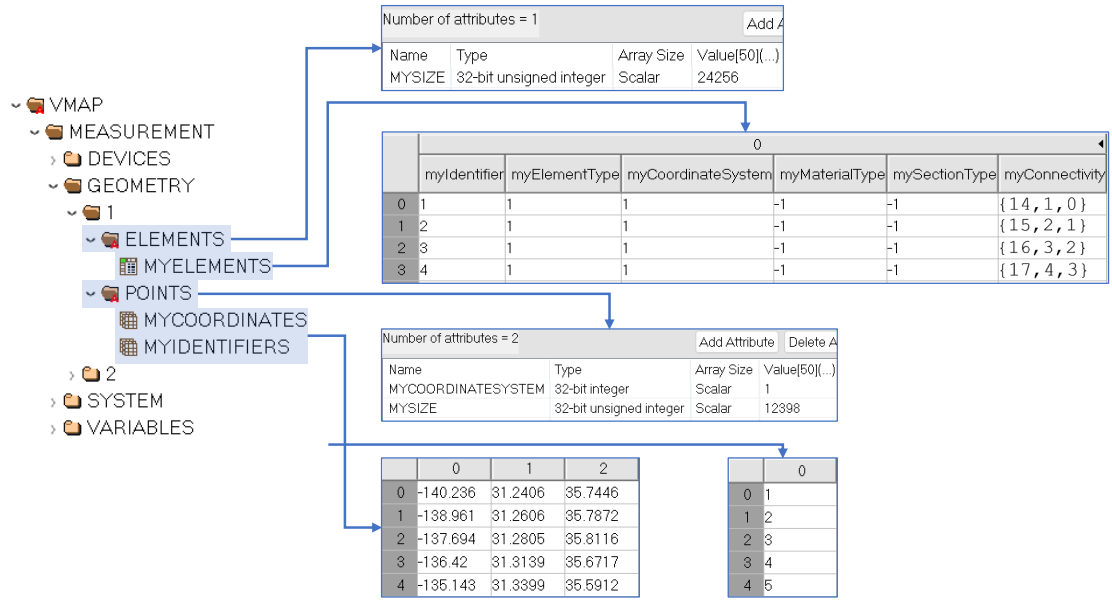


Figure 5.5: Stores the points and elements generated by the stereoscopic measuring system inside the `GEOMETRY` group

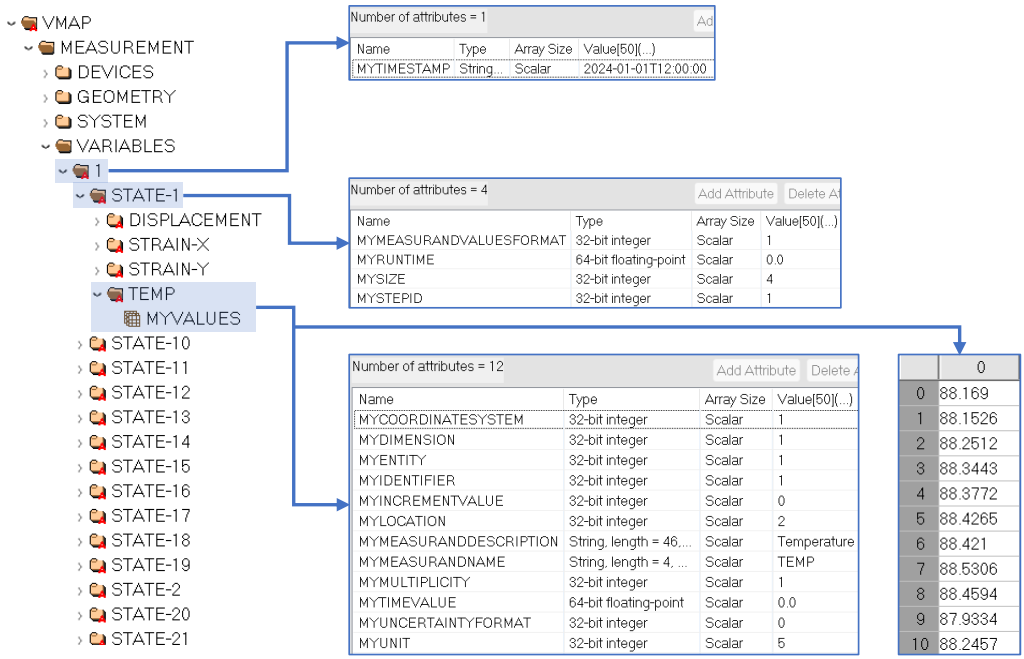


Figure 5.6: Store results from the stereoscopic measuring system inside the `VARIABLES` group

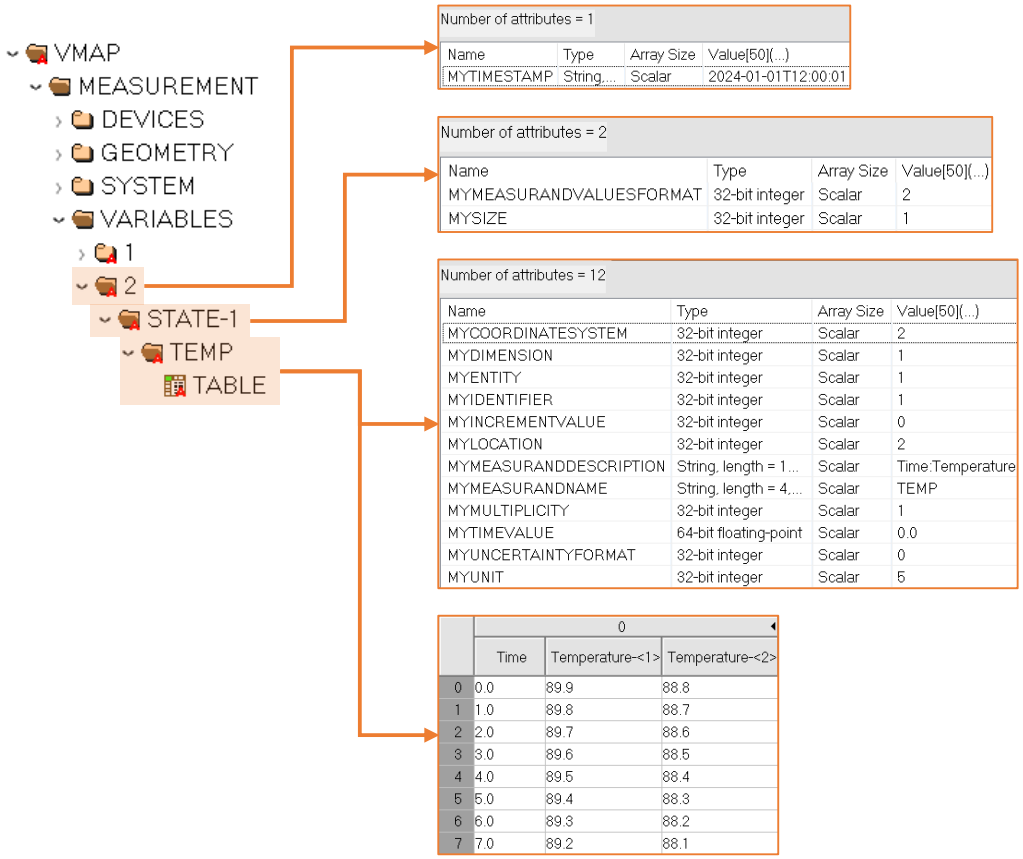


Figure 5.7: Store results from the temperature sensors system inside the `VARIABLES` group

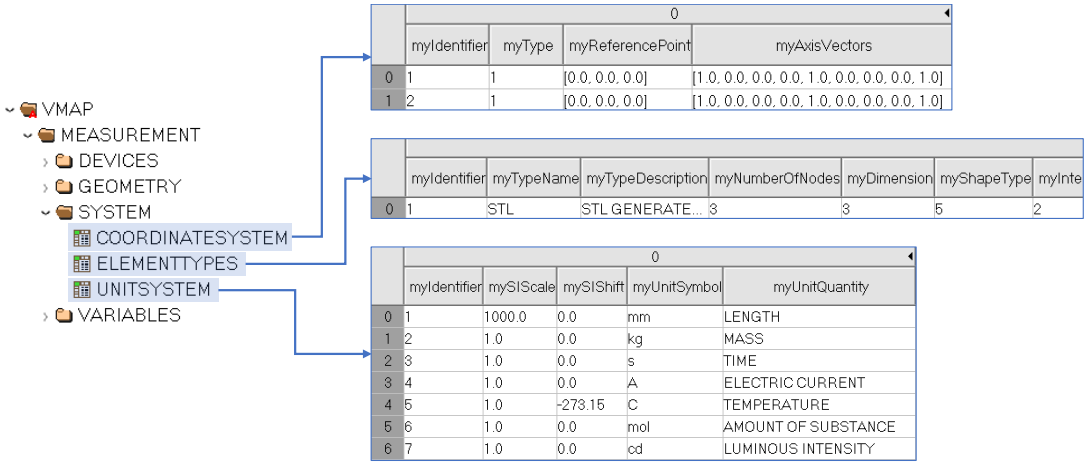


Figure 5.8: Store general information of the measurement trial inside the `SYSTEM` group

5.4 Run the Example

5.4.1 Setting Up the Virtual Environment

To set up the example follow these steps.

1. **Install Anaconda** (if you haven't already) and **open the Anaconda prompt**
2. **Create a virtual environment** (e. g. "VMAP" and Python 3.10):

   ```
   conda create --name VMAP python=3.10
   ```

3. **Activate the virtual environment**:

   ```
   conda activate VMAP
   ```

4. **Install required packages** in the "VMAP" environment:

   ```
   conda install -c anaconda numpy pandas
   ```

5. **Adjust the directories**:

   Open `BlowMoulding2VMAP.py` with an editor of your choice and

   - change the variable `vmap_lib_path` to your local VMAP directory,
   - change the variable `dataset_path` to your local "Dataset" directory.
   - change the variable `vmap_file_name` to the name of the resulting VMAP file.

   Save your changes and close the file.

5.4.2 Write the Measurement Data to VMAP

Once the environment is set up and the necessary packages are installed, you can run the script by following these steps (see Figure 5.9):

1. **Ensure the virtual environment is activated**:

   ```
   conda activate VMAP
   ```

2. **Navigate to the directory** containing `BlowMoulding2VMAP.py`:

   ```
   cd path/to/your/script
   ```

3. **Run the script**:

   ```
   python BlowMoulding2VMAP.py
   ```

```
(base) C:\Users\vlueddemann>activate VMAP

(VMAP) C:\Users\vlueddemann>cd C:\Users\vlueddemann\VMAP_WG_MD\UC\VMAP_UC_BLOW_MOULDING\Code

(VMAP) C:\Users\vlueddemann\VMAP_WG_MD\UC\VMAP_UC_BLOW_MOULDING\Code>python BlowMoulding2VMAP.py
Found 2 measuring systems in C:/Users/vlueddemann/VMAP_WG_MD/UC/VMAP_UC_BLOW_MOULDING/Dataset:
        -> C:/Users/vlueddemann/VMAP_WG_MD/UC/VMAP_UC_BLOW_MOULDING/Dataset\01_Camera_System
        -> C:/Users/vlueddemann/VMAP_WG_MD/UC/VMAP_UC_BLOW_MOULDING/Dataset\02_Thermocouple_System


Gather and write all element types ...
Gather and write all coordinate system ...


Write geometry of measuring system 1 ...
Write variables of measuring systems 1 ...
        -> Variable 1 : TEMP ...
        -> Variable 2 : STRAIN-X ...
        -> Variable 3 : STRAIN-Y ...
        -> Variable 4 : DISPLACEMENT ...
```

Figure 5.9: Run the Blow Moulding Example Script

You will then find the resulting VMAP file at your directory given in `vmap_file_name`. This file can be opened by using HDFView or visualized in ParaView.

5.4.3 Visualize the Data in ParaView

To visualize the blow moulding example follow these steps (see Figure 5.10):

1. **Download ParaView**
2. **Link the VMAP Plugin**:
    - Navigate to Tools/Manage Plugins...
    - Click on "Load New..."
    - Choose `VMAPReader.py`
3. **Open the VMAP example**:
    - Navigate to File/Open...
    - Choose the just created VMAP file in `vmap_file_name`

The Visualization the measurement data can be seen in Figure 5.11

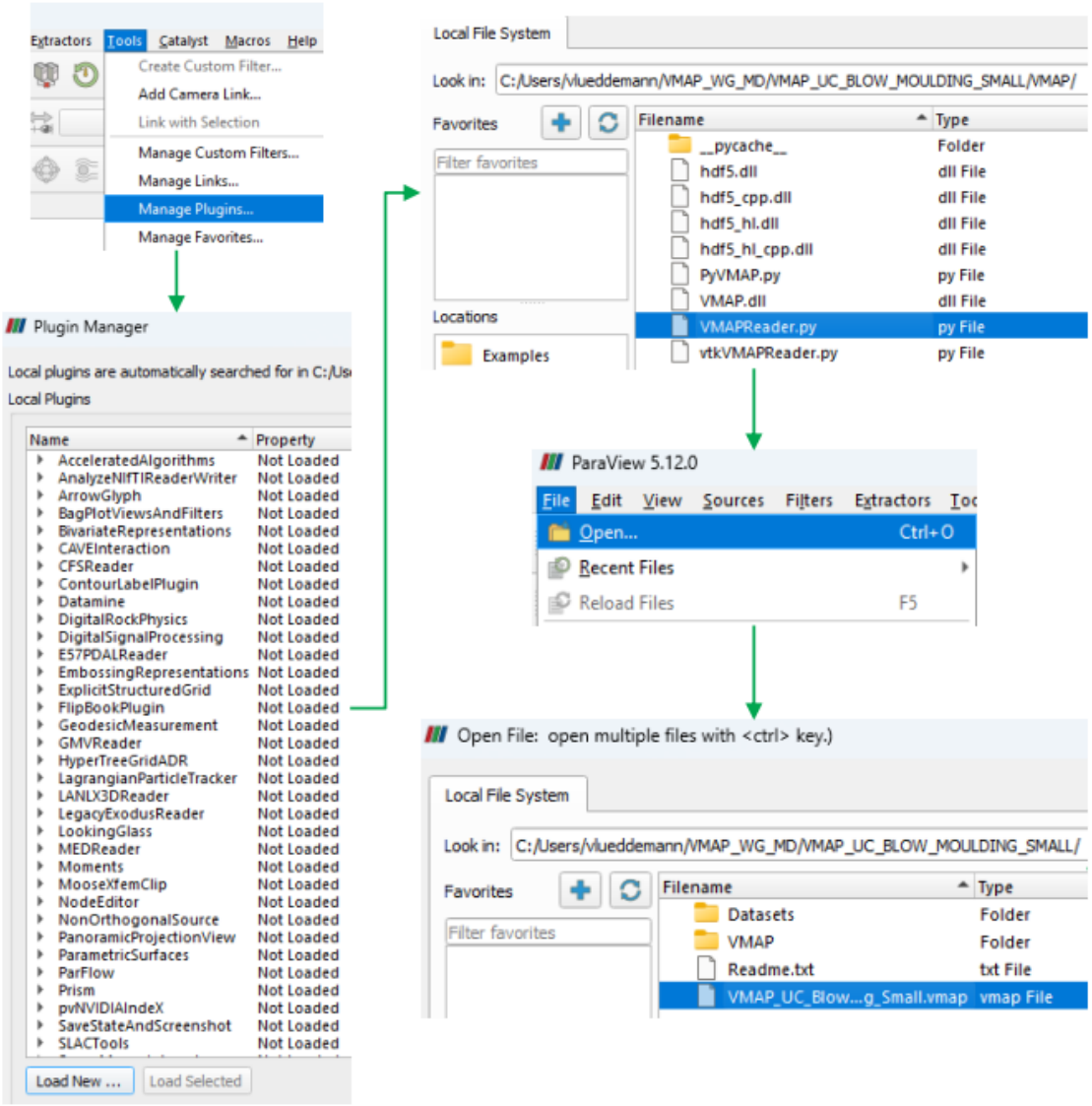


Figure 5.10: Open the Blow Moulding Example in ParaView

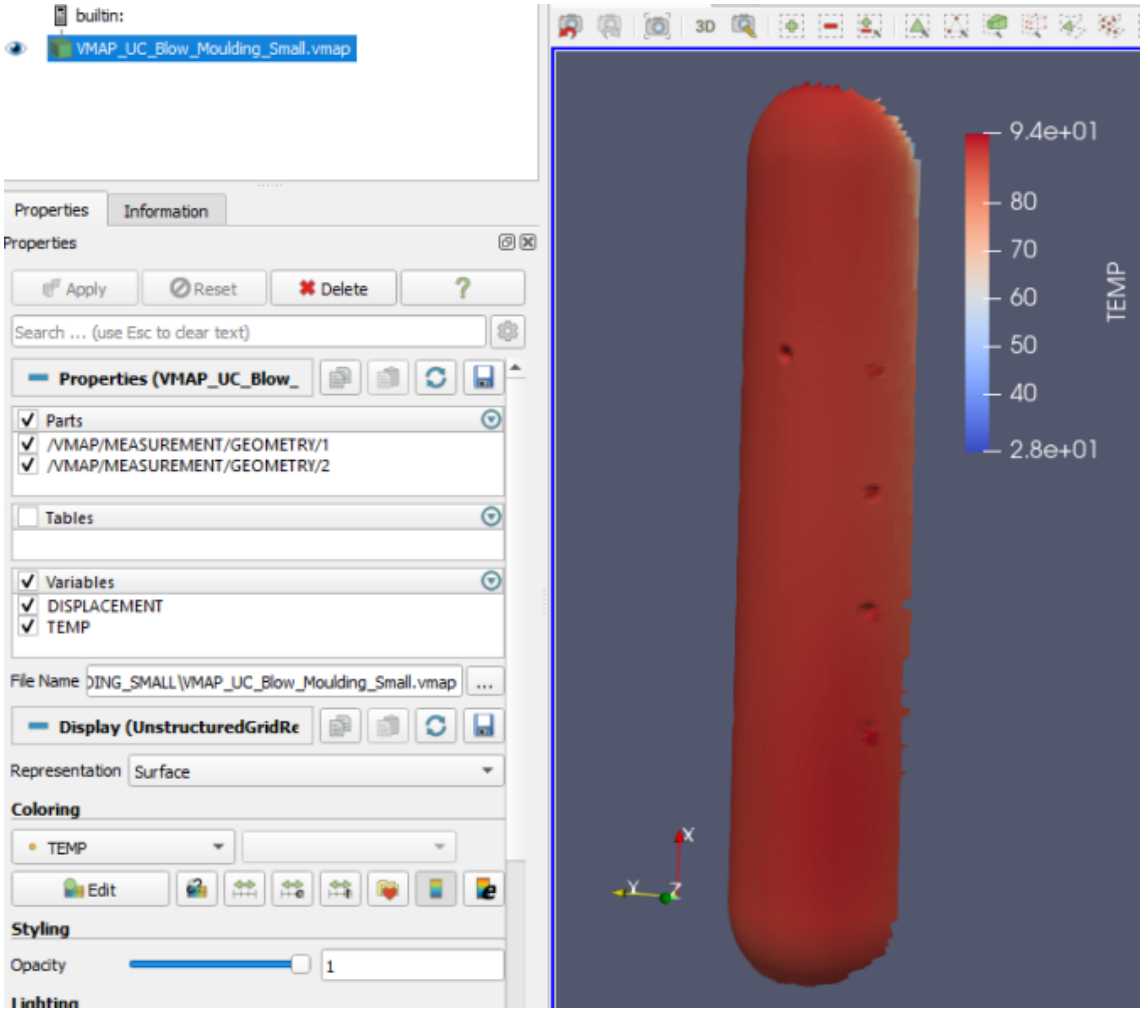


Figure 5.11: Visualize the Blow Moulding Example in ParaView

Chapter 6

Proposal for Including Measurement Uncertainties into VMAP

6.1 Requirements

As outlined in chapter 1.2, measurements go hand in hand with uncertainties. Typically, statistical methods are applied to determine these uncertainties. Therefore, the following chapter contains requirements that need to be considered when integrating full measurement results into VMAP. These requirements will then serve as a starting point for an implementation proposal.

In essence, the full measurement result includes, providing an estimate for the true value and indicating a potential deviation from this estimate. Ideally, this involves specifying a mean and an uncertainty gained from a sufficient number of measurement trials. In practice, however, it may also occur that an uncertainty is rather known from experience and is used in conjunction with the result of just a single or a few measurement trials.

To include estimates, uncertainties or error limits into VMAP, respective datasets are proposed in Figure 6.1. A dataset of the estimate and a dataset of an associated uncertainty dataset must have the same format. The values inside the uncertainty dataset are then considered as an interval around the estimated values. This principle is based on the specification of the full measurement result X from chapter 1.2

$$X = x \pm u$$

with an estimate x and a standard measurement uncertainty u. If only a single uncertainty is assigned to all measurement points (see type `a` and `c` in table 6.2), it is suggested to specify a `1x1` dataset with just this single value instead. In this case, this one value is assigned to the whole dataset of the measured value at the respective time step.

In VMAP, the `MYVALUES` dataset (or `TABLES` dataset) could be considered to be the estimated value. It can then either be a dataset containing the unprocessed result from a single measurement or the mean values from multiple measurements corrected with the systematic deviation.

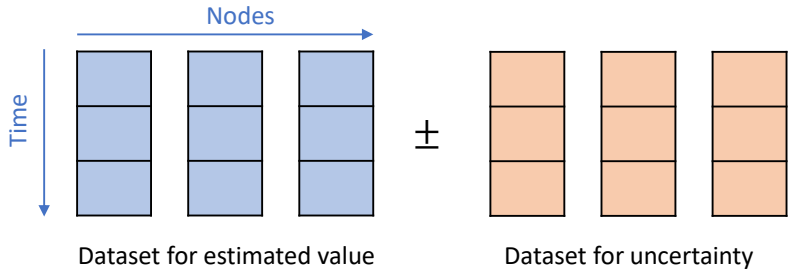


Figure 6.1: Proposal for uncertainty datasets

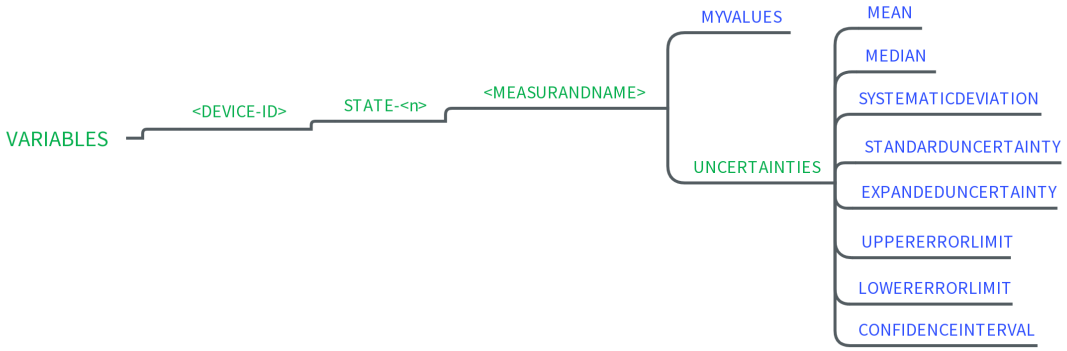


Figure 6.2: Proposal for including uncertainties into the VMAP structure

As an indication of a maximum deviation from that estimate, error limits can also be considered in addition to the different types of uncertainties. Therefore, datasets for the quantities listed in table 6.1a should exist in VMAP and be supplemented with the attributes from table 6.1b. A suggestion of how such datasets can be integrated into the VMAP structure is depicted in Figure 6.2.

6.2 Assignment of Uncertainties to the estimated Values.

Depending on the use case, uncertainties might refer to different ranges of the measurement data. Figure 6.3 illustrates possible ways how statistical datasets like uncertainties can be obtained from measurement data. An exemplary dataset is shown in blue, which consists of three measuring points, three time steps and two measurement trials. The format of the uncertainty data set for each option is shown in orange.

In 6.3a, the data from several measurement trials is used. Each orange data point is determined from the corresponding entries of the blue datasets. Therefore, for each measuring point and for each time step, a separate uncertainty is generated over all measurement trials. The resulting dataset has the same format as the original datasets.

In 6.3b, each orange data point is determined from all measuring points at the corresponding time step of one (ore more) blue dataset(s). Therefore, for each time step, a separate uncertainty is generated over all measurement points. The resulting dataset has the same

Table 6.1: Estimate & Uncertainty Datasets and associated Attributes in VMAP

(a) Datasets

Name	**Attributes (6.1b)**
Estimates	
Mean	1, 2, 3
Median	1, 2, 3
Systematic deviation	1, 2
Uncertainties	
Upper error limit	1, 2, 3
Lower error limit	1, 2, 3
Standard uncertainty	1, 2, 3
Expanded uncertainty	1, 2, 3, 4
Confidence interval	1, 2, 3, 6, 7

(b) Attributes

Nr.	**Attribute name**
1	Name
2	Description
3	Number of trials
4	k-Factor
5	t-Factor
6	Confidence level
7	Side

number of time steps as the original dataset.

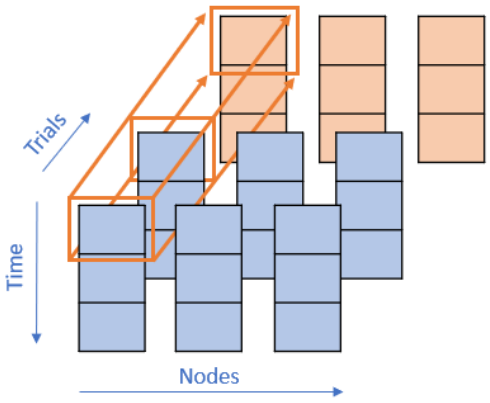
In 6.3c, each orange data point is determined from all time steps at the corresponding measuring point of one (or more) blue dataset(s). Therefore, for each measurement point, a separate uncertainty is generated over all time steps. The resulting dataset has the same number of measurement points as the original dataset.

In 6.3d, a single orange data point is determined from one (or more) blue dataset(s). Therefore, for the whole dataset, just a single uncertainty value is generated.

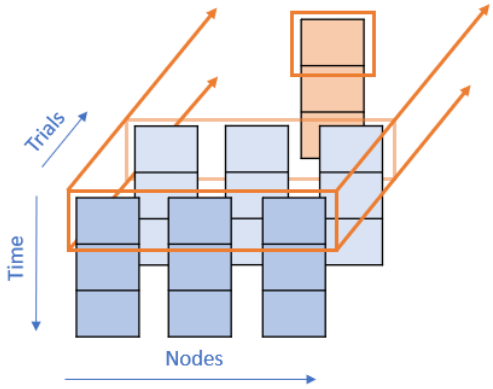
Table 6.2 and Figure 6.4 show how uncertainties can be assigned to the estimate dataset depending on the type from Figure 6.3. If assignment type `a` is present, then a separate uncertainty is assigned to each individual measurement point and for each individual time step. For assignment type `b`, a common uncertainty is assigned to all measurement points together but might vary over time. For assignment type `c`, there is a separate uncertainty for each individual measurement point, which does not differ over time. For assignment type `d`, the same measurement uncertainty is assigned to all measurement points over all time steps.

Table 6.2: Assigning uncertainties to a results dataset

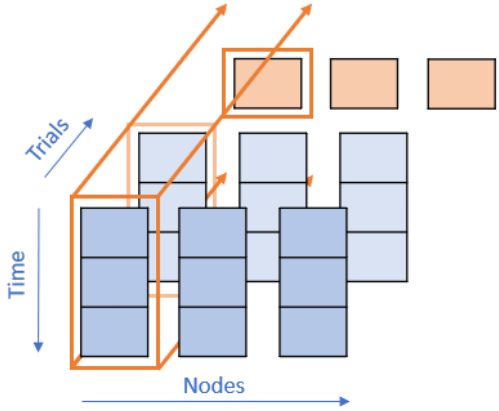
		Measurement points	
		Single	**Common**
Time step	**Single**	a	b
	Common	c	d



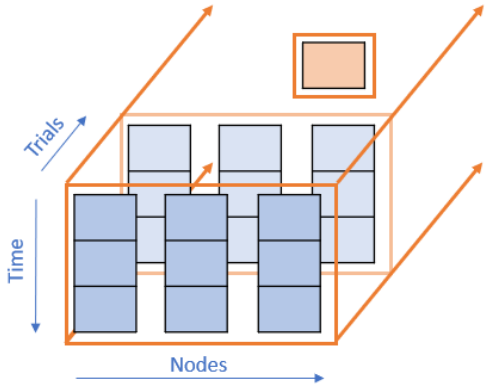
(a) Over multiple trials



(b) Over multiple points



(c) Over multiple time steps



(d) Over multiple points and time steps.

Figure 6.3: Different ways in which uncertainties or other statistical datasets (orange) can be gained from measurement results (blue)

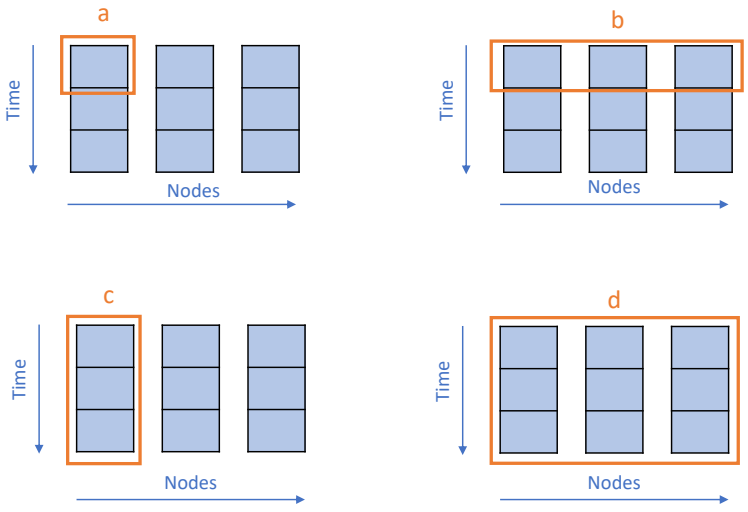


Figure 6.4: Different ways in which uncertainties (orange) can be assigned to a measurement results dataset (blue)

Appendix A

Support through third-party funded projects

The technical proposals, application examples and testing to extend the VMAP specification in the direction of physical measurement and sensor data are supported by several third-party funded projects. These very different R&D projects cover a wide range of industry-relevant applications and, thanks to the large number of partners from industry, software and research, enable a good penetration of all important aspects of standardisation.

PIONEER (Open Innovation platform for optimizing production systems) aims to develop and implement an interoperable material modeling and manufacturing ecosystem that enables multi-directional data flow along the material value chain by linking product design and distributed modeling data with information from material characterization, manufacturing processes, and product quality criteria. PIONEER combines a design-by-simulation approach with manufacturing and quality data to optimize product development strategies in high-mix/low-volume production systems.
Funded by the European Union / Grant Agreement No: 101091449; Project duration: January 2023 until December 2025

RESTORE (Sustainable remanufacturing solution with increased automation and recycled content in laser and plasma-based process) Remanufacturing boosts circular economies and job creation. RESTORE enhances this with sustainable processes and materials. It merges advanced technologies for better remanufacturing applications. The RESTORE platform will digitize and streamline the remanufacturing process. Fraunhofer SCAI standardizes data for remanufacturing efficiency.
Funded by the European Commission / Grant Agreement No: 101138842; Project duration: 01/2024 until 12/2027

ALABAMA (Adaptive Laser Beam for Additive Manufacturing) The ALABAMA project is developing adaptive laser techniques for improved additive manufacturing. The goal is to decrease material porosity and adjust microstructures.
The innovation includes the development of models for process optimization. Process parameters are optimized using multi-beam control and laser shaping. Advanced monitoring uses multispectral imaging for quality control. Material tests ensure compliance with requirements. The technology is being tested in aerospace, maritime, and automotive industries. These sectors face challenges regarding material quality. The project promises significant productivity increases and cost reductions. It also promotes the autonomy of the European industry.
Funded by the European Commission / Grant Agreement No: 101138842; Project duration: 01/2024 until 12/2027

BASE (Battery Passport for Resilient Supply Chain and Implementation of Circular Economy) The battery although is at its central role for green transitioning of the road transport, the current battery supply chain is lacking in traceability and sustainability, resiliency, and circularity aspects. Critical Raw Materials (CRM) are essential ingredients for battery manufacturing. The exploding growth of Electric vehicles driven by the climate neutrality policy objectives will create pressure on CRM supply chain and will increase the EU dependency for CRM on 3rd countries, resulting in decrease of competitiveness of the EU automotive and battery manufacturers. Implementation of the digital battery passport (DBP) concept in the battery value chain might resolve these issues.
Funded by the European Commission / Grant Agreement No: 101157200; Project duration: 06/2024 until 05/2028

VMAP Analytics (Smart Analytics for Multi-Scale Material and Manufacturing Modelling) The vision of VMAP analytics is to enable the realisation of smart Digital Twins for materials and manufacturing design tasks. The VMAP analytics interface standard is an extension of current standard that will accommodate multi-scale models, sensor and measurement data, and information from production machines. VMAP analytics will provide an open ontology for engineering processes in materials and manufacturing design. Both together – interface standard and open ontology – will enable broad use of semantic search and artificial intelligence methods in these classical engineering disciplines..
VMAP Analytics is an ITEA coordinated project with project ID 19007; Project duration: 10/2020 until 09/2023

SmartEM (Open reference architecture for engineering model spaces) aims to address the limitations of current engineering models by developing a reference architecture for engineering model spaces. The architecture will enable the reuse, exchange and integration of computational engineering models, reducing the need for costly design corrections and promoting early data and model exchanges. SmartEM will use AI-assisted methods to create surrogate models from heterogeneous data sources and allow their re-combination within a given engineering domain. The project will develop use-case model spaces to manage reusable and transferable engineering models for various domains and provide solutions for IP management to enable model exploitation in an increasingly digital engineering market.
SmartEM is an ITEA coordinated project with project ID 22009; Project duration: 10/2023 until 12/2026