FEBio Binary Database Specification

Last updated: June 16, 2020

Contact address
Musculoskeletal Research Laboratories, University of Utah
72 S. Central Campus Drive, Room 2646
Salt Lake City, Utah

Website
MRL: http://mrl.sci.utah.edu
FEBio: http://febio.org

Forum
http://mrl.sci.utah.edu/forums/

Development of the FEBio project is supported in part by a grant from the U.S. National Institutes of Health.

Table of Contents

1 Introduction

This document describes the structure of the FEBio binary database (that is, the FEBio plot file format), which stores the results of a FEBio analysis. The FEBio binary database format is both self-describing and extendible. A user can understand the contents of the file by simply parsing the block structure of the file. The file format is made extendible by providing an abstract layer between the data and the file system. This layer defines the file structure as a hierarchy of data blocks. The result is that each file is now structured similarly as a file folder. Each block can be viewed as a file in the folder and each branch in the hierarchy as a sub-folder.

The database consists of four parts: the *header* contains some general info that may be useful for parsing the rest of the file; the *dictionary* presents a textual description for each data field in the file. This can be used to identify the contents of each data field; the *mesh* section defines the mesh of the model; and finally, the *state* sections contain the actual data or results for the field variables.

The following sections describe the details of the database format. In the next section, the block-structure of the file is explained. The sections thereafter describe the different parts of the database.

1.1 Changes in this version

This version of the FEBio Binary Database Specification differs in a few important aspects from older version:

- The header structure now contains additional data fields, including author, software (that generated the file) and removes some fields such as number of nodes.

- The *root* section now only contains the header and the dictionary. The *Geometry* section is now a separate section and is renamed to *Mesh*. The *Materials* section was removed and replaced by a more general *Parts* section, now a subsection of *Mesh*, which defines an additional partitioning of the mesh (e.g. by material). The *Parts* section is optional. The domain structure defines the default partitioning.

- The format now supports multiple *Mesh* sections. Each *Mesh* section redefines the mesh. Subsequent *State* sections are defined relative to the last *Mesh* section. This is useful for analysis that modify the mesh structure at some point in the analysis, e.g. fracture, adaptive mesh refinement, etc.

2 Block structure

2.1 Overview

The FEBio binary database uses an abstract layer to communicate with the file system. This achieves two goals. The first is that the content of a file becomes independent of the

file system that wrote the file. Big endian systems can read files created with small endian systems and vice versa. The second goal is that the data is now stored in a hierarchical structure which is easy to search and modify. This means that future additions and changes can be made fairly easily without losing backward compatibility. In addition, the self-describing feature of the format even allows for some forward compatibility.

As mentioned above, the file is structured as a hierarchy of blocks. Each block consists of three fields: a DWORD[1] identifier, followed by a DWORD containing the size of the data chunk in bytes and then the actual data.

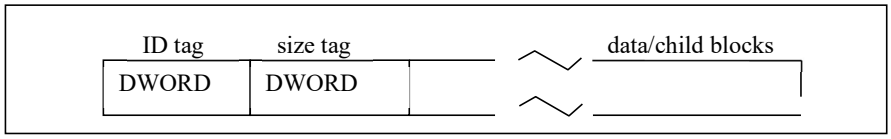

Figure 1. Each block in the plot file consists of three fields: a DWORD with an identifier, a DWORD containing the size of the block and finally the data of the block, which can be child blocks.

This data may be either numeric data (such as the data of a field variable) or child blocks. If the block has children, it is referred to as a *branch*. If the block only contains numeric data, it is referred to as a *leaf*.

2.2 Parsing the FEBio plot file

Although the FEBio plot file in essence is a hierarchy of blocks as described above, there are a few more caveats that are important for parsing the FEBio file.

The first DWORD of the file is a tag that identifies the FEBio plot file.

Tag	***ID***	***description***
FEBIO	0x00464542	FEBio identifier tag

Parsers should read this number and use it to identify whether the file is indeed a proper FEBio plot file. If the value differs, then that means that the file is either not a valid plot file, or that the endianess of the system that wrote the file is different than that of the system that is reading the file. In the latter case, a byte swap will be necessary when reading data from the file.

NOTE: Note that the FEBio tag is not followed by a size tag. The reason is because when FEBio is writing the file it does not know the length of the final file yet.

After the FEBio tag, the contents of the file follows, organized in the hierarchical block structure described above. At the highest level, the plot file has a single *root* block, followed by a *mesh* block and a series of *state* blocks.

[1] a DWORD is a "double word" meaning an unsigned integer of 4 bytes.

Tag	*ID*	*description*
ROOT	0x01000000	root block of FEBio plot file
STATE	0x02000000	state block containing results of a single time step

After the *root* block, the *mesh* section follows which defines the mesh of the model. Then, at least one *state* blocks follow. There will be one *state* block for each time step in the FEBio plot file. Note that these blocks are not child blocks of the *root* block. In the following figure, the high level structure of the FEBio plot file is depicted for a file that has two *state* blocks.

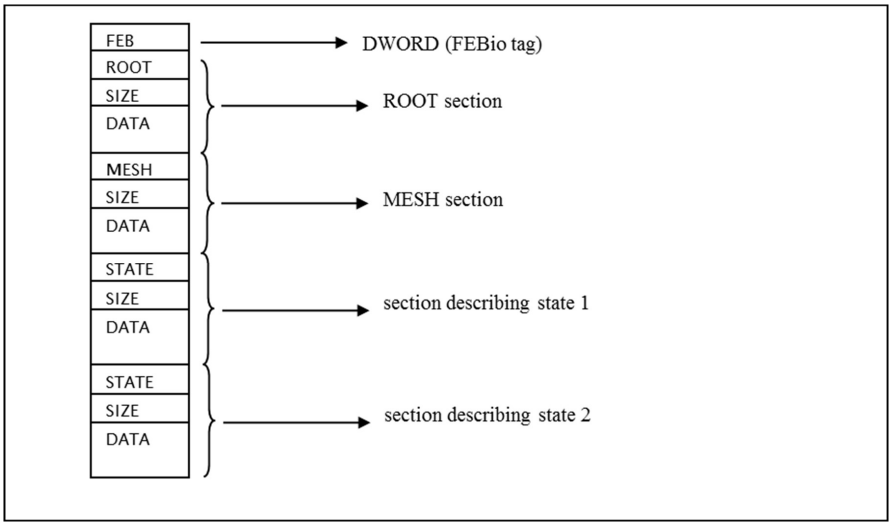

Figure 2. Example structure for a plot file containing a root section and two data sections. Each section is composed of three fields. A DWORD containing the section ID, followed by a DWORD containing the size of the section block and finally the actual data (which may be child blocks).

As explained above, the first DWORD will be the FEBio identifier. The first block in the file will always be the ROOT block, which will contain the header, dictionary, materials and geometry definitions. Then, the *state* blocks follow, which will contain the actual results. Each block has three fields. The first field is a DWORD, identifying the section. For the first block, this will be ROOT and for the other blocks this will be STATE. The next DWORD is the size of the entire block. Finally, the actual data will follow, which for the ROOT and STATE blocks will contain child blocks.

3 Root Section

The ROOT section is the first block in the file. This block contains the *header* and *dictionary* sections as child blocks.

Tag	*ID*	*description*
HEADER	0x01010000	contains the header section
DICTIONARY	0x01020000	contains the dictionary section

These sections will be detailed below.

4 Header Section

The first section of the *root* block is the *header* section. It stores the following data:

Tag	*ID*	*description*	*size*
VERSION	0x01010001	version of the file format	DWORD
MAX_FACET_NODES	0x01010003	max number of nodes per facet	DWORD
COMPRESSION	0x01010004	Compression flag	DWORD
AUTHOR	0x01010005	Author name	CHAR64
SOFTWARE	0x01010006	Software that generated this file	CHAR64

Currently, the VERSION will always be 0x0008. The MAX_FACET_NODES is the maximum number of nodes on any facet. To facilitate parsing surfaces, each surface facet writes MAX_FACET_NODES values when exporting its node list, even if it has fewer nodes. See section 6.3 for more details. The COMPRESSION flag indicates whether the state sections are compressed or not. Data compression is an optional feature for plot files, and will not be discussed in this document. The AUTHOR contains the name of the person who created the file. The SOFTWARE tag contains the name of the software that generated the file.

5 Dictionary Section

When running a FEBio analysis, FEBio will store the values of certain user-selected variables (e.g. stress, temperature, fluid pressure, etc.) to the plot file. In order for a parser to know which variables were written, it needs to read the *dictionary* section. This section stores a list of variables, including their names, which are stored in the plot file. Specifically, three attributes are stored for each data variable: a name that provides a description of the data, the data type (scalar, vector, tensor) and the storage format. In addition, the variables are grouped by category. Data variables can be defined for node sets, domains (element sets) and surfaces. In addition, global variables (which are not associated with any part of the model) can also be defined. Each of these categories has their own sub-section in the *dictionary*.

Tag	*ID*	*description*
GLOBAL_DATA	0x01021000	lists global data in model
NODESET_DATA	0x01023000	lists data associated with the node sets
DOMAIN_DATA	0x01024000	lists data associated with domains
SURFACE_DATA	0x01025000	lists data associated with surfaces

Note that all of these sections are optional. For example if a model does not define global data, that section will not be part of the plot file.

Each of these sub-sections defines a list of dictionary items defined by the following tag.

Tag	***ID***	***description***
DICTIONARY_ITEM	0x01020001	beginning of dictionary item

Each dictionary item contains three fields.

Tag	***ID***	***description***	***size***
ITEM_TYPE	0x01020002	type of data	DWORD
ITEM_FORMAT	0x01020003	storage format of data	DWORD
ITEM_NAME	0x01020004	textual description of data	CHAR64

The type of the data can be any of the following values.

ITEM_TYPE	***VALUE***	***description***
FLOAT	0	single precision (s.p.) floating point
VEC3F	1	3D vector of s.p. floats (stored in *x*, *y*, *z* order)
MAT3FS	2	symmetric 2nd order tensor of s.p. floats (stored in *xx*, *yy, zz*, *xy*, *yz*, *xz* order)
MAT3FD	3	diagonal matrix of s.p. floats (stored in *xx, yy, zz* order).
TENS4FS	4	symmetric fourth-order tensor of s.p. floats.

As explained below, data will be stored for different *regions* of the mesh, where a region can be a *node set*, a *surface* or a *domain* (element set). A *region* is composed of *items* where an item is a single shape in the region. For node sets, an item will refer to a node, for surfaces this will be a facet and for domains an item will refer to an element. The storage format defines how many values are written for each region and how the values relate to the geometry of the region or items. The following values are currently defined.

ITEM_FMT	***VALUE***	***description***
NODE	0	one value for each node of the region
ITEM	1	one value for each item
MULT	2	one value for each node for each item

The easiest format to understand is the *ITEM* format which simply stores one value for each item of the region. Thus, one value for each node or for each facet or for each element, depending on the region type. The *MULT* format defines a value for each node of each item. For example, for a surface of quads, the data will contain four values for each facet, one for each of the four nodes. The *NODE* format stores a single value for each *node* of the *region*. Each type of region (node set, surface or domain) implicitly also defines a set of nodes, namely all the nodes that are part of that region. With the *NODE* format, the user

defines a single value for each of the nodes in this implicit node set. This will be explained in more detail below in the state section.

Note that the storage formats are only important for surfaces and domains. For other categories (e.g. node sets), all formats are essentially equivalent, and can safely be ignored.

6 Mesh Section

The *Mesh* section defines the mesh of the model and its decomposition into separate regions. A region can be a *node set*, a *surface* (i.e. facet set) or a *domain* (element set). The mesh section is defined by the following sub-sections.

Tag	***ID***	***description***
NODE_SECTION	0x01041000	beginning of node section
DOMAIN_SECTION	0x01042000	beginning of domain section
SURFACE_SECTION	0x01043000	beginning of surface section
NODESET_SECTION	0x01044000	beginning of node set section
PARTS_SECTION	0x01045000	beginning of parts section

6.1 Node Section

The NODE section defines a set of nodes of the mesh. Each node section starts with a NODE_HEADER followed by a NODE_COORDS section.

Tag	***ID***	***description***
NODE_HEADER	0x01041100	node header
NODE_COORDS	0x01041200	node data list

The header contains the following information.

Tag	***ID***	***description***
NODES	0x01041101	number of nodes (N) in this NODE section
DIM	0x01041102	dimension of node set (i.e. nr. of coords)
NAME	0x01041103	name of node set.

The NODES parameter defines the number of nodes defined in this section. The DIM parameter sets the number of coordinates that will be defined for each node (e.g. 3 for 3D problems). Each NODE section also implicitly defines a node set. The NAME attribute defines the name of this node set.

Then the NODE_COORDS section follows, which defines the nodal data for each node. This data field stores the nodal IDs and coordinates for all the nodes in the mesh. The size of the field is defined by N*DWORD + DIM*N*FLOAT where N is the number of nodes

in the mesh (as defined in the header section), DIM is the dimension, and FLOAT is the size of a single precision floating point number (4 bytes). The order of the data is (e.g. if DIM equals 3),

ID1	*x1*	*y1*	*z1*	...	*IDN*	*xN*	*yN*	*zN*

Here, *x[i]* is the *x*-coordinate of node *i* and similarly for *y* and *z*.

6.2 Domain Section

The Domain section lists all domains (i.e. element sets) in the mesh. Each domain is identified by a DOMAIN section.

Tag	***ID***	***description***
DOMAIN	0x01042100	beginning of a domain section

Each domain is defined by two sub-sections, a *domain header* and an *element list*.

Tag	***ID***	***description***
DOMAIN_HEADER	0x01042101	beginning of domain header
ELEMENT_LIST	0x01042200	list of element connectivity

The *domain header* contains the following data fields.

Tag	***ID***	***description***	***size***
ELEM_TYPE	0x01042102	element type	DWORD
PART_ID	0x01042103	part ID	DWORD
ELEMENTS	0x01032104	number of elements	DWORD
NAME	0x01032105	an optional name for this domain	CHAR64

The *ELEM_TYPE* defines the type of elements stored in the domain. It can have one of the following values.

ELEM_TYPE	***VALUE***	***description***
HEX8	0	8-node hexahedron solid element
PENTA6	1	6-node pentahedron solid element
TET4	2	4-node tetrahedron solid element
QUAD4	3	4-node quadrilateral shell element
TRI3	4	3-node triangular shell element
TRUSS2	5	2-node linear truss element
HEX20	6	20-node quadratic hexahedral element
TET10	7	10-node quadratic tetrahedral element
TET15	8	15-node quadratic tetrahedral element
HEX27	9	27-node quadratic hexahedral element

The *PART_ID* corresponds to the ID of one of the parts defined in the *Parts* section.

The *ELEMENTS* field is the number of elements in the domain.

After the domain header the element list follows. For each element it defines an ELEMENT data field.

Tag	***ID***	***description***	***size***
ELEMENT	0x01042201	element data field	DWORD*(NE+1)

If NE is the number of nodes per element, then this data field stores NE+1 DWORDS. The first DWORD is the element ID, a unique number that identifies the element. The following NE DWORD's define the element connectivity.

6.3 Surface Section

The *surface* section defines the surfaces of the mesh for which data is stored in the plot file. Each surface begins with a SURFACE section.

Tag	***ID***	***description***
SURFACE	0x01043100	beginning of a surface section

The surface section follows a similar structure as the domain section, namely a *surface header* followed by a *facet list*.

Tag	***ID***	***description***
SURFACE_HEADER	0x01043101	beginning of surface header
FACET_LIST	0x01043200	facet connectivity list

The surface header contains the following data fields.

Tag	***ID***	***description***	***size***
SURFACE_ID	0x01043102	surface ID	DWORD
FACETS	0x01043103	Number of facets	DWORD
NAME	0x01043104	A name for this surface	CHAR64
MAX_FACET_NODES	0x01043105	max nodes per facet	DWORD

The SURFACE_ID is a unique identifier and FACETS is the number of facets in the surface.

The FACET_LIST follows the header and contains a FACET data field for each facet in the surface.

Tag	***ID***	***description***	***size***
FACET	0x01043201	facet data field	DWORD*NF

Here, NF = MAX_FACET_NODES + 2. The first DWORD is a unique identifier (which currently should be ignored). The second DWORD is the number of nodes for this facet. This also defines the facet type. (For instance, 3 nodes define a triangle, 4 nodes define a quadrilateral). Next, the node ID's follow.

6.4 NodeSet Section

The NodeSet section defines all the node sets where a node set is a named collection of nodes. Each nodeset begins with the NODESET section.

Tag	*ID*	*description*
NODESET	0x01044100	beginning of a nodeset section

Next, a header section and a node list section follow.

Tag	*ID*	*description*
NODESET_HEADER	0x01044101	beginning of nodeset header
NODE_LIST	0x01044200	node list

The header is composed of the following chunks.

Tag	*ID*	*description*	*size*
NODESET_ID	0x01044102	nodeset ID	DWORD
NODES	0x01044104	Number of nodes	DWORD
NAME	0x01044103	An optional name for this nodeset	CHAR64

The node list is a list of all the nodes that belong to this node set. All node indices are zero-based.

6.5 Parts Section

Commented [sm1]: Remove section and move to Geometry section. Maybe renaim it to Parts section.

The Parts section lists the parts defined in the plot file. For each part, a PART section is written.

Tag	*ID*	*description*
PART	0x01045100	beginning of a part definition

Then, for each part the following fields are written.

Tag	*ID*	*description*	*size*
PART_ID	0x01045101	ID of part	DWORD
PART_NAME	0x01045102	Name of part	CHAR64

The *ID* is a unique number that will be used in the domain definitions to refer to this part. The *NAME* is a textual description that can be used by the post-processor to present the part to the user.

7 State Section

The *state* section is where all the actual data is stored. FEBio will store one *state* section for each time step in the analysis. Each state defines two sub-sections, namely the *state header* and the *state data*.

Tag	*ID*	*description*
STATE_HEADER	0x02010000	state header section
STATE_DATA	0x02020000	state data section

7.1 State Header

The *STATE_HEADER* contains the following data field.

Tag	*ID*	*description*	*size*
STATE_TIME	0x02010002	time stamp of state	FLOAT

7.2 State Data Section

The STATE_DATA section stores the data for this state. It is composed of several sub-sections where each section corresponds to a data category as defined in the dictionary. The following sub-sections can thus be defined.

Tag	*ID*	*description*
GLOBAL_DATA	0x02020100	define global data section
NODE_DATA	0x02020300	define nodal data section
DOMAIN_DATA	0x02020400	define domain data section
SURFACE_DATA	0x02020500	define surface data section

Note that FEBio currently doesn't write global data sections.

Each of these sub-sections follows a similar structure. For each of the variables defined in the dictionary corresponding to the data category, a STATE_DATA section is defined.

Tag	*ID*	*description*
STATE_DATA	0x02020001	defines a data section

Each state data section has two fields.

Tag	*ID*	*description*	*size*

REGION_ID	0x02020002	ID of region	DWORD
DATA	0x02020003	actual data	?

The REGION_ID refers to the ID of the region for which this data variable is defined.

It is important to note that for node sets, a value of zero for the REGION_ID refers to the "master" node set which is the set containing all the nodes in the model. This node set is defined implicitly and will not be part of the list of node sets.

The DATA field contains the actual data. The size of this field is variable and is determined by the data type and storage format as defined in the dictionary as well as the number of items in the corresponding region. For example, for a domain that has NE elements, the size of the DATA field, using a type of FLOAT and a storage format of FMT_ITEM will be NE*FLOAT.

When a surface or domain stores its data in the FMT_NODE format, then the parser needs to figure out how many nodes are implicitly defined by the region and what the node order is. An implicit nodeset can be constructed by simple enumeration: each item of the region lists the nodes its visits and no nodes can be visited more than once. So for example, consider the surface of three faces shown in the figure below.

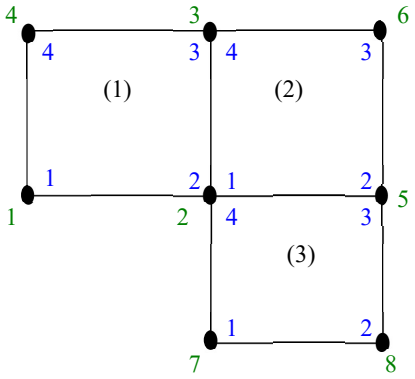

Figure 3. Example illustrating enumeration algorithm that defines the node ordering for the implicit node set defined by a surface.

Each facet has four nodes. The first four nodes of the implicit node set are simply the four nodes of the first element. The second element skips its first node, since it is already visited, add its second node (which becomes node 5) and its third (node 6) and skips its fourth node. Similarly, the third element adds its second (node 7) and third (node 8) and skips its third and fourth. Thus, this surface defines an implicit node set containing 6 nodes in the order as shown in figure 3. Consequently, if this surface stores its data in FMT_NODE

format, then the corresponding data section will contain six values, one for each of the nodes in the implicit node set.

8 APPENDIX: Tag Reference

The following table lists all the tags in the FEBio plot file format with their numerical ID and description.

- A size of 0 (zero) implies that the section contains only child sections.
- A size of ? implies the size is variable and needs to be calculated based on other information in the file.

Tag	*ID*	*description*	*size*
ROOT	0x01000000	root section	0
HEADER	0x01010000	header section	0
VERSION	0x01010001	version of file format	DWORD
NODES	0x01010002	number of nodes in mesh (N)	DWORD
MAX_FACET_NODES	0x01010003	max number of nodes per facet (NF)	DWORD
COMPRESSION	0x01010004	compression flag	DWORD
DICTIONARY	0x01020000	dictionary section	0
DICT_ITEM	0x01020001	dictionary item	0
ITEM_TYPE	0x01020002	type of data variable	DWORD
ITEM_FORMAT	0x01020003	storage format of data variable	DWORD
ITEM_NAME	0x01020004	name of variable	CHAR64
GLOBAL_VAR	0x01021000	global data dictionary section	0
MATERIAL_VAR	0x01022000	material data dictionary section	0
NODESET_VAR	0x01023000	node set data dictionary section	0
DOMAIN_VAR	0x01024000	domain data dictionary section	0
SURFACE_VAR	0x01025000	surface data dictionary section	0
MATERIALS	0x01030000	materials section	0
MATERIAL	0x01030001	material definition section	0
MATERIAL_ID	0x01030002	material ID	DWORD
MATERIAL_NAME	0x01030003	material name	CHAR64
GEOMETRY	0x01040000	geometry section	0
NODE_SECTION	0x01041000	node section	0
NODE_COORDS	0x01041001	nodal coordinates	3*N*FLOAT
DOMAIN_SECTION	0x01042000	domain section	0
DOMAIN	0x01042100	domain definition	0
DOMAIN_HEADER	0x01042101	domain header	0
ELEM_TYPE	0x01042102	element type	DWORD
MAT_ID	0x01042103	material ID	DWORD
ELEMENTS	0x01032104	number of elements in domain (NE)	DWORD
NAME	0x01032105	domain name	CHAR64
ELEMENT_LIST	0x01042200	element list	0
ELEMENT	0x01042201	element definition	DWORD*(NE+1)
SURFACE_SECTION	0x01043000	surface section	0
SURFACE	0x01043100	surface definition	0
SURFACE_HEADER	0x01043101	surface header	0
SURFACE_ID	0x01043102	surface ID	DWORD
FACES	0x01043103	number of faces	DWORD
NAME	0x01043104	surface name	CHAR64
FACE_LIST	0x01043200	face list	0
FACE	0x01043201	face definition	DWORD*(NF+2)
NODESET_SECTION	0x01044000	nodeset section	0
NODESET	0x01044100	nodeset definition	0

NODESET_HEADER	0x01044101	nodeset header	0
NODESET_ID	0x01044102	nodeset ID	DWORD
NODESET_NAME	0x01044103	nodeset name	CHAR64
NODESET_SIZE	0x01044104	number of nodes in nodeset (NN)	DWORD
NODELIST	0x01044200	list of nodes	DWORD*NN
STATE_SECTION	0x02000000	state section	0
STATE_HEADER	0x02010000	state header	0
TIME	0x02010002	time stamp of state	FLOAT
STATE_DATA	0x02020000	state data section	0
STATE_VAR	0x02020001	state variable	0
VARIABLE_ID	0x02020002	ID of variable	DWORD
VARIABLE_DATA	0x02020003	variable data section	?
GLOBAL_DATA	0x02020100	global data section	0
MATERIAL_DATA	0x02020200	material data section	0
NODE_DATA	0x02020300	nodal data section	0
DOMAIN_DATA	0x02020400	domain data section	0
SURFACE_DATA	0x02020500	surface data section	0