

VPS Solver Result File

ERF-HDF5 Result Specification

ESI Software

Solver / Application :	VPS Solver (PAM-CRASH)
Solver-Version :	2014
ERF-Version :	2.1
Release date :	March 2014

Contents

1	Purpose of the File	3
2	Filename Convention	3
3	Space Dimensions and Coordinates	3
4	Simulation States and Modes	3
5	Modular I/O	3
6	Solver Options	4
7	Entity Types	5
7.1	Introduction	5
7.2	Generic Entities	5
7.3	Finite Elements	5
7.3.1	Overview	5
7.3.2	0D Elements	8
7.3.3	1D Elements	8
7.3.4	2D Elements	8
7.3.5	3D Elements	8
7.3.6	4D Special Purpose Elements	9
7.4	Collectors	9
7.4.1	Introduction	9
7.4.2	Collectors for Kinematic Constraints	10
7.4.3	Collectors for Airbags	10
7.4.4	Muscle Collector	10
7.4.5	Tied Collector	10
7.4.6	PLINK Collectors	10
7.5	Zones	12
7.6	Local Frames	12
8	File Storage Hierarchy	13
9	References	13

1 Purpose of the File

The purpose of the described output file is to store simulation results for post-processing. This result specification is complementary to the ERF-HDF5 format specification [1]. The ERF file format is based on the open HDF5 binary format [2].

Note: The content described here refers to ERF format version 2.1.

2 Filename Convention

The file name consists of three components: *ROOT_NATURE.SUFFIX*

with: *ROOT* users simulation project name
 NATURE data indicator = RESULT
 SUFFIX file extension = erfh5

Note: Eigenmode bases are stored in a special file with NATURE=EGM. For detailed information about Eigenmode files, please refer to the VPS solver manuals [3].

3 Space Dimensions and Coordinates

A mechanical model is embedded in the three-dimensional Cartesian space. The deformed geometry can either be represented by:

- nodal coordinates per state or
- nodal coordinates of the initial model plus displacements per state:

result	type	representation	description
'COORDINATE'	vector \vec{x}	$[x \ y \ z]$	Cartesian coordinates
'Translational_Displacement'	vector \vec{u}	$[u_x \ u_y \ u_z]$	Displacement vector

4 Simulation States and Modes

All ERF blocks containing variable data are tagged with a state index. This state index is declared with the index definition block (ERF-block type 20). The index definition refers to the variable group 'INDEX', which contains the variable keys of all indices. The state index depends on the solver and analysis type (see section below) and may consist of time, frequency and/or load case values.

Note that the index values are stored in the two bijective arrays: *indexval* (type FLOAT) and *indexident* (type INT). For further details, please refer to the ERF-HDF5 format specification [1].

5 Modular I/O

ESI's VPS 2014 solver versions support modular I/O.

Modular I/O enables the user to easily exchange or include model components – called modules - without taking care of the uniqueness of the IDs (nodes, elements, parts etc.). The entities of each module are described by the user and modul IDs.

The solvers internally translate the user and modul IDs of all modules into sequential numbers. In the ERF file, these sequential numbers can be found as entity IDs in the ERF result blocks. In order to retranslate these sequential numbers into pairs of user and module IDs, one has to use the correspondance tables being stored in the new *identifiers* blocks (key=250).

In the file there are *identifiers* blocks containing pairs of user and module IDs for all types of entities. Additionally, the hierarchy of all modules is stored in a special *identifiers* block named 'MODULE_HIERARCHY'.

For further details about modular I/O, please refer to the VPS solver manuals [3].

6 Solver Options

The solver and analysis types are stored in the ERF system block under the parameter 'solver_name'. The string consists of two or three qualifiers, separated by colons (:) as follows:

solver_name = <Platform> : <Analysis_Type> [: <Application_Option>]

The following table lists all possible solver and analysis types.

ERF system block parameter solver_name	Description
'PAM-CSM:Explicit_Transient'	explicit nonlinear transient analysis
'PAM-CSM:Explicit_Transient:FPM'	FPM solver results embedded
'PAM-CSM:Explicit_Transient:Stamp'	stamping analysis
'PAM-CSM:Static_Linear'	implicit linear static analysis
'PAM-CSM:Static_Linear:Composite'	with multi-layered output for composites
'PAM-CSM:Static_Nonlinear'	implicit non-linear static implicit analysis
'PAM-CSM:Static_Nonlinear:Composite'	with multi-layered output for composites
'PAM-CSM:Transient_Linear'	implicit linear transient analysis
'PAM-CSM:Transient_Linear:Composite'	with multi-layered output for composites
'PAM-CSM:Transient_Nonlinear'	implicit non-linear transient implicit analysis
'PAM-CSM:Transient_Nonlinear:Composite'	with multi-layered output for composites
'PAM-CSM:Heat_Steady_Linear'	implicit heat linear steady analysis
'PAM-CSM:Heat_Steady_Nonlinear'	implicit heat non-linear steady analysis
'PAM-CSM:Heat_Transient_Linear'	implicit heat linear transient analysis
'PAM-CSM:Heat_Transient_Nonlinear'	implicit heat non-linear transient analysis
'PAM-CSM:Acoustic_Eigen_Modes'	acoustic mode extraction
'PAM-CSM:Eigen_Modes'	structure mode extraction
'PAM-CSM:Frequency_Response'	NVH and acoustics
'PAM-CSM:Transient_Modal'	dynamic analysis based on modal superposition
'PAM-CSM:Buckling_Modes'	buckling analysis

7 Entity Types

7.1 Introduction

An ERF entity can represent a structural element (e.g. a node, a finite-element or a finite-volume cell) as well as any other sort of objects or groups of entities (e.g. the whole model, a contact interface or a multi-body system). Each entity can be referenced by its entity type key.

Notes:

- 1) An entity key may be any string of ASCII characters not containing a slash or a dot ("/" and "."), which are reserved as HDF path separators) and starting with a non-space character. The use of punctuation, non-printing characters and spaces should be avoided, as they may create problems for other software.
- 2) All identifiers (e.g. nodal, elemental, part identifiers) are one-based (greater than zero, ID > 0).

7.2 Generic Entities

Entity type	Description	Visualization
NODE	FEM nodal points	point
PART	FEM parts	n/a
MODEL	the whole model	n/a
INPUT	stripped PAM-CRASH input deck	n/a
CDATA	comments (PAM-CRASH input CDATA)	n/a
FPMNODE	FPM points	point
FPMPART	FPM parts	n/a
SENPT	sensor points (attached to a node)	point
CONTACT	contact interfaces (section "Finite Elements")	contact elements
SECTION	sections for force output	n/a
ACOUSTIC_NODE	acoustic node with 1 DOF 'acoustic pressure' in the fluid domain	point
PANEL	interface between fluid and structure domains	facets
POROUS_NODE	porous node with 2 DOF 'pressure' and 'displacement'	point

7.3 Finite Elements


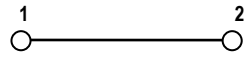
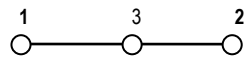
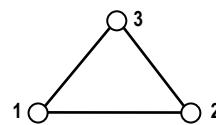
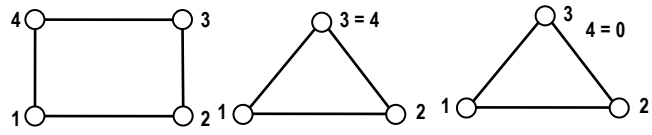
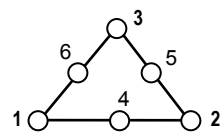
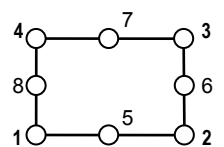
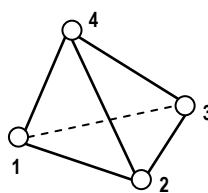
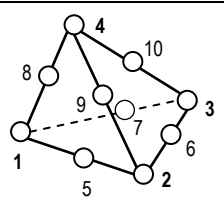
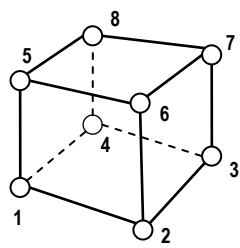
7.3.1 Overview

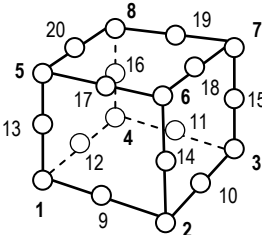
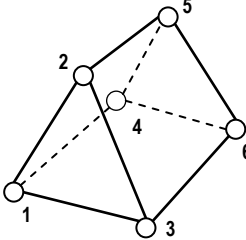
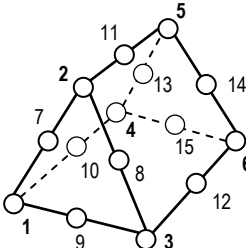
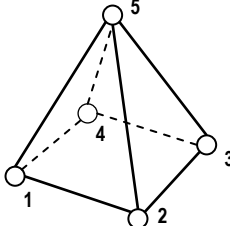
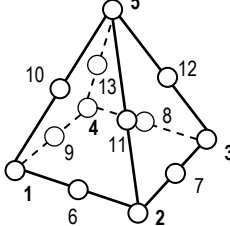
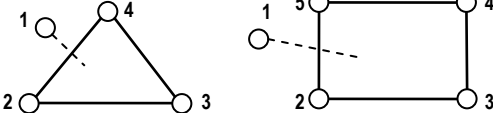
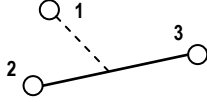
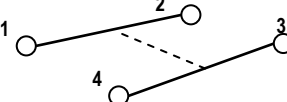
The finite elements are divided into four categories according to their dimensions in space. In addition to the classical bar and beam elements the structural link/joint elements are also considered as one-dimensional finite elements. The number of nodes and dimensions of each element type are stored as header parameter in the ERF connectivity blocks. It is highly recommended to use these parameters as primary information for visualization and extraction of faces. The purpose of the element key is to distinguish between different element formulations.

Note: The element identifiers (IDs) are unique over all types of elements. Therefore the generic type ELEMENT can also be used to refer to any type of element.

The following table shows all categories of elements used in CSM solvers. Note that an element family may consist of one or multiple element formulations, depending on the solver scheme.

Table: Types of elements

Dimensions	Family	Nodes	Connectivity
0D	point	1	
1D	line2	2	
	line3	3	
2D	tria3	3	
	quad4	4	
	tria6	6	
	quad8	8	
3D	tetra4	4	
	tetra10	10	
	hexa8	8	

	hexa20	20	
	penta6	6	
	penta15	15	
	pyramid5	5	
	pyramid13	13	
4D	node-to-segment	4 or 5	
	node-to-edge	3	
	edge-to-edge	4	

7.3.2 0D Elements

element type	generic type	family	description	visualization
SPH	PARTICLE	point	SPH particles	point
FPM	FPM	point	FPM points	point

7.3.3 1D Elements

element type	generic type	family	description	visualization
BEAM	BEAM	line2	beam C0	line
BAR	BAR	line2	bar	line
SPRING6DOF	SPRING	line2	spring with 6 DOF	line
SPHERICALJOINT	JOINT	line2	spherical joint (mat 221)	line
FLEXTORSJOINT	JOINT	line2	flexion-torsion joint (mat 222)	line
KJOINT	JOINT	line2	kinematic joint	line
PLINK	PLINK	line2	point link	line
MBSJOINT	JOINT	line2	multi-body-system joint	line
MBSSPRING	SPRING	line2	multi-body-system spring	line
SPRINGBEAM	SPRING	line2	spring-beam	line
DRAWBEAD	DRAWBEAD	line2	stamping drawbead	line
MTOJN	JOINT	line2	multiple-to-one node kinematic joint	line
MUSCLE	MUSCLE	line2	muscle	line
JET	JET	line2	gas jet	line
MPCPLINK	PLINK	line2	multi-point-constraint link	line
GAP	GAP	line2	contact gap element (implicit only)	line

7.3.4 2D Elements

element type	generic type	family	description	visualization
MEMBR	MEMBR	quad4	membrane (mat 150, 151, 152)	facet
THICKSHELL	SHELL	quad4	thick shell (mat 161, 162)	facet
SHELL	SHELL	quad4	4(3)-nodes shell element	facet
SHEL6	SHELL	tria6	6-nodes shell element	facet
SHEL8	SHELL	quad8	8-nodes shell element	facet

7.3.5 3D Elements

element type	generic type	family	description	visualization
HEXA8	SOLID	hexa8	8-node hexahedral FI	facets
BRICKSHELL	SOLID	hexa8	8-node brick shell	facets
TETRA10	SOLID	tetra10	10-node tetrahedron	facets
TETRA4	SOLID	tetra4	4-node tetrahedron	facets
PENTA6	SOLID	penta6	6-node penta	facets
PENTA15	SOLID	penta15	15-node penta	facets
HEXA20	SOLID	hexa20	20-node hexahedral	facets

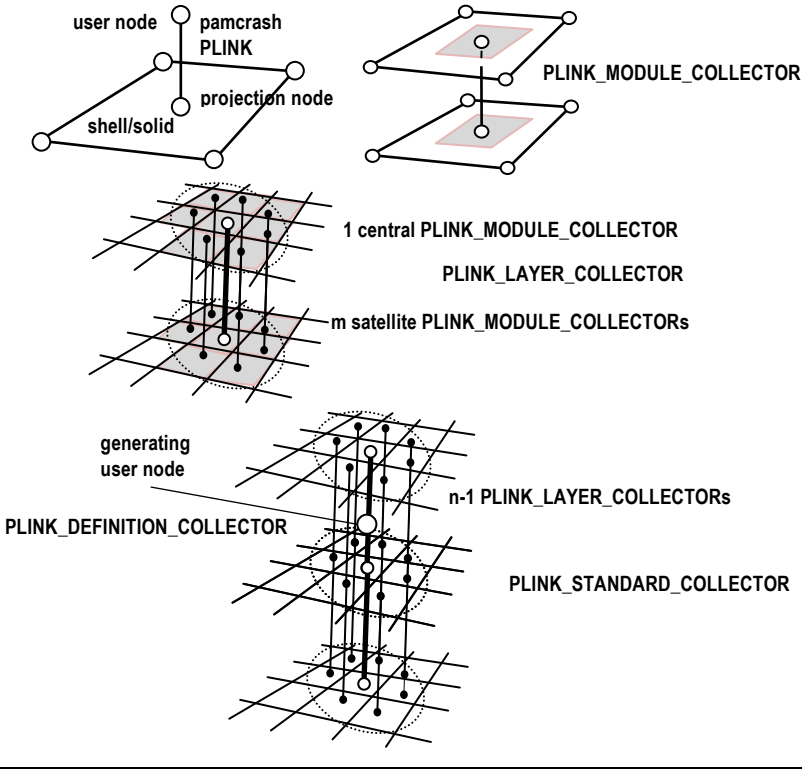
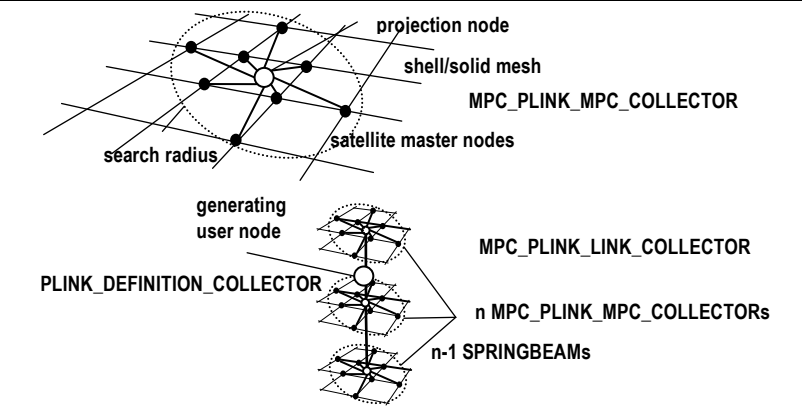
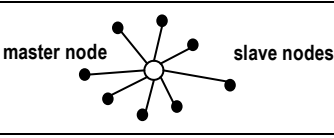
7.3.6 4D Special Purpose Elements

element type	generic type	family	description	visualization
CNT_ELE_NOD_SEG	CNT_ELE	node-to-segment	node-to-segment contact pair	point + facet
CNT_ELE_NOD_EDG	CNT_ELE	node-to-edge	node-to-edge contact pair	point + line
CNT_ELE_EDG_EDG	CNT_ELE	edge-to-edge	edge-to-edge contact pair	2 lines

7.4 Collectors

7.4.1 Introduction

A collector is a compound of entities of different types including the collector itself. Collectors are used to visualize and access associated nodes and elements, e.g. of spotwelds, airbags or rigid body definitions. The following table illustrates the visualization of collectors.

<p>PLINKs</p>	
<p>MPC_PLINKs</p>	
<p>RBODY, OTMCO_COLLECTOR</p>	

7.4.2 Collectors for Kinematic Constraints

collector type	description	visualization
RBODY	rigid body	spider between master and slave nodes
MTOCO_COLLECTOR	multiple-to-one kinematic constraint	spider between master and slave nodes
OTMCO_COLLECTOR	one-to-multiple constraint	spider between slave and master nodes

7.4.3 Collectors for Airbags

collector type	description	visualization
AIRBAG	Airbag	highlighted CHAMBER collectors
CHAMBER	Airbag chamber	highlighted elements, VENTs, WALLs
VENT	Airbag vent hole	highlighted elements and nodes
WALL	Airbag chamber wall	highlighted elements

7.4.4 Muscle Collector

collector type	description	visualization
MUSCLE_COLLECTOR	Human Model Muscle	highlighted MUSCLE elements

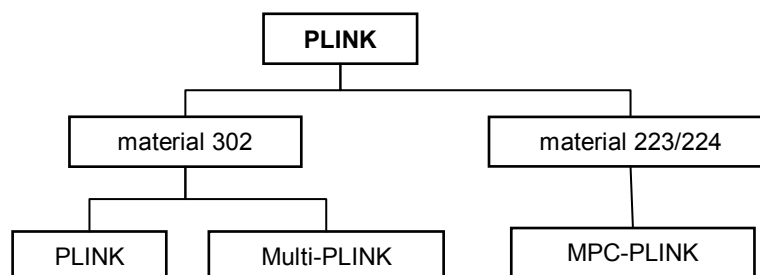
7.4.5 Tied Collector

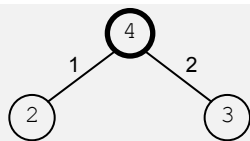
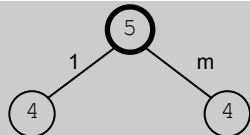
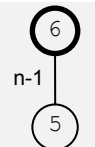
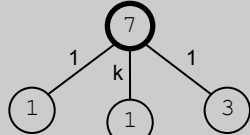
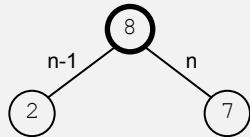
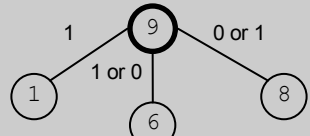
collector type	description	visualization
TIED_COLLECTOR	Tied	highlighted connected master element

7.4.6 PLINK Collectors










The purpose of PLINK collectors is to standardize the output of all PLINK entities. A schematic overview over the PLINK collectors is given below.

Note: The collector IDs are only unique for a given collector type.



collector type	description	dependencies (see legend below)
PLINK_MODULE_COLLECTOR	PLINK Module Definition	
PLINK_LAYER_COLLECTOR	Layer of a Standard-PLINK Definition	
PLINK_STANDARD_COLLECTOR	Standard-PLINK Definition	
MPC_PLINK_MPC_COLLECTOR	MPC-PLINK Constraint Definition	
MPC_PLINK_LINK_COLLECTOR	MPC-PLINK Link Definition	
PLINK_DEFINITION_COLLECTOR	MPC-/Multi-/Standard-PLINK Definition	

Legend for diagrams of collector dependencies:

symbol	entity type	definition	Identifier
	NODE	NODE	ID(1) = node ID
	<element>	1D element (BEAM or BAR)	ID(2) = element ID
	<element>	2D or 3D element (SHELL or SOLID)	ID(3) = element ID
	PLINK_MODULE_COLLECTOR	1 connecting BEAM or BAR element 2 SHELL or SOLID elements	ID(4) = element ID(2)
	PLINK_LAYER_COLLECTOR	1 central PLINK_MODULE_COLLECTOR m satellite PLINK_MODULE_COLLECTOR	ID(5) = central ID(4)
	PLINK_STANDARD_COLLECTOR	n-1 PLINK_LAYER_COLLECTORs	ID(6) = ID(9)
	MPC_PLINK_MPC_COLLECTOR	1 central NODE k satellite NODEs 1 element containing central node	ID(7) = central node ID(1)
	MPC_PLINK_LINK_COLLECTOR	n MPC_PLINK_MPC_COLLECTORs n-1 connecting BEAM elements	ID(8) = ID(9)
	PLINK_DEFINITION_COLLECTOR	1 generating NODE 0 or 1 PLINK_STANDARD_COLLECTOR 1 or 0 MPC_PLINK_LINK_COLLECTOR	ID(9) = PLINK ID

7.5 Zones

The purpose of zones is to divide a single entity (normally an element) into multiple domains to attach results to each of these domains. A typical example is a multi-layered shell element that is used to model a composite structure.

zone type	description	visualization
ZONE	default zone type – also used for single-zone entities	n/a
LAYER	Composite applications	n/a

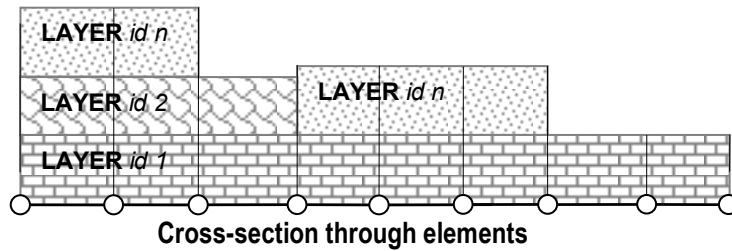


Figure: Multi-layered Shell Elements to model composite structures

Note: For detailed information about the simulation and output of composite structures please refer to the VPS solver manual [3] and the Functional Specification [4].

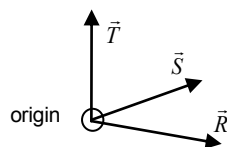
7.6 Local Frames

Vector or tensor coordinates may refer to local frames. A local frame can be referenced by the frame type key and an identifier.

frame type	description	visualization
FRAME	user-defined frame	lines
1D_ELEMENT_FRAME	local frames for beam elements	lines
2D_ELEMENT_FRAME	local frames for shell elements	lines

The three-dimensional vector basis \vec{R} , \vec{S} , \vec{T} of a local frame is stored as an entity result of the type “BASIS3X3”. In order to place the frame in space, the coordinates of the origin are stored as an result of the type “COORDINATE”.

Note: A frame is not necessarily orthonormal or orthogonal.



result	nature	ERF array storage order	description
COORDINATE	vector \vec{x}	$[x \ y \ z]$	Cartesian coordinates to store the origin
BASIS3X3	matrix \underline{B}	$[R_x \ R_y \ R_z \ S_x \ S_y \ S_z \ T_x \ T_y \ T_z]$	vector basis in Cartesian coordinates

8 File Storage Hierarchy

The hierarchy of ERF blocks is not fixed, i.e. it may vary depending on user and application settings.

Therefore, a reader must not rely on hard-coded paths to the data objects, but rather scan the file and iterate to find the data that is requested by the user.

9 References

- [1] ERF-HDF5 Specification Version 2.1, ESI Group, 2014
- [2] <http://www.hdfgroup.org/HDF5/>, HDF Group
- [3] VPS Solver Reference Manual Version 2014, ESI Group, 2014
- [4] Functional specification COMP09 Visual Composite Solution Phase2, ESI Group, 2009