

# **VTFx**

# **REFERENCE GUIDE**

## 1. General notes

- The VTFx file archive is a ZIP archive containing XML documents and binary data files
- Default extension : "\*.vtfx"

### 1.1. Terms and conventions

Term	Description
Database	A FEM analysis including geometry description and results
Case	Collection of visual settings related to result display, including view position, cut planes, visible parts, part colours, rendering method and model colours and more.
Archive	The ZIP file containing XML files and binary/text data files.

All references to files inside the VTFx ZIP archive are written using this style “*filename.ext*”.

Varying filename items, like sequence number, are indicated by <*varying items*>. A filename with four digits, “*filename0123*”, is written “*filename<four digits>*”.

To identify XML tags and attributes, prefixing using double colon of attribute is used. In the example <VTF Name=”Test”>, the tag is written “**VTF**”, and attribute “**VTF::Name.**”

## 2. Archive structure

### 2.1. ZIP archive

A VTFx file is a ZIP archive containing both XML documents and binary files. Metadata is stored in XML documents, and data blocks are stored in binary files. The VTFx file can be opened (and modified) using a standard ZIP archive application.

VTFx file format design is inspired by the open file formats used in Microsoft Office and OpenOffice.

### 2.2. Folder structure

The archive is organized with several folders. *VTF.xml* is the main file located in the root folder of the archive. This file contains a description of the current archive and references to cases and databases. Databases and cases are stored in folders with three digits used to identify the items.

A database folder has a *Database.xml* document located at the root, defining the database.

A case folder has a *Case.xml* document located at the root defining the case.

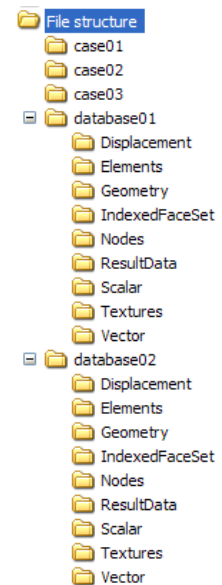
### 2.3. XML

XML documents are using the following XML declaration `version="1.0"`, `encoding="UTF-8"` and `standalone="yes"`.

All root tags in XML files are using the namespace `"http://ceetron.com"`.

#### 2.3.1. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<VTF xmlns="http://ceetron.com">
</VTF>
```



### 3. Item description

#### 3.1. Introduction

All items of the VTFx file are discussed in the following. Layout and attributes of each XML file are explained, and a short example is provided for clarity.

Data files in the database folder can be stored as binary files or readable text files. The user specifies this setting when exporting from GLviewInova. Binary files are small and compact, and text files can be used if the content of data files needs to be investigated. Binary data files have extension *.dat*, and text files have the extension *.txt*.

## 3.2. VTF

### 3.2.1. Description

Archive filename: *VTF.xml*

Main archive file with export date and digital signatures, and lists of analysis cases and databases.

### 3.2.2. Directives

Xml tag/attribute	Xml type	Required	Description
Fileinfo	Collection	Yes	
ExportDate	Date	Yes	File export date
ExportTime	Time	Yes	File export time
ExportApplication	String	Yes	Name of export application
VendorName	String	No	
ExpressSignature	String	Yes	Basic signature for the VTFx file. Used to validate file integrity when importing into <b>Gview Plugin</b> and <b>Gview Express</b> .
DigitalSignature	String	No	Advanced and highly secure digital signature for the VTFx file. Used to validate file integrity when importing into <b>Gview Plugin</b> and <b>Gview Express</b> .
Cases	Collection	Yes	
Case::ID	Int	Yes	Valid range is [0..MAX_INT]
Case::Name	String	Yes	Can be empty string
Case::Folder	String	Yes	Reference to archive folder
Case::DatabaseID	Int	Yes	Reference to database
Databases	Collection	Yes	
Database::ID	String	Yes	Valid range is [0..MAX_INT]
Database::Name	String	Yes	Can be empty string
Database::Folder	String	Yes	Reference to database folder in archive format is "Database<three digits>"

### 3.2.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<VTF xmlns="http://ceetron.com">
  <FileInfo>
    <ExportDate>2007-08-30</ExportDate>
    <ExportTime>11:21:57</ExportTime>
    <ExportApplication>GLview Inova</ExportApplication>
    <VendorName>Ceetron ASA</VendorName>
    <ExpressSignature>E8F2F77F6281AD87274B7B84E21B1F9C</ExpressSignature>
    <DigitalSignature />
  </FileInfo>
  <Cases>
    <Case ID="1" Name="Case 5" DatabaseID="1" Folder="Case001" />
  </Cases>
  <Databases>
    <Database ID="1" Name="desktop_computer.unv" Folder="Database001" />
  </Databases>
</VTF>
```

### 3.3. Case

#### 3.3.1. Description

Archive filename: *Case<three digits>\Case.xml*

A case defines a specific view of an analysis, including analysis database, visual properties, textual description and a snapshot.

#### 3.3.2. Directive

Xml tag/attribute	Xml type	Required	Description
Case::ID	Int	Yes	Valid range is [0..MAX_INT]
Case::Name	String	Yes	Can be empty string
Case::DatabaseID	Int	Yes	Reference to database
Case::Folder	String	Yes	Reference to archive folder
Case::Properties	String	Yes	Reference to case properties
Description	String	No	Reference to a HTML document with analysis details
Snapshot	String	No	Reference to a image file
Logo::ImageFilename	String	No	Reference to image file
Logo::Position	Int	No	Position in view
Logo::ScaleX	Float	No	Scale factor in x direction
Logo::ScaleY	Float	No	Scale factor in y direction
Logo::OffsetX	Int	No	Offset value in x direction
Logo::OffsetY	Int	No	Offset value in y direction

#### 3.3.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Case xmlns="http://ceetron.com" ID="1" Name="Case 5" DatabaseID="1" Folder="Case001"
  Properties="Properties.xml">
  <Logo ImageFilename="Case001\Logo.jpeg" Position="2" ScaleX="1.000000"
    ScaleY="1.000000" OffsetX="5" OffsetY="5" />
  <Snapshot>Snapshot.jpeg</Snapshot>
</Case>
```

### 3.4. Properties

#### 3.4.1. Description

Archive filename: *Case<three digits>\Properties.xml*

Collection of visual settings related to result display, including view position, cut planes, visible parts, part colours, rendering method and model colours and more.

#### 3.4.2. Directive

Xml tag/attribute	Xml type	Required	Description
VTPropertyGroupAPICollection	Collection	Yes	
VTPropertyGroupAPI::ID	Int	Yes	Valid range is [0..MAX_INT]
VTPropertyGroupAPI::Classtype	String	Yes	A string representation of the class type
Contextlist	Collection		
Contextlist Value	Int	No	Values interpreted by the given property type
Property::ID	String	Yes	A string representation of the property name
Property::Value	String	Yes	Property data value stored as string. See table below for string representation

Property type	String representation
Bool	TRUE or FALSE
Int	-10
Float	1.00000e+03
Vector	Three floats separated with a space
Matrix	16 floats separated with a space
String	
ByteColor	Three integer values in range [0..255] separated with a space
ResultID	Two integer values separated with space, ResultID and SectionID
RGBAColor	Four floats in range [0..1] separated with a space
PartItem	A set of IDs used to completely define a part item. The items are separated with space, and the values represent ID, GeometryID, StateID, ModelID, ItemIndex, and SubItemIndex

### 3.4.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<VTPropertyGroupAPICollection xmlns="http://ceetron.com">
  <VTPropertyGroupAPI ID="0" VTClassType="VT_CT_FRAME_SET_ATTRIBUTES">
    <Contextlist>
      <Value0>1</Value0>
    </Contextlist>
    <Property ID="VT_Pf_FS_OUTLINE_CREASE_ANGLE" Value="1.0472" />
    <Property ID="VT_Pf_FS_NORMAL_EDGE_THRESHOLD" Value="1.0472" />
    <Property ID="VT_Pb_FS_SHOW_UNDEFORMED_MODEL" Value="FALSE" />
    <Property ID="VT_Pc_FS_UNDEFORMED_COLOR" Value="0 0 255" />
    <Property ID="VT_Pb_FS_SHOW_BOUNDING_BOX" Value="FALSE" />
    <Property ID="VT_Pb_FS_SHOW_BBOX_ANNOTATIONS" Value="TRUE" />
    <Property ID="VT_Pc_FS_BOUNDING_BOX_COLOR" Value="255 255 255" />
    <Property ID="VT_Pi_FS_MAX_NUM_LABELS_TO_DRAW" Value="601" />
    <Property ID="VT_Pi_FS_MAX_NUM_LABELS_TO_SORT" Value="2000" />
    <Property ID="VT_Pi_FS_PART_HIGHLIGHT_MODE" Value="2" />
    <Property ID="VT_Pi_FS_MIN_NUM_PIXELS_PARTBB_CULL" Value="20" />
    <Property ID="VT_Pv_FS_USER_BOUNDING_BOX_MIN" Value="1.000000e+030
      1.000000e+030 1.000000e+030" />
    <Property ID="VT_Pv_FS_USER_BOUNDING_BOX_MAX" Value="-1.000000e+030 -1.000000e+030
      -1.000000e+030" />
  </VTPropertyGroupAPI>
</VTPropertyGroupAPICollection>
```

## 3.5. Database

### 3.5.1. Description

Archive filename: *Database<three digits>\Database.xml*

### 3.5.2. Directives

Xml tag/attribute	Xml type	Required	Description
Database::ID	String	Yes	Valid range is [0..MAX_INT]
Database::Name	String	Yes	Can be empty string
Database::Folder	String	Yes	Reference to database folder
Database::SourceName	String	No	Full filename of source analysis file

### 3.5.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Database xmlns="http://ceetron.com" ID="1" Name="desktop_computer.unv" Folder="Database001"
SourceName="C:\test\vtfx\desktop_computer.unv" />
```

## 3.6. Database table of content

### 3.6.1. Description

Archive filename: *Database<three digits>\Database-TOC.xml*

This file contains a table of content of all items in the database. This file is not present in the archive, it is automatically created. As this file gives a quick overview of all items, it is later used for fast access to database items.

### 3.6.2. Directive

Xml tag/attribute	Xml type	Required	Description
DatabaseTOC::ID	Int	Yes	Reference to database

The rest of this XML file includes reference to archive items with ID and filename.

### 3.6.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<DatabaseTOC xmlns="http://ceetron.com">
  <Nodes ID="1" File="Nodes\Nodes0001.xml" />
  <Elements ID="1" File="Elements\Elements0001.xml" />
  <Nodes ID="2" File="Nodes\Nodes0002.xml" />
  <Elements ID="2" File="Elements\Elements0002.xml" />
  <Nodes ID="3" File="Nodes\Nodes0003.xml" />
  <Elements ID="3" File="Elements\Elements0003.xml" />
  <Nodes ID="4" File="Nodes\Nodes0004.xml" />
  <Elements ID="4" File="Elements\Elements0004.xml" />
  <Nodes ID="5" File="Nodes\Nodes0005.xml" />
  <Elements ID="5" File="Elements\Elements0005.xml" />
  <Nodes ID="6" File="Nodes\Nodes0006.xml" />
  <Elements ID="6" File="Elements\Elements0006.xml" />
  <Geometry ID="1" File="Geometry\Geometry0001.xml" />
</DatabaseTOC>
```

## 3.7. Set

### 3.7.1. Description

Archive filename: *Database<three digits>\Set\Set.xml*

A set defines an element set (or group). A set can have items from multiple blocks (parts). A VTFx file can contain many sets, and the sets can be overlapping (one element can be included in more than one set).

### 3.7.2. Directives

Xml tag/attribute	Xml type	Required	Description
Set::ID	Int	Yes	Valid range is [0..MAX_INT]
Set::SetID	Int	Yes	Valid range is [0..MAX_INT]
Set::Name	String	No	Can be empty string
Set::ItemsSpecifiedAsIDs	Bool	Yes	If true, item ID is used to identify items If false, item index is used to identify items
Set::TotalNumItems	Int	Yes	Number of items in set
Set::ItemType	String	Yes	Supported type is "Element"
Items::BlockID	Int	Yes	Reference to item block
Items::BlockType	String	Yes	Supported type is "Element"
File::Filename	String	Yes	Reference to set data values
File::NumItems	Int	Yes	Number of items in the data file

### 3.7.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Set xmlns="http://ceetron.com" ID="1" SetID="0" Name="Set" ItemsSpecifiedAsIDs="false"
TotalNumItems="12312" ItemType="Element">
  <Items BlockID="1" BlockType="Elements">
    <File Filename="Set\Set0001-0001.dat" NumItems="182" />
  </Items>
  <Items BlockID="2" BlockType="Elements">
    <File Filename="Set\Set0001-0002.dat" NumItems="1271" />
  </Items>
  <Items BlockID="3" BlockType="Elements">
    <File Filename="Set\Set0001-0003.dat" NumItems="9" />
  </Items>
  <Items BlockID="4" BlockType="Elements">
    <File Filename="Set\Set0001-0004.dat" NumItems="54" />
  </Items>
</Set>
```

### 3.7.4. Data files

Archive filename (text): *Database<three digits>\Set\Set<four digits>-<four digits>.txt*

Archive filename (binary): *Database<three digits>\Set\Set<four digits>-<four digits>.dat*

## 3.8. Nodes

### 3.8.1. Description

Archive filename: *Database<three digits>\Nodes\Nodes<four digits>.xml*

Nodes are represented with tree xyz coordinate stored as tree float values. Nodes can be identified with ID. The node IDs will be stored in a separate data file.

### 3.8.2. Directive

Xml tag/attribute	Xml type	Required	Description
Nodes::ID	Int	Yes	Valid range is [0..MAX_INT]
Nodes::WithID	Bool	Yes	If true, node IDs are stored in data file If false, node indices used
File::Filename	String	Yes	Reference to data file with coordinates
File::NumItems	Int	Yes	Number of items in data file
File::IDs	String	Yes	Reference to data file with item IDs

### 3.8.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Nodes xmlns="http://ceetron.com" ID="1" WithID="1">
  <File Filename="Nodes\Nodes0001.txt" NumItems="149" IDs="Nodes\Nodes0001-IDs.txt" />
</Nodes>
```

### 3.8.4. Data files

Archive filename (text): *Database<three digits>\Nodes\Nodes<four digits>.txt*

Archive filename (text): *Database<three digits>\Nodes\Nodes<four digits>-IDs.txt*

Archive filename (binary): *Database<three digits>\Nodes\Nodes<four digits>.dat*

Archive filename (binary): *Database<three digits>\Nodes\Nodes<four digits>-IDs.dat*

### 3.9. IndexedFaceSet

#### 3.9.1. Description

Archive filename:

*Database<three digits>\IndexedFaceSet\IndexedFaceSet<four digits>.xml*

The indexed face set is used to define (parts of) a geometry. The indexed face set specifies polygons by connecting a series of nodes. The node coordinates are not contained in this block, indices are used to refer to nodes in a node block.

#### 3.9.2. Directive

Xml tag/attribute	Xml type	Required	Description
IndexedFaceSet::ID	Int	Yes	Valid range is [0..MAX_INT]
IndexedFaceSet::NodeBlock	Int	Yes	Reference to node block
IndexedFaceSet::MapToNodeIDS	Bool	Yes	If true, nodes are referenced using node IDs. If false, nodes are referenced using node indices.
IndexedFaceSet::WithID	Bool	Yes	IF true, polygon IDs are stored in a separate data file
IndexedFaceSet::NumPolygons	Int	Yes	Number of items in referenced archive file
File::Filename	String	Yes	Reference to data file
File::NumItems	Int	Yes	Number of items in data file

#### 3.9.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<IndexedFaceSet xmlns="http://ceetron.com" ID="145" NodeBlock="145" MapToNodeIDS="false"
  WithID="false" NumPolygons="13">
  <File Filename="IndexedFaceSet\IndexedFaceSet0145.dat" NumItems="26" />
</IndexedFaceSet>
```

#### 3.9.4. Data files

Archive filename (text):

*Database<three digits>IndexedFaceSet\IndexedFaceSet<four digits>.txt*

*Database<three digits>IndexedFaceSet\IndexedFaceSet<four digits>-IDs.txt*

Archive filename (binary):

*Database<three digits>IndexedFaceSet\IndexedFaceSet<four digits>.dat*

*Database<three digits>IndexedFaceSet\IndexedFaceSet<four digits>-IDs.dat*

### 3.10. Elements

#### 3.10.1. Description

Archive filename: *Database<three digits>\Elements\Element<four digit>.xml*

This block contains a collection of elements. The legal element types are listed in the next section.

#### 3.10.2. Directive

Xml tag/attribute	Xml type	Required	Description
Elements::ID	Int	Yes	Valid range is [0..MAX_INT]
Elements::NodeBlock	Int	Yes	Reference to node block
Elements::PartID	Int	Yes	Reference to PartID
Elements::MapToNodeIDS	Bool	Yes	If true, nodes are identified using IDs. If false, nodes are identified using node indices
Elements::WithID	Bool	Yes	If true, element IDs are stored in a separate data file
ElementGroup::Type	String	Yes	See section 4 for element details. Supported elements are: Beam Beam_3 Triangle Triangle_6 Quad Quad_8 Quad_9 Tetrahedron Tetrahedron_10 Hexahedron Hexahedron_20 Pentahedron Pentahedron_15 Brick Thick shell Shell Rigid Shell Point Pyramid Pyramid_13
File::Filename	String	Yes	Reference to data file
File::NumItems	Int	Yes	Number of items in data file

### 3.10.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  
<Elements xmlns="http://ceetron.com" ID="1" NodeBlock="1" MapToNodeIDs="false" WithID="false">  
  <ElementGroup Type="Triangle">  
    <File Filename="Elements\Elements0001-0001.dat" NumItems="182" />  
  </ElementGroup>  
</Elements>
```

### 3.10.4. Data files

Archive filename (text):

*Database<three digits>\Elements\Elements<four digits>-<four digits>.txt*

*Database<three digits>\Elements\Elements<four digits>-<four digits>-IDs.txt*

Archive filename (binary):

*Database<three digits>\Elements\Elements<four digits>-<four digits>.dat*

*Database<three digits>\Elements\Elements<four digits>-<four digits>-IDs.dat*

## 3.11. Geometry

### 3.11.1. Description

Archive filename: *Database<three digits>\Geometry\Geometry<four digits>.xml*

The geometry block is used for grouping various geometry blocks into a complete geometry. A geometry can contain many element blocks. The data part of this block refers to block ids (element blocks).

If the geometry is changing, geometry ID must be specified as part of the state.

### 3.11.2. Directive

Xml tag/attribute	Xml type	Required	Description
Geometry::ID	Int	Yes	Valid range is [0..MAX_INT]
Geometry::Name	String	No	
State::ID	Int	Yes	Reference to state
State::GeometryID	Int	No	Reference to geometry
Elements	Collection	Yes	
Elements::BlockID	Int	Yes	Reference to element block
Elements::PartID	Int	Yes	Reference to element part
Elements::PartName	String	Yes	Part name

### 3.11.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Geometry xmlns="http://ceetron.com" ID="1" Name="MASS FLUX ON NODES">
  <State ID="1000" GeometryID="1">
    <GLviewAPIGeometryIDs>
      <Value0>0</Value0>
    </GLviewAPIGeometryIDs>
    <Elements BlockID="1" PartID="0" PartName="Property ID 1" />
    <Elements BlockID="2" PartID="1" PartName="Property ID 2" />
    <Elements BlockID="3" PartID="2" PartName="Property ID 4" />
    <Elements BlockID="4" PartID="3" PartName="Property ID 5" />
    <Elements BlockID="5" PartID="4" PartName="Property ID 6" />
  </State>
</Geometry>
```

## 3.12. PrecomputedItem

### 3.12.1. Description

This block defines the pre-computed items cut planes, iso-surfaces, cut surfaces and particle traces.

Archive filename:

*Database<three digits>\PrecomputedItem\PrecomputedItem<four digits>.xml*

### 3.12.2. Directive

Xml tag/attribute	Xml type	Required	Description
PrecomputedItem::ID	Int	Yes	Valid range is [0..MAX_INT]
PrecomputedItem::ItemType	String	Yes	Supported types: Cutplane IsoSurface CutSurface ParticleTrace
PrecomputedItem::ItemID	Int	Yes	Reference to item of type specified by ItemType
State	Collection	Yes	
State::ID	Int	Yes	Reference to state
Part::IFSBBlockID	Int	Yes	Reference to indexed face set
Part::NodeBlockID	Int	Yes	Reference to node block
Part::FringeResultBlockID	Int	No	Reference to fringe result block
Part::ContourLinesResultBlockID	Int	No	Reference to contour lines result block
Part::VectorResultBlockID	Int	No	Reference to vector result block

### 3.12.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<PrecomputedItem xmlns="http://ceetron.com" ID="1" ItemType="Cutplane" ItemID="0">
  <State ID="1000">
    <Part IFSBlockID="6" NodeBlockID="6" FringesResultBlockID="6"
      ContourLinesResultBlockID="-1" VectorResultBlockID="-1" />
    <Part IFSBlockID="7" NodeBlockID="7" FringesResultBlockID="7"
      ContourLinesResultBlockID="-1" VectorResultBlockID="-1" />
  </State>
</PrecomputedItem>
```

### 3.13. Stateinfo

#### 3.13.1. Description

Archive filename: *Database<three digits>\StateInfo\StateInfo<four digits>.xml*

This block defines metadata for states on the VTFx file. For each step a state ID, name, ref. values etc. can be specified.

#### 3.13.2. Directive

Xml tag/attribute	Xml type	Required	Description
StateInfo	Collection		
StateInfo::ID	Int	Yes	Valid range is [0..MAX_INT]
State	Collection	Yes	
State::ID	Int	Yes	Valid range is [0..MAX_INT]
State::Name	String	No	
State::RefValue	String	Yes	A reference value for the state. See <b>State::RefType</b> for unit.
State::RefType	String	Yes	Unit used by the <b>State::RefValue</b> . Supported types are: Time Frequency LoadCase Other
State::ParentID	Int	Yes	ID of parent, -1 if the stateinfo item is root
State::Group	Bool	Yes	If true, this stateinfo item is a group item (has no step connection, but groups other states).
VTFxExportInfo::PolygonModel	Bool	No	If true, polygon model is exported without element data
VTFxExportInfo::VisiblePartsOnly	Bool	No	If true, only visible parts are exported
VTFxExportInfo::FirstOrderNodesOnly	Bool	No	If true, only first order nodes are exported
VisibleSetIDs	Collection		Collection of visible set IDs
PartFlags	Collection		Integer value per part used to define visual appearance

### 3.13.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<StateInfo xmlns="http://ceetron.com" ID="1">
  <State ID="1000" Name="Occurrence (1, 1)" RefValue="Undefined" RefType="Other"
    ParentID="-1" Group="false">
    <VTFxExportInfo PolygonModel="1" VisiblePartsOnly="0" FirstOrderNodesOnly="0">
      <VisibleSetIDs />
      <PartFlags>
        <Value0>1</Value0>
        <Value1>1</Value1>
        <Value2>1</Value2>
        <Value3>1</Value3>
        <Value4>1</Value4>
      </PartFlags>
    </VTFxExportInfo>
  </State>
</StateInfo>
```

### 3.14. Results

#### 3.14.1. Description

Archive filename: *Database<three digits>\Results\Results<four digits>.xml*

A result defines how result values are represented. This includes what type of result, the result mapping and a list of result values. Result values are stored in a separate folder, and referenced by ID.

#### 3.14.2. Directive

Xml tag/attribute	Xml type	Required	Description
Results	Collection		
Results::ID	Int	Yes	Valid range is [0..MAX_INT]
Results::Name	String	Yes	
Results::Description	String	Yes	
Results::ResultType	String	Yes	Supported types are: Scalar Vector Displacement Visibility Tensor Element
Results::ResultMapping	String	Yes	Supported mappings are: Node Element ElementNode Surface SurfaceNode Polygon PolygonNode
Results::ResultID	Int	Yes	Reference to result ID
Results::SectionID	Int	Yes	Reference to section ID
State::ID	Int	Yes	Reference to state ID
ResultValues	Collection		
Value<digit>	Int	Yes	Result value ID

### 3.14.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Results xmlns="http://ceetron.com" ID="1" Name="Temperature (ID 3) (nodes)" Description=""
  ResultType="Scalar" ResultMapping="Node" ResultID="19" SectionID="-1">
  <State ID="1000">
    <ResultValues>
      <Value0>1</Value0>
      <Value1>2</Value1>
      <Value2>3</Value2>
      <Value3>4</Value3>
      <Value4>5</Value4>
    </ResultValues>
  </State>
</Results>
```

### 3.15. ResultValues

#### 3.15.1. Description

Filename: *Database<three digits>\ResultValues\ResultValues<four digits>.xml*

This block contains a collection of results for nodes, polygons or elements. The results might be scalar results (1D) or vector results (3D). The block specifies which node, elements, or indexed face set the results belong to and how the results should be bound. The result block contains only results for one block for one time step. Multiple time steps are not allowed within one result block.

#### 3.15.2. Directive

Xml tag/attribute	Xml type	Required	Description
ResultValues::ID	Int	Yes	Valid range is [0..MAX_INT]
ResultValues::MapToBlockID	Int	Yes	Reference to node block
ResultValues::MapToBlockType	String	Yes	Can be either of the following: Nodes IndexedFaceSet Elements Geometry ResultValues Results Texture User CrossSection Directions StateInfo Set Plot2D PerElementProperties TransformationResultValues TransformationResults PrecomputedItem
ResultValues::NumDimensions	Int	Yes	Specifies the dimension of the results. 1 for scalar results and 3 for vector results.
ResultValues::WithID	Bool	Yes	If true, result value IDs are stored in a separate data file

File::Filename	String	Yes	Reference to data file
File::NumItems	Int	Yes	Number of items in data file

### 3.15.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<ResultValues xmlns="http://ceetron.com" ID="1" MapToBlockID="2" MapToBlockType="Nodes"
  NumDimensions="3" WithID="false">
  <File Filename="ResultValues\ResultValues0001.txt" NumItems="16" />
</ResultValues>
```

### 3.15.4. Data files

Archive filename (text):

*Database<three digits>\ResultValues\ResultValues <four digits>.txt*

*Database<three digits>\ResultValues\ResultValues <four digits>-IDs.txt*

Archive filename (binary):

*Database<three digits>\ResultValues\ResultValues <four digits>.dat*

*Database<three digits>\ResultValues\ResultValues <four digits>-IDs.dat*

### 3.16. TransformationResults

#### 3.16.1. Description

Filename:

*Database<three digits>\TransformationResults\ TransformationResult <four digits>.xml*

One transformation matrix can be applied to each element or indexed face set block per step. The transformations are specified as 3 by 4 matrices, one for each step. The transformation matrix is in the following format:

$$M = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \end{bmatrix}$$

This is equivalent to the standard four by four transformation matrices used with homogeneous coordinates, where the fourth column is taken to be  $[0 \ 0 \ 0 \ 1]^T$ . The coordinates  $(x_0, y_0, z_0)$  are transformed into  $(x,y,z)$  as specified in the following equation:

$$\begin{bmatrix} x & y & z \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & z_0 & 1 \end{bmatrix} \bullet M$$

#### 3.16.2. Directive

Xml tag/attribute	Xml type	Required	Description
TransformationResult::ID	Int	Yes	
TransformationResult::Name	String	Yes	
TransformationResult::ResultID	Int	Yes	Reference to result
State::ID	Int	Yes	Reference to state
ResultValues	Collection	Yes	
Value<digit>	Int	Yes	Result values

### 3.16.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<TransformationResult xmlns="http://ceetron.com" ID="1"
    Name="Rigid body transformation result for dies" ResultID="20">
  <State ID="0">
    <ResultValues>
      <Value0>1</Value0>
      <Value1>2</Value1>
      <Value2>3</Value2>
    </ResultValues>
  </State>
</TransformationResult>
```

### 3.16.4. Data files

Archive filename (text):

*Database<three digits>\ResultValues\ResultValues <four digits>.txt*

Archive filename (binary):

*Database<three digits>\ResultValues\ResultValues <four digits>.dat*

### 3.17. TransformationResultValues

#### 3.17.1. Description

Filename: *Database<three digits>\TransformationResultValues\  
TransformationResultValues<four digits>.xml*

This block contains a 4\*3 transformation matrix for one indexed face set or element block. The transformations are specified as 3 by 4 matrices, one for each step. The transformation matrix is in the following format:

$$M = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \end{bmatrix}$$

This is equivalent to the standard 4\*4 transformation matrices used with homogeneous coordinates, where the fourth column is taken to be  $[0 \ 0 \ 0 \ 1]^T$ . The coordinates  $(x_0, y_0, z_0)$  are transformed into  $(x,y,z)$  as specified in the following equation:

$$[x \ y \ z] = [x_0 \ y_0 \ z_0 \ 1] \bullet M$$

#### 3.17.2. Directives

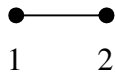
Xml tag/attribute	Xml type	Required	Description
TransformationResultValue::ID	Int	Yes	
TransformationResultValue::MapToBlockID	Int	Yes	
TransformationResultValue::MapToBlockType	String	Yes	Elements
TransformationResultValue::Row1	String	Yes	Three values representing row 1 of matrix M
TransformationResultValue::Row2	String	Yes	Three values representing row 2 of matrix M
TransformationResultValue::Row3	String	Yes	Three values representing row 3 of matrix M
TransformationResultValue::Row4	String	Yes	Three values representing row 4 of matrix M

### 3.17.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<TransformationResultValue xmlns="http://ceetron.com" ID="1" MapToBlockID="1"
MapToBlockType="Elements">
  <Row1>1.000000 0.000000 0.000000 0.000000</Row1>
  <Row2>0.000000 1.000000 0.000000 0.000000</Row2>
  <Row3>0.000000 0.000000 1.000000 0.000000</Row3>
  <Row4>0.000000 0.000000 0.000000 1.000000</Row4>
</TransformationResultValue>
```

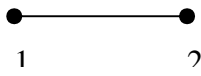
## 4. Supported element types

### 4.1. Points

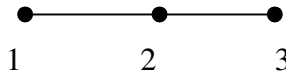


POINTS

### 4.2. Beams

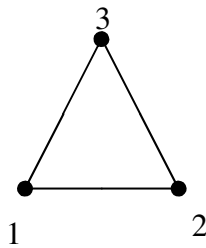


BEAMS

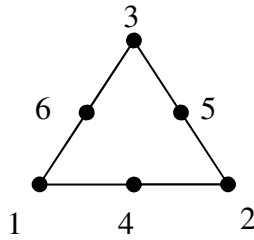


BEAMS\_3

### 4.3. Triangles

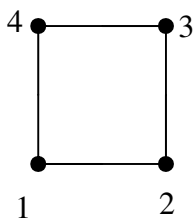


TRIANGLES

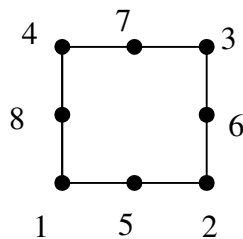


TRIANGLES\_6

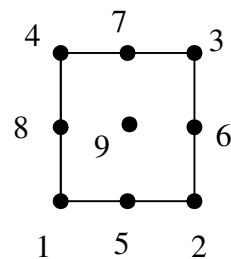
### 4.4. Quads



QUADS

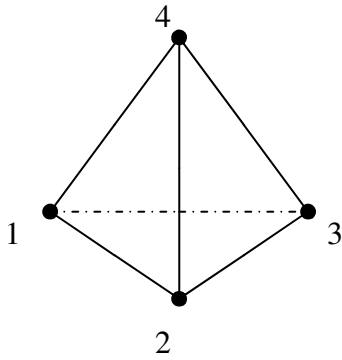


QUADS\_8

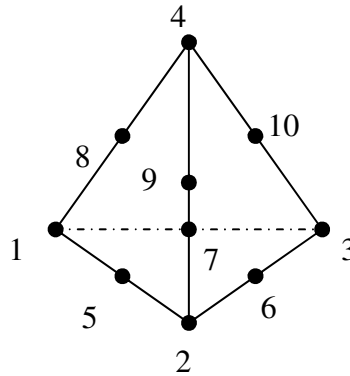


QUADS\_9

4.5. Tetrahedrons

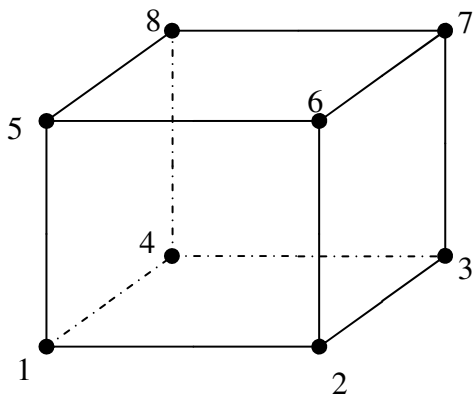


TETRAHEDRONS

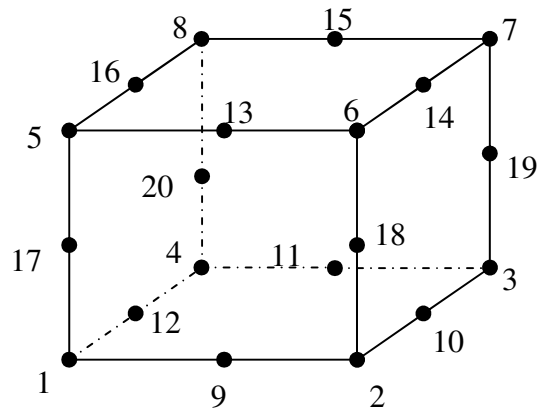


TETRAHEDRONS\_10

4.6. Hexahedrons

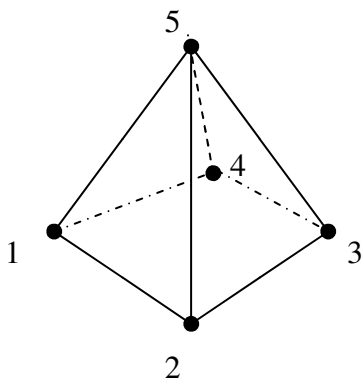


HEXAHEDRONS

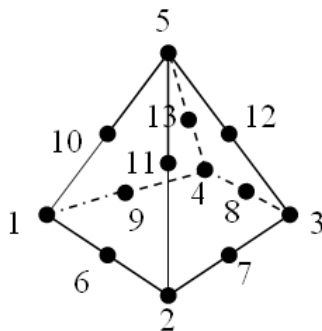


HEXAHEDRONS\_20

### 4.7. Pyramid

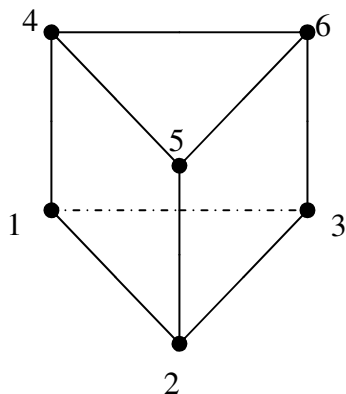


PYRAMIDS

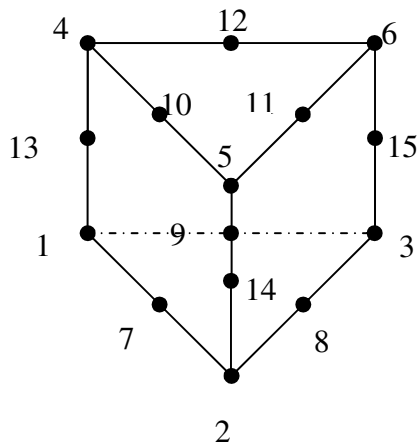


PYRAMIDS\_13

### 4.8. Pentahedrons



PENTAHEDRONS



PENTAHEDRONS\_15