VTFx

REFERENCE GUIDE

Table of Contents

1. General notes

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

1.1. Terms and conventions

Term	**Description**
Database	**A collection of files defining a single or multiple-state finite-element analysis. Each state contains geometries and results defined on these geometries. Each geometry is made of one or more parts, defined by a collection of finite elements built on a set of nodes.**
Case	Collection of visual settings related to the display of parts and results, including view position, cut planes, visible parts, part colors, draw styles and more.
Archive	The ZIP file containing XML files and binary/text data files.

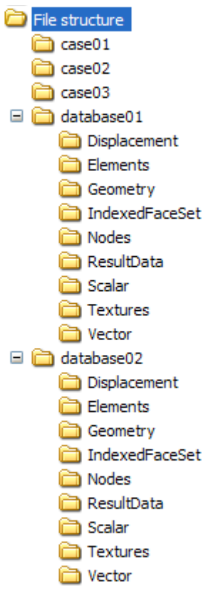
All references to files inside the VTFx ZIP archive are written using this style ***"filename.ext"***.

Varying filename items, like sequence number, are indicated by ***<varying items>***. A filename with four digits, ***"filname0123"***, 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 such as 7-zip.

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

2.2. Folder structure

The archive structure is a tree of 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 of the folder containing the files defining the database.

A case folder has a ***Case.xml*** document located at the root of the folder that contains the files defining the case.

2.3. XML

XML documents use the following XML declaration : version=”1.0”, encoding=”UTF-8” and standalone=”yes”.

All root tags in XML files use the namespace ”http://ceetron.com”.

2.3.1. Example

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

3. Item description

3.1. Introduction

This paragraph focuses on all items of a VTFx file. 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 the file, using the VTFx or Export Components. 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***

This is the main archive file. It contains the export date and digital signatures, and the lists of cases and databases.

3.2.2. Directives

Xml tag/attribute	**Xml type**	**Required**	**Description**
Fileinfo	**Collection**	**Yes**	**Container for the following tags:**
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 that the file has been exported by a licensed application when it is imported by **Ceetron 3D Viewer or Ceetron 3D Plugin for Microsoft Office**.
DigitalSignature	String	No	Advanced and highly secure digital signature for the VTFx file. Used to validate file integrity when it is imported by **Ceetron 3D Viewer** or **Ceetron 3D Plugin for Microsoft Office**.
Cases	**Collection**	**Yes**	**Container for the following tags:**
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**	**Container for the following tags:**
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
<?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 AS</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 its 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>\VTFxProperties.xml***

Collection of visual settings related to part and 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**
VTFxProperties	Collection	Yes	Root element of xml file
PropertySetCollection	Collection	Yes	
PropertySet	Collection	Yes	
PropertySet::classType	String	Yes	The name of the property group
PROPERTY_TYPE	Element		Element with a property of a given type (see below
PROPERTY_TYPE:key	Collection	Yes	Property name
PROPERTY_TYPE val	Element val	Yes	Property value

Property type	**Element value string representation**
Bool	true or false
Int	-10
Uint	10
Float	12.3 or 1.23e+01
Double	12.3 or 1.23e+01
Vec3d	Three floats separated with a space. E.g. 0.1 0.5 2.3
Color3f	0.0 -> 1.0 per component. E.g.: 1 0 1
String	The string
Array	Three integer values in range [0..255] separated with a space

The documentation of the properties in VTFx can be found in the online help of Ceetron 3D Components.

3.4.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<VTFxProperties xmlns="http://ceetron.com">
   <PropertySetCollection>
      <PropertySet classType="part_settings">
         <Color3f key="color">1 0 1</Color3f>
         <UInt key="context_geometry_index">0</UInt>
         <Int key="context_part_id">1</Int>
         <String key="draw_style">line</String>
      </PropertySet>
      <PropertySet classType="part_settings">
         <UInt key="context_geometry_index">0</UInt>
         <Int key="context_part_id">2</Int>
         <Bool key="visible">false</Bool>
      </PropertySet>
      <PropertySet classType="result_selection">
         <Int key="fringes_result_id">1</Int>
      </PropertySet>
      <PropertySet classType="result_selection">
         <Array key="vector_result_ids">
            <Int>1</Int>
         </Array>
      </PropertySet>
      <PropertySet classType="color_mapper_filled_contours_uniform">
         <String key="color_scheme">thermal_1</String>
         <Int key="context_result_id">1</Int>
         <Double key="range_max">5</Double>
         <Double key="range_min">1</Double>
      </PropertySet>
   </PropertySetCollection>
```

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. Nodes

3.7.1. Description

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

Nodes are defined by three xyz coordinates stored as three float values. Nodes can be identified with ID. The node IDs will be stored in a separate data file.

3.7.2. 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

3.7.3. Example

```xml
<?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>
```

3.7.4. 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***

3.8. Elements

3.8.1. 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.

3.8.2. 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 chapter 3.16 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`
			`Point`
			`Pyramid`
			`Pyramid_13`

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

3.8.3. 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>
```

3.8.4. 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

3.9. Geometry

3.9.1. Description

Archive filename: ***Database<three digits>\Geometry\Geometry0001.xml***

This block defines the geometries for the states in the database. If only one geometry is defined, it applies to all states. If not, a geometry must be specified for each and every state (adaptive models).

It is legal to specify the same element blocks in multiple states in order to optimize resource usage. This is useful when having a partially adaptive model (see example below).

There should be only one Geometry block in a database with block ID = 1.

3.9.2. Directive

Xml tag/attribute	Xml type	Required	Description
Geometry::ID	Int	Yes	Block ID. Always 1.
State::ID	Int	Yes	ID of the state. Needs to be defined in the StateInfo block.
Geometry::Index	Int	Yes	The zero based index of the geometry. Must match the geometries defined in the GeometryInfo block.
Elements::PartID	Int	Yes	The ID of the part. Must match the definition of the geometry in the GeometryInfo block.
Elements::BlockID	Int	Yes	The ID of the Elements block defining the part for this state.

3.9.3. Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Geometry xmlns="http://ceetron.com" ID="1">
  <State ID="0">
    <Geometry Index="0">
      <Elements PartID="0" BlockID="1"  />
    </Geometry>
    <Geometry Index="1">
      <Elements PartID="1000000" BlockID="10" />
      <Elements PartID="1000001" BlockID="11" />
    </Geometry>
  </State>
  <State ID="1">
    <Geometry Index="0">
      <Elements PartID="0" BlockID="2"  />
    </Geometry>
    <Geometry Index="1">
      <Elements PartID="1000000" BlockID="10" />
      <Elements PartID="1000001" BlockID="11" />
    </Geometry>
  </State>
</Geometry>
```

3.10. GeometryInfo

3.10.1. Description

Archive filename: ***Database<three digits>\GeometryInfo\GeometryInfo0001.xml***

This block defines metadata for the geometries in the database. All states must have the same number of geometries specified in the GeometryCountPerState attribute. Each geometry must be given a zero-based index. This index is used in the Geometry block when setting up the geometry contents in the various states.

This block defines the IDs and names of the part within the geometry. The IDs need only to be unique within a geometry. The names are optional.

There should be only one GeometryInfo block in a database with block ID = 1.

3.10.2. Directive

Xml tag/attribute	**Xml type**	**Required**	**Description**
GeometryInfo::ID	Int	Yes	Block ID. Always 1
GeometryInfo::GeometryCountPerState	Int	Yes	Number of geometries defined in this block
Geometry::GeometryIndex	Int	Yes	
Part::PartID	Int	Yes	The ID of the part. Has to be unique within the geometry.
Part::PartName	String	No	The name of the part

3.10.3. Example

```xml
<GeometryInfo xmlns="http://ceetron.com" ID="1" GeometryCountPerState="2">
  <Geometry GeometryIndex="0">
    <Part PartID="0" PartName="billet_" />
  </Geometry>
  <Geometry GeometryIndex="1">
    <Part PartID="1000000" PartName="upsetting - 1" />
    <Part PartID="1000001" PartName="upsetting - 2" />
  </Geometry>
</GeometryInfo>
```

3.11. Results

3.11.1. Description

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

This block defines the metadata of a result (Type, ID, name, mapping) and which ResultValues blocks that defines the result for the given states.

The result ID is used when referencing the result from properties (e.g. to show as fringes or map on a cutting plane) and is also used internally when loading the file in an application.

3.11.2. Directive

Xml tag/attribute	Xml type	Required	Description
Results::ID	Int	Yes	Block ID. Must be unique within the database.
Results::Name	String	Yes	The name of the result
Results::ResultType	String	Yes	Supported types are: • Scalar • Vector • Displacement
Results::ResultMapping	String	Yes	Supported mappings are: • Node • Element • ElementNode • Surface
Results::ResultID	Int	Yes	The ID of the result. Used when specifying properties.
State::ID	Int	Yes	ID of the state the subsequent ResultValues blocks applies to.
ResultValues	Collection		
Value<digit>	Int	Yes	ID of the ResultValues block with the results for the given state.

3.11.3. Example

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

3.12. ResultValues

3.12.1. 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), a vector results (3D) or a displacement result (3D). The block specifies the ID of the block in which to map the results to. This is either a Nodes or Elements block depending on the result mapping specified in the Results block.

The result values block contains only results for one time step.

3.12.2. Directive

Xml tag/attribute	**Xml type**	**Required**	**Description**
ResultValues::ID	Int	Yes	Block ID. Must be unique within the database.
ResultValues::MapToBlockID	Int	Yes	Reference to node or element block depending on the mapping specified in the Results block.
ResultValues::NumDimensions	Int	Yes	Specifies the dimension of the results. 1 for scalar results and 3 for vector or displacement 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.

3.12.3. Example

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

3.12.4. 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

3.13. TransformationResults

3.13.1. Description

Filename:
Database<three digits>\TransformationResults\ TransformationResult <four digits>.xml

This block is used to defined rigid body transformation of parts for each state.

You can specify per state which TransformationResultValues blocks to use.

3.13.2. Directives

Xml tag/attribute	**Xml type**	**Required**	**Description**
TransformationResult::ID	Int	Yes	Block ID. Must be unique within the database.
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

3.13.3. 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>
```

3.13.4. Data files

Archive filename (text):
Database<three digits>\ ResultValues\ResultValues <four digits>.txt
Archive filename (binary):
Database<three digits>\ ResultValues\ResultValues <four digits>.dat

3.14. TransformationResultValues

3.14.1. Description

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

This block contains a 4*4 transformation matrix for one element block (i.e. a part). The transformations are specified as 4 by 4 matrices, one for each step. The transformation matrix is in the following format:

$$M = \begin{bmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{bmatrix}$$

The coordinates (x_0, y_0, z_0) are transformed into (x,y,z) as specified in the following equation:

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

3.14.2. Directives

Xml tag/attribute	Xml type	Required	Description
TransformationResultValue::ID	Int	Yes	Block ID. Must be unique within the database.
TransformationResultValue::MapToBlockID	Int	Yes	Element block id to which the transformation applies
TransformationResultValue::MapToBlockType	String	Yes	Elements
TransformationResultValue::Row1	String	Yes	Values for row 1
TransformationResultValue::Row2	String	Yes	Values for row 2
TransformationResultValue::Row3	String	Yes	Values for row 3
TransformationResultValue::Row4	String	Yes	Values for row 4

3.14.3. 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>
```

3.15. Set

3.15.1. Description

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

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

3.15.2. 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

3.15.3. Example

```xml
<?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>
```

3.15.4. 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***

3.16. Stateinfo

3.16.1. Description

Archive filename: ***Database<three digits>\StateInfo\StateInfo0001.xml***

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

There should be only one StateInfo block in a database and its block ID must be 1.

3.16.2. Directive

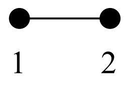
Xml tag/attribute	Xml type	Required	Description
StateInfo::ID	Int	Yes	Block ID. Always 1.
State::ID	Int	Yes	Valid range is [0..MAX_INT]
State::Name	String	No	
State::RefValue	String	Yes	A reference value for the state. See **State::RefType** for unit.
State::RefType	String	Yes	Unit used by the **State::RefValue.** 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).

3.16.3. 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">
  </State>
</StateInfo>
```

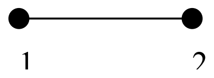
4. Supported element types

4.1. Points

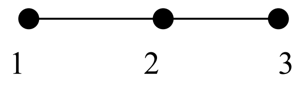


POINTS

4.2. Beams

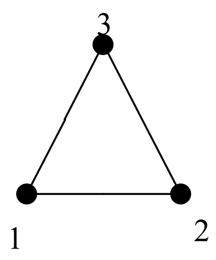


BEAMS

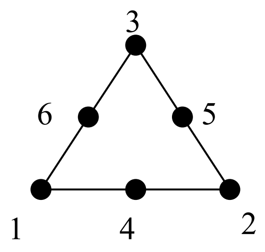


BEAMS_3

4.3. Triangles

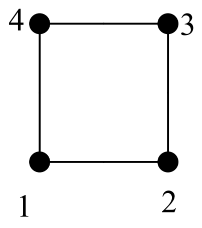


TRIANGLES

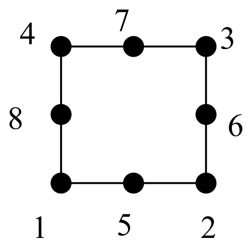


TRIANGLES_6

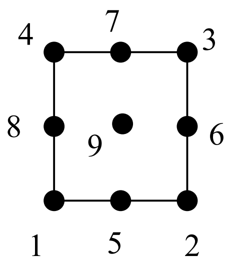
4.4. Quads



QUADS

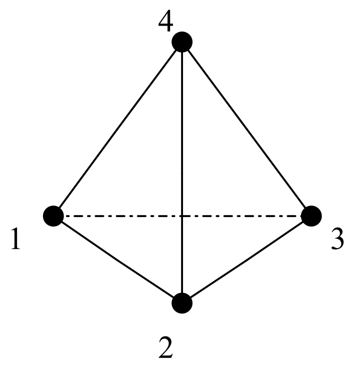


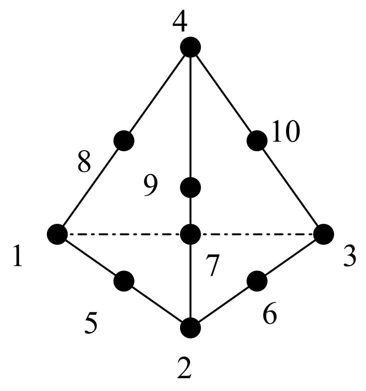
QUADS_8



QUADS_9

4.5. Tetrahedrons

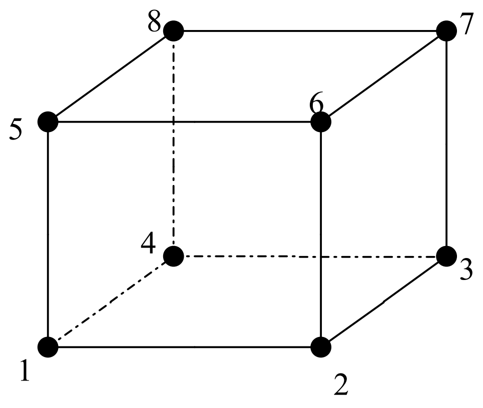


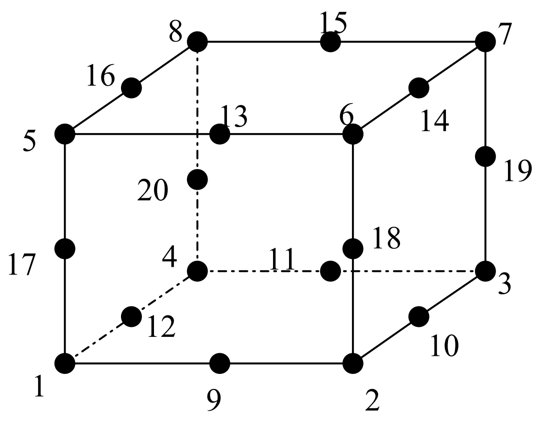


TETRAHEDRONS

TETRAHEDRONS_10

4.6. Hexahedrons

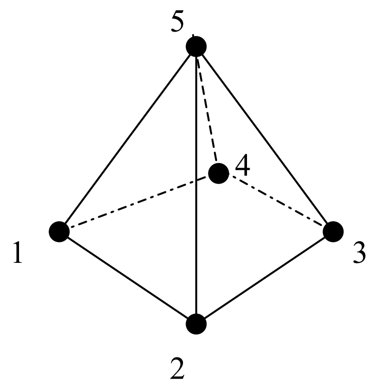




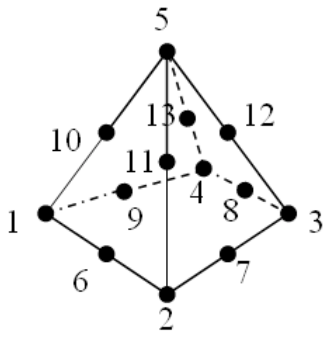
HEXAHEDRONS

HEXAHEDRONS_20

4.7. Pyramid

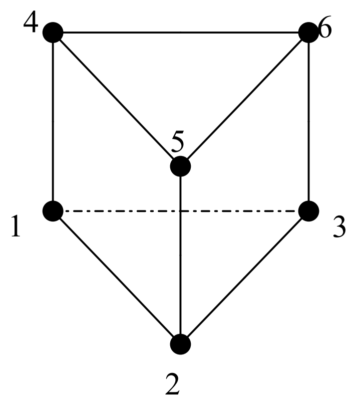


PYRAMIDS

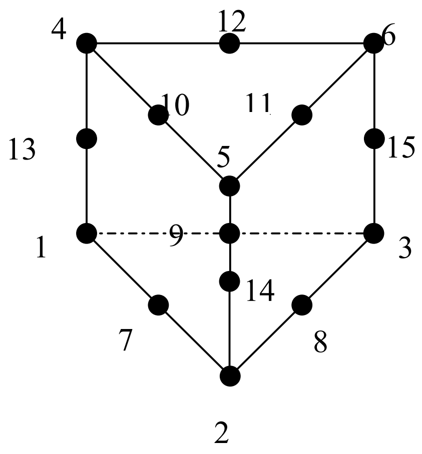


PYRAMIDS_13

4.8. Pentahedrons



PENTAHEDRONS



PENTAHEDRONS_15