Open CASCADE Technology
7.3.0


IGES Support


May 26, 2018

# Contents

# 1   Introduction

The IGES interface reads IGES files and translates them to Open CASCADE Technology models. The interface is able to translate one entity, a group of entities or a whole file. Before beginning a translation, you can set a range of parameters to manage the translation process. If you like, you can also check file consistency before translation. The IGES interface also translates OCCT models to IGES files.

Other kinds of data such as colors and names can be read or written with the help of XDE tools *IGESCAFControl↩_Reader* and *IGESCAFControl_Writer*.

Please, note:

- an IGES model is an IGES file that has been loaded into memory.

- an IGES entity is an entity in the IGES normal sense.

- a root entity is the highest level entity of any given type, e.g. type 144 for surfaces and type 186 for solids. Roots are not referenced by other entities.

This manual mainly explains how to convert an IGES file to an Open CASCADE Technology (**OCCT**) shape and vice versa. It provides basic documentation on conversion. For advanced information on conversion, see our E-learning & Training offerings.

IGES files produced in accordance with IGES standard versions up to and including version 5.3 can be read. IGES files that are produced by this interface conform to IGES version 5.3 (Initial Graphics Exchange Specification, IGES 5.3. ANS US PRO/IPO-100-1996).

This manual principally deals with two OCCT classes:

- The Reader class, which loads IGES files and translates their contents to OCCT shapes,

- The Writer class, which translates OCCT shapes to IGES entities and then writes these entities to IGES files.

File translation is performed in the programming mode, via C++ calls, and the resulting OCCT objects are shapes.

All definitions in IGES version 5.3 are recognized but only 3D geometric entities are translated. When the processor encounters data, which is not translated, it ignores it and writes a message identifying the types of data, which was not handled. This message can be written either to a log file or to screen output.

Shape Healing toolkit provides tools to heal various problems, which may be encountered in translated shapes, and to make them valid in Open CASCADE. The Shape Healing is smoothly connected to IGES translator using the same API, only the names of API packages change.

## 2 Reading IGES

### 2.1 Procedure

You can translate an IGES file to an OCCT shape by following the steps below:

1. Load the file,
2. Check file consistency,
3. Set the translation parameters,
4. Perform the file translation,
5. Fetch the results.

### 2.2 Domain covered

#### 2.2.1 Translatable entities

The types of IGES entities, which can be translated, are:

- Points
- Lines
- Curves
- Surfaces
- B-Rep entities
- Structure entities (groups). Each entity in the group outputs a shape. There can be a group of groups.
- Subfigures. Each entity defined in a sub-figure outputs a shape
- Transformation Matrix.

**Note** that all non-millimeter length unit values in the IGES file are converted to millimeters.

#### 2.2.2 Attributes

Entity attributes in the Directory Entry Section of the IGES file (such as layers, colors and thickness) are translated to Open CASCADE Technology using XDE.

#### 2.2.3 Administrative data

Administrative data, in the Global Section of the IGES file (such as the file name, the name of the author, the date and time a model was created or last modified) is not translated to Open CASCADE Technology. Administrative data can, however, be consulted in the IGES file.

### 2.3 Description of the process

#### 2.3.1 Loading the IGES file

Before performing any other operation, you have to load the file using the syntax below.

```
IGESControl_Reader reader;
IFSelect_ReturnStatus stat  = reader.ReadFile("filename.igs");
```

The loading operation only loads the IGES file into computer memory; it does not translate it.

### 2.3.2 Checking the IGES file

This step is not obligatory. Check the loaded file with:

```
Standard_Boolean ok =  reader.Check(Standard_True);
```

The variable "ok is True" is returned if no fail message was found; "ok is False" is returned if there was at least one fail message.

```
reader.PrintCheckLoad  (failsonly, mode);
```

Error messages are displayed if there are invalid or incomplete IGES entities, giving you information on the cause of the error.

```
Standard_Boolean failsonly  = Standard_True or Standard_False;
```

If you give True, you will see fail messages only. If you give False, you will see both fail and warning messages.

Your analysis of the file can be either message-oriented or entity-oriented. Choose your preference with *IFSelect↩ _PrintCount mode = IFSelect_xxx*, where *xxx* can be any of the following:

- *ItemsByEntity* gives a sequential list of all messages per IGES entity.

- *CountByItem* gives the number of IGES entities with their types per message.

- *ShortByItem* gives the number of IGES entities with their types per message and displays rank numbers of the first five IGES entities per message.

- *ListByItem* gives the number of IGES entities with their type and rank numbers per message.

- *EntitiesByItem* gives the number of IGES entities with their types, rank numbers and Directory Entry numbers per message.

### 2.3.3 Setting translation parameters

The following parameters can be used to translate an IGES file to an OCCT shape. If you give a value that is not within the range of possible values, it will be ignored.

**read.iges.bspline.continuity**

manages the continuity of BSpline curves (IGES entities 106, 112 and 126) after translation to Open CASCADE Technology (Open CASCADE Technology requires that the curves in a model be at least C1 continuous; no such requirement is made by IGES).

- 0: no change; the curves are taken as they are in the IGES file. C0 entities of Open CASCADE Technology may be produced.

- 1: if an IGES BSpline, Spline or CopiousData curve is C0 continuous, it is broken down into pieces of C1 continuous *Geom_BSplineCurve*.

- 2: This option concerns IGES Spline curves only. IGES Spline curves are broken down into pieces of C2 continuity. If C2 cannot be ensured, the Spline curves will be broken down into pieces of C1 continuity.

Read this parameter with:

```
Standard_Integer ic =  Interface_Static::IVal("read.iges.bspline.continuity");
```

Modify this value with:

```
if  (!Interface_Static::SetIVal ("read.iges.bspline.continuity",2))
.. error ..;
```

Default value is 1.

This parameter does not change the continuity of curves that are used in the construction of IGES BRep entities. In this case, the parameter does not influence the continuity of the resulting OCCT curves (it is ignored).

**read.precision.mode**

reads the precision value.

- File (0) the precision value is read in the IGES file header (default).

- User (1) the precision value is that of the read.precision.val parameter.

Read this parameter with:

```
Standard_Integer ic =  Interface_Static::IVal("read.precision.mode");
```

Modify this value with:

```
if  (!Interface_Static::SetIVal ("read.precision.mode",1))
.. error ..;
```

Default value is *File* (0).

**read.precision.val**

User defined precision value. This parameter gives the precision for shape construction when the read.precision.↩ mode parameter value is 1. By default it is 0.0001, but can be any real positive (non null) value.

This value is in the measurement unit defined in the IGES file header.

Read this parameter with:

```
Standard_Real rp =  Interface_Static::RVal("read.precision.val");
```

Modify this parameter with:

```
if  (!Interface_Static::SetRVal ("read.precision.val",0.001))
.. error ..;
```

Default value is 0.0001.

The value given to this parameter is a target value that is applied to *TopoDS_Vertex, TopoDS_Edge* and *TopoDS↩ _Face* entities. The processor does its best to reach it. Under certain circumstances, the value you give may not be attached to all of the entities concerned at the end of processing. IGES-to-OCCT translation does not improve the quality of the geometry in the original IGES file. This means that the value you enter may be impossible to attain the given quality of geometry in the IGES file.

Value of tolerance used for computation is calculated by multiplying the value of *read.precision.val* and the value of coefficient of transfer from the file units to millimeters.

**read.maxprecision.mode**

defines the mode of applying the maximum allowed tolerance. Its possible values are:

- *Preferred(0)* maximum tolerance is used as a limit but sometimes it can be exceeded (currently, only for deviation of a 3D curve of an edge from its pcurves and from vertices of such edge) to ensure shape validity;

- *Forced(1)* maximum tolerance is used as a rigid limit, i.e. it can not be exceeded and, if this happens, tolerance is trimmed to suit the maximum-allowable value.

Read this parameter with:

```
Standard_Integer mv =  Interface_Static::IVal("read.maxprecision.mode");
```

Modify this parameter with:

```
if  (!Interface_Static::SetIVal ("read.maxprecision.mode",1))
.. error ..;
```

Default value is *Preferred (0)*.

**read.maxprecision.val**

defines the maximum allowable tolerance (in mm) of the shape. It should be not less than the basis value of tolerance set in processor (either Resolution from the file or *read.precision.val*). Actually, the maximum between *read.maxprecision.val* and basis tolerance is used to define maximum allowed tolerance. Read this parameter with:

```
Standard_Real rp =  Interface_Static::RVal("read.maxprecision.val");
```

Modify this parameter with:

```
if  (!Interface_Static::SetRVal ("read.maxprecision.val",0.1))
.. error ..;
```

Default value is 1.

**read.stdsameparameter.mode**

defines the using of *BRepLib::SameParameter*. Its possible values are:

- 0 (Off) – *BRepLib::SameParameter* is not called,

- 1 (On) – *BRepLib::SameParameter* is called. *BRepLib::SameParameter* is used through *ShapeFix_Edge↩ ::SameParameter*. It ensures that the resulting edge will have the lowest tolerance taking pcurves either unmodified from the IGES file or modified by *BRepLib::SameParameter*. Read this parameter with:

  ```
  Standard_Integer mv =  Interface_Static::IVal("read.stdsameparameter.mode");
  ```

  Modify this parameter with:

  ```
  if  (!Interface_Static::SetIVal ("read.stdsameparameter.mode",1))
  .. error ..;
  ```

  Deafault value is 0 (Off).

**read.surfacecurve.mode**

preference for the computation of curves in case of 2D/3D inconsistency in an entity which has both 2D and 3D representations.

Here we are talking about entity types 141 (Boundary), 142 (CurveOnSurface) and 508 (Loop). These are entities representing a contour lying on a surface, which is translated to a *TopoDS_Wire*, formed by *TopoDS_Edges*. Each *TopoDS_Edge* must have a 3D curve and a 2D curve that reference the surface.

The processor also decides to re-compute either the 3D or the 2D curve even if both curves are translated successfully and seem to be correct, in case there is inconsistency between them. The processor considers that there is inconsistency if any of the following conditions is satisfied:

- the number of sub-curves in the 2D curve is different from the number of sub-curves in the 3D curve. This can be either due to different numbers of sub-curves given in the IGES file or because of splitting of curves during translation.

- 3D or 2D curve is a Circular Arc (entity type 100) starting and ending in the same point (note that this case is incorrect according to the IGES standard).

The parameter *read.surfacecurve.mode* defines which curve (3D or 2D) is used for re-computing the other one:

- *Default(0)* use the preference flag value in the entity's Parameter Data section. The flag values are:
    - 0: no preference given,
    - 1: use 2D for 142 entities and 3D for 141 entities,
    - 2: use 3D for 142 entities and 2D for 141 entities,
    - 3: both representations are equally preferred.
- *2DUse_Preferred (2)* : the 2D is used to rebuild the 3D in case of their inconsistency,
- *2DUse_Forced (-2)*: the 2D is always used to rebuild the 3D (even if 3D is present in the file),
- *3DUse_Preferred (3)*: the 3D is used to rebuild the 2D in case of their inconsistency,
- *3DUse_Forced (-3)*: the 3D is always used to rebuild the 2D (even if 2D is present in the file),

If no preference is defined (if the value of *read.surfacecurve.mode* is *Default* and the value of the preference flag in the entity's Parameter Data section is 0 or 3), an additional analysis is performed.

The 3D representation is preferred to the 2D in two cases:

- if 3D and 2D contours in the file have a different number of curves,
- if the 2D curve is a Circular Arc (entity type 100) starting and ending in the same point and the 3D one is not.

In any other case, the 2D representation is preferred to the 3D.

If either a 3D or a 2D contour is absent in the file or cannot be translated, then it is re-computed from another contour. If the translation of both 2D and 3D contours fails, the whole curve (type 141 or 142) is not translated. If this curve is used for trimming a face, the face will be translated without this trimming and will have natural restrictions.

Read this parameter with:

```
Standard_Integer ic =  Interface_Static::IVal("read.surfacecurve.mode");
```

Modify this value with:

```
if  (!Interface_Static::SetIVal ("read.surfacecurve.mode",3))
.. error ..;
```

Default value is Default (0).

**read.encoderegularity.angle**

This parameter is used within the *BRepLib::EncodeRegularity()* function which is called for a shape read from an IGES or a STEP file at the end of translation process. This function sets the regularity flag of an edge in a shell when this edge is shared by two faces. This flag shows the continuity, which these two faces are connected with at that edge.

Read this parameter with:

```
Standard_Real era =  Interface_Static::RVal("read.encoderegularity.angle");
```

Modify this parameter with:

```
if  (!Interface_Static::SetRVal ("read.encoderegularity.angle",0.1))
.. error ..;
```

Default value is 0.01.

**read.iges.bspline.approxd1.mode**

This parameter is obsolete (it is rarely used in real practice). If set to True, it affects the translation of bspline curves of degree 1 from IGES: these curves (which geometrically are polylines) are split by duplicated points, and the translator attempts to convert each of the obtained parts to a bspline of a higher continuity.

Read this parameter with:

```
Standard_Real bam =   Interface_Static::CVal("read.iges.bspline.approxd1.mode");
```

Modify this parameter with:

```
if  (!Interface_Static::SetRVal ("read.encoderegularity.angle","On"))
.. error ..;
```

Default value is Off.

**read.iges.resource.name and read.iges.sequence**

These two parameters define the name of the resource file and the name of the sequence of operators (defined in that file) for Shape Processing, which is automatically performed by the IGES translator. The Shape Processing is a user-configurable step, which is performed after the translation and consists in application of a set of operators to a resulting shape. This is a very powerful tool allowing to customize the shape and to adapt it to the needs of a receiving application. By default, the sequence consists of a single operator *ShapeFix* that calls Shape Healing from the IGES translator.

Please find an example of the resource file for IGES (which defines parameters corresponding to the sequence applied by default, i.e. if the resource file is not found) in the Open CASCADE Technology installation, by the path *CASROOT%/src/XSTEPResource/IGES* .

In order for the IGES translator to use that file, you have to define the environment variable *CSF_IGESDefaults*, which should point to the directory where the resource file resides. Note that if you change parameter *read.↩ iges.resource.name*, you should change the name of the resource file and the name of the environment variable correspondingly. The variable should contain a path to the resource file.

Default values:

- read.iges.resource.name – IGES,

- read.iges.sequence – FromIGES.

**xstep.cascade.unit**

This parameter defines units to which a shape should be converted when translated from IGES or STEP to C↩ ASCADE. Normally it is MM; only those applications that work internally in units other than MM should use this parameter.

Default value is MM.

### 2.3.4   Selecting entities

A list of entities can be formed by invoking the method *IGESControl_Reader::GiveList*.

```
Handle(TColStd_HSequenceOfTransient)  list = reader.GiveList();
```

Several predefined operators can be used to select a list of entities of a specific type. To make a selection, you use the method *IGESControl_Reader::GiveList* with the selection type in quotation marks as an argument. You can also make cumulative selections. For example, you would use the following syntax:

1. Requesting the faces in the file:

```
faces =   Reader.GiveList("iges-faces");
```

2. Requesting the visible roots in the file:

```
visibles =   Reader.GiveList(iges-visible-roots);
```

3. Requesting the visible faces:

```
visfac =   Reader.GiveList(iges-visible-roots,faces);
```

Using a signature, you can define a selection dynamically, filtering the string by means of a criterion. When you request a selection using the method GiveList, you can give either a predefined selection or a selection by signature. You make your selection by signature using the predefined signature followed by your criterion in parentheses as shown in the example below. The syntaxes given are equivalent to each other.

```
faces =   Reader.GiveList("xst-type(SurfaceOfRevolution)");
faces =   Reader.GiveList("iges-type(120)");
```

You can also look for:

- values returned by your signature which match your criterion exactly

  ```
  faces =   Reader.GiveList("xst-type(=SurfaceOfRevolution)");
  ```

- values returned by your signature which do not contain your criterion

  ```
  faces = Reader.GiveList("xst-type(!SurfaceOfRevolution)");
  ```

- values returned by your signature which do not exactly match your criterion.

  ```
  faces =   Reader.GiveList("xst-type(!=SurfaceOfRevolution)");
  ```

**List of predefined operators that can be used:**

- *xst-model-all* – selects all entities.
- *xst-model-roots* – selects all roots.
- *xst-transferrable-all* – selects all translatable entities.
- *xst-transferrable-roots* – selects all translatable roots (default).
- *xst-sharing + <selection>* – selects all entities sharing at least one entity selected by <selection>.
- *xst-shared + <selection>* – selects all entities shared by at least one entity selected by <selection>.
- *iges-visible-roots* – selects all visible roots, whether translatable or not.
- *iges-visible-transf-roots* – selects all visible and translatable roots.
- *iges-blanked-roots* – selects all blank roots, whether translatable or not.
- *iges-blanked-transf-roots* – selects all blank and translatable roots.
- *iges-status-independant* – selects entities whose IGES Subordinate Status = 0.
- *iges-bypass-group* – selects all root entities. If a root entity is a group (402/7 or 402/9), the entities in the group are selected.
- *iges-bypass-subfigure* – selects all root entities. If a root entity is a subfigure definition (308), the entities in the subfigure definition are selected.
- *iges-bypass-group-subfigure* – selects all root entities. If a root entity is a group (402/7 or 402/9) or a subfigure definition (308), the entities in the group and in the subfigure definition are selected.
- *iges-curves-3d* – selects 3D curves, whether they are roots or not (e.g. a 3D curve on a surface).
- *iges-basic-geom* – selects 3D curves and untrimmed surfaces.
- *iges-faces* – selects face-supporting surfaces (trimmed or not).
- *iges-surfaces* – selects surfaces not supporting faces (i.e. with natural bounds).
- *iges-basic-curves-3d* – selects the same entities as iges-curves-3d. Composite Curves are broken down into their components and the components are selected.

### 2.3.5 Performing the IGES file translation

Perform translation according to what you want to translate:

1. Translate an entity identified by its rank with:

   ```
   Standard_Boolean ok =  reader.Transfer (rank);
   ```

2. Translate an entity identified by its handle with:

   ```
   Standard_Boolean ok =  reader.TransferEntity (ent);
   ```

3. Translate a list of entities in one operation with:

   ```
   Standard_Integer nbtrans =  reader.TransferList (list);
   reader.IsDone();
   ```

   where *nbtrans* returns the number of items in the list that produced a shape and *reader.IsDone()* indicates whether at least one entity was translated.

4. Translate a list of entities, entity by entity:

   ```
   Standard_Integer i,nb =  list-Length();
   for (i = 1; i  <= nb; i ++) {
       Handle(Standard_Transient) ent = list-Value(i);
       Standard_Boolean OK = reader.TransferEntity (ent);
   }
   ```

5. Translate the whole file (all entities or only visible entities) with:

   ```
   Standard_Boolean  onlyvisible = Standard_True or Standard_False;
   reader.TransferRoots(onlyvisible)
   ```

### 2.3.6 Getting the translation results

Each successful translation operation outputs one shape. A series of translations gives a series of shapes. Each time you invoke *TransferEntity, Transfer* or *Transferlist*, their results are accumulated and NbShapes increases. You can clear the results (Clear function) between two translation operations, if you do not do this, the results from the next translation will be added to the accumulation. *TransferRoots* operations automatically clear all existing results before they start.

```
Standard_Integer nbs =  reader.NbShapes();
```

returns the number of shapes recorded in the result.

```
TopoDS_Shape shape =  reader.Shape(num);,
```

returns the result *num*, where *num* is an integer between 1 and *NbShapes*.

```
TopoDS_Shape shape =  reader.Shape();
```

returns the first result in a translation operation.

```
TopoDS_Shape shape =  reader.OneShape();
```

returns all results in a single shape which is:

- a null shape if there are no results,

- in case of a single result, a shape that is specific to that result,

- a compound that lists the results if there are several results.

```
reader.Clear();
```

erases the existing results.

```
reader.PrintTransferInfo (failsonly, mode);
```

displays the messages that appeared during the last invocation of *Transfer* or *TransferRoots*.

If *failsonly* is *IFSelect_FailOnly*, only fail messages will be output, if it is *IFSelect_FailAndWarn*, all messages will be output. Parameter "mode" can have *IFSelect_xxx* values where *xxx* can be:

- *GeneralCount* – gives general statistics on the transfer (number of translated IGES entities, number of fails and warnings, etc)

- *CountByItem* – gives the number of IGES entities with their types per message.

- *ListByItem* – gives the number of IGES entities with their type and DE numbers per message.

- *ResultCount* – gives the number of resulting OCCT shapes per type.

- *Mapping* – gives mapping between roots of the IGES file and the resulting OCCT shape per IGES and OCCT type.

## 2.4 Mapping of IGES entities to Open CASCADE Technology shapes

*NOTE* that IGES entity types that are not given in the following tables are not translatable.

### 2.4.1 Points

| IGES entity type | CASCADE shape | Comments |
|---|---|---|
| 116: Point | TopoDS_Vertex | |

### 2.4.2 Curves

Curves, which form the 2D of face boundaries, are translated as *Geom2D_Curves* (Geom2D circles, etc.).

| IGES entity type | CASCADE shape | Comments |
|---|---|---|
| 100: Circular Arc | TopoDS_Edge | The geometrical support is a *Geom_Circle* or a *Geom_TrimmedCurve* (if the arc is not closed). |
| 102: Composite Curve | TopoDS_Wire | The resulting shape is always a *TopoDS↵_Wire* that is built from a set of *TopoDS_↵Edges*. Each *TopoDS_Edge* is connected to the preceding and to the following edge by a common *TopoDS_Vertex*. |
| 104: Conic Arc | TopoDS_Edge | The geometric support depends on whether the IGES entity's form is 0 (*Geom_Circle*), 1 (*Geom_Ellipse*), 2 (*Geom_Hyperbola*), or 3 (*Geom_Parabola*). A *Geom_TrimmedCurve* is output if the arc is not closed. |

| IGES entity type | CASCADE shape | Comments |
|---|---|---|
| 106: Copious Data | TopoDS_Edge or TopoDS_Wire | IGES entity Copious Data (type 106, forms 1-3) is translated just as the IGES entities Linear Path (106/11-13) and the Simple Closed Planar Curve (106/63). Vectors applying to forms other than 11,12 or 63 are ignored. The *Geom_BSplineCurve* (geometrical support) has C0 continuity. If the Copious Data has vectors (DataType = 3) they will be ignored. |
| 110: Line | TopoDS_Edge | The supporting curve is a *Geom_Trimmed↩ Curve* whose basis curve is a *Geom_Line*. |
| 112: Parametric Spline Curve | TopoDS_Edge or TopoDS_Wire | The geometric support is a Geom_Bspline↩ Curve. |
| 126: BSpline Curve | TopoDS_Edge or TopoDS_Wire | |
| 130: Offset Curve | TopoDS_Edge or TopoDS_Wire | The resulting shape is a *TopoDS_Edge* or a *TopoDS_Wire* (depending on the translation of the basis curve) whose geometrical support is a *Geom_OffsetCurve* built from a basis *Geom_Curve*. Limitation: The IGES Offset Type value must be 1. |
| 141: Boundary | TopoDS_Wire | Same behavior as for the Curve On Surface (see below). The translation of a non-referenced Boundary IGES entity in a *BoundedSurface* IGES entity outputs a *TopoDS_Edge* or a *TopoDS_Wire* with a *Geom_Curve*. |
| 142: Curve On Surface | TopoDS_Wire | Each *TopoDS_Edge* is defined by a 3D curve and by a 2D curve that references the surface. |

The type of OCCT shapes (either *TopoDS_Edges* or *TopoDS_Wires*) that result from the translation of IGES entities 106, 112 and 126 depends on the continuity of the curve in the IGES file and the value of the *read.iges.bspline.↩ continuity* translation parameter.

### 2.4.3   Surfaces

Translation of a surface outputs either a *TopoDS_Face* or a *TopoDS_Shell*. If a *TopoDS_Face* is output, its geometrical support is a *Geom_Surface* and its outer and inner boundaries (if it has any) are *TopoDS_Wires*.

| IGES entity type | CASCADE shape | Comments |
|---|---|---|
| 108: Plane | TopoDS_Face | The geometrical support for the *Topo↩ DS_Face* is a *Geom_Plane* and the orientation of its *TopoDS_Wire* depends on whether it is an outer *TopoDS_Wire* or whether it is a hole. |
| 114: Parametric Spline Surface | TopoDS_Face | The geometrical support of a *TopoDS_↩ Face* is a *Geom_BSplineSurface*. |
| 118: Ruled Surface | TopoDS_Face or TopoDS_Shell | The translation of a Ruled Surface outputs a *TopoDS_Face* if the profile curves become *TopoDS_Edges*, or a *TopoDS_Shell* if the profile curves become *TopoDS_↩ Wires*. Limitation: This translation cannot be completed when these two *TopoDS_↩ Wires* are oriented in different directions. |

| IGES entity type | CASCADE shape | Comments |
|---|---|---|
| 120: Surface Of Revolution | TopoDS_Face or TopoDS_Shell | The translation of a Surface Of Revolution outputs: a *TopoDS_Face* if the generatrix becomes a *TopoDS_Edge*, a *TopoDS_Shell* if the generatrix becomes a *TopoDS_Wire*. The geometrical support may be: *Geom_CylindricalSurface, Geom_ConicalSurface, Geom_SphericalSurface, Geom_ToroidalSurface* or a *Geom_SurfaceOfRevolution* depending on the result of the CASCADE computation (based on the generatrix type). |
| 122: Tabulated Cylinder | TopoDS_Face or TopoDS_Shell | The translation outputs a *TopoDS_Face* if the base becomes a *TopoDS_Edge* or a *TopoDS_Shell* if the base becomes a *TopoDS_Wire*. The geometrical support may be *Geom_Plane, Geom_Cylindrical Surface* or a *Geom_SurfaceOfLinearExtrusion* depending on the result of the CASCADE computation (based on the generatrix type). The *Geom_Surface* geometrical support is limited according to the generatrix. |
| 128: BSpline Surface | TopoDS_Face | The geometrical support of the *TopoDS_Face* is a *Geom_BsplineSurface*. |
| 140: Offset Surface | TopoDS_Face | The translation of an Offset Surface outputs a *TopoDS_Face* whose geometrical support is a *Geom_OffsetSurface*. Limitations: For OCCT algorithms, the original surface must be C1-continuous so that the *Geom_OffsetSurface* can be created. If the basis surface is not C1-continuous, its translation outputs a *TopoDS_Shell* and only the first *TopoDS_Face* in the *TopoDS_Shell* is offset. |
| 143: Bounded Surface | TopoDS_Face or TopoDS_Shell | If the basis surface outputs a *TopoDS_Shell* (that has more than one *TopoDS_Face*), the IGES boundaries are not translated. Limitations: If the bounding curves define holes, natural bounds are not created. If the orientation of the contours is wrong, it is not corrected. |
| 144: Trimmed Surface | TopoDS_Face or TopoDS_Shell | For the needs of interface processing, the basis surface must be a face. Shells are only processed if they are single-face. The contours (wires that are correctly oriented according to the definition of the IGES 142: Curve On Surface entity) are added to the face that is already created. If the orientation of the contours is wrong, it is corrected. |
| 190: Plane Surface | TopoDS_Face | This type of IGES entity can only be used in BRep entities in place of an IGES 108 type entity. The geometrical support of the face is a *Geom_Plane*. |

### 2.4.4 Boundary Representation Solid Entities

| IGES entity type | CASCADE shape | Comments |
|---|---|---|
| 186: ManifoldSolid | TopoDS_Solid | |
| 514: Shell | TopoDS_Shell | |
| 510: Face | TopoDS_Face | This is the lowest IGES entity in the BRep structure that can be specified as a starting point for translation. |
| 508: Loop | TopoDS_Wire | |
| 504: Edge List | | |
| 502: Vertex List | | |

### 2.4.5 Structure Entities

| IGES entity type | CASCADE shape | Comments |
|---|---|---|
| 402/1: Associativity Instance: Group with back pointers | TopoDS_Compound | |
| 402/7: Associativity Instance: Group without back pointers | TopoDS_Compound | |
| 402/9: Associativity Instance: Single Parent | TopoDS_Face | The translation of a *SingleParent* entity is only performed for 402 form 9 with entities 108/1 and 108/-1. The geometrical support for the *TopoDS_Face* is a *Geom_Plane* with boundaries: the parent plane defines the outer boundary; the child planes define the inner boundaries. |

### 2.4.6 Subfigures

| IGES entity type | CASCADE shape | Comments |
|---|---|---|
| 308: Subfigure Definition | TopoDS_Compound | This IGES entity is only translated when there are no Singular Subfigure Instance entities. |
| 408: Singular Subfigure Instance | TopoDS_Compound | This shape has the Subfigure Definition Compound as its origin and is positioned in space by its translation vector and its scale factor. |

### 2.4.7 Transformation Matrix

| IGES entity type | CASCADE shape | Comments |
|---|---|---|
| 124: Transformation Matrix | Geom_Transformation | This entity is never translated alone. It must be included in the definition of another entity. |

## 2.5 Messages

Messages are displayed concerning the normal functioning of the processor (transfer, loading, etc.). You must declare an include file:

```
#include \<Interface_DT.hxx\>
```

You have the choice of the following options for messages:

```
IDT_SetLevel (level);
```

level modifies the level of messages:

- 0: no messages

- 1: raise and fail messages are displayed, as are messages concerning file access,

- 2: warnings are also displayed.

  ```
  IDT_SetFile ("tracefile.log");
  ```

  prints the messages in a file,

  ```
  IDT_SetStandard();
  ```

  restores screen output.

## 2.6 Tolerance management

### 2.6.1 Values used for tolerances during reading IGES

During the transfer of IGES to Open CASCADE Technology several parameters are used as tolerances and precisions for different algorithms. Some of them are computed from other using specific functions.

**3D (spatial) tolerances**

- Package method *Precision::Confusion* equal to $10^{-7}$ is used as a minimal distance between points, which are considered distinct.

- Resolution in the IGES file is defined in the Global section of an IGES file. It is used as a fundamental value of precision during the transfer.

- User-defined variable *read.precision.val* can be used instead of resolution from the file when parameter *read.precision.mode* is set to 1 ("User").

- Field *EpsGeom* of the class *IGESToBRep_CurveAndSurface* is a basic precision for translating an IGES object. It is set for each object of class *IGESToBRep_CurveAndSurface* and its derived classes. It is initialized for the root of transfer either by value of resolution from the file or by value of *read.precision.val*, depending on the value of *read.precision.mode* parameter. It is returned by call to method *IGESToBRep_CurvAnd↩Surface::GetEpsGeom*. As this value belongs to measurement units of the IGES file, it is usually multiplied by the coefficient *UnitFactor* (returned by method *IGESToBRep_CurvAndSurface::GetUnitFactor*) to convert it to Open CASCADE Technology units.

- Field *MaxTol* of the class *IGESToBRep_CurveAndSurface* is used as the maximum tolerance for some algorithms. Currently, it is computed as the maximum between 1 and *GetEpsGeom* ∗ *GetUnitFactor*. This field is returned by method *IGESToBRep_CurvAndSurface::GetMaxTol*.

**2D (parametric) tolerances**

- Package method *Precision::PConfusion* equal to *0.01∗Precision::Confusion*, i.e. $10^{-9}$. It is used to compare parametric bounds of curves.

- Field *EpsCoeff* of the class *IGESToBRep_CurveAndSurface* is a parametric precision for translating an IGES object. It is set for each object of class *IGESToBRep_CurveAndSurface* and its derived classes. Currently, it always has its default value $10^{-6}$. It is returned by call to method *IGESToBRep_CurvAndSurface::GetEps↩Coeff*. This value is used for translating 2d objects (for instance, parametric curves).

- Methods *UResolution(tolerance3d)* and *VResolution(tolerance3d)* of the class *GeomAdaptor_Surface* or *B↩RepAdaptor_Surface* return tolerance in parametric space of a surface computed from 3D tolerance. When one tolerance value is to be used for both U and V parametric directions, the maximum or the minimum value of *UResolution* and *VResolution* is used.

- Methods *Resolution(tolerance3d)* of the class *GeomAdaptor_Curve* or *BRepAdaptor_Curve* return tolerance in the parametric space of a curve computed from 3d tolerance.

**Zero-dimensional tolerances**

- Field *Epsilon* of the class *IGESToBRep_CurveAndSurface* is set for each object of class *IGESToBRep_↩CurveAndSurface* and returned by call to method *GetEpsilon*. It is used in comparing angles and converting transformation matrices. In most cases, it is reset to a fixed value ($10^{-5}$ - $10^{-3}$) right before use. The default value is $10^{-4}$.

### 2.6.2 Initial setting of tolerances in translating objects

Transfer starts from one entity treated as a root (either the actual root in the IGES file or an entity selected by the user). The function which performs the transfer (that is *IGESToBRep_Actor::Transfer* or *IGESToBRep_Reader::↩Transfer*) creates an object of the type *IGESToBRep_CurveAndSurface*, which is intended for translating geometry.

This object contains three tolerances: *Epsilon, EpsGeom* and *EpsCoeff*.

Parameter *Epsilon* is set by default to value $10^{-4}$. In most cases when it is used in the package *IGESToBRep*, it is reset to a fixed value, either $10^{-5}$ or $10^{-4}$ or $10^{-3}$. It is used as precision when comparing angles and transformation matrices and does not have influence on the tolerance of the resulting shape.

Parameter *EpsGeom* is set right after creating a *IGESToBRep_CurveAndSurface* object to the value of resolution, taken either from the Global section of an IGES file, or from the *XSTEP.readprecision.val* parameter, depending on the value of *XSTEP.readprecision.mode*.

Parameter *EpsCoeff* is set by default to $10^{-6}$ and is not changed.

During the transfer of a shape, new objects of type *IGESToBRep_CurveAndSurface* are created for translating subshapes. All of them have the same tolerances as the root object.

### 2.6.3 Transfer process

**Translating into Geometry**

Geometrical entities are translated by classes *IGESToBRep_BasicCurve* and *IGESToBRep_BasicSurface*. Methods of these classes convert curves and surfaces of an IGES file to Open CASCADE Technology geometry objects↩: *Geom_Curve, Geom_Surface,* and *Geom_Transformation*.

Since these objects are not BRep objects, they do not have tolerances. Hence, tolerance parameters are used in these classes only as precisions: to detect specific cases (e.g., to distinguish a circle, an ellipse, a parabola and a hyperbola) and to detect bad cases (such as coincident points).

Use of precision parameters is reflected in the following classes:

- *IGESToBRep_BasicCurve* – all parameters and points are compared with precision *EpsGeom*. All transformations (except *IGESToBRep_BasicCurve::TransferTransformation*) are fulfilled with precision *Epsilon* which is set to $10^{-3}$ (in the *IGESToBRep_BasicCurve::TransferTransformation* the value $10^{-5}$ is used).

- *IGESToBRep_BasicCurve::TransferBSplineCurve* – all weights of *BSplineCurve* are assumed to be more than *Precision::PConfusion* (else the curve is not translated).

- *IGESToBRep_BasicSurface* – all parameters and points are compared with precision *EpsGeom*. All transformations are fulfilled with precision *Epsilon*, which is set to $10^{-3}$.

- *IGESToBRep_BasicSurface::TransferBSplineSurface* – all weights of *BSplineSurface* are assumed to be more than *Precision::PConfusion* (else the surface is not translated).

**Translating into Topology**

IGES entities represented as topological shapes and geometrical objects are translated into OCCT shapes by use of the classes *IGESToBRep_TopoCurve, IGESToBRep_TopoSurface, IGESToBRep_BRepEntity* and *ShapeFix_Wire*.

Class *IGESToBRep_BRepEntity* is intended for transferring BRep entities (IGES version is 5.1 or greater) while the two former are used for translating geometry and topology defined in IGES versions prior to 5.1. Methods

from *IGESToBRep_BRepEntity* call methods from *IGESToBRep_TopoCurve* and *IGESToBRep_TopoSurface*, while those call methods from *IGESToBRep_BasicCurve* and *IGESToBRep_BasicSurface* to translate IGES geometry into OCCT geometry.

Although the IGES file contains only one parameter for tolerance in the Global Section, OCCT shapes are produced with different tolerances. As a rule, updating the tolerance is fulfilled according to local distances between shapes (distance between vertices of adjacent edges, deviation of edge's 3D curve and its parametric curve and so on) and may be less or greater than precision in the file.

The following classes show what default tolerances are used when creating shapes and how they are updated during transfer.

**Class IGESToBRep_TopoCurve**

All methods are in charge of transferring curves from IGES curve entities *(TransferCompositeCurve, Transfer2d↩ CompositeCurve, TransferCurveOnFace, TransferBoundaryOnFace, TransferOffsetCurve, TransferTopoBasic↩ Curve)* if an entity has transformation call to *IGESData_ToolLocation::ConvertLocation* with *Epsilon* value set to $10^{-4}$.

- *IGESToBRep_TopoCurve::TransferPoint* – vertex is constructed from a Point entity with tolerance *Eps↩ Geom∗UnitFactor*.

- *IGESToBRep_TopoCurve::Transfer2dPoint* – vertex is constructed from a Point entity with tolerance *EpsCoeff*.

- *IGESToBRep_TopoCurve::TransferCompositeCurveGeneral* – obtains shapes (edges or wires) from other methods and adds them into the resulting wire. Two adjacent edges of the wire can be connected with tolerance up to *MaxTol*.

- *IGESToBRep_TopoCurve::TransferCurveOnFace* and *IGESToBRep_TopoCurve::TransferBoundaryOnFace* build a wire from 3D and 2D representations of a curve on surface. Edges and vertices of the wire cannot have tolerance larger than *MaxTol*. The value *EpsGeom∗UnitFactor* is passed into *ShapeFix_Wire::Set↩ Precision* and *MaxTol* is passed into *ShapeFix_Wire::MaxTolerance*. To find out how these parameters affect the resulting tolerance changes, please, refer to class *ShapeFix_Wire*.

- *IGESToBRep_TopoCurve::TransferTopoBasicCurve* and *IGESToBRep_TopoCurve::Transfer2dTopoBasic↩ Curve* – the boundary vertices of an edge (or a wire if a curve was of C0 continuity) translated from a basis IGES curve *(BSplineCurve, CopiousData, Line,* etc.) are built with tolerance *EpsGeom∗UnitFactor*, the edge tolerance is *Precision::Confusion*. If a curve was divided into several edges, the common vertices of such adjacent edges have tolerance *Precision::Confusion*.

**Class IGESToBRep_TopoSurface**

All faces created by this class have tolerance *Precision::Confusion*.

**Class IGESToBRep_BRepEntity**

- *IGESToBRep_BRepEntity::TransferVertex* – the vertices from the *VertexList* entity are constructed with tolerance *EpsGeom∗UnitFactor*.

- *IGESToBRep_BRepEntity::TransferEdge* – the edges from the *EdgeList* entity are constructed with tolerance *Precision::Confusion*.

- *IGESToBRep_BRepEntity::TransferLoop* – this function works like *IGESToBRep_TopoCurve::TransferCurve↩ OnFace* and *IGESToBRep_TopoCurve::TransferBoundaryOnFace*.

- *IGESToBRep_BRepEntity::TransferFace* – the face from the *Face* IGES entity is constructed with tolerance *Precision::Confusion*.

**Shape Healing classes**

After performing a simple mapping, shape-healing algorithms are called (class *ShapeFix_Shape*) by *IGESToBRep↩ _Actor::Transfer()*. Shape-healing algorithm performs the correction of the resulting OCCT shape. Class *ShapeFix↩ _Wire* can increase the tolerance of a shape. This class is used in *IGESToBRep_BRepEntity::TransferLoop*, *IGES↩ ToBRep_TopoCurve::TransferBoundaryOnFace* and *IGESToBRep_TopoCurve::TransferCurveOnFace* for correcting

a wire. The maximum possible tolerance applied to the edges or vertices after invoking the methods of this class is *MaxTolerance* (set by method *ShapeFix_Wire::MaxTolerance()* ).

## 2.7 Code architecture

The following diagram illustrates the structure of calls in reading IGES. The highlighted classes produce OCCT geometry.
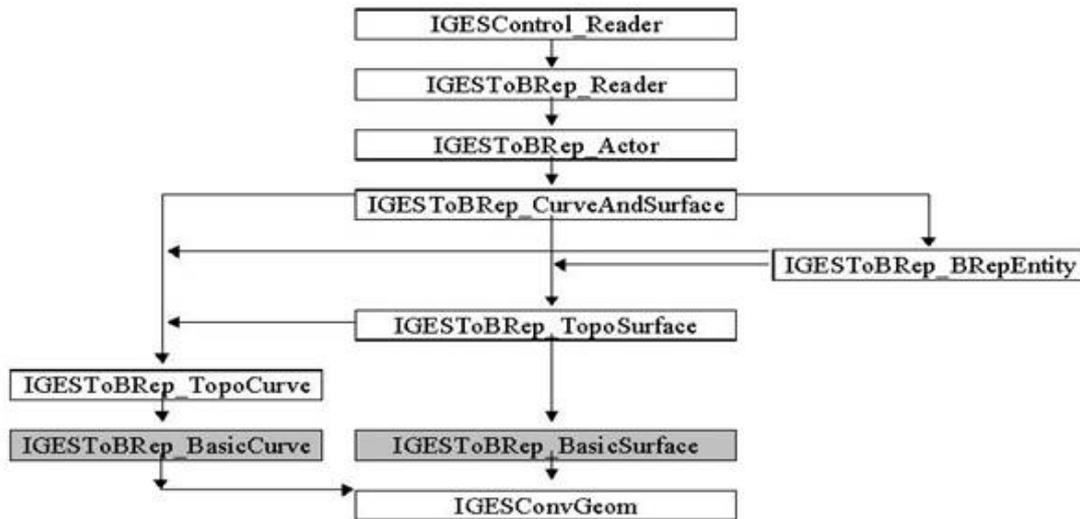


Figure 1: The structure of calls in reading IGES

## 2.8 Example

```
#include "IGESControl_Reader.hxx"
#include "TColStd_HSequenceOfTransient.hxx"
#include "TopoDS_Shape.hxx"
{
IGESControl_Reader myIgesReader;
Standard_Integer nIgesFaces,nTransFaces;

myIgesReader.ReadFile ("MyFile.igs");
//loads file MyFile.igs

Handle(TColStd_HSequenceOfTransient) myList =  myIgesReader.GiveList("iges-faces");
//selects all IGES faces in the file and puts them into a list  called //MyList,

nIgesFaces = myList-Length();
nTransFaces = myIgesReader.TransferList(myList);
//translates MyList,

cout<<"IGES Faces: "<<nIgesFaces<<"   Transferred:"<<nTransFaces<<endl;
TopoDS_Shape sh = myIgesReader.OneShape();
//and obtains the results in an OCCT shape.
}
```

# 3 Writing IGES

## 3.1 Procedure

You can translate OCCT shapes to IGES entities in the following steps:

1. Initialize the process.
2. Set the translation parameters,
3. Perform the model translation,
4. Write the output IGES file.

You can translate several shapes before writing a file. Each shape will be a root entity in the IGES model.

## 3.2 Domain covered

There are two families of OCCT objects that can be translated:

- geometrical,
- topological.

## 3.3 Description of the process

### 3.3.1 Initializing the process

Choose the unit and the mode you want to use to write the output file as follows:

- *IGESControl_Controller::Init* performs standard initialization. Returns False if an error occurred.
- *IGESControl_Writer writer* uses the default unit (millimeters) and the default write mode (Face).
- *IGESControl_Writer writer (UNIT)* uses the Face write mode and any of the units that are accepted by IGES.
- *IGESControl_Writer writer (UNIT,modecr)* uses the unit (accepted by IGES) and the write mode of your choice.
    - 0: Faces,
    - 1: BRep The result is an *IGESControl_Writer* object.

### 3.3.2 Setting the translation parameters

The following parameters are used for the OCCT-to-IGES translation.

- *write.iges.brep.mode:* allows choosing the write mode:
    - "Faces" (0): OCCT *TopoDS_Faces* will be translated into IGES 144 (Trimmed Surface) entities, no BRep entities will be written to the IGES file,
    - "BRep" (1): OCCT *TopoDS_Faces* will be translated into IGES 510 (Face) entities, the IGES file will contain BRep entities. Read this parameter with:

        ```
        Standard_Integer byvalue =  Interface_Static::IVal("write.iges.brep.mode");
        ```

        Modify this parameter with:

        ```
        Interface_Static::SetIVal  ("write.iges.brep.mode", 1);
        ```

Default value is "Faces" (0).

- *write.convertsurface.mode* when writing to IGES in the BRep mode, this parameter indicates whether elementary surfaces (cylindrical, conical, spherical, and toroidal) are converted into corresponding IGES 5.3 entities (if the value of a parameter value is On), or written as surfaces of revolution (by default).

- *write.iges.unit:* allows choosing the unit. The default unit for Open CASCADE Technology is "MM" (millimeter). You can choose to write a file into any unit accepted by IGES.

  – Read this parameter with *Standard_String byvalue = Interface_Static::CVal("write.iges.unit")*;
  – Modify this parameter with *Interface_Static::SetCVal ("write.iges.unit", "INCH");*

- *write.iges.header.autor:* gives the name of the author of the file. The default value is the system name of the user.

  – Read this parameter with *Standard_String byvalue = Interface_Static::CVal("write.iges.header.author")*;
  – Modify this value with *Interface_Static::SetCVal ("write.iges.header.author", "name")*;

- *write.iges.header.company:* gives the name of the sending company. The default value is "" (empty).

  – Read this parameter with *Standard_String byvalue = Interface_Static::CVal("write.iges.header.↩ company")*;
  – Modify this value with *Interface_Static::SetCVal ("write.iges.header.company", "Open CASCADE");*

- *write.iges.header.product:* gives the name of the sending product. The default value is "CAS.CADE IGES processor Vx.x", where *x.x* means the current version of Open CASCADE Technology.

  – Read this parameter with *Standard_String byvalue = Interface_Static::CVal("write.iges.header.↩ product")*;
  – Modify this value with *Interface_Static::SetCVal ("write.iges.header.product", "product name")*;

- *write.iges.header.receiver:* – gives the name of the receiving company. The default value is "" (empty).

  – Read this parameter with *Standard_String byvalue = Interface_Static::CVal("write.iges.header.↩ receiver")*;
  – Modify this value with *Interface_Static::SetCVal ("write.iges.header.receiver", "reciever name");*

- *write.precision.mode:* specifies the mode of writing the resolution value into the IGES file.

  – "Least" (-1): resolution value is set to the minimum tolerance of all edges and all vertices in an OCCT shape.
  – "Average" (0): resolution value is set to average between the average tolerance of all edges and the average tolerance of all vertices in an OCCT shape. This is the default value.
  – "Greatest" (1): resolution value is set to the maximum tolerance of all edges and all vertices in an OCCT shape.
  – "Session" (2): resolution value is that of the write.precision.val parameter.
  – Read this parameter with *Standard_Integer ic = Interface_Static::IVal("write.precision.mode")*;
  – Modify this parameter with *if (!Interface_Static::SetIVal("write.precision.mode",1)) .. error ..*

- *write.precision.val:* is the user precision value. This parameter gives the resolution value for an IGES file when the *write.precision.mode* parameter value is 1. It is equal to 0.0001 by default, but can take any real positive (non null) value.

Read this parameter with:

```
Standard_Real rp =  Interface_Static::RVal(;write.precision.val;);
```

Modify this parameter with:

```
if  (!Interface_Static::SetRVal(;write.precision.val;,0.01))
.. error ..
```

Default value is 0.0001.

**write.iges.resource.name**

and

**write.iges.sequence**

are the same as the corresponding read.iges.∗ parameters, please, see above.  Note that the default sequence for writing contains *DirectFaces* operator, which converts elementary surfaces based on left-hand axes (valid in CASCADE) to right-hand axes (which are valid only in IGES).

Default values :

```
write.iges.resource.name - IGES,
write.iges.sequence - ToIGES.
```

### 3.3.3  Performing the Open CASCADE Technology shape translation

You can perform the translation in one or several operations. Here is how you translate topological and geometrical objects:

```
Standard_Boolean ok =  writer.AddShape (TopoDS_Shape);
```

*ok* is True if translation was correctly performed and False if there was at least one entity that was not translated.

```
Standard_Boolean ok =  writer.AddGeom (geom);
```

where *geom* is *Handle(Geom_Curve)* or *Handle(Geom_Surface)*; *ok* is True if the translation was correctly performed and False if there was at least one entity whose geometry was not among the allowed types.

### 3.3.4  Writing the IGES file

Write the IGES file with:

```
Standard_Boolean ok =  writer.Write ("filename.igs");
```

to give the file name.

```
Standard_Boolean ok =  writer.Write (S);
```

where *S* is *Standard_OStream ok* is True if the operation was correctly performed and False if an error occurred (for instance, if the processor could not create the file).

## 3.4  Mapping Open CASCADE Technology shapes to IGES entities

Translated objects depend on the write mode that you chose. If you chose the Face mode, all of the shapes are translated, but the level of topological entities becomes lower (geometrical one).  If you chose the BRep mode, topological OCCT shapes become topological IGES entities.

### 3.4.1  Curves

| CASCADE shape | IGES entity type | Comments |
|---|---|---|
| Geom_BsplineCurve | 126: BSpline Curve | |
| Geom_BezierCurve | 126: BSpline Curve | |

| CASCADE shape | IGES entity type | Comments |
|---|---|---|
| Geom_TrimmedCurve | All types of translatable IGES curves | The type of entity output depends on the type of the basis curve. If the curve is not trimmed, limiting points will be defined by the CASCADE RealLast value. |
| Geom_Circle | 100: Circular Arc or 126: BSpline Curve | A BSpline Curve is output if the *Geom_↩ Circle* is closed |
| Geom_Ellipse | 104: Conic Arc or 126: BSpline Curve | A Conic Arc has Form 1. A BSpline Curve is output if the *Geom_Ellipse* is closed. |
| Geom_Hyperbola | 104: Conic Arc | Form 2 |
| Geom_Parabola | 104: Conic Arc | Form 3 |
| Geom_Line | 110: Line | |
| Geom_OffsetCurve | 130: Offset Curve | |

### 3.4.2   Surfaces

| CASCADE shapes | IGES entity type | Comments |
|---|---|---|
| Geom_BSplineSurface | 128: BSpline Surface | |
| Geom_BezierSurface | 128: BSpline Surface | |
| Geom_RectangularTrimmed↩ Surface | All types of translatable IGES surfaces. | The type of entity output depends on the type of the basis surface. If the surface is not trimmed and has infinite edges/sides, the coordinates of the sides in IGES will be limited to the CASCADE *RealLast* value. |
| Geom_Plane | 128: BSpline Surface or 190: Plane Surface | A BSpline Surface (of degree 1 in U and V) is output if you are working in the face mode. A Plane Surface is output if you are working in the BRep mode. |
| Geom_CylindricalSurface | 120: Surface Of Revolution | |
| Geom_ConicalSurface | 120: Surface Of Revolution | |
| Geom_SphericalSurface | 120: Surface Of Revolution | |
| Geom_ToroidalSurface | 120: Surface Of Revolution | |
| Geom_SurfaceOfLinearExtrusion | 122: Tabulated Cylinder | |
| Geom_SurfaceOfRevolution | 120: Surface Of Revolution | |
| Geom_OffsetSurface | 140: Offset Surface | |

### 3.4.3   Topological entities -- Translation in Face mode

| CASCADE shapes | IGES entity type | Comments |
|---|---|---|
| Single TopoDS_Vertex | 116: 3D Point | |
| TopoDS_Vertex in a TopoDS_Edge | No equivalent | Not transferred. |
| TopoDS_Edge | All types of translatable IGES curves | The output IGES curve will be the one that corresponds to the Open CASCADE Technology definition. |
| Single TopoDS_Wire | 102: Composite Curve | Each *TopoDS_Edge* in the *Topo↩ DS_Wire* results in a curve. |

| CASCADE shapes | IGES entity type | Comments |
|---|---|---|
| TopoDS_Wire in a TopoDS_Face | 142: Curve On Surface | Both curves (3D and pcurve) are transferred if they are defined and result in a simple curve or a composite curve depending on whether there is one or more edges in the wire.Note: if the basis surface is a plane (108), only the 3D curve is used. |
| TopoDS_Face | 144: Trimmed Surface | |
| TopoDS_Shell | 402: Form 1 Group or no equivalent | Group is created only if *TopoD↩ S_Shell* contains more than one *TopoDS_Face*. The IGES group contains Trimmed Surfaces. |
| TopoDS_Solid | 402: Form 1 Group or no equivalent | Group is created only if *TopoD↩ S_Solid* contains more than one *TopoDS_Shell*. One IGES entity is created per *TopoDS_Shell*. |
| TopoDS_CompSolid | 402: Form 1 Group or no equivalent | Group is created only if *TopoDS_↩ CompSolid* contains more than one *TopoDS_Solid*. One IGES entity is created per *TopoDS_Solid*. |
| TopoDS_Compound | 402: Form 1 Group or no equivalent | Group is created only if *TopoD↩ S_Compound* contains more than one item. One IGES entity is created per *TopoDS_Shape* in the *TopoDS_Compound*. If *TopoDS↩ _Compound* is nested into another *TopoDS_Compound*, it is not mapped. |

### 3.4.4   Topological entities -- Translation in BRep mode

| CASCADE shapes | IGES entity type | Comments |
|---|---|---|
| Single TopoDS_Vertex | No equivalent | Not transferred. |
| TopoDS_Vertex in a TopoDS_Edge | One item in a 502: *VertexList* | |
| TopoDS_Edge | No equivalent | Not transferred as such. This entity serves as a part of a Loop entity. |
| TopoDS_Edge in a TopoDS_Wire | One item in a 504: EdgeList | |
| TopoDS_Wire | 508: Loop | |
| TopoDS_Face | 510: Face | If the geometrical support of the face is a plane, it will be translated as a 190 entity *PlaneSurface*. |
| TopoDS_Shell | 514: Shell | |
| TopoDS_Solid | 186: Manifold Solid | |
| TopoDS_CompSolid | 402 Form1 Group or no equivalent | Group is created only if *TopoDS_↩ Compound* contains more than one item. One IGES Manifold Solid is created for each *TopoDS_Solid* in the *TopoDS_CompSolid*. |

| CASCADE shapes | IGES entity type | Comments |
|---|---|---|
| TopoDS_Compound | 402 Form1 Group or no equivalent | Group is created only if *TopoD↩S_Compound* contains more than one item.   One IGES entity is created per *TopoDS_Shape* in the *TopoDS_Compound*.   If *TopoD↩S_Compound* is nested into another *TopoDS_Compound* it is not mapped. |

## 3.5   Tolerance management

### 3.5.1   Setting resolution in an IGES file

There are several possibilities to set resolution in an IGES file. They are controlled by write.precision.mode parameter; the dependence between the value of this parameter and the set resolution is described in paragraph Setting the translation parameters.

If the value of parameter *write.precision.mode* is -1, 0 or 1, resolution is computed from tolerances of sub-shapes inside the shape to be translated. In this computation, only tolerances of *TopoDS_Edges* and *TopoDS_Vertices* participate since they reflect the accuracy of the shape. *TopoDS_Faces* are ignored in computations since their tolerances may have influence on resulting computed resolution while IGES resolution mainly concerns points and curves but not surfaces.

## 3.6   Code architecture

### 3.6.1   Graph of calls

The following diagram illustrates the class structure in writing IGES. The highlighted classes are intended to translate geometry.
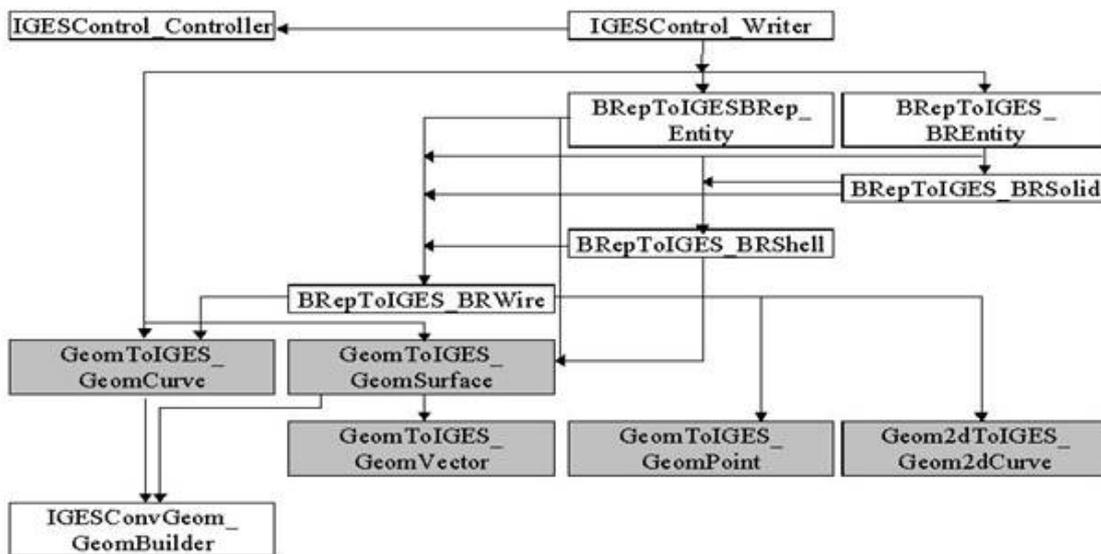


Figure 2: The class structure in writing IGES

## 3.7 Example

```c++
{c++}
#include <IGESControl_Controller.hxx>
#include <IGESControl_Writer.hxx>
#include <TopoDS_Shape.hxx>
Standard_Integer main()
{
  IGESControl_Controller::Init();
  IGESControl_Writer ICW (;MM;, 0);
  //creates a writer object for writing in Face mode with  millimeters
  TopoDS_Shape sh;
  ICW.AddShape (sh);
  //adds shape sh to IGES model
  ICW.ComputeModel();
  Standard_Boolean OK = ICW.Write (;MyFile.igs;);
  //writes a model to the file MyFile.igs
}
```

# 4   Using XSTEPDRAW

XSTEPDRAW UL is intended for creating executables for testing XSTEP interfaces interactively in the DRAW environment. It provides an additional set of DRAW commands specific for the data exchange tasks, which allow loading and writing data files and analysis of resulting data structures and shapes.

In the description of commands, square brackets ([]) are used to indicate optional parameters. Parameters given in the angle brackets (<>) and sharps (#) are to be substituted by an appropriate value. When several exclusive variants are possible, vertical dash (|) is used.

## 4.1   Setting interface parameters

A set of parameters for importing and exporting IGES files is defined in the XSTEP resource file. In XSTEPDRAW, these parameters can be viewed or changed using command

```
Draw> param [<parameter_name> [<value>]]
```

Command *param* with no arguments gives a list of all parameters with their values. When argument *parameter_↵ name* is specified, information about this parameter is printed (current value and short description).

The third argument is used to set a new value of the given parameter. The result of the setting is printed immediately.

During all interface operations, the protocol of the process (fail and warning messages, mapping of the loaded entities into OCCT shapes etc.) can be output to the trace file. Two parameters are defined in the DRAW session: trace level (integer value from 0 to 9, default is 0), and trace file (default is a standard output).

Command *xtrace* is intended to view and change these parameters:

- *Draw> xtrace* – prints current settings (e.g.: "Level=0 - Standard Output");

- *Draw> xtrace #* – sets the trace level to the value #;

- *Draw> xtrace tracefile.log* – sets the trace file as *tracefile.log*;

- *Draw xtrace* – directs all messages to the standard output.

## 4.2   Reading IGES files

For a description of parameters used in reading an IGES file refer to Setting the translation parameters.

These parameters are set by command *param* :

| Description | Name | Values |
|---|---|---|
| Precision for input entities | read.precision.mode | 0 or 1 |
|  | read.precision.val | real |
| Continuity of B splines | read.iges.bspline.continuity | 0-2 |
| Surface curves | read.surfacecurve.mode | 2, 3 or 0 |

It is possible either only to load an IGES file into memory (i.e. to fill the model with data from the file), or to read it (i.e. to load and convert all entities to OCCT shapes).

Loading is done by the command

```
Draw> xload <file_name>
```

Once the file is loaded, it is possible to investigate the structure of the loaded data. To learn how to do it see Analyzing the transferred.

Reading of an IGES file is done by the command

```
Draw> igesbrep <file_name> <result_shape_name> [<selection>]
```

Here a dot can be used instead of a filename if the file is already loaded by *xload* or *igesbrep* command. In that case, only conversion of IGES entities to OCCT shapes will be done.

Command *igesbrep* will interactively ask the user to select a set of entities to be converted:

| N | Mode | Description |
|---|------|-------------|
| 0 | End | finish conversion and exit igesbrep |
| 1 | Visible roots | convert only visible roots |
| 2 | All roots | convert all roots |
| 3 | One entity | convert entity with number provided by the user |
| 4 | Selection | convert only entities contained in selection |

After the selected set of entities is loaded the user will be asked how loaded entities should be converted into OCCT shapes (e.g., one shape per root or one shape for all the entities). It is also possible to save loaded shapes in files, and to cancel loading.

The second parameter of the *igesbrep* command defines the name of the loaded shape. If several shapes are created, they will get indexed names. For instance, if the last parameter is 's', they will be *s_1, ... s_N.*

<*selection*> specifies the scope of selected entities in the model, it is *xst-transferrable-roots* by default. An asterisk "∗" can be specified instead of *iges-visible-transf-roots*. For possible values of *selection* refer to Selecting entities section.

Instead of *igesbrep* it is possible to use commands:

```
Draw> trimport <file_name> <result_shape_name> <selection>
```

which outputs the result of translation of each selected entity into one shape, or

```
Draw> trimpcomp <file_name> <result_shape_name> <selection>
```

which outputs the result of translation of all selected entities into one shape (*TopoDS_Compound* for several entities).

An asterisk "∗" can be specified instead of *selection*, it means *xst-transferrable-roots*.

During the IGES translation, a map of correspondence between IGES entities and OCCT shapes is created. The following commands are available:

- *Draw> tpent #* – provides information on the result of translation of the given IGES entity;

- *Draw> tpdraw #* –creates an OCCT shape corresponding to an IGES entity;

- *Draw> fromshape <shape_name>* – provides the number of an IGES entity corresponding to an OCCT shape;

- *Draw> tpclear* – clears the map of correspondences between IGES entities and OCCT shapes.

## 4.3    Analyzing the transferred data

The procedure of analysis of the data import can be divided into two stages:

1. Checking the file contents;

2. Estimation of translation results (conversion and validated ratios).

### 4.3.1  Checking file contents

General statistics on the loaded data can be obtained by using command

```
Draw> data <symbol>
```

The information printed by this command depends on the symbol specified:

| Symbol | Output |
|--------|--------|
| g | Prints information contained in the header of the file (Start and Global sections) |
| c or f | Runs check procedure of the integrity of the loaded data and prints the resulting statistics (f works only with fails while c with both fail and warning messages) |
| t | The same as c or f, with a list of failed or warned entities |
| m or l | The same as t but also prints a status for each entity |
| e | Lists all entities of the model with their numbers, types, status of validity etc. |
| r | The same as e but lists only root entities |

There is a set of special objects, which can be used to operate with the loaded model. They can be of the following types:

| Special object type | Operation |
|---------------------|-----------|
| Selection Filters | allow selecting subsets of entities of the loaded model |
| Counters | Calculate statistics on the model data |

A list of these objects defined in the current session can be printed in DRAW by command

```
Draw> listitems
```

In the following commands if several <*selection*> arguments are specified the results of each following selection are applied to the results of the previous one.

```
Draw> givelist <selection_name> [<selection_name>]
```

prints a list of loaded entities defined by selection argument.

```
Draw> givecount <selection_name> [<selection_name>]
```

prints a number of loaded entities defined by *selection* argument.

Three commands are used to calculate statistics on the entities in the model:

- *Draw> count <counter> [<selection> ...]* – prints only a number of entities per each type matching the criteria defined by arguments.

- *Draw> sumcount <counter> [<selection> ...]* – prints the total number of entities of all types matching the criteria defined by arguments and the largest number corresponding to one type.

- *Draw> listcount <counter> [<selection> ...]* – prints a list of entities per each type matching the criteria defined by arguments.

Optional <*selection*> argument, if specified, defines a subset of entities, which are to be taken into account. Argument <*counter*> should be one of the currently defined counters:

| Counter | Operation |
|---------|-----------|
| xst-types | Calculates how much entities of each OCCT type exist |
| iges-types | Calculates how much entities of each IGES type and form exist |
| iges-levels | Calculates how much entities lie in different IGES levels |

The command:

```
Draw> listtypes <selection_name> ...
```

gives a list of entity types which were encountered in the last loaded file (with a number of IGES entities of each type). The list can be shown not for all entities but for a subset of them. This subset is defined by an optional selection argument.

Entities in the IGES file are numbered in the succeeding order. An entity can be identified either by its number (#) or by its label. Label is the letter 'D' followed by the index of the first line with the data for this entity in the Directory Entry section of the IGES file. The label can be calculated on the basis of the number as 'D(2∗# -1)'. For example, entity # 6 has label D11.

- *Draw> elab #* – provides a label for an entity with a known number;

- *Draw> enum #* – prints a number for an entity with the given label;

- *Draw> entity # <level_of_information>* – gives the content of an IGES entity;

- *Draw> estat #* – provides the list of entities referenced by a given entity and the list of entities referencing to it.

### 4.3.2 Estimating the results of reading IGES

All of the following commands are available only after the data are converted into OCCT shapes (i.e. after command **igesbrep**).

```
Draw> tpstat [*|?]<symbol> [<selection>]
```

provides all statistics on the last transfer, including the list of transferred entities with mapping from IGES to OCCT types, as well as fail and warning messages. The parameter *<symbol>* defines what information will be printed:

- G – General statistics (list of results and messages)

- C – Count of all warning and fail messages

- C – List of all warning and fail messages

- F – Count of all fail messages

- F – List of all fail messages

- N – List of all transferred roots

- S – The same, with types of source entity and result type

- B – The same, with messages

- T – Count of roots for geometrical types

- R – Count of roots for topological types

- I – The same, with a type of the source entity

The sign '∗' before the parameters **n**, **s**, **b**, **t**, **r** makes it work on all entities (not only on roots). The sign '?' before **n**, **s**, **b**, **t** limits the scope of information to invalid entities.

Optional argument *<selection>* can limit the action of the command with a selected subset of entities. To get help, run this command without arguments.

For example, to get translation ratio on IGES faces, you can use.

```
Draw:> tpstat *l iges-faces
```

The second version of the same command is TPSTAT (not capital spelling).

```
Draw:> TPSTAT <symbol>
```

Symbol can be of the following values:

- g – General statistics (list of results and messages)

- c – Count of all warning and fail messages

- C – List of all warning and fail messages

- r – Count of resulting OCCT shapes per each type

- s – Mapping of IGES roots and resulting OCCT shapes

Sometimes the trimming contours of IGES faces (i.e., entity 141 for 143, 142 for 144) can be lost during translation due to fails.

The number of lost trims and the corresponding IGES entities can be obtained by the command:

```
Draw> tplosttrim [<IGES_type>]
```

It outputs the rank and DE numbers of faces that lost their trims and their numbers for each type (143, 144, 510) and their total number. If a face lost several of its trims it is output only once.

Optional parameter <*IGES_type*> can be *TrimmedSurface, BoundedSurface* or *Face* to specify the only type of IGES faces.

For example, to get untrimmed 144 entities, use command

```
Draw> tplosttrim TrimmedSurface
```

To get the information on OCCT shape contents, use command

```
Draw> statshape <shape_name>
```

It outputs the number of each kind of shapes (vertex, edge, wire, etc.) in a shape and some geometrical data (number of C0 surfaces, curves, indirect surfaces, etc.).

Note. The number of faces is returned as a number of references. To obtain the number of single instances the standard command (from TTOPOLOGY executable) **nbshapes** can be used.

To analyze the internal validity of a shape, use command

```
Draw> checkbrep <shape_name> <expurged_shape_name>
```

It checks the geometry and topology of a shape for different cases of inconsistency, like self-intersecting wires or wrong orientation of trimming contours. If an error is found, it copies bad parts of the shape with the names "expurged_subshape_name _#" and generates an appropriate message. If possible, this command also tries to find IGES entities the OCCT shape was produced from.

<*expurged_shape_name*> will contain the original shape without invalid subshapes.

To get information on tolerances of subshapes, use command

```
Draw> tolerance <shape_name> [<min> [<max>] [<symbol>]]
```

It outputs maximum, average and minimum values of tolerances for each kind of subshapes having tolerances or it can output tolerances of all subshapes of the whole shape.

When specifying *min* and *max* arguments this command outputs shapes with names <*shape_name*>... and their total number with tolerances in the range *[min, max]*.

<*Symbol*> is used for specifying the kind of sub-shapes to analyze:

- v – for vertices,

- e – for edges,

- f – for faces,

- c – for shells and faces.

## 4.4 Writing an IGES file

Refer to Setting the translation parameters for a description of parameters used in reading an IGES file. The parameters are set by command *param*:

| Description | Name | Values |
|---|---|---|
| Author | XSTEP.iges.header.author | String |
| Company | XSTEP.iges.header.company | String |
| Receiver | XSTEP.iges.header.receiver | String |
| Write mode for shapes | XSTEP.iges.writebrep.mode | 0/Faces or 1/BRep |
| Measurement units | XSTEP.iges.unit | 1-11 (or a string value) |

Several shapes can be written in one file. To start writing a new file, enter command

```
Draw> newmodel
```

This command clears the *InterfaceModel* to make it empty.

```
Draw> brepiges <shape_name_1> [<filename.igs>]
```

Converts the specified shapes into IGES entities and puts them into the *InterfaceModel*.

```
Draw> writeall <filename.igs>
```

Allows writing the prepared model to a file with name *filename.igs*.

# 5    Reading from and writing to IGES

## 5.1    Reading from IGES

**Load an IGES file**

Before performing any other operation, you must load an IGES file with:

```
IGESCAFControl_Reader reader(XSDRAW::Session(),  Standard_False);
IFSelect_ReturnStatus stat = reader.ReadFile("filename.igs");
```

Loading the file only memorizes, but does not translate the data.

**Check the loaded IGES file**

This step is not obligatory. See the description of Checking the IGES file above.

**Set parameters for translation to XDE**

See the description of Setting translation parameters above.

In addition, the following parameters can be set for XDE translation of attributes:

- For transferring colors:

  ```
  reader.SetColorMode(mode);
  // mode can be Standard_True or Standard_False
  ```

- For transferring names:

  ```
  reader.SetNameMode(mode);
  // mode can be Standard_True or Standard_False
  ```

**Translate an IGES file to XDE**

The following function performs a translation of the whole document:

```
Standard_Boolean ok = reader.Transfer(doc);
```

where *doc* is a variable which contains a handle to the output document and should have a type *Handle(TDocStd↩_Document)*.

## 5.2    Writing to IGES

The translation from XDE to IGES can be initialized as follows:

```
IGESCAFControl_Writer aWriter(XSDRAW::Session(),Standard_False);
```

**Set parameters for translation from XDE to IGES**

The following parameters can be set for translation of attributes to IGES:

- For transferring colors:

  ```
  aWriter.SetColorMode(mode);
  // mode can be Standard_True or Standard_False
  ```

- For transferring names:

  ```
  aWriter.SetNameMode(mode);
  // mode can be Standard_True or Standard_False
  ```

**Translate an XDE document to IGES**

You can perform the translation of a document by calling the function:

```
IFSelect_ReturnStatus aRetSt = aWriter.Transfer(doc);
```

where "doc" is a variable which contains a handle to the input document for transferring and should have a type *Handle(TDocStd_Document)*.

**Write an IGES file**

Write an IGES file with:

```
IFSelect_ReturnStatus statw =  aWriter.WriteFile("filename.igs");
```

or

```
IFSelect_ReturnStatus statw = writer.WriteFile (S);
```

where S is OStream.

```
IFSelect_ReturnStatus aRetSt = aWriter.Transfer(doc);
```