Data Exchange

IGES FORMAT
User's Guide

Version 6.6.0 / April 2013

NOTICE FOR USERS:

This User Guide is a general instruction for Open CASCADE Technology study. It may be incomplete and even contain occasional mistakes, particularly in examples, samples, etc.

OPEN CASCADE S.A.S. bears no responsibility for such mistakes. If you find any mistakes or imperfections in this document, or if you have suggestions for improving this document, please, contact us and contribute your share to the development of Open CASCADE Technology: bugmaster@opencascade.com

http://www.opencascade.com/contact/

Table of Contents

1. Introduction

1.1. The IGES-Open CASCADE Technology processor

This manual 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 offerings on our web site at www.opencascade.org/support/training/

IGES files up to and including IGES 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.

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 subfigure outputs a shape
- Transformation Matrix.

NOTE

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.

NOTE

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 precision value. This parameter gives the precision used during translation when the read.precision.mode parameter value is 1.

0.0001: default.

any real positive (non null) value.

This value is a basis value for computation tolerances for TopoDS_Vertex, TopoDS_Edge and TopoDS_Face entities.

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.

NOTE

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.

NOTE

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.

Functionality of 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:

1. "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.
2. **"2DUSE_PREFERRED" (2): THE 2D IS USED TO REBUILD THE 3D IN CASE OF THEIR INCONSISTENCY,**
3. "2DUse_Forced" (-2): the 2D is always used to rebuild the 3D (even if 2D is present in the file),
4. "3DUse_Preferred" (3): the 3D is used to rebuild the 2D in case of their inconsistency,
5. "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

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 is how Shape Healing is called 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 ($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.

read.scale.unit

This parameter is obsolete (the parameter xstep.cascade.unit should be used instead when necessary). If it is set to 'M', the shape is scaled 0.001 times (as if it were in meters) after translation from IGES or STEP.

Default value is MM.

xstep.cascade.unit

This parameter defines units to which a shape should be converted when translated from IGES or STEP to CASCADE. 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();
```

`nbtrans` returns the number of items in the list that produced a shape.

`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

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. A Geom_TrimmedCurve is output 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.
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_TrimmedCurve whose basis curve is a Geom_Line.
112: Parametric Spline Curve	TopoDS_Edge or TopoDS_Wire	The geometric support is a Geom_BsplineCurve.
126: BSpline Curve	TopoDS_Edge or TopoDS_Wire	
130: Offset Curve	TopoDS_Edge or	The resulting shape is a TopoDS_Edge or a

	TopoDS_Wire	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.

NOTE

The type of OCCT shapes (either TopDS_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 TopoDS_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, - 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.
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: - a Geom_CylindricalSurface, - a Geom_ConicalSurface, - a Geom_SphericalSurface, - a 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, - a TopoDS_Shell if the base becomes a TopoDS_Wire. The geometrical support may be: - a Geom_Plane, - a Geom_Cylindrical Surface, - 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	For the needs of interface processing, the basis surface must be a face.

	TopoDS_Shell	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, - 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

The value is 10^{-7}. It is used as a minimal distance between points, which are considered distinct.

Resolution in the IGES file

This parameter 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

It is to be used instead of resolution from the file when parameter read.precision.mode is 1 (“User”).

Field EpsGeom of the class IGESToBRep_CurveAndSurface

This value 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. Returned by call to method IGESToBRep_CurvAndSurface::GetEpsGeom.

NOTE: Since this value is in 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

This value 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

This is value 0.01*Precision::Confusion = 10^{-9}. It is used to compare parametric bounds of curves.

Field EpsCoeff of the class IGESToBRep_CurveAndSurface

This value 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::GetEpsCoeff. This value is used for translating 2d objects (for instance, parametric curves).

Methods UResolution(tolerance3d), VResolution(tolerance3d) of the class GeomAdaptor_Surface or BRepAdaptor_Surface

Return tolerance in parametric space of a surface computed from 3d tolerance.

NOTE

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

Value is set for each object of class IGESToBRep_CurveAndSurface. 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. 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,
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:

Class 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).

Class 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 following classes:

IGESToBRep_TopoCurve,
IGESToBRep_TopoSurface,
IGESToBRep_BRepEntity,
ShapeFix_Wire

Class IGESToBRep_BRepEntity is intended for transferring BRep entities (IGES version ≥ 5.1) while the two former are used for translating geometry and topology defined in IGES < 5.1. Methods from IGESToBRep_BRepEntity call methods from IGESToBRep_TopoCurve and IGESToBRep_TopoSurface, while those call methods from IGESToBRep_BasicCurve and IGESToBRep_BasicSurface in order 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 the methods which are in charge of transferring curves from IGES curve entities (TransferCompositeCurve, Transfer2dCompositeCurve, TransferCurveOnFace, TransferBoundaryOnFace, TransferOffsetCurve, TransferTopoBasicCurve) 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 EpsGeom*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

This method builds 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::SetPrecision and MaxTol - 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::Transfer2dTopoBasicCurve

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 tolerance of the edge(s) is (are) 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 the 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::TransferCurveOnFace 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(). A shape-healing algorithm performs the correction of a resulting OCCT shape.

Class ShapeFix_Wire can increase the tolerance of a shape. This class is used in IGESToBRep_BRepEntity::TransferLoop, IGESToBRep_TopoCurve::TransferBoundaryOnFace and IGESToBRep_TopoCurve::TransferCurveOnFace for correcting a wire. The maximum possible tolerance which edges or vertices will have after invoking the methods of this class is MaxTolerance (set by method ShapeFix_Wire::MaxTolerance()).

2.7. Code architecture

2.7.1. List of the classes

Package IGESControl

IGESControl_Reader

Package IGESToBRep

IGESToBRep_Reader

IGESToBRep_Actor

IGESToBRep_CurveAndSurface

IGESToBRep_BasicCurve

IGESToBRep_BasicSurface

IGESToBRep_TopoCurve

IGESToBRep_TopoSurface

IGESToBRep_BRepEntity

Package IGESConvGeom

For description of classes, refer to CDL.

2.7.2. List of API classes

package IGESControl

IGESControl_Reader

package IGESToBRep

IGESToBRep_Reader

package IGESData

class IGESData_IGESModel

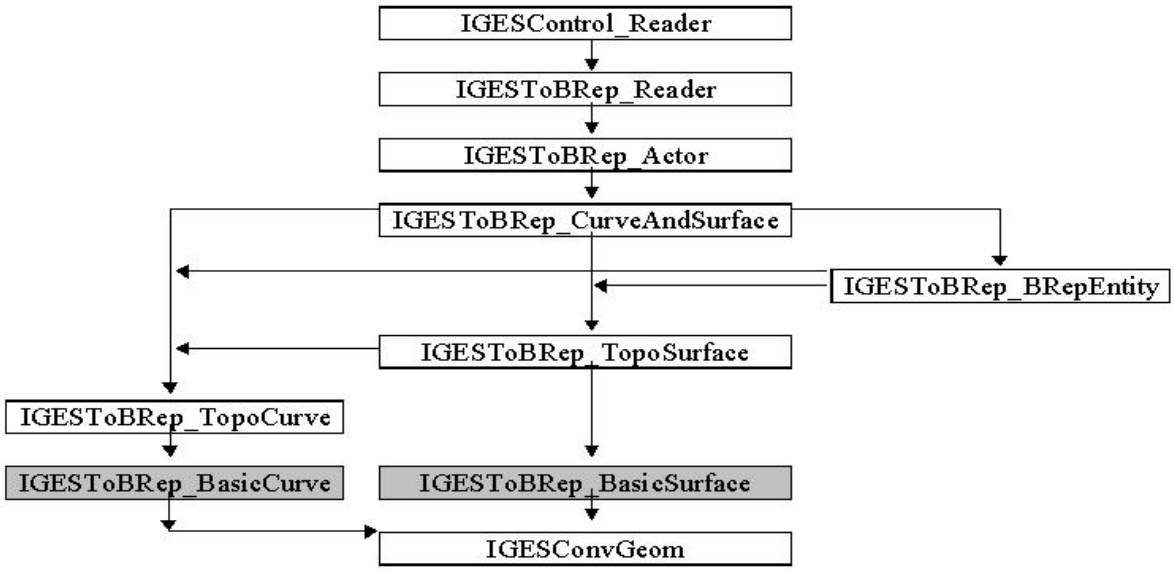
class IGESData_IGESEntity

For details, refer to 4 API for reading/writing IGES and CDL.

2.7.3. Graph of calls

The following diagram illustrates the structure of calls in reading IGES.

The highlighted classes produce OCCT geometry.



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:

gives the choice of the write mode. You can choose the following write modes:

"Faces" (0): OCCT TopoDS_Faces will be translated into IGES 144 (Trimmed Surface) entities, no B-Rep 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 B-Rep 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

For writing to IGES in the BRep mode (see parameter write.iges.brep.mode), this parameter indicates whether elementary surfaces (cylindrical, conical, spherical, and toroidal) are converted into corresponding IGES 5.3 entities (if parameter's value is On), or written as surfaces of revolution (by default).

Default value is Off.

write.iges.unit:

gives the choice of the unit. The default unit for Open CASCADE Technology is the millimeter. You can choose to write your file in any of the units that are 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");
```

Default value is "MM".

write.iges.header.autor:

gives the name of the author of the file.

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");
```

Default value is the system name of the user.

write.iges.header.company:

gives the name of the sending company.

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", "MDTV");
```

Default value is "" (empty).

write.iges.header.product:

gives the name of the sending product.

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");
```

Default value is "CAS.CADE IGES processor Vx.x" where x.x means the current version of Open CASCADE Technology.

write.iges.header.receiver:

gives the name of the receiving company.

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");
```

Default value is "" (empty).

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 (default),

"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 ..
```

Default value is "Average" (0).

write.precision.val:

user precision value. This parameter gives the resolution value for an IGES file when the write.precision.mode parameter value is 1.

0.0001: default

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

write.iges.sequence

The same as read.iges.*, please see above. Note that the default sequence for writing contains one operator – DirectFaces - 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 (shape);
```

where shape is a 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 either 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	
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_Rectangular TrimmedSurface	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_SurfaceOfLinear Extrusion	122: Tabulated Cylinder	
Geom_SurfaceOf Revolution	120: Surface Of Revolution	
Geom_OffsetSurface	140: Offset Surface	

3.4.3. Topological entities

Translation in Face mode

CASCADE shape	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 TopoDS_Wire results in a curve.
TopoDS_Wire in a TopoDS_Face	142: Curve On Surface	Both of the 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 TopoDS_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 TopoDS_Solid contains more than one TopoDS_Shell. One IGES entity is created per TopoDS_Shell.
TopoDS_Comp Solid	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_Comp ound	402: Form 1 Group or no equivalent	Group is created only if TopoDS_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.

Translation in BRep mode

CASCADE shape	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.
TopoDS_Compound	402 Form1 Group or no equivalent	Group is created only if TopoDS_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.

3.5. 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 3.3.2 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. List of the classes

package IGESControl

IGESControl_Controller

IGESControl_Writer

package BRepToIGES

BRepToIGES_BREntity

BRepToIGES_BRWire

BRepToIGES_BRShell

BRepToIGES_BRSolid

package BRepToIGESBRep

BRepToIGESBRep_Entity

package GeomToIGES

GeomToIGES_GeomPoint

GeomToIGES_GeomVector

GeomToIGES_GeomCurve

GeomToIGES_GeomSurface

package Geom2dToIGES

Geom2dToIGES_Geom2dCurve

package IGESConvGeom

IGESConvGeom_GeomBuilder

For description of classes refer to CDL.

3.6.2. List of API classes

package IGESControl

- IGESControl_Controller
- IGESControl_Writer

package IGESData

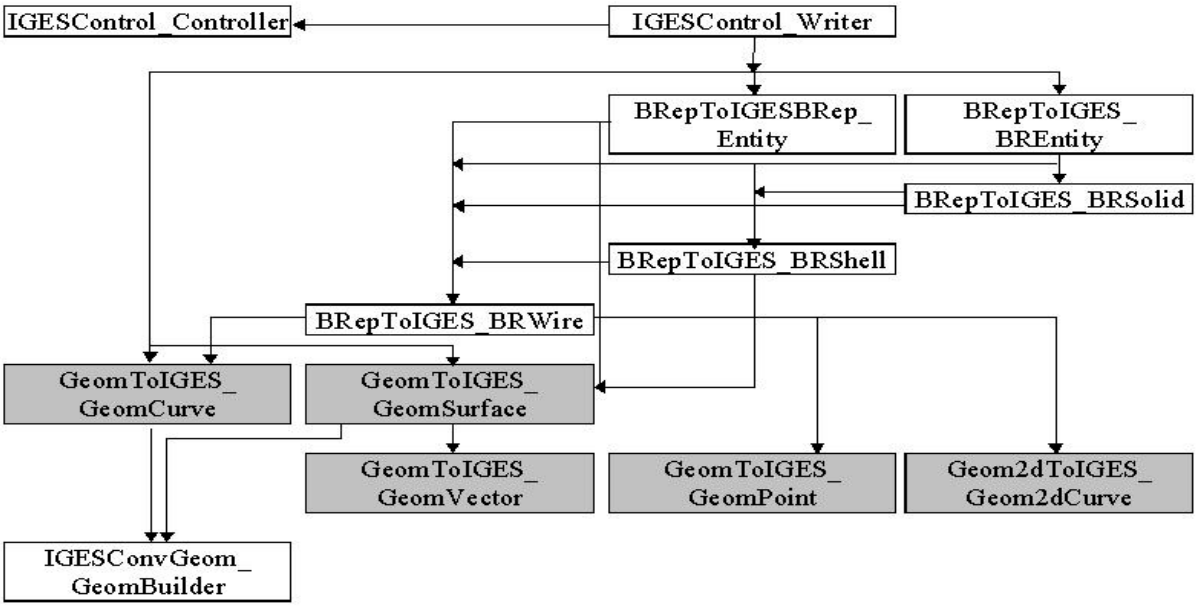
- class IGESData_IGESModel
- class IGESData_IGESEntity

For details refer to 4. API for reading/writing IGES and CDL.

3.6.3. Graph of calls

The following diagram illustrates the class structure in writing IGES.

The highlighted classes are intended to translate geometry.



3.7. Example

```
#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. API for reading/writing IGES

4.1. Overview

API classes provides the following tools:

- loading IGES files into memory,
- checking IGES files consistency,
- translating IGES files into OCCT shapes,
- translating OCCT shapes into IGES files,
- accessing the IGES model (which is an image of the IGES file in memory),
- selecting entities from the IGES model,
- accessing each entity in the IGES model.

4.2. Package IGESControl

4.2.1. General description

This package is intended to provide a tool to convert IGES-format entities to OCCT shapes and vice versa.

The package allows the end-user to perform both import from and export to an IGES file.

IGES files up to and including IGES version 5.3 can be read.

IGES files that are produced by this component conform to IGES version 5.3.

The result of reading IGES files is either a single Open CASCADE Technology shape or a set of them, the result of exporting Open CASCADE Technology geometric or topologic objects is an IGES file which may include one or several root entities (the ones not referenced by others).

4.2.2. Class IGESControl_Controller

General description

This class controls the IGES norm.

This class is intended to provide an appropriate initialization of the IGES norm, namely it includes a set of necessary parameters for IGES translation and declaration of possible selections for IGES entities.

After the execution of initialization procedures, the use of IGES norm becomes available.

```
Inheritance
        Standard_Transient
        MMgt_TShared
        XSControl_Controller
```

Methods

Constructors

IGESControl_Controller(const Standard_Boolean modefnes = Standard_False);

Purpose: Initializes the use of IGES (if <modefnes> is False) or FNES (if <modefnes> is True) norm.

Method for performing initialization

IGESControl:: Init

```
static  Standard_Boolean Init() ;
```

Purpose: Performs standard initialization creating controller objects for both IGES and FNES norm.

Returns True when done, False if an error occurred.

Method for creating IGES model

IGESControl:: NewModel

```
Handle_Interface_InterfaceModel NewModel() const;
```

Purpose: Creates a new empty model ready to receive data of the norm. The Global section is filled with static parameters (receiver, author, company and unit).

Method for getting the actor object

IGESControl:: ActorRead

```
Handle_Transfer_ActorOfTransientProcess ActorRead(
const Handle(Interface_InterfaceModel)& model) const;
```

Purpose: Returns the actor object for reading (actually, it is of type IGESToBRep_Actor) with a set parameter of spline continuity taken from static parameter.

Method for translating an Open CASCADE Technology shape

IGESControl:: TransferWriteShape

```
virtual IFSelect_ReturnStatus TransferWriteShape(const TopoDS_Shape&
                            shape,
                            const Handle(Transfer_FinderProcess)&
                            FP,       const
                            Handle(Interface_InterfaceModel)& model,
                            const Standard_Integer modetrans = 0)
                            const;
```

Purpose: Translates <shape> into the interface model.

<modetrans>: 0 - group of Faces (IGES < 5.1) , 1 - for BRep (IGES >= 5.1)

Returns:

`IFSelect_RetDone`: OK,

`IFSelect_RetError`: if <modetrans> is not equal to 0 or 1, or <model> is not an IGES model.

`IFSelect_Fail`: if <shape> is null.

4.2.3. Class IGESControl_Reader

General description

This object reads IGES files and translates their contents into OCCT shapes.

All definitions in IGES version 5.3 are recognized but only geometric and topologic entities can be translated. Data, which cannot be translated, is loaded with the file but during translation it is ignored.

The translation of the IGES model into the OCCT model goes through the following steps:

- loading a file into memory,
- checking file consistency,
- setting translation parameters,
- performing the translation itself,
- fetching the results.

The resulting OCCT objects belong to topologic shapes. The geometric objects (2D and 3D) are constructed in intermediate steps and serve as a support for topologic entities.

Each successful translation operation outputs one shape. A series of translations gives a list of shapes.

```
Inheritance
        IGESToBRep_Reader
```

This class complements IGESToBRep_Reader class:

- deals directly with WorkSession object,
- computes the list of IGES entities matching specified criteria,
- performs translation of a list of entities and the ones specified by handle,
- outputs the results of checking and translating.

Methods

Constructors:

- IGESControl_Reader ();

Purpose: Creates a reader from scratch and with a new WorkSession object.

- IGESControl_Reader (const Handle(XSControl_WorkSession)& WS,
  const Standard_Boolean scratch);

Purpose: Defines work session for the reader. If <scratch> is True the new model will be created in the work session.

Methods for dealing with WorkSession object

- IGESControl_Reader::SetWS

```
void SetWS ( const Handle(XSControl_WorkSession)& WS,
             const Standard_Boolean scratch = Standard_True);
```

Purpose: Defines the work session for the reader.

If <scratch> is True the new model will be created in the work session object.

- IGESControl_Reader::WS

```
Handle_XSControl_WorkSession() const;
```

Purpose: Returns the used work session object.

Method for loading an IGES file into memory

- IGESControl_Reader::ReadFile

```
IFSelect_ReturnStatus ReadFile(const Standard_CString filename);
```

Purpose: Loads and memorizes an IGES file in memory.

Returns:

`IFSelect_RetDone`: the file was successfully read

`IFSelect_RetVoid`: no file found

`IFSelect_RetError`: an error occurred during reading

See also:

IGESToBRep_Reader::LoadFile()

Methods for selecting entities to transfer

- IGESControl_Reader::GiveList

```
Handle_TColStd_HSequenceOfTransient GiveList( const
Standard_CString first = "", const Standard_CString second =
"");
```

Purpose: Returns a list of entities from the model according to the following rules:

- if <first> and <second> are empty - the list of roots for transfer,
- if <first> is a number or label of an entity - this entity itself,
- if <first> is a list of numbers/labels separated by commas - these entities,
- if <first> is a name of a selection in work session and <second> is not defined - the standard result of this selection,
- if <first> is a name of a selection and <second> is defined - the criterion defined by <second> is applied to result of <first> selection

Remarks:

if <second> is erroneous it is ignored.

```
Handle_TColStd_HSequenceOfTransient GiveList( const
Standard_CString first, const Handle(Standard_Transient)& ent) ;
```

Purpose: Returns a list of entities from the model according to the following rules:

- if <first> is a selection name and <second> is an entity or a list of entities (as a HSequenceOfTransient) - the standard result of this selection is applied to this list.

Remarks:

if <first> is erroneous, a null handle is returned.

Methods for performing translation

- IGESControl_Reader::TransferEntity

```
Standard_Boolean TransferEntity(const
Handle(Standard_Transient)& start) ;
```

Purpose: Performs the translation of the entity specified by its handle.

Returns False if an entity is not in the Model, else returns the result of the transfer.

- IGESControl_Reader:: TransferList

```
Standard_Integer TransferList(
const Handle(TColStd_HSequenceOfTransient)& list) ;
```

Purpose: Performs the translation of the list of entities.

Returns the number of successful transfers.

Methods for printing statistics

- IGESControl_Reader:: PrintCheckLoad

```
void PrintCheckLoad(
const Standard_Boolean failsonly,
const IFSelect_PrintCount mode) const ;
```

Purpose: Displays the check results on file entities.

If <failsonly> is True prints only « Fail » messages, otherwise all messages.

<mode> determines the contents and the order of messages:

`IFSelect_ItemsByEntity` - sequential list of messages per entity,

`IFSelect_CountByItem` - counts the number of entities per message,

`IFSelect_ShortByItem` - the same function but also of the first five entities,

`IFSelect_ListByItem` - the same but displays the rank numbers of all (not only five) entities,

`IFSelect_EntitiesByItem` - the same plus it displays the Directory Entry number for each entity

- IGESControl_Reader:: PrintCheckTransfer

```
void PrintCheckTransfer(
const Standard_Boolean failsonly,
const IFSelect_PrintCount mode) const;
```

Purpose: Displays the checking results of the last transfer.

The parameters play the same role as in `PrintCheckLoad`.

- IGESControl_Reader:: PrintStatsTransfer

```
void PrintStatsTransfer(
const Standard_Integer what,
const Standard_Integer mode = 0) const;
```

Purpose: Displays all available statistics of the last transfer on the default trace file. The returned information is filtered by giving parameters.

<what> defines what kind of statistics are to be printed:

0 - basic figures,

1 - root results,

2 - all recorded (roots, intermediate, checked entities),

3 - abnormal records,

4 - warnings and fails messages,

5 - only fail messages

<mode> is used according to <what>:

if <what> is 0 <mode> is ignored

if <what> is 1, 2 or 3 <mode> defines the following:

0 - lists numbers of concerned entities in the model,

1 - for each entity, gives the number, label, type and result type and/or status (fail / warning...),

2 - for each entity, gives the maximum information (check result),

3 - counts per type of starting entity (class type),

4 - counts per result type and/or status,

5 - counts per couple (starting type / result type/status),

6 – does the same thing plus gives for each item, the list of numbers of entities in the starting model

if <what> is 4 or 5 - <mode> is treated as enumeration `IFSelect_PrintCount`.

- IGESControl_Reader:: PrintTransferInfo

```
void PrintTransferInfo(
const IFSelect_PrintFail failwarn,
const IFSelect_PrintCount mode) const;
```

Purpose: Displays information concerning the last transfer on the default trace file according to the given parameters:

<mode> defines what will be printed:

`IFSelect_GeneralCount` - general statistics (number of selected IGES entities, number of root IGES entities, number of resulting OCCT shapes, number of fail and warning messages),

`IFSelect_CountByItem` - number of IGES entities per each message type and IGES type and form,

`IFSelect_ListByItem` - number and a complete list of DE numbers of IGES entities per each message type and IGES type and form,

`IFSelect_ResultCount` - number of resulting OCCT shapes per each type of the shape,

`IFSelect_Mapping` - mapping of root IGES entities into OCCT shapes per IGES type and form and OCCT shape type.

<failwarn> defines if only fail messages (if <failwarn> is IFSelect_FailOnly) or both fail and warning messages (if it is IFSelect_FailAndWarn) will be printed if <mode> is IFSelect_CountByItem or IFSelect_ListByItem.

4.2.4. Class IGESControl_Writer

General description

This class is intended to create and write an IGES file out of OCCT models.

IGES files produced by this component conform to IGES version 5.3.

This component gives a possibility to write an IGES file containing either geometric entities (conformant to IGES version less than 5.1) only or BRep entities (conformant to IGES version up to and including 5.3) as well. The writing mode is chosen by specifying the appropriate parameter.

The translation of an OCCT model (which can be a 2D or 3D geometric object or a topologic shape) into an IGES file is fulfilled in the following steps:

1. initializing the file,
2. setting the translation parameters,
3. performing the translation itself,
4. writing the IGES file.

Export to the IGES file can be performed on the basis of either an already existing IGES model (representation of the IGES file in memory) or a new one. The former case gives an opportunity to add geometric/topologic OCCT objects into an IGES model (file) that already exists.

Methods

Constructors:

- IGESControl_Writer() ;

Purpose: Creates a writer object with the default unit and write mode (Face).

- IGESControl_Writer(const Standard_CString unit, const Standard_Integer modecr = 0);

Purpose: Creates a writer object with the given values for the unit and write mode.

<unit> is the name of the units that are accepted by IGES in the upper case (« IN » or « INCH » for inches, « MM » for millimeters and so on),

<modecr> corresponds to write mode:

0 - Face (default),

1 - BRep

- IGESControl_Writer(const Handle(IGESData_IGESModel)& model,
  const Standard_Integer modecr = 0);

Purpose: Creates a writer object with an already prepared IGES model and write mode.

Methods dealing with IGES models

- IGESControl_Writer:: Model

```
Handle_IGESData_IGESModel Model() const;
```

Purpose: Returns the produced model.

- IGESControl_Writer:: ComputeModel() ;

```
void ComputeModel() ;
```

Purpose: Prepares the model before writing by setting the required statuses inside the model.

Methods dealing with transfer processes

- IGESControl_Writer:: SetTransferProcess

```
void SetTransferProcess(const Handle(Transfer_FinderProcess)&
TP) ;
```

Purpose: Sets the FinderProcess object for the writer.

- IGESControl_Writer:: TransferProcess

```
Handle_Transfer_FinderProcess TransferProcess() const;
```

Purpose: Returns the FinderProcess object (containing final results and messages if any).

Methods for performing translation

- IGESControl_Writer:: AddShape

```
Standard_Boolean AddShape(const TopoDS_Shape& sh) ;
```

Purpose: Translates a shape <sh> to IGES entities and adds them to the model.

Returns True if done, False if <sh> is not suitable for IGES or is null.

- IGESControl_Writer:: AddGeom

```
Standard_Boolean AddGeom(const Handle(Standard_Transient)& geom)
;
```

Purpose: Translates <geom> (which must be a curve or a surface) to IGES entities and adds them to the model.

Returns True if done, False if <geom> is neither a surface nor a curve suitable for IGES or is null.

- IGESControl_Writer:: AddEntity

```
Standard_Boolean AddEntity(const Handle(IGESData_IGESEntity)&
ent) ;
```

Purpose: Adds an IGES entity (and the ones it references) to the model.

Returns False if <ent> is null.

Methods for writing an IGES file

- IGESControl_Writer:: Write

```
Standard_Boolean Write( Standard_OStream& S,
const Standard_Boolean fnes = Standard_False) ;

Standard_Boolean Write( const Standard_CString file,
const Standard_Boolean fnes = Standard_False) ;
```

Purpose: Prepares (call ComputeModel()) and writes the model to the stream <S> or to the file <file>.

Returns True if the operation was correctly performed, False in case of error.

If mode <fnes> is equal to True, the resulting file will be written in the FNES format.

Method for obtaining statistics

- IGESControl_Writer:: PrintStatsTransfer

```
void PrintStatsTransfer( const Standard_Integer what,
const Standard_Integer mode = 0) const;
```

Purpose: Intended to display all statistics on the last performed translation.

Remarks: At the present moment does nothing (an empty method).

Package IGESToBRep

4.2.5. General description

Performs the actual translation of IGES entities into OCCT objects.

This package recognizes an IGES entity, performs its translation into an OCCT object (which can be 2D or 3D geometric objects, topologic shapes or transformation matrices) and returns the resulting shapes with associated messages (if there are any) occurred during the translation.

Those IGES entities that can be translated into OCCT objects by this package are given in the following table:

IGES entity	Type and Form number	OCCT class representing entity	OCCT class performing translation
Circular Arc	Type 100	IGESGeom_CircularArc	IGESToBRep_BasicCurve
Composite Curve	Type 102	..._CompositeCurve	..._TopoCurve
Conic Arc	Type 104	..._ConicArc	..._BasicCurve
Copious Data	Type 106, Form 1-3	..._CopiousData	..._BasicCurve
Linear Path	Type 106, Form 11-13	..._CopiousData	..._BasicCurve
Simple Closed Planar Curve	Type 106, Form 63	..._CopiousData	..._BasicCurve
Plane	Type 108	..._Plane	..._TopoSurface
Line	Type 110	..._Line	..._BasicCurve
Parametric Spline Curve	Type 112	..._SplineCurve	..._BasicCurve

Parametric Spline Surface	Type 114	..._SplineSurface	..._BasicSurface
Point	Type 116	..._Point	..._TopoCurve
Ruled Surface	Type 118	..._RuledSurface	..._TopoSurface
Surface of Revolution	Type 120	..._SurfaceOfRevolution	..._TopoSurface
Tabulated Cylinder	Type 122	..._TabulatedCylinder	..._TopoSurface
Transformation Matrix	Type 124	..._TransformationMatrix	..._BasicCurve
Rational B-Spline Curve	Type 126	..._BSplineCurve	..._BasicCurve
Rational B-Spline Surface	Type 128	..._BSplineSurface	..._BasicSurface
Offset Curve	Type 130	..._OffsetCurve	..._TopoCurve
Offset Surface	Type 140	..._OffsetSurface	..._TopoSurface
Boundary	Type 141	..._Boundary	..._TopoCurve
Curve on a Parametric Surface	Type 142	..._CurveOnSurface	..._TopoCurve
Bounded Surface	Type 143	..._BoundedSurface	..._TopoSurface
Trimmed (Parametric) Surface	Type 144	..._TrimmedSurface	..._TopoSurface
Manifold Solid B-Rep Object	Type 186	IGESSolid_ManifoldSolid	..._BRepEntity
Plane Surface	Type 190	..._PlaneSurface	..._TopoSurface
Subfigure Definition	Type 308	IGESBasic_SubfigureDef	..._CurveAndSurface
Group	Type 402, Form 1	..._Group	..._CurveAndSurface
Group Without Back Pointers	Type 402, Form 7	..._GroupWithoutBackP	..._CurveAndSurface
Single Parent	Type 402, Form 9	..._SingleParent	..._TopoSurface
Singular Subfigure Instance	Type 408	..._SingularSubfigure	..._CurveAndSurface
Vertex List	Type 502, Form 1	IGESSolid_VertexList	..._BRepEntity
Edge List	Type 504, Form 1	..._EdgeList	..._BRepEntity
Loop	Type 508	..._Loop	..._BRepEntity
Face	Type 510	..._Face	..._BRepEntity
Shell	Type 514	..._Shell	..._BRepEntity

Finally, all geometric IGES entities (curves and surfaces) are translated into topologic shapes. OCCT geometric objects serve as a support for topology.

4.2.6. Class IGESToBRep_Reader

General description

This class reads IGES files and translates their contents into OCCT shapes.

This class provides basic tools for loading, checking and translating IGES files into OCCT topologic shapes. It is complemented with more high-level features by class IGESControl_Reader.

The functionalities provided by this class are the following:

- loading a file into memory,
- checking an IGES model in memory,
- translating all root entities or one entity specified by its rank number into OCCT shapes,
- fetching the results.

Methods

Constructors:

- IGESToBRep_Reader();

Purpose: Performs initialization calling IGESAppli::Init() and IGESSolid::Init(), creates a new Actor object for transfer.

Method for loading an IGES file into memory

- IGESToBRep_Reader:: LoadFile

```
Standard_Integer LoadFile(const Standard_CString filename) ;
```

Purpose: Loads an IGES file <filename> into memory calling IGESFile_Read(), sets the returned IGES model (representing the loaded IGES file) calling SetModel().

Method for checking an IGES file

- IGESToBRep_Reader:: Check

```
Standard_Boolean Check(const Standard_Boolean withprint) const;
```

Purpose: Performs checking of a loaded IGES file calling Interface_CheckTool and Interface_CheckIterator. If <withprint> is True outputs the results of checking to the default trace file.

Methods for preparing the transfer process

- IGESToBRep_Reader:: SetModel

```
void SetModel(const Handle(IGESData_IGESModel)& model) ;
```

Purpose: Sets a new IGES model object. Clears the list of translated shapes (if there are any), sets a new transfer process object.

- IGESToBRep_Reader:: Model

```
Handle_IGESData_IGESModel Model() const;
```

Purpose: Returns the used IGES model object.

- IGESToBRep_Reader:: SetTransientProcess

```
void SetTransientProcess(const
Handle(Transfer_TransientProcess)& TP) ;
```

Purpose: Sets the transfer process object.

- IGESToBRep_Reader:: TransientProcess

```
Handle_Transfer_TransientProcess TransientProcess() const;
```

Purpose: Returns the used transfer process object.

- IGESToBRep_Reader:: Actor

```
Handle_IGESToBRep_Actor Actor() const;
```

Purpose: Returns the used actor object.

- IGESToBRep_Reader::Clear

```
void Clear() ;
```

Purpose: Clears the list of translated shapes.

Methods for translation

- IGESToBRep_Reader:: TransferRoots

```
void TransferRoots(const Standard_Boolean onlyvisible =
Standard_True)
```

Purpose: Performs the translation of root entities (ones that are not referenced by others). If <onlyvisible> is True, translates only visible entities (with Blank status equal to 0). Sets the continuity in accordance with the static parameter read.iges.bspline.continuity.

If parameter read.maxprecision.mode is set to 1, calls to ShapeTool_Utils::LimitTolerance() for the resulting shape with parameters 0 and the maximum between read.maxprecision.val and the basis tolerance of processor.

- IGESToBRep_Reader:: Transfer

```
Standard_Boolean Transfer(const Standard_Integer num) ;
```

Purpose: Performs the translation of an entity specified by its rank number.

Creates an object of class IGESToBRep_CurveAndSurface and sets:

3D precision (taking its value either from the file or from the work session in accordance with the static parameter read.precision.mode),

the approximation mode parameter in accordance with static the parameter read.iges.bspline.approxd1.mode,

the mode for a preferred computation of curves on a surface in accordance with the static parameter read.surfacecurve.mode,

the spline continuity parameter in accordance with the static parameter read.iges.bspline.continuity,

the transfer process object taken from itself.

Once all the fields have been filled out this method calls method TransferGeometry() with the IGES entity calculated by its rank number to obtain the OCCT shape.

Like method TransferRoots() this one also limits the tolerance if the static parameter read.maxprecision.mode is set to 1.

Returns False if <num> is greater than the number of entities in the model or less than 1, otherwise returns True even if there was an exception during the transfer.

Methods for fetching the results

- IGESToBRep_Reader:: IsDone

```
Standard_Boolean IsDone() const;
```

Purpose: Returns True if the last transfer was successful.

- IGESToBRep_Reader:: NbShapes

```
Standard_Integer NbShapes() const;
```

Purpose: Returns the number of shapes recorded in the result.

- IGESToBRep_Reader:: Shape

```
TopoDS_Shape Shape(const Standard_Integer num = 1) const;
```

Purpose: Returns the result number <num> where <num> is an integer between 1 and NbShapes(). If not returns a null shape.

- IGESToBRep_Reader:: OneShape

```
TopoDS_Shape OneShape() const;
```

Purpose: Returns all results in a single shape, which is:

- a null shape if there are no results,
- in the case of a single result, only that shape,
- a compound that lists all the results if there are several resulting shapes.

4.3. Package IGESData

4.3.1. General description

This package defines general objects for dealing with the IGES interface.

It gives a basic description of the IGES interface:

- defines the Model for IGES (class IGESData_IGESModel),
- defines the Protocol tool specific for IGES (class IGESData_Protocol)
- defines the basic class IGESData_IGESEntity describing abstract IGES entity
- defines classes derived from IGESEntity and representing general IGES entities (IGESData_LineFontEntity, IGESData_TransfEntity, IGESData_SingleParentEntity, etc.),

4.3.2. Class IGESData_IGESModel

General description

Gives an access to the general data in the Start and the Global sections of an IGES file.

Defines a model specific for IGES.

An IGES file includes the following sections:

Start,

Global,

Directory Entry,

Parameter Data,

Terminate

```
Inheritance:
        Interface_InterfaceModel
        MMgt_TShared
        Standard_Transient
```

Methods

Constructor

- IGESData_IGESModel ();

Purpose: Creates an empty IGES Model.

Methods for initializing

- IGESData_IGESModel::ClearHeader

```
void ClearHeader() ;
```

Purpose: Erases all the data in the Start and Global sections.

- IGESData_IGESModel::NewEmptyModel

```
Handle_Interface_InterfaceModel NewEmptyModel() const;
```

Purpose: Returns a new Empty Model of the same type as this object, i.e. of type IGESData_IGESModel.

Methods for dealing with the Start and the Global sections

- IGESData_IGESModel::DumpHeader

```
void DumpHeader(Standard_OStream& S,
const Standard_Integer level = 0) const;
```

Remark: the Integer parameter is intended to be used as a level indicator, but not used for the moment.

- IGESData_IGESModel::StartSection

```
Handle_TColStd_HSequenceOfHAsciiString StartSection() const;
```

Purpose: Returns the Start section of the Model as a list of lines.

- IGESData_IGESModel::NbStartLines

```
Standard_Integer NbStartLines() const;
```

Purpose: Returns the number of the lines in the Start section.

- IGESData_IGESModel::StartLine

```
Standard_CString StartLine(const Standard_Integer num) const;
```

Purpose: Returns a line from the Start section specified by number `num`.

Remark: An empty string is returned if number `num` is out of range [1, NbStartLines()].

- IGESData_IGESModel::ClearStartSection

```
void ClearStartSection() ;
```

Purpose: Clears the Start section.

- IGESData_IGESModel::SetStartSection

```
void SetStartSection(const
Handle(TColStd_HSequenceOfHAsciiString)& list, const
Standard_Boolean copy = Standard_True) ;
```

Purpose: Sets a new Start section from the list of strings <list>, copying it if <copy> is True (by default) or pointing to the <list> if <copy> is False.

- IGESData_IGESModel::AddStartLine

```
void AddStartLine(const Standard_CString line,
const Standard_Integer atnum = 0) ;
```

Purpose: Adds a new string to the end of the existing Start section if <atnum> is 0 or not given, or before the <atnum>-th line.

Remark: If a number is out of range [0, NbStartLines()], the line is added at the end of section.

- IGESData_IGESModel::GlobalSection

```
const IGESData_GlobalSection& GlobalSection() const;
```

Purpose: Returns the Global section of the Model.

- IGESData_IGESModel::SetGlobalSection.

```
void SetGlobalSection(const IGESData_GlobalSection& header) ;
```

Purpose: Sets the Model's Global section.

- IGESData_IGESModel::ApplyStatic

```
Standard_Boolean ApplyStatic(const Standard_CString param = "")
;
```

Purpose: Sets some parameters of the Global section to those defined by static parameters (see parameters of translation). The allowed values for <param> (all by default) are: receiver, author and company (these are acronyms of static parameters). Returns True when done and if <param> is given, False if <param> is unknown or empty.

Remark: To set a unit into the Global section use the IGESData_BasicEditor class.

See also: User's Guide: Parameters of translation.

- IGESData_IGESModel::GetFromAnother

```
void GetFromAnother(const Handle(Interface_InterfaceModel)&
other) ;
```

Purpose: Takes the Global section from another Model.

- IGESData_IGESModel::VerifyCheck

```
virtual void VerifyCheck(Interface_Check& ach) const;
```

Purpose: Checks whether the Global section contains valid data according to the IGES specification. If the Global section is correct this method adds nothing into ach, but if not the method adds fail messages.

- IGESData_IGESModel::SetLineWeights

```
void SetLineWeights(const Standard_Real defw) ;
```

Purpose: Sets LineWeights of entities according to the Global section (MaxLineWeight and LineWeightGrad values) or to a default value (<defw>) for undefined weights.

Methods for dealing with IGES entities

- IGESData_IGESModel::ClearLabels() ;

```
void ClearLabels() ;
```

Purpose: Erases labels. Not yet implemented.

- IGESData_IGESModel::PrintLabel

```
void PrintLabel(const Handle(Standard_Transient)& ent,
Standard_OStream& S) const;
```

Purpose: Prints the Directory Entry number of a given entity, i.e. 'Dnn' where Dnn=2*number-1on the stream S.

- IGESData_IGESModel::StringLabel

```
Handle_TCollection_HAsciiString StringLabel
(const Handle(Standard_Transient)& ent) const;
```

Purpose: Returns a string with a Directory Entry number of a given entity, i.e. a string 'Dnn' where Dnn=2*number-1.

- IGESData_IGESModel::Entity

```
Handle_IGESData_IGESEntity Entity(const Standard_Integer num)
const;
```

Purpose: Returns an entity given by its rank number.

- IGESData_IGESModel::DNum

```
Standard_Integer DNum(const Handle(IGESData_IGESEntity)& ent)
const;
```

Purpose: Returns the DE Number of an entity, i.e. 2*Number(ent)-1, or 0 if <ent> is unknown from this Model.

4.3.3. Class IGESData_IGESEntity

General description

Represents an abstract IGES entity.

This class provides an access to common IGES entity fields (TypeNumber, TransformationMatrix,

etc.).

This class is a basic one for other classes complementing it to represent a certain IGES entity.

Refer to the IGES specification for more details.

```
Inheritance
MMgt_TShared
Standard_Transient
```

Methods

Constructors:

- IGESData_IGESEntity();

Purpose: Creates an empty object. Sets all values to defaults (calls Clear()).

Methods for initializing fields of object.

- IGESData_IGESEntity::Clear

```
void Clear() ;
```

Purpose: Clears all fields of the object.

- IGESData_IGESEntity::InitTypeAndForm

```
void InitTypeAndForm( const Standard_Integer typenum, const
Standard_Integer formnum) ;
```

Purpose: Sets the Type and Form Numbers to new values.

Remarks: Private method. Reserved for special use.

- IGESData_IGESEntity::InitDirFieldEntity

```
void InitDirFieldEntity( const Standard_Integer fieldnum, const
Handle(IGESData_IGESEntity)& ent) ;
```

Purpose: Sets a directory field to an <ent> of any kind (see DirFieldEntity() for more details).

Remarks: If <fieldnum> is not equal to values listed in DirFieldEntity(), this method does nothing.

- IGESData_IGESEntity::InitTransf

```
void InitTransf(const Handle(IGESData_TransfEntity)& ent) ;
```

Purpose: Sets the Transf or erases it if <ent> is null.

- IGESData_IGESEntity::InitView

```
void InitView(const Handle(IGESData_ViewKindEntity)& ent) ;
```

Purpose: Sets the View or erases it if <ent> is null.

- IGESData_IGESEntity::InitLineFont

```
void InitLineFont( const Handle(IGESData_LineFontEntity)& ent,
const Standard_Integer rank = 0) ;
```

Purpose: Sets the LineFont. If <ent> is null the RankLineFont is set to <rank>, otherwise it is set to a negative value.

- IGESData_IGESEntity::InitLevel

```
void InitLevel( const Handle(IGESData_LevelListEntity)& ent,
const Standard_Integer val = 0) ;
```

Purpose: Sets the Level. If <ent> is null the DefLevel is set to <val>, otherwise it is set to a negative value.

- IGESData_IGESEntity::InitColor

```
void InitColor( const Handle(IGESData_ColorEntity)& ent, const
Standard_Integer rank = 0) ;
```

Purpose: Sets the Color. If <ent> is null the DefColor is set to <rank>, otherwise it is set to a negative value.

- IGESData_IGESEntity::InitStatus

```
void InitStatus( const Standard_Integer blank,
          const Standard_Integer subordinate,
          const Standard_Integer useflag,
          const Standard_Integer hierarchy) ;
```

Purpose: Sets the flags of the Directory Part.

- IGESData_IGESEntity::SetLabel

```
void SetLabel( const Handle(TCollection_HAsciiString)& label,
const Standard_Integer sub = -1) ;
```

Purpose: Sets a new Label to an Entity. If <sub> is given, it sets the value of SubScriptNumber, else SubScriptNumber is erased.

- IGESData_IGESEntity::InitMisc

```
void InitMisc( const Handle(IGESData_IGESEntity)& str,
          const Handle(IGESData_LabelDisplayEntity)& lab,
          const Standard_Integer weightnum) ;
```

Purpose: Sets data or erases it if it is given as null (zero for <weightnum>):

<str> for Structure,

<lab> for LabelDisplay,

<weightnum> for WeightNumber

- IGESData_IGESEntity::SetLineWeight

```
void SetLineWeight( const Standard_Real defw,
                    const Standard_Real maxw,
                    const Standard_Integer gradw) ;
```

Purpose: Computes and sets the "true" line weight according to IGES rules from the global data MaxLineWeight (<maxw>) and LineWeightGrad (<gradw>), or sets it to <defw> (Default) if LineWeightNumber is null

Remarks: If gradw is zero, there is division by zero in this method.

Methods for querying the corresponding fields of an IGES entity.

- IGESData_IGESEntity::IGESType

```
IGESData_IGESType IGESType() const;
```

Purpose: Returns information on the IGES type of an entity including the type and the form of that entity.

- IGESData_IGESEntity::TypeNumber

```
Standard_Integer TypeNumber() const;
```

Purpose: Returns the IGES Type number.

- IGESData_IGESEntity::FormNumber

```
Standard_Integer FormNumber() const;
```

Purpose: Returns the IGES Form number.

- IGESData_IGESEntity::DirFieldEntity

```
Handle_IGESData_IGESEntity DirFieldEntity(const Standard_Integer
fieldnum) const;
```

Purpose: Returns the Entity that is recorded for a given Field Number <fieldnum> where:

3 - Structure

4 - LineFont

5 - LevelList

6 - View

7 - Transf(ormation Matrix)

8 - LabelDisplay

13 - Color.

In a case of other values it returns a null handle.

- IGESData_IGESEntity::HasStructure

```
Standard_Boolean HasStructure() const;
```

Purpose: Returns True if an IGES entity is defined with a structure (it is normally reserved for certain classes, such as Macros).

- IGESData_IGESEntity::Structure

```
Handle_IGESData_IGESEntity Structure() const;
```

Purpose: Returns the Structure (used by some types of IGES entities only), returns a null handle if Structure is not defined.

- IGESData_IGESEntity::DefLineFont

```
IGESData_DefType DefLineFont() const;
```

Purpose: Returns the definition status of LineFont.

- IGESData_IGESEntity::RankLineFont

```
Standard_Integer RankLineFont() const;
```

Purpose: Returns LineFont definition as an integer if it is defined as Rank. If LineFont is defined as an Entity, returns a negative value

- IGESData_IGESEntity::LineFont

```
Handle_IGESData_LineFontEntity LineFont() const;
```

Purpose: Returns LineFont as an entity if it is defined as Reference. Returns a null handle if DefLineFont is not "DefReference".

- IGESData_IGESEntity::DefLevel

```
IGESData_DefList DefLevel() const;
```

Purpose: Returns the definition status of Level.

- IGESData_IGESEntity::Level

```
Standard_Integer Level() const;
```

Purpose: Returns Level definition as an integer.

- IGESData_IGESEntity::LevelList

```
Handle_IGESData_LevelListEntity LevelList() const;
```

Purpose: Returns LevelList if Level is defined as List. Returns a null handle if DefLevel is not "DefSeveral".

- IGESData_IGESEntity::DefView

```
IGESData_DefList DefView() const;
```

Purpose: Returns the definition status of View (None,One or Several).

- IGESData_IGESEntity::View

```
Handle_IGESData_ViewKindEntity View() const;
```

Purpose: Returns the View (either Single or List) if it is defined. Returns a null handle if it is not defined.

- IGESData_IGESEntity::SingleView

```
Handle_IGESData_ViewKindEntity SingleView() const;
```

Purpose: Returns View as Single, if defined as One. Returns a null handle if DefView is not "DefOne".

- IGESData_IGESEntity::ViewList

```
Handle_IGESData_ViewKindEntity ViewList() const;
```

Purpose: Returns View as a List. Returns a null handle if DefView is not "DefSeveral".

- IGESData_IGESEntity::HasTransf

```
Standard_Boolean HasTransf() const;
```

Purpose: Returns True if a Transformation Matrix is defined.

- IGESData_IGESEntity::Transf

```
Handle_IGESData_TransfEntity Transf() const;
```

Purpose: Returns the Transformation Matrix (under IGES definition). Returns a null handle if there is none.

Remarks: For a more complete use, see Location & CompoundLocation.

- IGESData_IGESEntity::HasLabelDisplay

```
Standard_Boolean HasLabelDisplay() const;
```

Purpose: Returns True if the LabelDisplay mode is defined for this entity.

- IGESData_IGESEntity::LabelDisplay

```
Handle_IGESData_LabelDisplayEntity LabelDisplay() const;
```

Purpose: Returns the LabelDisplay, if there is one; else returns a null handle.

- IGESData_IGESEntity::BlankStatus

```
Standard_Integer BlankStatus() const;
```

Purpose: Returns the Blank Status (0 - visible, 1 - blanked).

- IGESData_IGESEntity::SubordinateStatus

```
Standard_Integer SubordinateStatus() const;
```

Purpose: Returns the Subordinate Switch (0-1-2-3)

- IGESData_IGESEntity::UseFlag

```
Standard_Integer UseFlag() const;
```

Purpose: Returns the Use Flag (0 to 5) of an entity.

- IGESData_IGESEntity::HierarchyStatus

```
Standard_Integer HierarchyStatus() const;
```

Purpose: Returns the Hierarchy status (0-1-2).

- IGESData_IGESEntity::LineWeightNumber

```
Standard_Integer LineWeightNumber() const;
```

Purpose: Returns the LineWeight Number (0 if it is not defined).

See also: LineWeight.

- IGESData_IGESEntity::LineWeight

```
Standard_Real LineWeight() const;
```

Purpose: Returns "true" LineWeight, computed from LineWeightNumber and the global parameter of the Model by call to SetLineWeight.

- IGESData_IGESEntity::DefColor

```
IGESData_DefType DefColor() const;
```

Purpose: Returns the definition status of Color.

- IGESData_IGESEntity::RankColor

```
Standard_Integer RankColor() const;
```

Purpose: Returns the Color definition as an Integer (if defined as Rank). If Color is defined as an Entity, returns a negative value.

- IGESData_IGESEntity::Color

```
Handle_IGESData_ColorEntity Color() const;
```

Purpose: Returns the Color as an Entity (if defined as Reference) or a null handle if Color Definition is not "DefReference".

- IGESData_IGESEntity::CResValues

```
Standard_Boolean CResValues( const Standard_CString res1,
          const Standard_CString res2) const;
```

Purpose: Fills <res1> and <res2> with inner "reserved" alphanumeric fields theRes1 and theRes2. Returns False if both are blank, otherwise returns True.

Warning: Both must be of a length equal to at least 9 characters. The contents of <res1> and <res2> are modofied. The 9-th character becomes null.

- IGESData_IGESEntity::HasShortLabel

```
Standard_Boolean HasShortLabel() const;
```

Purpose: Returns True if ShortLabel is not null.

- IGESData_IGESEntity::ShortLabel

```
Handle_TCollection_HAsciiString ShortLabel() const;
```

Purpose: Returns label value as a string (null if ShortLabel is blank).

- IGESData_IGESEntity::HasSubScriptNumber

```
virtualStandard_Boolean HasSubScriptNumber() const;
```

Purpose: Returns True if SubScript Number is defined.

- IGESData_IGESEntity::SubScriptNumber

```
Standard_Integer SubScriptNumber() const;
```

Purpose: Returns SubScript Number as an integer (0 if not defined).

- IGESData_IGESEntity::HasOneParent()

```
Standard_Boolean HasOneParent() const;
```

Purpose: Returns True if an entity has one and only one parent, defined by a SingleParentEntity Type Associativity (explicit sharing).

Remarks: Thus, implicit sharing remains defined at the model level.

See class ToolLocation.

- IGESData_IGESEntity::UniqueParent() const;

```
Handle_IGESData_IGESEntity UniqueParent() const;
```

Purpose: Returns the Unique Parent (if it is the one).

Exceptions: <Interface_InterfaceError> if there are either several or no parents.

- IGESData_IGESEntity::Location()

```
gp_GTrsf Location() const;
```

Purpose: Returns the entity Location given by Transf in the Directory Part (see above). Considers local location only (not taking into account the parent's one - see CompoundLocation for that). If no Transf is defined, returns Identity.

- IGESData_IGESEntity::VectorLocation()

```
gp_GTrsf VectorLocation() const;
```

Purpose: Returns the Translation part of a local location (as for Location).

- IGESData_IGESEntity::CompoundLocation()

```
gp_GTrsf CompoundLocation() const;
```

Purpose: Returns the location of this object combined with CompoundLocation of its Parent (i.e. can be recursive). If the Parent is not single (see HasOneParent) returns Location.

- IGESData_IGESEntity::HasName()

```
Standard_Boolean HasName() const;
```

Purpose: Says if a Name is defined as Short Label or as Name Property. (Property is looked for first, otherwise ShortLabel is considered).

- IGESData_IGESEntity::NameValue()

```
Handle_TCollection_HAsciiString NameValue() const;
```

Purpose: Returns the Name value as a String (Property Name or ShortLabel). If SubNumber is defined, it is concatenated after ShortLabel as follows - label (number). Ignored in case of Property Name.

Methods for dealing with associativities and properties.

- IGESData_IGESEntity::ArePresentAssociativities()

```
Standard_Boolean ArePresentAssociativities() const;
```

Purpose: Returns True if the Entity is defined with an Associativity list, even an empty one (i.e., the file is of 0 length). Otherwise returns False (the file contains no identification concerning this list at all).

- IGESData_IGESEntity::NbAssociativities

```
Standard_Integer NbAssociativities() const;
```

Purpose: Returns the number of recorded associativities (0 if no list is defined).

- IGESData_IGESEntity::Associativities

```
Interface_EntityIterator Associativities() const;
```

Purpose: Returns the Associativity List in the form of an EntityIterator.

- IGESData_IGESEntity::NbTypedAssociativities

```
Standard_Integer NbTypedAssociativities
          const Handle(Standard_Type)& atype) const;
```

Purpose: Returns information on how many Associativities have the given type.

- IGESData_IGESEntity::TypedAssociativity

```
Handle_IGESData_IGESEntity TypedAssociativity
          (const Handle(Standard_Type)& atype) const;
```

Purpose: Returns the Associativity of a given Type (if one exists)

Exceptions: <Interface_InterfaceError> if there is none or more than one associativity.

- IGESData_IGESEntity::AddAssociativity

```
void AddAssociativity(const Handle(IGESData_IGESEntity)& ent) ;
```

Purpose: Adds an Associativity to the list (called by Associate only).

Exceptions: <Standard_NullObject> if <ent> is null.

- IGESData_IGESEntity::RemoveAssociativity

```
void RemoveAssociativity(const Handle(IGESData_IGESEntity)& ent)
;
```

Purpose: Removes an Associativity from the list (called by Dissociate).

Exceptions: <Standard_NullObject> if ent is null.

- IGESData_IGESEntity::LoadAssociativities

```
void LoadAssociativities(const Interface_EntityList& list) ;
```

Purpose: Loads a complete List of Asociativities (used during Read or Copy operations).

- IGESData_IGESEntity::ClearAssociativities

```
void ClearAssociativities() ;
```

Purpose: Removes all associativities at once.

- IGESData_IGESEntity::Associate

```
void Associate(const Handle(IGESData_IGESEntity)& ent) const;
```

Purpose: Sets this object to the Associativity list of another Entity. If <ent> is a null object, method does nothing.

- IGESData_IGESEntity::Dissociate

```
void Dissociate(const Handle(IGESData_IGESEntity)& ent) const;
```

Purpose: Removes this object from the Associativity list of another Entity. If <ent> is a null object, method does nothing.

- IGESData_IGESEntity::ArePresentProperties

```
Standard_Boolean ArePresentProperties() const;
```

Purpose: Returns True if the Entity is defined with a Property list, even an empty one (i.e., the file is of 0 length). Otherwise, returns False (file contains no identification concerning this list at all).

- IGESData_IGESEntity::NbProperties()

```
Standard_Integer NbProperties() const;
```

Purpose: Returns the number of recorded properties (0 if no list is defined)

- IGESData_IGESEntity::Properties()

```
Interface_EntityIterator Properties() const;
```

Purpose: Returns the Property List in the form of an EntityIterator

- IGESData_IGESEntity::NbTypedProperties

```
Standard_Integer NbTypedProperties
          (const Handle(Standard_Type)& atype) const;
```

Purpose: Returns information on how many Properties have a given type

- IGESData_IGESEntity::TypedProperty

```
Handle_IGESData_IGESEntity TypedProperty
          (const Handle(Standard_Type)& atype) const;
```

Purpose: Returns the Property of a given Type (if only one exists)

Exceptions: <Interface_InterfaceError> if there is none or more than one Properties.

- IGESData_IGESEntity::AddProperty

```
void AddProperty(const Handle(IGESData_IGESEntity)& ent) ;
```

Purpose: Adds a Property to the list.

Exceptions: <Standard_NullObject> if <ent>is null.

- IGESData_IGESEntity::RemoveProperty

```
void RemoveProperty(const Handle(IGESData_IGESEntity)& ent) ;
```

Purpose: Removes a Property from the list.

Exceptions: <Standard_NullObject> if <ent>is null.

- IGESData_IGESEntity::LoadProperties

```
void LoadProperties(const Interface_EntityList& list) ;
```

Purpose: Loads a complete List of Properties (used during Read or Copy operations).

- IGESData_IGESEntity::ClearProperties() ;

```
void ClearProperties() ;
```

Purpose: Removes all properties at once

5. Using XSTEPDRAW

5.1. XSDRAWIGES Overview

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.

This paragraph 5 is divided into several sections. Sections 5.3 and 5.5 deal with reading and writing of IGES files and are intended specifically for the IGES processor, sections 5.2 and 5.4 describe some general tools for setting parameters and analyzing the data. Most of them are independent of the norm being tested. Additionally, a table of mentioned DRAW commands is provided.

NOTE

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.

5.2. 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; and

```
Draw> xtrace .
```

- directs all messages to the standard output.

5.3. Reading IGES files

For a description of parameters used in reading an IGES file refer to 2.3.3 "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 5.4 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 (refer to 2.3.4)

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 was ‘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 for <selection> refer to 2.3.4.

Instead of igesbrep the following commands can be used:

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

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

```
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.

To get information on the result of translation of the given IGES entity the command

```
Draw> tpent #
```

is used.

To create an OCCT shape corresponding to an IGES entity the command

```
Draw> tpdraw #
```

is used.

To get the number of an IGES entity corresponding to an OCCT shape the command

```
Draw> fromshape <shape_name>
```

is used.

To clear the map of correspondences between IGES entities and OCCT shapes the command

```
Draw> tpclear
```

is used.

5.4. 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).

5.4.1. 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 to select subsets of entities of the loaded model
Counters	Calculate some 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 those of the one preceding it.

Command

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

prints a list of loaded entities defined by selection argument. For possible values of <selection_name> please refer to 2.3.4.

Command

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

prints a number of loaded entities defined by selection argument. For possible values of <selection_name> please refer to 2.3.4.

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

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. To get a label for an entity with a known number, command

```
Draw> elab #
```

can be used.

In the same way, command

```
Draw> enum D#
```

prints a number for an entity with the given label.

The content of an IGES entity can be obtained by using command

```
Draw> entity # <level_of_information>
```

The list of entities referenced by a given entity and the list of entities referencing to it can be obtained by command

```
Draw> estat #
```

5.4.2. 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**).

Command

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

is provided to get 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:

Symbol	Output
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.

Example. Translation ratio on IGES faces.

```
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:

Symbol	Output
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. To obtain the number of lost trims and the number of corresponding IGES entities the command

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

is used. 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.

Example. Untrimmed 144 entities.

```
Draw> tplosttrim TrimmedSurface
```

To get information on OCCT shape contents the command

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

is used.

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, command

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

is used. 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 the command

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

is used. 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.

5.5. Writing an IGES file

For a description of parameters used in reading an IGES file refer to 3.3.2 Setting the translation parameters.

These 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
```

Actually, command **newmodel** will clear the InterfaceModel to make it empty, and the next command will convert the specified shapes to IGES entities and put them into the InterfaceModel:

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

To write the prepared model to a file with name **<filename.igs>**, enter

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

5.6. Index of useful commands

Command	Description
Setting general parameters	
Xtrace [#|<file>|.]	View and set parameters of the trace file
Param [<param> [<val>]]	View and set parameters of transfer
Reading and writing an IGES file	
Xload <file>	Load a file into memory
Igesbrep {<file>|.} <name>	Translates an IGES file into a shape
Newmodel	Start new writing
Brepiges <shape_1> [<filename.igs>]	Translates a shape into an IGES model
Writeall <file>	Write an IGES model in memory to a file
Checking the results of the load procedure	

data <symbol>	Get statistics on the loaded file
listitems	Get list of all defined special objects like selections and counters
count <counter> [<selection>]	Count entities by counter
listcount <counter> [<selection>]	Count entities by counter and list them
givelist <selection>	Get a list of the subset of loaded entities defined by selection
listtypes [<selection>]	Get statistics on the types of the entities loaded
elab #	Get the label of an item with a given number
enum D#	Get the number of an item with a given label
entity {#|D#}	Dump data loaded for a particular entity
estatus {#|D#}	Get the load status of an entity and its references
Checking the translation results	
tpstate [*| ?]<symbol>	Get statistics on the entities converted
tplosttrim	Get statistics on the lost trimmings of IGES faces (entities 143, 144, 510)
tpdraw #	Get an OCCT shape as a result of translation for a given entity
tpent #	Outputs statistics of translation for a given entity
fromshape <shape>	Get the source IGES entity for the specified shape
tpclear	Clears the map of correspondences between IGES entities and OCCT shapes
Analysis of loaded shapes	
checkbrep <shape> <expurged_shape>	Check a shape for internal errors
statshape <shape>	Get statistics on a shape
tolerance <shape>	Calculate tolerances for a given shape

6. Reading from and writing to XDE

6.1. Description of the process

6.1.1. Loading 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.

6.1.2. Checking the loaded IGES file

This step is not obligatory. See the description of this step below in paragraph 2.3.2.

6.1.3. Setting parameters for translation to XDE

See the description of this step below in paragraph 2.3.3.

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

• Parameter for transferring colors:

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

• Parameter for transferring names:

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

6.1.4. Performing the translation of 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)`.

6.1.5. Initializing the process of translation from XDE to IGES

Here is how the process is initialized:

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

6.1.6. Setting parameters for translation from XDE to IGES

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

• Parameter for transferring colors:

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

• Parameter for transferring names:

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

6.1.7. Performing the translation of 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)`.

6.1.8. Writing 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