TiGL

2.2.1

Generated by Doxygen 1.8.12

Contents

1 Overview

The TiGL Geometry Library can be used for easy processing of geometric data stored inside CPACS data sets. Ti↩ GL offers query functions for the geometry structure. These functions can be used for example to detect how many segments are attached to a certain segment, which indices these segments have, or how many wings and fuselages the current airplane configuration contains. This functionality is necessary because not only the modeling of simple wings or fuselages but also the description of quite complicated structures with branches or flaps is targeted. The developed library uses the Open Source software OpenCASCADE to represent the airplane geometry by B-spline surfaces in order to compute surface points and also to export the geometry in the IGES/STEP/STL/VTK format. The library provides external interfaces for C/C++, Python, MATLAB and FORTRAN.

1.1 What is TiGL

In order to perform the modeling of wings and fuselages as well as the computation of surface points effectively, a geometry library was developed in C++. The library provides external interfaces for C and FORTRAN. Some of the requirements of the library were:

- Ability to read and process the information stored in a CPACS file for wings and fuselages,
- Possibility to extend to engine pods, landing gear and other geometrical characteristics,
- Ability to build up the three-dimensional airplane geometry for further processing,
- Ability to compute surface points in Cartesian coordinates by using common aircraft parameters,
- Possibility to be expanded by additional functions such as area or volume computations,
- Possibility to export the airplane geometry in the IGES format.

The developed library uses the Open Source software OpenCASCADE to represent the airplane geometry by B-spline surfaces in order to compute surface points and also to export the geometry in the IGES format. OpenCAS↩ CADE is a development platform written in C++ for CAD, CAM, and CAE applications which has been continuously developed for more than ten years. The functionality covers geometrical primitives (for example points, vectors, matrix operations), the computation of B-spline surfaces and boolean operations on volume models.

Apart from the already specified requirements above, the geometry library offers query functions for the geometry structure. These functions can be used for example to detect how many segments are attached to a certain segment, which indices these segments have, or how many wings and fuselages the current airplane configuration contains. This functionality is necessary because not only the modeling of simple wings or fuselages but also the description of quite complicated structures with branches or flaps is targeted.

1.2 TiGL Viewer

In order to review the geometry information of the central data set a visualization tool, TiGL Viewer, was developed. The TiGL Viewer allows the visualization of the used airfoils and fuselage profiles as well as of the surfaces and the entire airplane model. Furthermore, the TiGL Viewer can be used to validate and test the implemented functions of the geometry library, for example the calculation of points on the surface or other functions to check data that belong to the geometry structure.

2 Usage

In order to use the TiGL library in a C program, you have to link against the TiGL as well as the TiXI library. The TiXI (TiXI Xml Interface) is used to read in the XML-based CPACS files. It can be obtained from the project site http↩ ://github.com/DLR-SC/tixi. In addition to the libraries itself, one needs to include the corresponding header files:

```
#include "tigl.h"
#include "tixi.h"
```

For example, to open a CPACS configuration, one can use

```
TixiDocumentHandle tixiHandle;
TiglCPACSConfigurationHandle tiglHandle;
tixiOpenDocument("aFileName", &tixiHandle);
tiglOpenCPACSConfiguration(tixiHandle, "aConfigurationUID", &tiglHandle);
```

For more details and for the usage from Python or Matlab, see the examples in the

```
TIGL_INSTALL_DIR/share/doc/tigl/examples
```

directory.

3 Examples

TiGL ships with three examples of how to use the API from different programming languages: C/C++, Python and MATLAB. All examples can be found in the

```
TIGL_INSTALL_DIR/share/doc/tigl/examples
```

directory.

In order to run a TiGL based project, make sure to add

- TIGL_INSTALL_DIR/bin and TIXI_INSTALL_DIR/bin to *PATH on Windows*
- TIGL_INSTALL_DIR/lib(64) and TIXI_INSTALL_DIR/lib(64) to *LD_LIBRARY_PATH on Linux*
- TIGL_INSTALL_DIR/lib and TIXI_INSTALL_DIR/lib to *DYLD_LIBRARY_PATH on Mac*

C-Demo

This program is given by the file

```
c_demo.c
```

and an be build using the makefiles shipped with the example.

On *Windows*, open a visual studio command prompt, go into the

```
TIGL_INSTALL_DIR/share/doc/tigl/examples
```

directory. Open makefile.msvc in a text editor and change TIGL_HOME and TIXI_HOME to the appropriate directories. Then enter

```
nmake /f makefile.msvc
```

to build the c-demo.exe.

On *Linux*, open a shell and go into the directory

```
TIGL_INSTALL_DIR/share/doc/tigl/examples
```

Open makefile.gnu in a text editor and change TIGL_HOME and TIXI_HOME to the appropriate directories. Then type

```
make -f makefile.gnu
```

to build the c-demo.

To build this example program by yourself without a makefile, one has to link against the following libaries:

- (lib)TIGL
- (lib)TIXI

The OpenCASCADE libraries should be automatically linked by the TiGL shared library.

To run this program, call it with a CPACS file, e.g.

```
c-demo simpletest.cpacs.xml
```

Python-Demo

The Python demo is similar to the C demo. In order to make it work, one needs to add

```
TIGL_INSTALL_DIR/share/tigl/python
TIXI_INSTALL_DIR/share/tixi/python
```

to the PYTHONPATH environment variable. Afterwards, test it with

```
python python_demo.py simpletest.cpacs.xml
```

MATLAB-Demo

To run the MATLAB demo, one has to make sure, that the TiGL and TiXI libraries can be found by setting the path variables as described above.

The MATLAB bindings can be found in the following directories:

```
TIGL_INSTALL_DIR/share/tigl/matlab
TIXI_INSTALL_DIR/share/tixi/matlab
```

Please add both directories to the MATLAB path using the 'path' command of MATLAB.

The demo can be called as a function by entering

```
matlab_demo('simpletest.cpacs.xml')
```

JAVA-Demo

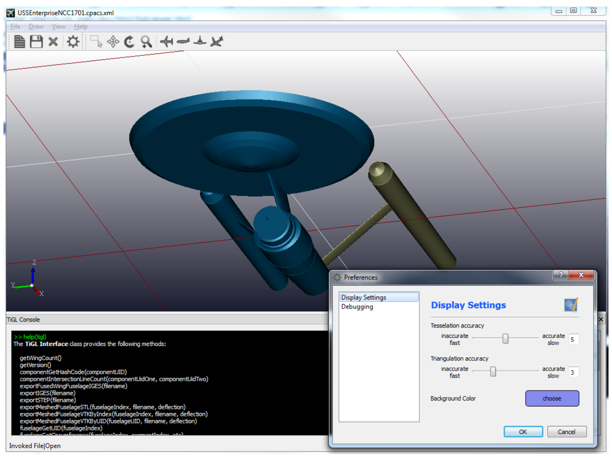
To compile the Java Demo (JavaDemo.java), you have to link against the following libraries

- commons-logging (Apache Commons logging)
- jna and jna-platform (Sun Java Native Access)
- tigl (the JAR file can be found in TIGL_INSTALL_DIR/share/tigl/java)

Under Windows it is important to chose the right TiGL library matching to the Java Virtual Machine. That means if you use a 64 Bit JVM, use the TiGL 64 bit library and vice versa.

4 TiGL Viewer

The TiGL Viewer is a 3D viewer for CPACS geometries. In addition, it also allows opening standard CAD file formats like IGES, STEP and BREP. Historically, TiGL Viewer was created as a TiGL debugging tool to visualize the modeled geometries.



```
>> help(tigl)
The TiGL Interface class provides the following methods:

   getWingCount()
   getVersion()
   componentGetHashCode(componentUID)
   componentIntersectionLineCount(componentUidOne, componentUidTwo)
   exportFusedWingFuselageIGES(filename)
   exportIGES(filename)
   exportSTEP(filename)
   exportMeshedFuselageSTL(fuselageIndex, filename, deflection)
   exportMeshedFuselageVTKByIndex(fuselageIndex, filename, deflection)
   exportMeshedFuselageVTKByUID(fuselageUID, filename, deflection)
   fuselageGetUID(fuselageIndex)
```

Invoked File|Open

Display Settings

Tesselation accuracy: inaccurate fast — accurate slow: 5

Triangulation accuracy: inaccurate fast — accurate slow: 3

Background Color

Figure 1 The TiGL Viewer showing a CPACS model of the Enterprise NCC 1701

Features of TiGL Viewer

- 3D Visualization CPACS, STEP, IGES, BREP, and STL files
- Creation of screen shots
- Conversion of geometries into standard CAD file formats
- Conversion into mesh formats such as STL, VTK, and COLLADA (for Blender rendering)
- A powerful scripting console that allows automatizing typical workflows, such as creating screen shots or making some debug plots.
- CPACS specific:
  - Display single CPACS entities such as wings, fuselages, profiles and guide curves
  - Compute and export a trimmed aircraft configuration
  - Compute discrete points on wings/fuselages using TiGL functions

4.1 Navigation in the 3D view

The most often used navigation functions are included in the tool bar

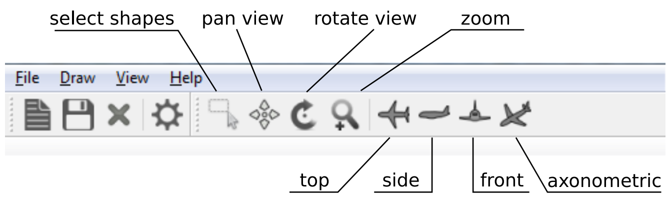


Figure 2 The navigation panel

Here, one can use

- "Select shapes" to select some parts ("shapes") of the geometry with the mouse.
- "Pan view" to translate the whole geometry with the mouse
- "Rotate view" to rotate the view point with the mouse
- "Zoom" to zoom in or out with the mouse
- "Top" to see the top view of the geometry
- "Side" to see the side view of the geometry
- "Front" to see the front view of the geometry
- "Axonometric" to see the axonometric view of the geometry

More functions can be found in the "View" menu.

4.2 Keyboard shortcuts

All actions in the TiGL Viewer can be accessed using the application menu or the context menu inside the 3D view. To improve usability, some the actions can also be executed with keyboard shortcuts. The most practical ones are:

Basic actions

Keys	Action
Ctrl+O	open file
Ctrl+S	save file
Ctrl+Q	close program

Change view

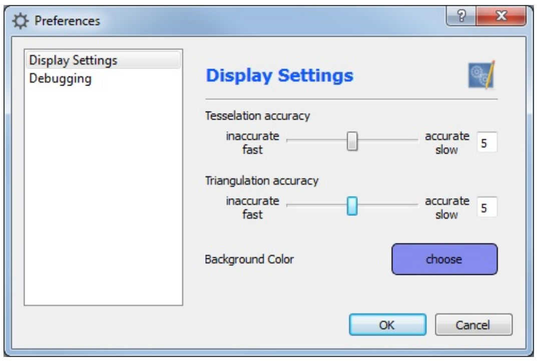
Keys	Action
1	Front view
2	Back view
3	Top view
4	Bottom view
5	Left view
6	Right view
Ctrl-D	Axonometric view
+	Zoom in
-	Zoom out
Ctrl-W	Toggle display of wireframe
Alt-C	Toggle display of console
Ctrl-E	Enable zoom mode
Ctrl-T	Enable pan view mode
Ctrl-R	Enable rotate view mode
Ctrl-A	Fits all objects into view
Ctrl-G	Toggle display of grid

4.3 The Settings

The settings dialog allows some customization of the visualization of objects and the export of meshes. It is opened by clicking on File -> Settings.

Display Settings

- Tesselation accuracy: the accuracy how the mathematical geometries are converted to triangles. The higher this setting is, the more triangles are created. High values typical require more computation time. **Default: 5**.
- Triangulation accuracy: similar to tesselation accuracy, but only used for export of triangular meshes (VTK, COLLADA, STL). **Default: 5**.
- Background color: base color of the 3D viewer's background gradient.


Preferences

Display Settings
Debugging

Display Settings

Tesselation accuracy
inaccurate fast — accurate slow: 5

Triangulation accuracy
inaccurate fast — accurate slow: 5

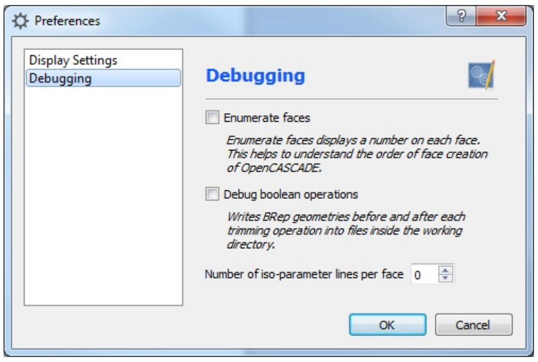
Background Color: choose

OK Cancel

Figure 3 The display settings dialog

Debugging

- Enumerate faces: If enabled, a number will be displayed next to each face in the viewer. This helps to understand the order of face creation. Mostly useful for TiGL developers. **Default: off**
- Debug boolean operations: Boolean operations tend to be quite unstable due to the problems in the Open↩ CASCADE kernel. To improve the debugging of such operations, TiGL Viewer can export intermediate geometries as BREP files to disk. These files can again be displayed in the TiGL Viewer. In case of an error, these files should be send to the TiGL developers. The files are placed inside the current working directory. **Default: off**


Preferences

Display Settings
Debugging

Debugging

☐ Enumerate faces

Enumerate faces displays a number on each face. This helps to understand the order of face creation of OpenCASCADE.

☐ Debug boolean operations

Writes BRep geometries before and after each trimming operation into files inside the working directory.

Number of iso-parameter lines per face 0

OK Cancel

Figure 4 The debugging dialog

4.4 Graphics Customization

The rendering off all geometrical objects can be be customized to some extends. In order modify some rendered objects, select the objects of interest and **press the right mouse button** inside the 3D view. The following actions are available:

- Setting the material (which affects the shading behavior)
- Setting the object color and transparency
- Toggle between wireframe and shading rendering

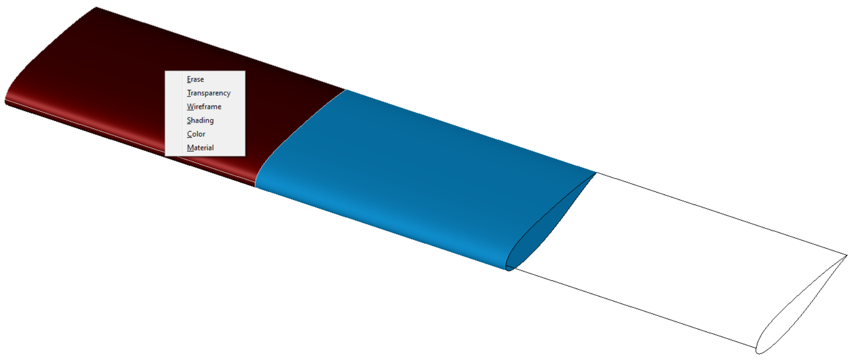


Figure 5 Different shading settings (glossy red, plastic blue, wireframe)

4.5 Mobile platforms

A simplified version of the TiGL Viewer is also available for mobile devices based on Android. It can be downloaded from the `Google play store`.

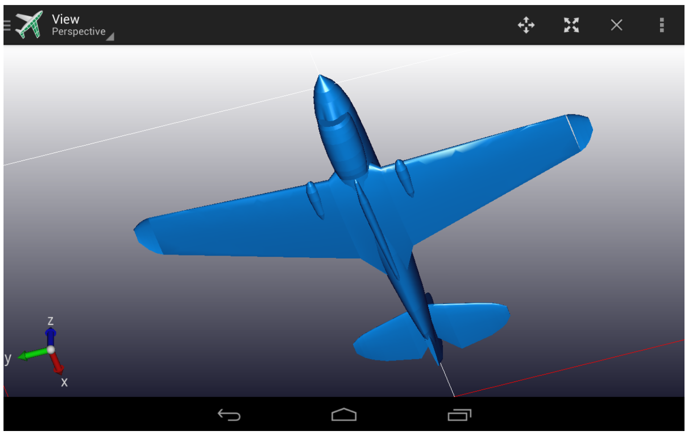


Figure 6 The TiGL Viewer App for Android

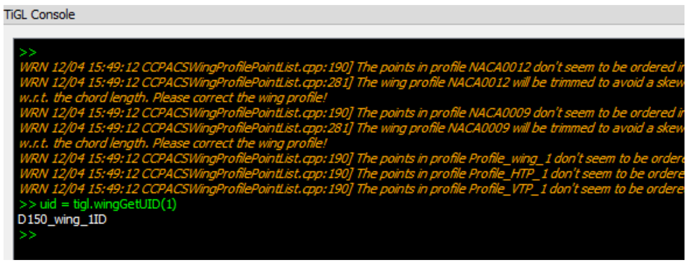
4.6 The TiGL Viewer Console

TiGL Viewer comes with a powerful JavaScript console that allows automatizing small workflows or helps debugging of geometries. To show and hide the console, press **ALT+C**.

The console has two purposes

1. to show all kinds of error messages, warnings etc. from the TiGL and OpenCASCADE engine
2. to enter user code to steer the TiGL Viewer application

In many cases, the user wants to open a CPACS file and execute TiGL function to compute and draw some points on the aircraft surface. Another use case would be to automatize the creation of screenshots of a changing aircraft geometry.



```
>>
WRN 12/04 15:49:12 CCPACSWingProfilePointList.cpp:190] The points in profile NACA0012 don't seem to be ordered i
WRN 12/04 15:49:12 CCPACSWingProfilePointList.cpp:281] The wing profile NACA0012 will be trimmed to avoid a skew
w.r.t. the chord length. Please correct the wing profile!
WRN 12/04 15:49:12 CCPACSWingProfilePointList.cpp:190] The points in profile NACA0009 don't seem to be ordered i
WRN 12/04 15:49:12 CCPACSWingProfilePointList.cpp:281] The wing profile NACA0009 will be trimmed to avoid a skew
w.r.t. the chord length. Please correct the wing profile!
WRN 12/04 15:49:12 CCPACSWingProfilePointList.cpp:190] The points in profile Profile_wing_1 don't seem to be order
WRN 12/04 15:49:12 CCPACSWingProfilePointList.cpp:190] The points in profile Profile_HTP_1 don't seem to be ordere
WRN 12/04 15:49:12 CCPACSWingProfilePointList.cpp:190] The points in profile Profile_VTP_1 don't seem to be ordere
>> uid = tigl.wingGetUID(1)
D150_wing_1ID
>>
```

Figure 7 The scripting console

We copied some practical features from the famous bash terminal to improve the usability of the console. This includes a history of the recently typed commands. Just type

```
history
```

To navigate through the recent commands from the history, use the up and down arrow keys.

To clear the console, enter

```
clear
```

For a general help, type in

```
help
```

4.6.1 Basic use

The console understands all kinds of JavaScript elements. Thus, typical constructs like loops, functions, and even classes can be defined. Instead of typing the code into the console, TiGL Viewer can also load and execute a script file. If you want to load a script file during the startup of TiGL Viewer, use the `--script` option:

```
 TIGLViewer --script myscript.js [--filename aircraft.cpacs.xml]
```

The `ans` object: The result of the last command will be stored in a variable `ans`. This can be sometimes quite practical:

```
>> 2+3
5
>> ans*ans
25
>> ans - 5
20
```

In addition to pure JavaScript, TiGL Viewer offers some function to draw points, vectors and TiGL shapes.

Draw a point at (0,0,0)

```
drawPoint(new Point3d(0,0,0));
```

Draw an arrow in z direction

```
pnt = new Point3d(0,0,0);
dir = new Point3d(0,0,5);
drawVector(pnt, dir);
```

Draw the first fuselage of a cpacs file

```
uid = tigl.fuselageGetUID(1);
shape = tigl.getShape(uid);
drawShape(shape);
```

In case you want to do some vector arithmetic, we defined the `Point3d` class, which offers some basic vector functionality:

```
>> p = new Point3d(10,0,0)
Point3d(10,0,0)
>> p.length()
10
>> p.add(p)
Point3d(20,0,0)
>> p.normalized()
Point3d(1,0,0)
>> p.dot(p)
100
>> p.mult(0.5)
Point3d(5,0,0)
>> q = new Point3d(0,10,0)
Point3d(0,10,0)
>> p.cross(q)
Point3d(0,0,100)
```

To get a list of all `Point3d` methods, enter

```
help(new Point3d);
```

4.6.2 The main application objects

The TiGL Viewer scripting engine consists of four main objects. The whole application can be controlled with only these four objects. These are app, app.viewer, app.scene, and tigl.

The scripting engine contains a *help()* function that shows the available methods and properties for each object. To show the help for e.g. the tigl object, enter

```
help(tigl);
```

4.6.2.1 The app object

This is the main object of the application. It is used to load and save files, load scripts, control the application's window size, and also close the application. Some methods of the app object are

method	description
openFile	opens a file (CPACS, STEP, IGES, BREP)
saveFile	saves all visible objects to a file (STEP, IGES, BREP, STL)
openScript	executes a TiGL Viewer script file
closeConfiguration	closes the current CPACS configuration and clears the scene
close	terminates TiGL Viewer

To get a list of all methods and properties, enter help(app) in the TiGL Viewer console.

4.6.2.2 The app.viewer object

This object controls the rendering of the 3D view. It can be used to

- change the camera position,
- make screen shots,
- set the background color or a background image,
- and fit all objects into the view.

There are different options, to make a screen shot of the viewer. The simplest is

```
app.viewer.makeScreenshot("myfile-white.jpg");
```

This will create a screenshot with a white background color and the size of the image will be the size of the viewer's window. To disable the white background, type

```
app.viewer.makeScreenshot("myfile.jpg", false);
```

In addition, you can also control the size of the resulting image file. However, **this feature does not work in Linux or Max OS X**. The syntax is the following:

```
app.viewer.makeScreenshot("myfile-large.jpg", true, 1500, 1000);
```

The most important methods of the *app.viewer* object are:

method	description
setBackgroundColor	sets a solid background color (RGB triple, 0..255)
setBackgroundGradient	sets a background with gradually darkening color(RGB triple, 0..255)
viewTop	moves camera to the top view
viewFront	moves camera to the front view
viewLeft	moves camera to the left view
viewAxo	changes camera to axonometrix view (top, left, front)
fitAll	fits all objects into the viewer window
zoomIn	zooms in the camera
zoomOut	zooms out the camera

To get a list of all methods and properties, enter

```
help(app.viewer);
```

in the TiGL Viewer console.

4.6.2.3 The app.scene object

The application scene controls, which objects are shown in the 3D view. Thus, it offers methods to display objects and points, to clear the whole scene and modifying the grid display.

The most important *app.scene* methods are:

method	description
wireFrame	if argument is true, all objetcs will be displayed in wireframe mode
displayShape	displays a TiGL geometry (e.g. got from tigl.getShape("wingUID"))
drawPoint	draws a point (e.g. app.scene.drawPoint(0,0,10))
drawVector	draws a vector/arrow (e.g. app.scene.drawVector(x,y,z,dx,dy,dz))
deleteAllObjects	clears the scene
gridOn	displays the grid
gridOff	turns of the grid display
gridXY	Sets the grid into the X-Y plane
gridXZ	Sets the grid into the X-Z plane
gridYZ	Sets the grid into the Y-Z plane
gridCirc	Switches the grid into polar form
gridRect	Swictehs the grid into cartesian form

To get a list of all app.scene methods and properties, enter:

```
help(app.scene);
```

4.6.2.4 The tigl object

The tigl object is used to access the TiGL library functions. Currently, not all functions are wrapped.

These are the functions currently wrapped by the scripting interface

- getWingCount()
- getVersion()
- componentGetHashCode(componentUID)
- componentIntersectionLineCount(componentUidOne, componentUidTwo)
- exportFusedWingFuselageIGES(filename)
- exportIGES(filename)
- exportSTEP(filename)
- exportMeshedFuselageSTL(fuselageIndex, filename, deflection)
- exportMeshedFuselageVTKByIndex(fuselageIndex, filename, deflection)
- exportMeshedFuselageVTKByUID(fuselageUID, filename, deflection)
- fuselageGetUID(fuselageIndex)
- fuselageGetCircumference(fuselageIndex, segmentIndex, eta)
- fuselageGetPoint(fuselageIndex, segmentIndex, eta, zeta)
- fuselageGetSegmentUID(fuselageIndex, segmentIndex)
- fuselageGetSegmentVolume(fuselageIndex, segmentIndex)
- getFuselageCount()
- fuselageGetSegmentCount(fuselageIndex)
- wingGetUpperPoint(wingIndex, segmentIndex, eta, xsi)
- wingGetUID(wingIndex)
- wingGetLowerPoint(wingIndex, segmentIndex, eta, xsi)
- wingGetUpperPointAtDirection(wingIndex, segmentIndex, eta, xsi, dirx, diry, dirz)
- wingGetLowerPointAtDirection(wingIndex, segmentIndex, eta, xsi, dirx, diry, dirz)
- wingGetChordPoint(wingIndex, segmentIndex, eta, xsi)
- wingGetChordNormal(wingIndex, segmentIndex, eta, xsi)
- wingGetSegmentCount(wingIndex)
- wingGetSegmentUID(wingIndex, segmentIndex)
- getErrorString(errorCode)
- getShape(uid)

In comparison to the C/C++ API, the scripting functions don't return error codes. Instead, an exception is thrown in case of an error, e.g. if no CPACS file is currently open.

The use of these functions is analog to the TiGL functions of the C API. The only exception is the `tigl.get↩ Shape(uid)` method. It can be used to return the CAD representation of a CPACS entity. Currently, only wings, fuselages, wing segments, and fuselage segments can be returned by `getShape`. The returned shape can then be rendered using the `drawShape` command.

The point sampling functions (e.g. `fuselageGetPoint`, `wingGetChordPoint`, ...) return a Point3d object.

Example

```
app.openFile("aircraft.cpacs.xml");
p = tigl.wingGetUpperPoint(1, 1, 0.5, 0.2);
drawPoint(p);
```

4.6.3 Examples

Here are some real life examples how to use the scripting engine:

Open a file

```
app.openFile("airplane.cpacs.xml");
```

Change to top view and save screenshot to image.png

```
app.viewer.viewTop();
app.viewer.fitAll();
app.viewer.zoomOut();
app.viewer.makeScreenshot("image.png");
```

*Convert a **STEP** file into **IGES** format and close TiGL Viewer*

```
app.openFile("part.stp");
app.saveFile("part.igs");
app.close();
```

Display all wings of an aircraft (which is already opened)

```
for (iwing = 1; iwing <= tigl.getWingCount(); ++iwing) {
  uid = tigl.wingGetUID(iwing);
  shape = tigl.getShape(uid);
  drawShape(shape);
}
```

Switch to fullscreen view

```
app.showFullScreen();
```

Delete all visible objects

```
app.scene.deleteAllObjects();
```

Disable the coordinate grid

```
app.scene.gridOff();
```

Show polar grid in y-z plane

```
app.scene.gridCirc();
app.scene.gridYZ();
app.scene.gridOn();
```

5 Latest Changes

Version 2.2.1

18/08/2017

- General changes:
  - Improved calculation time of `tiglFuselageGetPointAngle` by roughly a factor of 30. The results might be a different than in previous versions, but the function should be more robust now.
  - Improved calculation time of `tiglFuselageGetPoint` by applying caching. This leads only to a benefit in case of a large number of GetPoint calls (∼30) per fuselage segment. This will be even improved in TiGL 3.
- New API functions:
  - New API function `tiglExportVTKSetOptions`. This function can be used e.g to disable normal vector writing in the VTK export.
- Changed API:
  - Ignore Symmetry face in `tiglFuselageGetSurfaceArea` for half fuselages
  - In `tiglFuselageGetPointAngle` the cross section center is used as starting point of the angle rather than the origin of the yz-plane
- Fixes:
  - Fixed bug, where the VTK export showed no geometry in ParaView
  - Improved accuracy of the VTK export. The digits of points are not truncated anymore to avoid duplicate points
  - Triangles with zero surface are excluded from the VTK export
  - Fixed incorrect face name ordering in WingComponentSegment

Version 2.2.0

23/12/2016

- Major changes:
  - Added modelling of the wing structure, including ribs and spars . This code was part of a large pull request by Airbus D&S. Currently, the structure is only accessible by the TiGL Viewer. In future releases, we plan to make the structure accessible from the API.
  - Improved Collada export: The export is now conforming with the collada schema and can be displayed with OS X preview.
  - External shapes are added to the exports.
  - Added writing of modified CPACS files. Still, we do not offer yet API functions for modifications. Using the internal API, modifications are already possible.
- New API functions:
  - New API function `tiglSaveCPACSConfiguration` for writing the CPACS configuration into a file.
- TiGLViewer:
  - Visualization of the wing structure.
  - Improved linking and compilation with Qt.
  - Added STL export of the whole aircraft configuration.

Version 2.1.7

22/09/2016

- General changes:
  - Support for generic aircraft systems (by Jonas Jepsen).
  - External components are now conforming with the CPACS 2.3 standard (by Jonas Jepsen).
  - Support for rotorcraft (by Philipp Kunze).
  - Improved IGES export: Added support for long names in IGES (more than 8 characters as before).
  - Removed support for RedHat 5 and Ubuntu 13.10.
  - Added support for RedHat 7.
  - Ported to OpenCASCADE 6.9.0 and 7.0.0.
- New API functions:
  - `tiglWingComponentSegmentComputeEtaIntersection`: This function should be used to compute points on the wing that lie on a straight line between two given points.
  - `tiglFuselageGetIndex` to compute the index based on the fuselage UID.
  - Added new API functions for rotors and rotor blades:
    - `tiglGetRotorCount`
    - `tiglRotorGetUID`
    - `tiglRotorGetIndex`
    - `tiglRotorGetRadius`
    - `tiglRotorGetReferenceArea`
    - `tiglRotorGetTotalBladePlanformArea`
    - `tiglRotorGetSolidity`
    - `tiglRotorGetSurfaceArea`
    - `tiglRotorGetVolume`
    - `tiglRotorGetTipSpeed`
    - `tiglRotorGetRotorBladeCount`
    - `tiglRotorBladeGetWingIndex`
    - `tiglRotorBladeGetWingUID`
    - `tiglRotorBladeGetAzimuthAngle`
    - `tiglRotorBladeGetRadius`
    - `tiglRotorBladeGetPlanformArea`
    - `tiglRotorBladeGetSurfaceArea`
    - `tiglRotorBladeGetVolume`
    - `tiglRotorBladeGetTipSpeed`
    - `tiglRotorBladeGetLocalRadius`
    - `tiglRotorBladeGetLocalChord`
    - `tiglRotorBladeGetLocalTwistAngle`
- Fixes:
  - Fixed parent-child transformations in case of multiple root components.
  - Fixed an error in `tiglWingComponentSegmentGetPoint` in case of multiple intersections of the eta plane with the wing. This was the case for e.g. box wings. (issue #176).
  - Fixed bug `tiglWingGetSpan` in wing span computation when no wing symmetry is given (e.g. for a VTP) (issue #185 and #195).
  - Fixed another bug in `tiglWingGetSpan` when the symmetry plane was the Y-Z plane (issue #174).

  - Fixed incorrect result in `tiglWingGetSegmentEtaXsi` near wing sections, returning the wrong section (issue#187).
  - Fixed an issue in `tiglWingGetSegmentEtaXsi` in case the airfoil is completely above the chord surface.
  - Fixed point projection on the geometry for large scale data (eg a factor of 1000) by making the convergence criterium size dependent (issue #203).

- TiGLViewer:
  - Visualization of rotorcraft and rotorcraft specific menus.
  - Visualization of generic aircraft systems.
  - Added `wingGetSpan` function to TiGLViewer scripting.
  - Added script function `wingComponentSegmentGetPoint`.
  - Fixed tiglviewer.sh script loading wrong OpenCASCADE libraries.
- Language bindings:
  - Started experimental python bindings for the internal API. This allows a direct manipulation of the geometry objects from python together with the OpenCASCADE python bindings (pythoncc).
  - The source code of the matlab bindings is now shipped on all systems. In addition, we distribute a Makefile which can be used to compile the Matlab bindings when needed.
  - Added function `tiglWingComponentSegmentComputeEtaIntersection` to java bindings.

Version 2.1.6

15/07/2015

- TiGL Viewer:
  - Fixed critical crash on Windows 64 bit systems that occured sporadically on some systems

Version 2.1.5

01/07/2015

- Changed API:
  - Added an output argument in the function `tiglWingComponentSegmentPointGetSegment↩EtaXsi` that returns the error of the computation.
  - The function `tiglWingComponentSegmentGetSegmentIntersection` uses a new algorithm that should ensure straight flap leading edges and straight spars. Also, a new parameter `has↩Warning` was added, to inform the user, that the returned segment xsi value is not in the valid range [0,1]. This might be the case, if a spar is partially located outside the wing.
  - Removed macros TIGL_VERSION and TIGL_VERSION_MAJOR from tigl.h. Please use tigl_version.h instead.
- General changes:
  - External geometries can be included into the CPACS file using a link to a STEP file. The allows e.g. the use of engines and nacelles. (Note: this is not yet included in the CPACS standard)
  - Improved computation of half model fuselage profiles. Now, fuselages are c2 continuous at the symmetry plane.

  - Improved computation speed of `tiglWingComponentSegmentGetPoint` by a factor of 30 to 600 (depending on the geometry).
  - Reduced execution time of `tiglOpenCPACSConfiguration`.
  - All TiXI messages (errors/warnings) are now printed to the TiGL log.
  - Ported to OpenCASCADE 6.8.0.
- New API functions:
  - `tiglExportFusedBREP` to export the fused configuration to the BRep file format.
- Fixes:
  - Fixed bug, where guide curves on half model fuselages were not touching the symmetry plane.
  - Fixed a TIGL_MATH_ERROR bug in `tiglWingComponentSegmentGetSegmentIntersection`.
- TiGL Viewer:
  - Ported to Qt 5.
  - OpenGL accelerated rubber band selection. This fixes the slow rubber band selection on Linux and the invisible rubber band selection on Mac.
  - Fixed bug when loading a CPACS file with multiple models (thanks Jonas!).
  - Removed support for legacy VRML and CSDFB files.

Version 2.1.4

06/02/2015

- Changed API:
  - Added an output argument in the functions`tiglWingGetUpperPointAtDirection` and `tiglWingGetLowerPointAtDirection` that returns the error of the computation
- New API functions:
  - `tiglFuselageGetIndex` and `tiglFuselageGetSegmentIndex` to retrieve the index of a fuselage and fuselage segment given a CPACS UID
- Fixes:
  - Fixed some warnings using CMake 3
- TiGL Viewer:
  - Fixed a crash in case of a missing model UID
  - Fixed a bug, where debugging BREP files where always created on Linux
  - Improved scripting interface for wingGetLower/UpperPointAtDirection to return the error
- Language bindings:
  - Completed the new Java bindings for TiGL
  - Removed Fortran bindings since nobody is using them

Version 2.1.3

08/12/2014

- Changed API:
  - Changed functions `tiglWingGetSegmentSurfaceArea` to exclude side faces and trailing edges
  - Removed functions `tiglWingGetUpperPointAtAngle` and `tiglWingGetLowerPoint←AtAngle`. These functions were replaced by `tiglWingGetUpperPointAtDirection` and `tiglWingGetLowerPointAtDirection`.
- General changes:
  - Support for global transformation (translation refType="absGlobal"). Notice: some (incorrect) CPACS models will now look differently.
  - Accurate B-Spline approximation of CST curves using Chebychev approximation.
  - Implemented recursive fusing trimming for the use of e.g. bellyfairings. Added small cpacs example how to model a bellyfairing.
  - Implemented trimming of intersection curves with parent bodies and far fields
  - Improved computation of fuselage positionings. This should improve loading times of configurations with large number of sections, since the algorithmic complexity is reduced.
  - IGES + STEP export: All wing faces are now classified as Top-Wing, Bottom-Wing, or Trailing-Edge.
  - IGES export: Changed units to mm.
  - IGES export: Implemented layers/levels .
  - The OCAF framework is no longer required to build TiGL.
  - The build system now uses the cmake config-style TiXI and OCE search mechanisms.
- New API functions:
  - Added functions to get the B-Spline paramterization of fuselage and wing profiles:
    * `tiglProfileGetBSplineCount`, returns the of B-Splines a profile is built of.
    * `tiglProfileGetBSplineDataSizes`, returns the size of the knot vector and the number of control points of one profile B-Spline.
    * `tiglProfileGetBSplineData`, returns the knot vector and the control points.
  - Added functions `tiglWingGetSectionCount` and `tiglFuselageGetSectionCount`
  - Added functions `tiglWingGetSegmentUpperSurfaceAreaTrimmed` and `tiglWingGet←SegmentLowerSurfaceAreaTrimmed` to e.g. compute the surface area of a control device.
  - Added functions `tiglWingGetChordPoint` and `tiglWingGetChordNormal` to query points on the wing chord surface.
  - Added functions `tiglWingGetUpperPointAtDirection` and `tiglWingGetLower←PointAtDirection`.
- Fixes:
  - Fixed `tiglWingComponentSegmentGetPoint` in case of global wing transformations.
  - Fixed `tiglWingComponentSegmentGetPoint` bug, returning eta values $>$ 1 (issue 107).
  - Fixed numerical inaccuracy of `tiglWingGetPointDirection`
  - Fixed incorrect CST curves at for N2 $<$ 1.
  - Fixed null pointer bug in IGES export.
- TiGLViewer:
  - Highly improved scripting console:
    * Script file can be given as command line argument (using option –script)

        * Added function to export all objects to file
        * The main application objects can be scripted (i.e. app, app.viewer, app.scene)
        * Draw shapes, points and vectors from script
        * Make screenshots by command
        * Context menu for copy-paste actions
        * History with recent commands
        * Mouse support
        * More wrapped tigl functions
        * Improved stability
        * Exception handling
        * TiXI errors and warnings are printed on the console now
    - Added dialog to draw points and vectors.
    - Added dialog for screenshot settings, including option for white background.
    - JPEG and PNG support for background images.
    - Collada export for the complete configuration (i.e. support for multiple objects).
    - Cleanup of menu entries.
    - Fixed multiple opened CPACS documents in TiGLViewer.
    - Fixed 3D view flickering on Mac OS X with Qt 4.8.6.

- Language bindings:
    - Added python 3 support of `tiglwrapper.py`
    - Improved TiGL library loading error messages in python wrapper
    - New Java bindings. Hand-written high level API not yet complete (not all TiGL functions wrapped). Low level API (autogenerated) can be used instead for unimplemented functions.
    - Added Java example (see share/doc/tigl/examples/JavaDemo.java)

- Documentation:
    - Added chapter for TiGL Viewer and the TiGL Viewer scripting console.

Version 2.1.2

17/04/2014

- Changed API:
    - The returned UID strings of the following functions must not be freed by the user anymore:
        * `tiglWingGetOuterSectionAndElementIndex`
        * `tiglWingGetInnerSectionAndElementUID`
        * `tiglFuselageGetStartSectionAndElementUID`
        * `tiglFuselageGetEndSectionAndElementUID`
        * `tiglWingComponentSegmentPointGetSegmentEtaXsi`
        * `tiglWingComponentSegmentFindSegment`
    - Changed behavior of `tiglWingComponentSegmentFindSegment`. In case the specified point does not lie within 1 cm of any segment, `TIGL_NOT_FOUND` is returned.
    - Changed behavior of `tiglWingComponentSegmentPointGetSegmentEtaXsi`. If the specified point lies outside any segment so that the transformation can not be executed, `TIGL_MATH_E↩RROR` is returned.
    - The following API functions `tiglComponentIntersectionPoint`, `tiglComponent↩IntersectionPoints` and `tiglComponentIntersectionLineCount` are deprecated and will be removed in future releases. These functions are replaced by new intersection routines.

  - Removed functions `tiglExportStructuredIGES` and `tiglExportStructuredSTEP`
- General changes:
  - Completely reworked boolean operations with the following effects:
    * Fusing the whole plane is faster in many cases and more reliable
    * We keep track of the origins of each trimmed face which helps for the IGES and STEP exports
    * The boolean operations can be debugged now by setting the environment variable TIGL_DEBU↩ G_BOP
  - Completely rewritten STEP and IGES exports:
    * Each face has now an identifier name
    * Inclusion of intersection curves
    * Inclusion of far fields into exports
    * IGES export in non-BREP mode since it is not supported by CATIA
  - Added new demos for C, Python and MATLAB. Please look into the documentation how to run them.
  - Added function for the intersection computation of a geometrical shape with a plane
  - Experimental implementation of wing and fuselage guide curves
  - The mathematical orientation of wing profiles is checked for correctness
  - TiGL requires now at least OpenCASCADE 6.6.0
  - Ported to OpenCASCADE 6.7.0
- New API functions:
  - Added new API functions for shape/shape and shape/plane intersections:
    * `tiglIntersectComponents`
    * `tiglIntersectWithPlane`
    * `tiglIntersectGetLineCount`
    * `tiglIntersectGetPoint`
- Fixes:
  - Fixed a bug in `tiglWingComponentSegmentFindSegment`
  - Fixed numerical inaccuracy in the projection of points to a wing segment
  - Fixed crash in case of too long UIDs
  - Fixed some memory leaks
- TiGLViewer:
  - Improved IGES import:
    * Multiple shapes are now imported as separate shapes from IGES files.
    * Fixed wrong scaling when importing IGES files.
  - Improved STEP export:
    * Files are now exported in units of meters instead of millimeters.
  - Improved BREP export and import: compound objects are now decomposed to separate objects.
  - Added method to export fused aircraft to BREP
  - Added method to export wing and fuselage profiles and guide curves to BREP
  - Fixed incorrect units in IGES and STEP export from save-as dialog.
  - General save-as dialog: All visible objects are saved to the file, if no shape is selected. Otherwise, only the selected objects are exported.
  - Added debugging parameters to settings dialog. This includes:
    * Adapting number of displayed iso lines per face
    * Display face numbers

    * Debug boolean operations (if enabled, debugging shapes in BRep format are stored to the current working directory)
  - Added new dialog for shape/shape and shape/plane intersections.
  - Added drag and drop support to TiGLViewer. Files (CPACS, IGES, STEP, BREP) are opened when they are dragged into TiGLViewer.
  - TiGLViewer displays far field after calculating the trimmed aircraft
  - Iso U/V lines are no longer displayed by default. This can be changed in the settings dialog.
  - The fused/trimmed aircraft geometry is displayed using a different color for each component.
  - Added visualization of wing and fuselage guide curves.
  - Fix: Only the first intersection curve was displayed. This is fixed now.
  - New icons

Version 2.1.1

Released: 28/01/2014

- Changed API:
  - In previous TiGL version, some strings had to be freed manually after calling some functions. These strings must not be freed anymore. Following functions are affected:
    * `tiglWingGetOuterSectionAndElementIndex`
    * `tiglWingGetInnerSectionAndElementUID`
    * `tiglFuselageGetStartSectionAndElementUID`
    * `tiglFuselageGetEndSectionAndElementUID`
    * `tiglWingComponentSegmentPointGetSegmentEtaXsi`
    * `tiglWingComponentSegmentFindSegment`
    * Changed return value of `tiglWingComponentSegmentFindSegment`: In case the given point is located more than 1 cm away from any segment, TIGL_NOT_FOUND is returned.
    * Changed return value of `tiglWingComponentSegmentPointGetSegmentEtaXsi`: If the given point lies outside any segment so that the transformation can not be executed, TIG←L_MATH_ERROR is returned.
- General changes:
  - Changed console logging to include errors and warnings by default
  - Wing profiles are automatically trimmed at their trailing edge to ensure, that the trailing edge is always perpendicular to the chord line. This is required by the `wingGetUpperPoint` and `wingGet←LowerPoint` functions.
- Fixes:
  - Fixed a bug in `tiglWingGetUpperPoint` and `tiglWingGetLowerPoint` in which some points could not be calculated
  - Fixed two memory leaks

Version 2.1.0

Released: 17/01/2014

- Changed API:
  - Added argument for `tiglWingGetReferenceArea` to define the projection plane for reference area calculations
- General Changes:
  - Support for parametric CST wing profiles
  - Logging improvements. The console verbosity can now be set independent of file logging.
- New API functions:
  - `tiglWingGetMAC`, computes the mead aerodynamic chord length and position (thanks to Arda!)
- Fixes:
  - Fixed crash in case of missing wing and fuselage profiles
  - Fixed accuracy errors in `tiglWingSegmentPointGetComponentSegmentEtaXsi` and `tiglWingComponentSegmentPointGetSegmentEtaXsi`
  - Fixed a warning when including `tigl.h`
  - Fixed numerical bug `tiglWingComponentSegmentGetSegmentIntersection`
- TiGLViewer:
  - Improved dialog for displaying wing component segment points
  - Added BRep export
  - Fixed crash on some Linux systems with strange LANG settings
  - Added dialog showing log history in case of an error

Version 2.0.7

Released: 21/11/2013

- Changed API:
  - replaced `tiglWingComponentSegmentGetMaterialUIDs` with `tiglWingComponent↩ SegmentGetMaterialUID` and `tiglWingComponentSegmentGetMaterialCount`
- General Changes:
  - Implementation of far fields
  - More work on STEP and IGES export. The geometry is now exported as faces instead of solids in the STEP export.
  - Verification of airfoils during CPACS loading, this fixes twisted wing segments for some CPACS files
  - Improved modelling of leading and trailing edge
  - New Logging Framework
- New API functions:
  - `tiglWingGetSegmentEtaXsi`, to transform global x,y,z coordinates into wing segment coordinates
  - `tiglExportFusedSTEP`, exports the trimmed/fused geometry as a step file

  - `tiglWingComponentSegmentGetMaterialUID`, to get the material UID at a point of the component segment
  - `tiglWingComponentSegmentGetMaterialThickness`, to get the material thickness at a point of the component segment
  - `tiglWingComponentSegmentGetMaterialCount`, to get the number of materials defined at a point of the component segment
  - `tiglLogSetFileEnding`, `tiglLogSetTimeInFilenameEnabled`, `tiglLogToFile↵Disabled`, `tiglLogToFileEnabled`, `tiglLogToFileStreamEnabled`, to modify logging settings

- Fixes:
  - fixed a bug in `tiglWingComponentSegmentGetSegmentIntersection`

- TiGLViewer:
  - Loading of HOTSOSE mesh files
  - Auto-reload of non-CPACS files
  - Improved display of airfoils
  - Improved error dialogs

Version 2.0.6

Released: 27/08/2013

- General Changes:
  - Improved loading times
  - Improved speed of intersection calculations by caching and reusing results
  - Switched to faces instead of solids in STEP Export
  - Switched to const `char * API`
  - Prepared TiGL for Android (currently working but still experimental)

- New API functions:
  - `tiglExportMeshedWingSTLByUID` and `tiglExportMeshedFuselageSTLByUID` for S↵TL export (Hello 3D printing!)
  - `tiglWingComponentSegmentGetSegmentIntersection`

- Fixes:
  - Fixed `tiglFuselageGetPointAngle` and `tiglFuselageGetPointAngleTranslated` giving wrong results (issues 89 and 92)
  - TiGL required positionings for all fuselage segments, this is now fixed (issue 57)
  - Fixed error when opening CPACS files with composite materials
  - Fixed wrong units in iges export (issue 78)
  - Silenced Error messages that weren't errors
  - Fixed different errors in `tiglWingComponentSegmentPointGetSegmentEtaXsi`
  - Fixed duplicate log files

Version 2.0.5

Released: 11/06/2013

- Changed API:
  - Switched to const `char * strings`
  - `tiglWingGetSegmentIndex` returns now also the wing index instead of asking for it
  - Removed TiGL wire algorithms switching functions. Now a Bspline wire is used for each profile.
- General Changes:
  - Symmetry modeling
  - Explicit modeling of the wing leading edge to improve tesselated output (no more leading edge bumps)
  - Explicit modeling of the wing upper and lower shape
  - The fuselage lofting now creates a smooth surface without hard edges (due to fixed OpenCASCADE bug)
  - Completely rewritten VTK export:
    - Full body export including wing segment metadata
    - Fixed lots of errors.
    - Large improvement of calculation times.
    - Provides wing segment metadata in a proper VTK-way, to allow visualization of these data in VTK viewers. Calculation and export of the proper normal vectors.
  - IGES and STEP export with CPACS metadata
  - Collada export for use in 3D rendering programs like Blender
  - MATLAB bindings for TiGL!
  - Completeley rewritten python bindings (e.g. better support for arrays)
  - Added logging framework to write outputs into log files (google-glog)
  - Ported to OpenCASCADE 6.6.0
- New API functions:
  - `tiglWingComponentSegmentGetMaterialUIDs`, to query materials on component segment
  - `tiglWingComponentSegmentGetPoint`, to query cartesian point on the wing component segment (on chord surface)
  - `tiglWingSegmentPointGetComponentSegmentEtaXsi`, to compute segment to component segment coordinates
  - `tiglWingComponentSegmentGetSegmentUID`, queries the uids of the ith segment of the component segment
  - `tiglWingComponentSegmentGetNumberOfSegments`, queries the number of segments belonging to a component segment
  - `tiglExportFuselageColladaByUID`
  - `tiglExportWingColladaByUID`
  - `tiglExportStructuredIGES`
  - `tiglExportStructuredSTEP`
  - `tiglConfigurationGetLength`, returns the length of the airplane
  - `tiglWingGetSpan`, returns the wing span
  - `tiglComponentIntersectionPoints` (convenience function, vectorizes `tiglComponent↩ IntersectionPoint` to improve speed)
  - `tiglExportMeshedGeometryVTKSimple` and `tiglExportMeshedGeometryVTK↩ Simple` (replaced dummy implementation)

- Fixes:
  - Intersection calculation used by `tiglComponentIntersectionPoint`
  - Fixed incorrectly placed fuselage positionings
  - Fixed `tiglWingComponentSegmentFindSegment` returning segments that don't belong to the component segment
  - Workaround to buggy OpenCASCADE boolean fuse algorithms (which seems to be non-commutative). As a result, the fusing of the whole plane can be slower than before.
  - Removed warning about missing component segments
- TiGLViewer:
  - Display user defined component segment point
  - Improved speed of intersection calculation
  - Improved speed of shape triangulation
  - Display upper and lower shape of the wing
  - Display of the full model incorporating the symmetry properties of the CPACS components
  - Fixed detection, if CPACS file is changed while displayed in TiGLViewer
  - An optional control file can steer some basic settings like tesselation/triangulation accuracy (e.g. to tune speed of cpacs file opening)
  - STEP export
  - Settings Dialog (tesslation accuracy settings, background...)
  - New icon and color scheme

Version 2.0.4

Released: 17/01/2013

- New API Functions:
  - Added function `tiglGetErrorString`
- Fixes:
  - Fixed bug in `tiglWingComponentSegmentPointGetSegmentEtaXsi`
  - Fixed camberline / chordline bug
  - Fixed exception, when no airfoils are available
- TiGLViewer:
  - Added scripting interface to TIGLViewer. It is now possible to call TIGL/TIXI functions from within the TIGLViewer. The output is displayed in the console-view. Find more information here.
  - Added context-popup menu on right click. Available actions:
    - Removing of geometric shapes
    - Set transparency level of geometric shapes
    - Set color of selected shapes
    - Set material of selected shapes
    - Set wireframe/shading of selected shapes
  - Works now also in Mac OSX (since opencascade commit 4fe5661)
  - Added toolbars
  - Added option to load a background image
  - Added autosave of user settings
  - Menus are disabled/enabled depending on dataset
  - Script could be loaded from file via File->open Script
  - Fixed opening from command line
  - Fixed bad font rendering for OpenCascade? > - 6.4.0

Version 2.0.3

Released: 17/01/2013

- General Changes:
  - Added `CCPACSWingSegment::GetChordPoint` to internal API
  - Added a simple cpacs data set for accuracy testing
  - Added some accuracy tests for getPoint functions
  - Added support for visual leak detector
  - Changed UnitTesting? Framework to google-test
  - Changed TIGL linking to static for unit tests
  - Added coverage with gcov (gcc only) to project
- New API Functions:
  - `tiglWingGetIndex` Returns the wing index given a wing UID
  - `tiglWingGetSegmentIndex` Returns the segment index given a wing segment UID
  - `tiglWingGetComponentSegmentCount` Returns the number of component segments for a specific wing (selected by wing index)
  - `tiglWingGetComponentSegmentIndex` Translates component segment UID into component segment index
  - `tiglWingGetComponentSegmentUID` Translates component segment index into component segment UID
- Fixes:
  - Fixed `tiglWingComponentSegmentPointGetSegmentEtaXsi`
  - Fixed accuracy of `CCPACSWingComponentSegment::getPoint`
  - Fixed accuracy of `CCPACSWingSegment::getEta`
  - Fixed memory management handling of `tiglWingGetProfileName`
  - Fixed memory leaks in VTK export
  - Fixed memory leak in `CCPACSFuselages`
- TIGLViewer:
  - Menus are enabled/disabled depending on number of wings/fuselages
  - Added close configuration menu entry
  - improved view rotation with middle mouse button

Version 2.0.2

Released: 16/10/2012

- General Changes:
  - Fixed wing translation bug in TIGLViewer and export Functions
  - Implemented STL import
  - The result of the fused plane calculation is now chached
- TiGLViewer:
  - Fixed triangulation algorithm (should not crash anymore)
  - Added full plane triangulation
  - Checks in all Selection-Dialogs, if cancel was pressed
  - Added calculation of intersection line of Wing and Fuselage
  - The recently opened folder is now saved

Version 2.0.1

Released: 02/10/2012

- General Changes:
  - Fixed geometry transformations like sweep angle and dihedral angle rotating sections
  - Fixed scalings that lead to a translation of sections
  - Datasets with CPACS Version $<$ 2.0 are now rejected
  - Calculation of fused fuselage, wing, and airplane should not crash anymore
  - Removed memory leaks
  - Cmake based project files
  - Prepared for OpenCascade? 6.5.3
  - Windows 64 Bit builds available
- TiGLViewer:
  - Fixed inconsistent hot keys and menu entries for views
  - Console window showing outputs of TIGL and OpenCascade?
  - Wireframe mode
  - Recent Documents menu entry

Version 2.0

- General Changes:
  - Compatible with CPACS 2.0
  - Corrected the implicit rotation of fuselage and wing profiles. They shouldn't turn from x-y to x-z by TIGL
  - TIGL now uses the coordinate system of parent components for child components. This is only done when parent –> child relations could be figured out via UIDs.
  - Profile points could now also be stored as the vector (x ,y, and z) containing all profile points.
  - Some XPath have changed in CPACS 2.0, for example "sweepangle" is now "sweepAngle". TIGL is taking care of these changes.
- New API Functions:
  - `tiglFuselageGetSymmetry` and `tiglWingGetSymmetry` for querying symmetry information
  - `tiglWingGetReferenceArea` gives the reference area of a wing
  - `tiglWingComponentSegmentFindSegment` returns the segmentUID and wingUID for a given point on a componentSegment
  - `tiglWingGetWettedArea` a new function for caluclating wetted area of a wing
  - `tiglWingComponentSegmentPointGetSegmentEtaXsireturns` eta, xsi, segmentUID and wingUID for a given eta and xsi on a componentSegment
- TiGLViewer:
  - Complete rebuild of TIGLViewer. It is now QT based and platform independent
  - TIGLViewer now updates the view when the cpacs file is changed.

Version 1.0

Released: 31/08/2011

- Changed API:
  - `tiglOpenCpacsConfiguration` now opens a configuration without specification if the uid if it is the only one in the data set. Simply take NULL or en empty string as uid argument.
  - Added a Python wrapper for the C-code and DLL handling (no need to manually convert cpython variables to python)
- New API Functions
  - `tiglComponentIntersectionPoint` Returns a point on the intersection line of two geometric components. Often there are more one intersection line, therefore you need to specify the line.
  - `tiglComponentIntersectionLineCount` Returns the number if intersection lines of two geometric components.
  - `tiglComponentGetHashCode` Computes a hash value to represent a specific shape. The value is computed from the value of the underlying shape reference and the Orientation is not taken into account.
  - `tiglFuselageGetMinumumDistanceToGround` Returns the point where the distance between the selected fuselage and the ground is at minimum. Fuselage could be turned with a given angle at at given axis, specified by a point and a direction.
- TIGLViewer
  - now could draw the componentSegment of a selected wing.

Version 0.9

Released: 29/04/2011

- General Changes:
  - Fixed a bug that leads to strange errors when the first positioning of a Wing is not in the origin and has not innerSectionUID-element.
  - Removed annoying strErr message when a point miss one parameter.
  - TIGL 64-Bit libs are available for linux.
  - TIGL is now able to open rotocraft configurations as well as aircraft

6 Deprecated List

Global tiglComponentIntersectionLineCount (TiglCPACSConfigurationHandle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, int ∗numWires)

This is a deprecated function and will be removed in future releases. Use tiglIntersectGetLineCount in combination with tiglIntersectGetPoint instead.

Global tiglComponentIntersectionPoint (TiglCPACSConfigurationHandle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, int lineID, double eta, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

This is a deprecated function and will be removed in future releases. Use tiglIntersectComponents and tigl↩ IntersectGetPoint instead.

Global tiglComponentIntersectionPoints (TiglCPACSConfigurationHandle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, int lineID, const double ∗etaArray, int number↩ OfPoints, double ∗pointXArray, double ∗pointYArray, double ∗pointZArray)

This is a deprecated function and will be removed in future releases. Use tiglIntersectComponents and tigl↩ IntersectGetPoint instead.

7 Module Index

7.1 Modules

Here is a list of all modules:

8 File Index

8.1 File List

Here is a list of all files with brief descriptions:

9 Module Documentation

9.1 Enumerations

Typedefs

- typedef unsigned int TiglGeometricComponentType

Enumerations

- enum TiglAlgorithmCode { TIGL_INTERPOLATE_LINEAR_WIRE = 0, TIGL_INTERPOLATE_BSPLINE_↩ WIRE = 1, TIGL_APPROXIMATE_BSPLINE_WIRE = 2 }
- enum TiglBoolean { TIGL_FALSE = 0, TIGL_TRUE = 1 }
- enum TiglImportExportFormat { TIGL_IMPORTEXPORT_IGES = 0, TIGL_IMPORTEXPORT_STEP = 1, T↩ IGL_IMPORTEXPORT_STL = 2, TIGL_IMPORTEXPORT_VTK = 3 }
- enum TiglLogLevel {
  TILOG_SILENT =0, TILOG_ERROR =1, TILOG_WARNING =2, TILOG_INFO =3,
  TILOG_DEBUG =4, TILOG_DEBUG1 =5, TILOG_DEBUG2 =6, TILOG_DEBUG3 =7,
  TILOG_DEBUG4 =8 }
- enum TiglReturnCode {
  TIGL_SUCCESS = 0, TIGL_ERROR = 1, TIGL_NULL_POINTER = 2, TIGL_NOT_FOUND = 3,
  TIGL_XML_ERROR = 4, TIGL_OPEN_FAILED = 5, TIGL_CLOSE_FAILED = 6, TIGL_INDEX_ERROR = 7,
  TIGL_STRING_TRUNCATED = 8, TIGL_WRONG_TIXI_VERSION = 9, TIGL_UID_ERROR = 10, TIGL_↩ WRONG_CPACS_VERSION = 11,
  TIGL_UNINITIALIZED = 12, TIGL_MATH_ERROR = 13, TIGL_WRITE_FAILED = 14 }
- enum TiglSymmetryAxis { TIGL_NO_SYMMETRY = 0, TIGL_X_Y_PLANE = 1, TIGL_X_Z_PLANE = 2, TI↩ GL_Y_Z_PLANE = 3 }

9.1.1 Detailed Description

9.1.2 Typedef Documentation

9.1.2.1 TiglGeometricComponentType

```
typedef unsigned int TiglGeometricComponentType
```

Definition of possible types for geometric components. Used for calculations where the type if the component changes the way of behavior.

9.1.3 Enumeration Type Documentation

9.1.3.1 TiglAlgorithmCode

```
enum TiglAlgorithmCode
```

Definition of possible algorithms used in calculations. Use these constants to define the algorithm to be used e.g. in interpolations.

Enumerator

TIGL_INTERPOLATE_LINEAR_WIRE	Use a linear interpolation between the points of a wire
TIGL_INTERPOLATE_BSPLINE_WIRE	Use a BSpline interpolation between the points of a wire
TIGL_APPROXIMATE_BSPLINE_WIRE	Use a BSpline approximation for the points of a wire

9.1.3.2 TiglBoolean

```
enum TiglBoolean
```

Definition of boolean values used in TIGL.

Enumerator

TIGL_FALSE	
TIGL_TRUE	

9.1.3.3 TiglImportExportFormat

```
enum TiglImportExportFormat
```

Definition of the different file formats used for import/export used in TIGL.

Enumerator

TIGL_IMPORTEXPORT_IGES	Use IGES format for geometry import/export
TIGL_IMPORTEXPORT_STEP	Use STEP format for geometry import/export
TIGL_IMPORTEXPORT_STL	Use STL format for geometry import/export
TIGL_IMPORTEXPORT_VTK	Use VTK (XML/VTP) format for geometry import/export

9.1.3.4 TiglLogLevel

```
enum TiglLogLevel
```

Definition of possible logging levels

Enumerator

TILOG_SILENT	No messages are printed. TiGL is completely silent, even in case of errors.
TILOG_ERROR	Only error messages are printed.
TILOG_WARNING	Only errors and warnings are printed on console. This is the default log level of TiGL.
TILOG_INFO	In addition to TILOG_WANING, also informative messages are printed.
TILOG_DEBUG	Also debug messages are printed. Enable this if you want to track down potential errors in TiGL.
TILOG_DEBUG1	This level is only interesting for TiGL developers
TILOG_DEBUG2	This level is only interesting for TiGL developers
TILOG_DEBUG3	This level is only interesting for TiGL developers
TILOG_DEBUG4	This level is only interesting for TiGL developers

9.1.3.5 TiglReturnCode

```
enum TiglReturnCode
```

Definition of possible error return codes of some functions.

Enumerator

TIGL_SUCCESS	
TIGL_ERROR	
TIGL_NULL_POINTER	
TIGL_NOT_FOUND	

Enumerator

TIGL_XML_ERROR	
TIGL_OPEN_FAILED	
TIGL_CLOSE_FAILED	
TIGL_INDEX_ERROR	
TIGL_STRING_TRUNCATED	
TIGL_WRONG_TIXI_VERSION	
TIGL_UID_ERROR	
TIGL_WRONG_CPACS_VERSION	
TIGL_UNINITIALIZED	
TIGL_MATH_ERROR	
TIGL_WRITE_FAILED	

9.1.3.6 TiglSymmetryAxis

```
enum TiglSymmetryAxis
```

Definition of Symmetry Axis used in TIGL.

Enumerator

TIGL_NO_SYMMETRY	
TIGL_X_Y_PLANE	
TIGL_X_Z_PLANE	
TIGL_Y_Z_PLANE	

9.2 General TIGL handling functions

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglCloseCPACSConfiguration (TiglCPACSConfigurationHandle cpacsHandle)

  Closes a CPACS configuration and cleans up all memory used by the configuration. After closing a configuration the associated configuration handle is no longer valid. When the CPACS configuration has been closed, the companion tixi document can also be closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglGetCPACSTixiHandle (TiglCPACSConfigurationHandle cpacsHandle, TixiDocumentHandle ∗tixiHandlePtr)

  Returns the underlying TixiDocumentHandle for a given CPACS configuration handle.
- TIGL_COMMON_EXPORT const char ∗ tiglGetVersion ()

  Returns the version number of this TIGL version.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIsCPACSConfigurationHandleValid (TiglCPACSConfiguration↩ Handle cpacsHandle, TiglBoolean ∗isValidPtr)

  Checks if a given CPACS configuration handle is a valid.
- TIGL_COMMON_EXPORT TiglReturnCode tiglOpenCPACSConfiguration (TixiDocumentHandle tixiHandle, const char ∗configurationUID, TiglCPACSConfigurationHandle ∗cpacsHandlePtr)

  Opens a CPACS configuration and builds up the data and geometry structure in memory.
- TIGL_COMMON_EXPORT TiglReturnCode tiglSaveCPACSConfiguration (const char ∗configurationUID, TiglCPACSConfigurationHandle cpacsHandle)

  Writes a CPACS configuration based on the data and geometry structure in memory.

9.2.1 Detailed Description

Function to open, create, and close CPACS-files.

9.2.2 Function Documentation

9.2.2.1 tiglCloseCPACSConfiguration()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglCloseCPACSConfiguration (
            TiglCPACSConfigurationHandle cpacsHandle )
```

Closes a CPACS configuration and cleans up all memory used by the configuration. After closing a configuration the associated configuration handle is no longer valid. When the CPACS configuration has been closed, the companion tixi document can also be closed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration.

Returns

- TIGL_SUCCESS if successfully opened the CPACS configuration file
- TIGL_CLOSE_FAILED if closing of the CPACS configuration failed
- TIGL_NOT_FOUND if handle ist not found in handle container
- TIGL_ERROR if some other kind of error occurred

9.2.2.2 tiglGetCPACSTixiHandle()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglGetCPACSTixiHandle (
            TiglCPACSConfigurationHandle cpacsHandle,
            TixiDocumentHandle * tixiHandlePtr )
```

Returns the underlying TixiDocumentHandle for a given CPACS configuration handle.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
out	*tixiHandlePtr*	Handle for the TIXI document associated with the CPACS configuration

Returns

- TIGL_SUCCESS if the TIXI handle was found
- TIGL_NOT_FOUND if the TIXI handle was not found
- TIGL_NULL_POINTER if tixiHandlePtr is an invalid null pointer
- TIGL_ERROR if some other kind of error occurred

9.2.2.3 tiglGetVersion()

```
TIGL_COMMON_EXPORT const char* tiglGetVersion ( )
```

Returns the version number of this TIGL version.

Returns

- char∗ A string with the version number.

9.2.2.4 tiglIsCPACSConfigurationHandleValid()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglIsCPACSConfigurationHandleValid (
            TiglCPACSConfigurationHandle cpacsHandle,
            TiglBoolean * isValidPtr )
```

Checks if a given CPACS configuration handle is a valid.

Parameters

in	*cpacsHandle*	CPACS configuration handle to check
out	*isValidPtr*	Contains TIGL_TRUE, if handle is valid, otherwise contains TIGL_FALSE

Returns

- TIGL_SUCCESS if no error occurred

9.2.2.5 tiglOpenCPACSConfiguration()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglOpenCPACSConfiguration (
            TixiDocumentHandle tixiHandle,
            const char * configurationUID,
            TiglCPACSConfigurationHandle * cpacsHandlePtr )
```

Opens a CPACS configuration and builds up the data and geometry structure in memory.

Parameters

in	*tixiHandle*	Handle to a TIXI document. The TIXI document should not be closed until the CPACS configuration is closed. First close the CPACS configuration, then the TIXI document.
in	*configurationUID*	The UID of the configuration that should be loaded by TIGL. Could be NULL or an empty string if the data set contains only one configuration.
out	*cpacsHandlePtr*	Handle to the CPACS configuration. This handle is used in calls to other TIGL functions.

Returns

- TIGL_SUCCESS if successfully opened the CPACS configuration
- TIGL_XML_ERROR if file is not well-formed or another XML error occurred
- TIGL_OPEN_FAILED if some other error occurred
- TIGL_NULL_POINTER if cpacsHandlePtr is an invalid null pointer
- TIGL_ERROR if some other kind of error occurred
- TIGL_INVALID_UID is the UID does not exist or an error orrcured with this configuration

9.2.2.6 tiglSaveCPACSConfiguration()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglSaveCPACSConfiguration (
            const char * configurationUID,
            TiglCPACSConfigurationHandle cpacsHandle )
```

Writes a CPACS configuration based on the data and geometry structure in memory.

Parameters

in	*configurationUID*	The UID of the configuration that should be written.
in	*cpacsHandle*	Handle to the CPACS configuration. This handle is used in calls to other TIGL functions.

Returns

- TIGL_SUCCESS if the CPACS configuration was successfully written
- TIGL_NULL_POINTER if cpacsHandle is an invalid null pointer
- TIGL_UNINITIALIZED if cpacsHandle is not managed by the CCPACSConfigurationManager
- TIGL_ERROR if some other kind of error occurred

9.3 Functions for wing calculations

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglGetWingCount (TiglCPACSConfigurationHandle cpacs↩ Handle, int ∗wingCountPtr)

  Returns the number of wings in a CPACS configuration.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentComputeEtaIntersection (TiglCP↩ ACSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, double csEta1, double csXsi1, double csEta2, double csXsi2, double eta, double ∗xsi, TiglBoolean ∗hasWarning)

  Given a straight line in space defined by a pair of component segment (eta,xsi) coordinates, the function computes the intersection of the line with a component segment iso-eta line.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentFindSegment (TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, double x, double y, double z, char ∗∗segmentUID, char ∗∗wingUID)

  Returns the segmentUID and wingUID for a given point on a componentSegment. The returned strings must not be freed by the user anymore.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetNumberOfSegments (TiglCP↩ ACSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, int ∗nsegments)

  Returns the number of segments belonging to a component segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetPoint (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗componentSegmentUID, double eta, double xsi, double ∗x, double ∗y, double ∗z)

  Returns x,y,z koordinates for a given eta and xsi on a componentSegment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetSegmentIntersection (TiglCP↩ ACSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, const char ∗segmentUID, double csEta1, double csXsi1, double csEta2, double csXsi2, double segmentEta, double ∗segmentXsi, Tigl↩ Boolean ∗hasWarning)

  Computes the intersection of a line (defined by component segment coordinates) with an iso-eta line on a specified wing segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetSegmentUID (TiglCPA↩ CSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, int segmentIndex, char ∗∗segmentUID)

  Returns the segment UID of a segment belonging to a component segment. The segment is specified with its index, which is in the 1...nsegments. The number of segments nsegments can be queried with tiglWingComponent↩ SegmentGetNumberOfSegments.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentPointGetSegmentEtaXsi (TiglCP↩ ACSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, double eta, double xsi, char ∗∗wingUID, char ∗∗segmentUID, double ∗segmentEta, double ∗segmentXsi, double ∗errorDistance)

  Returns eta, xsi, segmentUID and wingUID for a given eta and xsi on a componentSegment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetChordNormal (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double ∗normalXPtr, double ∗normal↩ YPtr, double ∗normalZPtr)

  Returns a normal vector on the wing chord surface for a a given wing and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetChordPoint (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on the wing chord surface for a a given wing and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetComponentSegmentCount (TiglCPACSConfiguration↩ Handle cpacsHandle, int wingIndex, int ∗compSegmentCountPtr)

  Returns the number of component segments for a wing in a CPACS configuration.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetComponentSegmentIndex (TiglCPACSConfiguration↩ Handle cpacsHandle, int wingIndex, const char ∗compSegmentUID, int ∗segmentIndexPtr)

  Returns the Index of a component segment of a wing.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetComponentSegmentUID (TiglCPACSConfiguration←→Handle cpacsHandle, int wingIndex, int compSegmentIndex, char ∗∗uidNamePtr)

    Returns the UID of a component segment of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetIndex (TiglCPACSConfigurationHandle cpacs←→Handle, const char ∗wingUID, int ∗wingIndexPtr)

    Returns the Index of a wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerConnectedSegmentCount (TiglCPACS←→ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int ∗segmentCountPtr)

    Returns the count of wing segments connected to the inner section of a given segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerConnectedSegmentIndex (TiglCPACS←→ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int n, int ∗connectedIndexPtr)

    Returns the index (number) of the n-th wing segment connected to the inner section of a given segment. n starts at 1.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerSectionAndElementIndex (TiglCPACS←→ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int ∗sectionIndexPtr, int ∗element←→IndexPtr)

    Returns the section index and section element index of the inner side of a given wing segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerSectionAndElementUID (TiglCPACS←→ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, char ∗∗sectionUIDPtr, char ∗∗element←→UIDPtr)

    Returns the section UID and section element UID of the inner side of a given wing segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetLowerPoint (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

    Returns a point on the lower wing surface for a a given wing and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetLowerPointAtDirection (TiglCPACSConfiguration←→Handle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double dirx, double diry, double dirz, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr, double ∗errorDistance)

    Returns a point on the lower wing surface for a a given wing and segment index. This function is different from tigl←→WingGetLowerPoint: First, a point on the wing chord surface is computed (defined by segment index and eta,xsi). Then, a line is constructed, which is defined by this point and a direction given by the user. The intersection of this line with the lower wing surface is finally returned. The point is returned in absolute world coordinates.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterConnectedSegmentCount (TiglCPACS←→ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int ∗segmentCountPtr)

    Returns the count of wing segments connected to the outer section of a given segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterConnectedSegmentIndex (TiglCPACS←→ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int n, int ∗connectedIndexPtr)

    Returns the index (number) of the n-th wing segment connected to the outer section of a given segment. n starts at 1.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterSectionAndElementIndex (TiglCPACS←→ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int ∗sectionIndexPtr, int ∗element←→IndexPtr)

    Returns the section index and section element index of the outer side of a given wing segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterSectionAndElementUID (TiglCPACS←→ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, char ∗∗sectionUIDPtr, char ∗∗element←→UIDPtr)

    Returns the section UID and section element UID of the outer side of a given wing segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetProfileName (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int sectionIndex, int elementIndex, char ∗∗profileNamePtr)

    Returns the name of a wing profile. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSectionCount (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int ∗sectionCount)

    Returns the number of sections of a wing.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSectionUID (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int sectionIndex, char ∗∗uidNamePtr)

    Returns the UID of a section of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentCount (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int ∗segmentCountPtr)

    Returns the number of segments for a wing in a CPACS configuration.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentEtaXsi (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, double pointX, double pointY, double pointZ, int ∗segmentIndex, double ∗eta, double ∗xsi, int ∗isOnTop)

    Inverse function to tiglWingGetLowerPoint and tiglWingGetLowerPoint. Calculates to a point (x,y,z) in global coordinates the wing segment coordinates and the wing segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentIndex (TiglCPACSConfigurationHandle cpacsHandle, const char ∗segmentUID, int ∗segmentIndexPtr, int ∗wingIndexPtr)

    Returns the Index of a segment of a wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentUID (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, char ∗∗uidNamePtr)

    Returns the UID of a segment of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSymmetry (TiglCPACSConfigurationHandle cpacs↩Handle, int wingIndex, TiglSymmetryAxis ∗symmetryAxisPtr)

    Returns the Symmetry Enum if the wing has symmetry-axis.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetUID (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, char ∗∗uidNamePtr)

    Returns the UID of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetUpperPoint (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

    Returns a point on the upper wing surface for a a given wing and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetUpperPointAtDirection (TiglCPACSConfiguration↩Handle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double dirx, double diry, double dirz, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr, double ∗errorDistance)

    Returns a point on the upper wing surface for a a given wing and segment index. This function is different from tiglWingGetUpperPoint: First, a point on the wing chord surface is computed (defined by segment index and eta,xsi). Then, a line is constructed, which is defined by this point and a direction given by the user. The intersection of this line with the upper wing surface is finally returned. The point is returned in absolute world coordinates.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingSegmentPointGetComponentSegmentEtaXsi (TiglCP↩ACSConfigurationHandle cpacsHandle, const char ∗segmentUID, const char ∗componentSegmentUID, double segmentEta, double segmentXsi, double ∗eta, double ∗xsi)

    Returns eta, xsi coordinates of a componentSegment given segmentEta and segmentXsi on a wing segment.

9.3.1 Detailed Description

Function to handle wing geometry's with TIGL.

9.3.2 Function Documentation

9.3.2.1 tiglGetWingCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglGetWingCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            int * wingCountPtr )
```

Returns the number of wings in a CPACS configuration.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
out	*wingCountPtr*	Pointer to the number of wings

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if wingCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.2 tiglWingComponentSegmentComputeEtaIntersection()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentComputeEtaIntersection (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentSegmentUID,
            double csEta1,
            double csXsi1,
            double csEta2,
            double csXsi2,
            double eta,
            double * xsi,
            TiglBoolean * hasWarning )
```

Given a straight line in space defined by a pair of component segment (eta,xsi) coordinates, the function computes the intersection of the line with a component segment iso-eta line.

The function is similar to tiglWingComponentSegmentGetSegmentIntersection, with the difference, that an iso-line of the component segment is used instead of an iso-line of a segment. The line is defined by its inner and outer point, both given in component segment coordinates. Typically, these might be spar positions or leading edge coordinates of flaps. The eta value for the iso-eta line should be in the range [csEta1, csEta2]. The function returns the xsi coordinate (depth coordinate) of the intersection point. This coordinate is given in the component segment coordinate system. See image below for details.

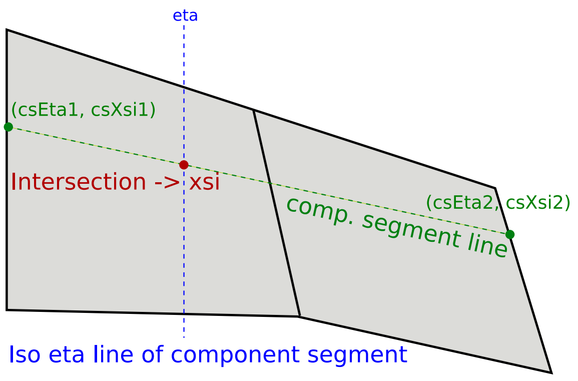


Figure 8 Computation of the component segment interpolation point.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentSegmentUID*	UID of the componentSegment

Parameters

in	*csEta1,csEta2*	Start and end eta coordinates of the intersection line (given as component segment coordinates)
in	*csXsi1,csXsi2*	Start and end xsi coordinates of the intersection line (given as component segment coordinates)
in	*eta*	Eta coordinate of the iso-eta component segment intersection line
out	*xsi*	Xsi coordinate of the intersection point on the wing component segment
out	*hasWarning*	The hasWarning flag is true (1), if the resulting xsi value is either outside the valid range [0,1]. It is up to the user to handle these cases properly. This flag is only valid, if the function returns TIGL_SUCCESS.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if the segment or the component segment does not exist
- TIGL_MATH_ERROR if the intersection could not be computed (e.g. if no intersection exists)
- TIGL_NULL_POINTER if componentSegmentUID or xsi are null pointers
- TIGL_ERROR if some other error occurred

9.3.2.3 tiglWingComponentSegmentFindSegment()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentFindSegment (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentSegmentUID,
            double x,
            double y,
            double z,
            char ** segmentUID,
            char ** wingUID )
```

Returns the segmentUID and wingUID for a given point on a componentSegment. The returned strings must not be freed by the user anymore.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentSegmentUID*	UID of the componentSegment to search for
in	*x,y,z*	Coordinates of the point of the componentSegment
out	*segmentUID*	UID of the segment that fits to the given point and componentSegment. In contrast to old releases, the returned string **must not be freed** by the user!
out	*wingUID*	UID of the wing that fits to the given point and componentSegment In contrast to old releases, the returned string **must not be freed** by the user!

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NOT_FOUND if the point does not lie on the wing component segment within 1cm tolerance.

- TIGL_INDEX_ERROR if wingIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.3.2.4 tiglWingComponentSegmentGetNumberOfSegments()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetNumberOfSegments (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentSegmentUID,
            int * nsegments )
```

Returns the number of segments belonging to a component segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentSegmentUID*	UID of the componentSegment
out	*nsegments*	Number of segments belonging to the component segment

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if the component segment does not exist
- TIGL_ERROR if some other error occurred

9.3.2.5 tiglWingComponentSegmentGetPoint()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetPoint (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentSegmentUID,
            double eta,
            double xsi,
            double * x,
            double * y,
            double * z )
```

Returns x,y,z koordinates for a given eta and xsi on a componentSegment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentSegmentUID*	UID of the componentSegment to search for
in	*eta,xsi*	Eta and Xsi of the point of the componentSegment
out	*x*	X coordinate of the point on the corresponding segment.
out	*y*	Y coordinate of the point on the corresponding segment.
out	*z*	Z coordinate of the point on the corresponding segment.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if the componentSegment does not exist
- TIGL_ERROR if some other error occurred

9.3.2.6 tiglWingComponentSegmentGetSegmentIntersection()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetSegmentIntersection (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentSegmentUID,
            const char * segmentUID,
            double csEta1,
            double csXsi1,
            double csEta2,
            double csXsi2,
            double segmentEta,
            double * segmentXsi,
            TiglBoolean * hasWarning )
```

Computes the intersection of a line (defined by component segment coordinates) with an iso-eta line on a specified wing segment.

The component segment line is defined by its inner and outer point, both defined in component segment coordinates. Typically, these might be spar positions or leading edge coordinates of flaps. The segment line is defined by a iso-eta line. Typically, the intersection with a wing section would be computed (i.e. eta=1 or eta=0). The function returns the xsi coordinate (depth coordinate) of the intersection point. This coordinate is given in the segment coordinate system. See image below for details.

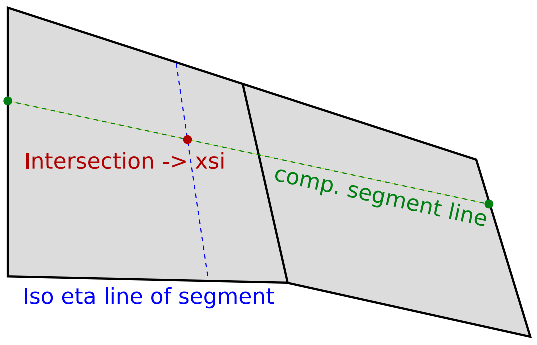


Figure 9 Computation of the component segment - segment intersection point.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentSegmentUID*	UID of the componentSegment
in	*segmentUID*	UID of the segment, the intersection should be calculated with
in	*csEta1,csEta2*	Start and end eta coordinates of the intersection line (given as component segment coordinates)
in	*csXsi1,csXsi2*	Start and end xsi coordinates of the intersection line (given as component segment coordinates)
in	*segmentEta*	Eta coordinate of the iso-eta segment intersection line
out	*segmentXsi*	Xsi coordinate of the intersection point on the wing segment
out	*hasWarning*	The hasWarning flag is true (1), if the resulting xsi value is outside the valid range [0,1]. It is up to the user to handle these cases properly. This flag is only valid, if the function returns TIGL_SUCCESS.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if the segment or the component segment does not exist
- TIGL_MATH_ERROR if the intersection could not be computed (e.g. if no intersection exists)
- TIGL_ERROR if some other error occurred

9.3.2.7 tiglWingComponentSegmentGetSegmentUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetSegmentUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentSegmentUID,
            int segmentIndex,
            char ** segmentUID )
```

Returns the segment UID of a segment belonging to a component segment. The segment is specified with its index, which is in the 1...nsegments. The number of segments nsegments can be queried with tiglWingComponent↵ SegmentGetNumberOfSegments.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentSegmentUID*	UID of the componentSegment
in	*segmentIndex*	Index of the segment (1 $<=$ index $<=$ nsegments)
out	*segmentUID*	UID of the segment

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if the component segment does not exist
- TIGL_INDEX_ERROR if the segment index is invalid
- TIGL_ERROR if some other error occurred

9.3.2.8 tiglWingComponentSegmentPointGetSegmentEtaXsi()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentPointGetSegmentEtaXsi (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentSegmentUID,
            double eta,
            double xsi,
            char ** wingUID,
            char ** segmentUID,
            double * segmentEta,
            double * segmentXsi,
            double * errorDistance )
```

Returns eta, xsi, segmentUID and wingUID for a given eta and xsi on a componentSegment.

If the given component segment point lies outside the wing chord surface, the function returns an error distance $>$ 0. If this distance is larger than 1 mm, the point is first projected onto the segment (see image). Then, this point is transformed into segment coordinates. It is up to the user to handle this case correctly.

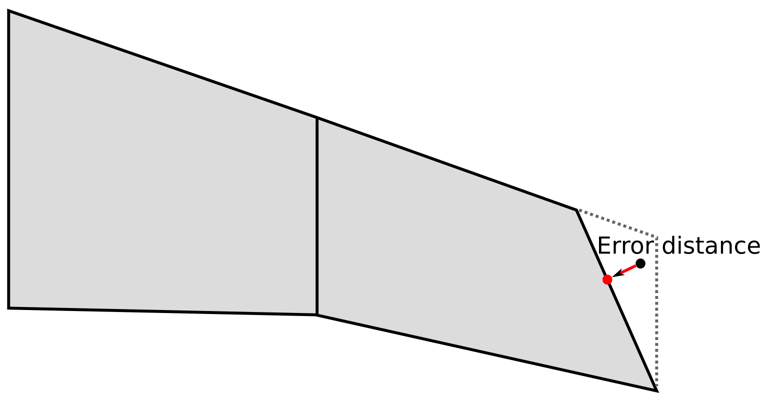


Figure 10 If the given point (black) lies outside the wing chord surface, it will be projected onto the wing (red).

Parameters

`in`	*cpacsHandle*	Handle for the CPACS configuration
`in`	*componentSegmentUID*	UID of the componentSegment to search for
`in`	*eta,xsi*	Eta and Xsi of the point of the componentSegment
`out`	*wingUID*	UID of the wing that fits to the given point and componentSegment. In contrast to old releases, the returned string **must not be freed** by the user!
`out`	*segmentUID*	UID of the segment that fits to the given point and componentSegment. In contrast to old releases, the returned string **must not be freed** by the user!
`out`	*segmentEta*	Eta of the point on the corresponding segment.
`out`	*segmentXsi*	Xsi of the point on the corresponding segment.
`out`	*errorDistance*	If the point given in component segment lies outside the wing chord surface this paramter returns the distance of the point to the nearest point on the wing. Normally, this should be zero! In older releases, an error was generated if this distance was larger than 1 cm. Now, it is up to the user to handle this case.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if the componentSegment does not exist
- TIGL_MATH_ERROR if the given wing component segment point can not be transformed into a segment point. This might happen, if the component segment contains twisted section.
- TIGL_ERROR if some other error occurred

9.3.2.9 tiglWingGetChordNormal()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetChordNormal (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            double eta,
            double xsi,
            double * normalXPtr,
            double * normalYPtr,
            double * normalZPtr )
```

Returns a normal vector on the wing chord surface for a a given wing and segment index.

Returns a normal vector on the wing chord surface in dependence of parameters eta and xsi, which range from 0.0 to 1.0.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of the wing, starting at 1
in	*segmentIndex*	The index of the segment of the wing, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*xsi*	xsi in the range 0.0 <= xsi <= 1.0
out	*normalXPtr*	Pointer to the x-coordinate of the resulting normal vector
out	*normalYPtr*	Pointer to the y-coordinate of the resulting normal vector
out	*normalZPtr*	Pointer to the z-coordinate of the resulting normal vector

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if cpacs handle is not valid
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if normalXPtr, normalYPtr or normalZPtr are null pointers
- TIGL_ERROR if some other error occured

9.3.2.10 tiglWingGetChordPoint()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetChordPoint (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            double eta,
            double xsi,
            double * pointXPtr,
            double * pointYPtr,
            double * pointZPtr )
```

Returns a point on the wing chord surface for a a given wing and segment index.

Returns a point on the wing chord surface in dependence of parameters eta and xsi, which range from 0.0 to 1.0. For eta = 0.0, xsi = 0.0 the point is equal to the leading edge on the inner section of the given segment. For eta = 1.0, xsi = 1.0 the point is equal to the trailing edge on the outer section of the given segment. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of the wing, starting at 1
in	*segmentIndex*	The index of the segment of the wing, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*xsi*	xsi in the range 0.0 <= xsi <= 1.0
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if cpacs handle is not valid
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occured

9.3.2.11 tiglWingGetComponentSegmentCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetComponentSegmentCount (
             TiglCPACSConfigurationHandle cpacsHandle,
             int wingIndex,
             int * compSegmentCountPtr )
```

Returns the number of component segments for a wing in a CPACS configuration.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of the wing, starting at 1
out	*compSegmentCountPtr*	Pointer to the number of component segments

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex is not valid
- TIGL_NULL_POINTER if compSegmentCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.12 tiglWingGetComponentSegmentIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetComponentSegmentIndex (
             TiglCPACSConfigurationHandle cpacsHandle,
             int wingIndex,
             const char * compSegmentUID,
             int * segmentIndexPtr )
```

Returns the Index of a component segment of a wing.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*compSegmentUID*	The uid of the wing
out	*segmentIndexPtr*	The index of a segment, starting at 1

Usage example:

```
TiglReturnCode returnCode;
int segmentIndex;
returnCode = tiglWingGetComponentSegmentIndex(cpacsHandle, wing, uidName, &segmentIndex);
printf("The Index of the component segment of wing %d is %d\n", wing, segmentIndex);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex is not valid
- TIGL_UID_ERROR if the compSegmentUID does not exist
- TIGL_NULL_POINTER if compSegmentUID is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.13 tiglWingGetComponentSegmentUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetComponentSegmentUID (
             TiglCPACSConfigurationHandle cpacsHandle,
             int wingIndex,
             int compSegmentIndex,
             char ** uidNamePtr )
```

Returns the UID of a component segment of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*compSegmentIndex*	The index of a segment, starting at 1
out	*uidNamePtr*	The uid of the wing

Usage example:

```
TiglReturnCode returnCode;
char* uidPtr = 0;
returnCode = tiglWingGetComponentSegmentUID(cpacsHandle, wing, segmentID, &uidPtr);
printf("The UID of the component segment of wing %d is %s\n", wing, uidPtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex or the compSegmentIndex are not valid
- TIGL_NULL_POINTER if uidNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.14 tiglWingGetIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetIndex (
             TiglCPACSConfigurationHandle cpacsHandle,
             const char * wingUID,
             int * wingIndexPtr )
```

Returns the Index of a wing.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingUID*	The uid of the wing
out	*wingIndexPtr*	The index of a wing, starting at 1

Usage example:

```
TiglReturnCode returnCode;
int wingIndex;
returnCode = tiglWingGetUID(cpacsHandle, wingUID, &wingIndex);
printf("The Index of the wing is %d\n", wingIndex);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if wingUID does not exist
- TIGL_NULL_POINTER if wingUID is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.15 tiglWingGetInnerConnectedSegmentCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerConnectedSegmentCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            int * segmentCountPtr )
```

Returns the count of wing segments connected to the inner section of a given segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*segmentCountPtr*	Pointer to the count of connected segments

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if segmentCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.16 tiglWingGetInnerConnectedSegmentIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerConnectedSegmentIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            int n,
            int * connectedIndexPtr )
```

Returns the index (number) of the n-th wing segment connected to the inner section of a given segment. n starts at 1.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
in	*n*	n-th segment searched, 1 <= n <= tiglWingGetInnerConnectedSegmentCount(...)
out	*connectedIndexPtr*	Pointer to the segment index of the n-th connected segment

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle or no n-th connected segment was found
- TIGL_INDEX_ERROR if wingIndex, segmentIndex or n are not valid
- TIGL_NULL_POINTER if segmentIndexPtr is a null pointer
- TIGL_ERROR if some other error occured

9.3.2.17 tiglWingGetInnerSectionAndElementIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerSectionAndElementIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            int * sectionIndexPtr,
            int * elementIndexPtr )
```

Returns the section index and section element index of the inner side of a given wing segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*sectionIndexPtr*	The section index of the inner side
out	*elementIndexPtr*	The section element index of the inner side

Returns

- TIGL_SUCCESS if no error occurred

- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if sectionIndexPtr or elementIndexPtr are a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.18 tiglWingGetInnerSectionAndElementUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerSectionAndElementUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            char ** sectionUIDPtr,
            char ** elementUIDPtr )
```

Returns the section UID and section element UID of the inner side of a given wing segment.

Important change: The memory necessary for the two UIDs must not be freed by the user anymore.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*sectionUIDPtr*	The section UID of the inner side
out	*elementUIDPtr*	The section element UID of the inner side

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if sectionIndexPtr or elementIndexPtr are a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.19 tiglWingGetLowerPoint()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetLowerPoint (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            double eta,
            double xsi,
            double * pointXPtr,
            double * pointYPtr,
            double * pointZPtr )
```

Returns a point on the lower wing surface for a a given wing and segment index.

Returns a point on the lower wing surface in dependence of parameters eta and xsi, which range from 0.0 to 1.0. For eta = 0.0, xsi = 0.0 the point is equal to the leading edge on the inner section of the given segment. For eta = 1.0, xsi = 1.0 the point is equal to the trailing edge on the outer section of the given segment. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of the wing, starting at 1
in	*segmentIndex*	The index of the segment of the wing, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*xsi*	xsi in the range 0.0 <= xsi <= 1.0
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no intersection point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occured

9.3.2.20 tiglWingGetLowerPointAtDirection()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetLowerPointAtDirection (
             TiglCPACSConfigurationHandle cpacsHandle,
             int wingIndex,
             int segmentIndex,
             double eta,
             double xsi,
             double dirx,
             double diry,
             double dirz,
             double * pointXPtr,
             double * pointYPtr,
             double * pointZPtr,
             double * errorDistance )
```

Returns a point on the lower wing surface for a a given wing and segment index. This function is different from tigl↩WingGetLowerPoint: First, a point on the wing chord surface is computed (defined by segment index and eta,xsi). Then, a line is constructed, which is defined by this point and a direction given by the user. The intersection of this line with the lower wing surface is finally returned. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of the wing, starting at 1
in	*segmentIndex*	The index of the segment of the wing, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0; eta = 0 for inner section , eta = 1 for outer section
in	*xsi*	xsi in the range 0.0 <= xsi <= 1.0; xsi = 0 for Leading Edge, xsi = 1 for Trailing Edge
in	*dirx*	X-component of the direction vector.
in	*diry*	Y-component of the direction vector.
in	*dirz*	Z-component of the direction vector.
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates

Parameters

out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates
out	*errorDistance*	If the lower surface is missed by the line, the absolute distance between line and the nearest point on the surface is returned. The distance is zero in case of successful intersection. **It's up to the user to decide, if the distance is too large and the result has to be treated as an error.**

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.3.2.21 tiglWingGetOuterConnectedSegmentCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterConnectedSegmentCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            int * segmentCountPtr )
```

Returns the count of wing segments connected to the outer section of a given segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*segmentCountPtr*	Pointer to the count of connected segments

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if segmentCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.22 tiglWingGetOuterConnectedSegmentIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterConnectedSegmentIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            int n,
            int * connectedIndexPtr )
```

Returns the index (number) of the n-th wing segment connected to the outer section of a given segment. n starts at 1.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
in	*n*	n-th segment searched, 1 <= n <= tiglWingGetOuterConnectedSegmentCount(...)
out	*connectedIndexPtr*	Pointer to the segment index of the n-th connected segment

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle or no n-th connected segment was found
- TIGL_INDEX_ERROR if wingIndex, segmentIndex or n are not valid
- TIGL_NULL_POINTER if segmentIndexPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.23 tiglWingGetOuterSectionAndElementIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterSectionAndElementIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            int * sectionIndexPtr,
            int * elementIndexPtr )
```

Returns the section index and section element index of the outer side of a given wing segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*sectionIndexPtr*	The section index of the outer side
out	*elementIndexPtr*	The section element index of the outer side

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if sectionIndexPtr or elementIndexPtr are a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.24 tiglWingGetOuterSectionAndElementUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterSectionAndElementUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            char ** sectionUIDPtr,
            char ** elementUIDPtr )
```

Returns the section UID and section element UID of the outer side of a given wing segment.

Important change: The memory necessary for the two UIDs must not be freed by the user anymore.

Parameters

`in`	*cpacsHandle*	Handle for the CPACS configuration
`in`	*wingIndex*	The index of a wing, starting at 1
`in`	*segmentIndex*	The index of a segment, starting at 1
`out`	*sectionUIDPtr*	The section UID of the outer side
`out`	*elementUIDPtr*	The section element UID of the outer side

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if sectionIndexPtr or elementIndexPtr are a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.25 tiglWingGetProfileName()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetProfileName (
             TiglCPACSConfigurationHandle cpacsHandle,
             int wingIndex,
             int sectionIndex,
             int elementIndex,
             char ** profileNamePtr )
```

Returns the name of a wing profile. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Parameters

`in`	*cpacsHandle*	Handle for the CPACS configuration
`in`	*wingIndex*	The index of a wing, starting at 1
`in`	*sectionIndex*	The index of a section, starting at 1
`in`	*elementIndex*	The index of an element on the section
`out`	*profileNamePtr*	The name of the wing profile

Usage example:

```
TiglReturnCode returnCode;
char* namePtr = 0;
returnCode = tiglWingGetProfileName(cpacsHandle, wing, section, element, &namePtr);
printf("Profile name is %s\n", namePtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if profileNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.26 tiglWingGetSectionCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSectionCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int * sectionCount )
```

Returns the number of sections of a wing.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
out	*sectionCount*	The number of sections of the wing

Usage example:

```
TiglReturnCode returnCode;
int sectionCount = 0;
returnCode = tiglWingGetSectionUID(cpacsHandle, wingIndex, &sectionCount);
printf("The Number of sections of wing %d is %d\n", wingIndex, sectionCount);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex is not valid
- TIGL_NULL_POINTER if sectionCount is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.27 tiglWingGetSectionUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSectionUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int sectionIndex,
            char ** uidNamePtr )
```

Returns the UID of a section of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*sectionIndex*	The index of a section, starting at 1
out	*uidNamePtr*	The uid of the wing

Usage example:

```
TiglReturnCode returnCode;
char* uidPtr = 0;
returnCode = tiglWingGetSectionUID(cpacsHandle, wing, sectionUID, &uidPtr);
printf("The UID of the section of wing %d is %s\n", wing, uidPtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if profileNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.28 tiglWingGetSegmentCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int * segmentCountPtr )
```

Returns the number of segments for a wing in a CPACS configuration.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of the wing, starting at 1
out	*segmentCountPtr*	Pointer to the number of segments

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex is not valid
- TIGL_NULL_POINTER if segmentCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.29 tiglWingGetSegmentEtaXsi()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentEtaXsi (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            double pointX,
            double pointY,
            double pointZ,
            int * segmentIndex,
            double * eta,
            double * xsi,
            int * isOnTop )
```

Inverse function to tiglWingGetLowerPoint and tiglWingGetLowerPoint. Calculates to a point (x,y,z) in global coordinates the wing segment coordinates and the wing segment index.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of the wing, starting at 1
in	*pointX*	X-Coordinate of the global point
in	*pointY*	Y-Coordinate of the global point
in	*pointZ*	Z-Coordinate of the global point
out	*segmentIndex*	The index of the segment of the wing, starting at 1
out	*eta*	Eta value in segment coordinates
out	*xsi*	Xsi value in segment coordinates
out	*isOnTop*	isOnTop is 1, if the point lies on the upper wing face, else 0.

Returns

- TIGL_SUCCESS if a solution was found
- TIGL_NOT_FOUND if the point does not lie on the wing
- TIGL_INDEX_ERROR if wingIndex is not valid
- TIGL_NULL_POINTER if eta, xsi or isOnTop are null pointers
- TIGL_ERROR if some other error occurred

9.3.2.30 tiglWingGetSegmentIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * segmentUID,
            int * segmentIndexPtr,
            int * wingIndexPtr )
```

Returns the Index of a segment of a wing.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*segmentUID*	The uid of the wing
out	*segmentIndexPtr*	The index of a segment, starting at 1
out	*wingIndexPtr*	The index of a wing, starting at 1

Usage example:

```
TiglReturnCode returnCode;
int segmentIndex, wingIndex;
returnCode = tiglWingGetSegmentIndex(cpacsHandle, segmentUID, &segmentIndex, &wingIndex);
printf("The Index of the segment of wing %d is %d\n", wingIndex, segmentIndex);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex is not valid
- TIGL_UID_ERROR if the segmentUID does not exist
- TIGL_NULL_POINTER if segmentUID is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.31 tiglWingGetSegmentUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            char ** uidNamePtr )
```

Returns the UID of a segment of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*uidNamePtr*	The uid of the wing

Usage example:

```
TiglReturnCode returnCode;
char* uidPtr = 0;
returnCode = tiglWingGetSegmentUID(cpacsHandle, wing, segmentID, &uidPtr);
printf("The UID of the segment of wing %d is %s\n", wing, uidPtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if profileNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.32 tiglWingGetSymmetry()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSymmetry (
             TiglCPACSConfigurationHandle cpacsHandle,
             int wingIndex,
             TiglSymmetryAxis * symmetryAxisPtr )
```

Returns the Symmetry Enum if the wing has symmetry-axis.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	Index of the Wing to export
out	*symmetryAxisPtr*	Returning TiglSymmetryAxis enum pointer

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.3.2.33 tiglWingGetUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            char ** uidNamePtr )
```

Returns the UID of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
out	*uidNamePtr*	The uid of the wing

Usage example:

```
TiglReturnCode returnCode;
char* uidPtr = 0;
returnCode = tiglWingGetUID(cpacsHandle, wing, &uidPtr);
printf("The UID of the wing is %s\n", uidPtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if profileNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.3.2.34 tiglWingGetUpperPoint()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetUpperPoint (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            double eta,
            double xsi,
            double * pointXPtr,
            double * pointYPtr,
            double * pointZPtr )
```

Returns a point on the upper wing surface for a a given wing and segment index.

Returns a point on the upper wing surface in dependence of parameters eta and xsi, which range from 0.0 to 1.0. For eta = 0.0, xsi = 0.0 the point is equal to the leading edge on the inner section of the given segment. For eta = 1.0, xsi = 1.0 the point is equal to the trailing edge on the outer section of the given segment. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of the wing, starting at 1
in	*segmentIndex*	The index of the segment of the wing, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0; eta = 0 for inner section , eta = 1 for outer section
in	*xsi*	xsi in the range 0.0 <= xsi <= 1.0; xsi = 0 for Leading Edge, xsi = 1 for Trailing Edge
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.3.2.35 tiglWingGetUpperPointAtDirection()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetUpperPointAtDirection (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            double eta,
            double xsi,
            double dirx,
            double diry,
            double dirz,
            double * pointXPtr,
            double * pointYPtr,
            double * pointZPtr,
            double * errorDistance )
```

Returns a point on the upper wing surface for a a given wing and segment index. This function is different from tigl↵ WingGetUpperPoint: First, a point on the wing chord surface is computed (defined by segment index and eta,xsi). Then, a line is constructed, which is defined by this point and a direction given by the user. The intersection of this line with the upper wing surface is finally returned. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of the wing, starting at 1
in	*segmentIndex*	The index of the segment of the wing, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0; eta = 0 for inner section , eta = 1 for outer section
in	*xsi*	xsi in the range 0.0 <= xsi <= 1.0; xsi = 0 for Leading Edge, xsi = 1 for Trailing Edge
in	*dirx*	X-component of the direction vector.
in	*diry*	Y-component of the direction vector.
in	*dirz*	Z-component of the direction vector.
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates
out	*errorDistance*	If the upper surface is missed by the line, the absolute distance between line and the nearest point on the surface is returned. The distance is zero in case of successful intersection. **It's up to the user to decide, if the distance is too large and the result has to be treated as an error.**

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no intersection point was found or the cpacs handle is not valid
- TIGL_MATH_ERROR if the given direction is a null vector (which has zero length)
- TIGL_INDEX_ERROR if wingIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.3.2.36 tiglWingSegmentPointGetComponentSegmentEtaXsi()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingSegmentPointGetComponentSegmentEtaXsi (
             TiglCPACSConfigurationHandle cpacsHandle,
             const char * segmentUID,
             const char * componentSegmentUID,
             double segmentEta,
             double segmentXsi,
             double * eta,
             double * xsi )
```

Returns eta, xsi coordinates of a componentSegment given segmentEta and segmentXsi on a wing segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*segmentUID*	UID of the wing segment to search for
in	*componentSegmentUID*	UID of the associated componentSegment
in	*segmentEta,segmentXsi*	Eta and Xsi coordinates of the point on the wing segment
out	*eta*	Eta of the point on the corresponding component segment.
out	*xsi*	Xsi of the point on the corresponding component segment.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if the segment or the component segment does not exist
- TIGL_ERROR if some other error occurred

9.4 Functions for fuselage calculations

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetCircumference (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double ∗circumferencePtr)

  Returns the circumference of a fuselage surface for a given fuselage and segment index and an eta.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndConnectedSegmentCount (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int ∗segmentCountPtr)

  Returns the count of segments connected to the end section of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndConnectedSegmentIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int n, int ∗connectedIndexPtr)

  Returns the index (number) of the n-th segment connected to the end section of a given fuselage segment. n starts at 1.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndSectionAndElementIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int ∗sectionIndexPtr, int ∗element↩ IndexPtr)

  Returns the section index and section element index of the end side of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndSectionAndElementUID (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, char ∗∗sectionUIDPtr, char ∗∗elementUIDPtr)

  Returns the section UID and section element UID of the end side of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetIndex (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗fuselageUID, int ∗fuselageIndexPtr)

  Returns the index of a fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetMinumumDistanceToGround (TiglCPACS↩ ConfigurationHandle cpacsHandle, char ∗fuselageUID, double axisPntX, double axisPntY, double axisPntZ, double axisDirX, double axisDirY, double axisDirZ, double angle, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns the point where the distance between the selected fuselage and the ground is at minimum. The Fuselage could be turned with a given angle at at given axis, specified by a point and a direction.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetNumPointsOnXPlane (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double xpos, int ∗numPointsPtr)

  Returns the number of points on a fuselage surface for a given fuselage and a give x-position.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetNumPointsOnYPlane (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double ypos, int ∗numPointsPtr)

  Returns the number of points on a fuselage surface for a given fuselage and a give y-position.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPoint (TiglCPACSConfigurationHandle cpacs↩ Handle, int fuselageIndex, int segmentIndex, double eta, double zeta, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on a fuselage surface for a given fuselage and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointAngle (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double alpha, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on a fuselage surface for a given fuselage and segment index and an angle alpha (degree).
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointAngleTranslated (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double alpha, double y_cs, double z_cs, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on a fuselage surface for a given fuselage and segment index and an angle alpha (degree). 0 degree of the angle alpha is meant to be "up" in the direction of the positive z-axis like specifies in cpacs. The origin of the line that will be rotated with the angle alpha could be translated via the parameters y_cs and z_cs.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointOnXPlane (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double xpos, int pointIndex, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

Returns a point on a fuselage surface for a given fuselage and segment index.

- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointOnYPlane (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double ypos, int pointIndex, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on a fuselage surface for a given fuselage and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetProfileName (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int sectionIndex, int elementIndex, char ∗∗profileNamePtr)

  Returns the name of a fuselage profile. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSectionCount (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int ∗sectionCount)

  Returns the number of sections of a fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSectionUID (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int sectionIndex, char ∗∗uidNamePtr)

  Returns the UID of a section of a fuselage. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentCount (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, int ∗segmentCountPtr)

  Returns the number of segments for a fuselage in a CPACS configuration.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentIndex (TiglCPACSConfigurationHandle cpacsHandle, const char ∗segmentUID, int ∗segmentIndexPtr, int ∗fuselageIndexPtr)

  Returns the Index of a segment of a fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentUID (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, char ∗∗uidNamePtr)

  Returns the UID of a segment of a fuselage. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartConnectedSegmentCount (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int ∗segmentCountPtr)

  Returns the count of segments connected to the start section of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartConnectedSegmentIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int n, int ∗connectedIndexPtr)

  Returns the index (number) of the n-th segment connected to the start section of a given fuselage segment. n starts at 1.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartSectionAndElementIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int ∗sectionIndexPtr, int ∗element↩ IndexPtr)

  Returns the section index and section element index of the start side of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartSectionAndElementUID (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, char ∗∗sectionUIDPtr, char ∗∗elementUIDPtr)

  Returns the section UID and section element UID of the start side of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSymmetry (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, TiglSymmetryAxis ∗symmetryAxisPtr)

  Returns the Symmetry Enum if the fuselage has symmetry-axis.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetUID (TiglCPACSConfigurationHandle cpacs↩ Handle, int fuselageIndex, char ∗∗uidNamePtr)

  Returns the UID of a fuselage. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglGetFuselageCount (TiglCPACSConfigurationHandle cpacs↩ Handle, int ∗fuselageCountPtr)

  Returns the number of fuselages in a CPACS configuration.

9.4.1 Detailed Description

Function to handle fuselage geometry's with TIGL.

9.4.2 Function Documentation

9.4.2.1 tiglFuselageGetCircumference()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetCircumference (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int segmentIndex,
             double eta,
             double ∗ circumferencePtr )
```

Returns the circumference of a fuselage surface for a given fuselage and segment index and an eta.

Returns the circumference of a fuselage segment of a given fuselage in dependence of parameters eta with 0.0 <= eta <= 1.0. For eta = 0.0 the point lies on the start profile of the segment, for eta = 1.0 it lies on the end profile of the segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of the fuselage, starting at 1
in	*segmentIndex*	The index of the segment of the fuselage, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0
out	*circumferencePtr*	The Circumference of the fuselage at the given position

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.4.2.2 tiglFuselageGetEndConnectedSegmentCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndConnectedSegmentCount (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int segmentIndex,
             int ∗ segmentCountPtr )
```

Returns the count of segments connected to the end section of a given fuselage segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*segmentCountPtr*	Pointer to the count of connected segments

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if segmentCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.3 tiglFuselageGetEndConnectedSegmentIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndConnectedSegmentIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            int n,
            int * connectedIndexPtr )
```

Returns the index (number) of the n-th segment connected to the end section of a given fuselage segment. n starts at 1.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
in	*n*	n-th segment searched, 1 <= n <= tiglFuselageGetEndConnectedSegmentCount(...)
out	*connectedIndexPtr*	Pointer to the segment index of the n-th connected segment

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle or no n-th connected segment was found
- TIGL_INDEX_ERROR if fuselageIndex, segmentIndex or n are not valid
- TIGL_NULL_POINTER if segmentIndexPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.4 tiglFuselageGetEndSectionAndElementIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndSectionAndElementIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            int * sectionIndexPtr,
            int * elementIndexPtr )
```

Returns the section index and section element index of the end side of a given fuselage segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*sectionIndexPtr*	The section index UID the end side
out	*elementIndexPtr*	The section element UID of the end side

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if sectionIndexPtr or elementIndexPtr are a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.5 tiglFuselageGetEndSectionAndElementUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndSectionAndElementUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            char ** sectionUIDPtr,
            char ** elementUIDPtr )
```

Returns the section UID and section element UID of the end side of a given fuselage segment.

Important change: The memory necessary for the two UIDs must not to be freed by the user anymore.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*sectionUIDPtr*	The section UID the end side
out	*elementUIDPtr*	The section element UID of the end side

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_ERROR if some other error occurred

9.4.2.6 tiglFuselageGetIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * fuselageUID,
            int * fuselageIndexPtr )
```

Returns the index of a fuselage.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageUID*	The uid of the fuselage
out	*fuselageIndexPtr*	The index of a fuselage, starting at 1

Usage example:

```
TiglReturnCode returnCode;
int fuselageIndex;
returnCode = tiglFuselageGetIndex(cpacsHandle, fuselageUID, &fuselageIndex);
printf("The Index of the fuselage is %d\n", fuselageIndex);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if fuselageUID does not exist
- TIGL_NULL_POINTER if fuselageUID is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.7 tiglFuselageGetMinumumDistanceToGround()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetMinumumDistanceToGround (
             TiglCPACSConfigurationHandle cpacsHandle,
             char * fuselageUID,
             double axisPntX,
             double axisPntY,
             double axisPntZ,
             double axisDirX,
             double axisDirY,
             double axisDirZ,
             double angle,
             double * pointXPtr,
             double * pointYPtr,
             double * pointZPtr )
```

Returns the point where the distance between the selected fuselage and the ground is at minimum. The Fuselage could be turned with a given angle at at given axis, specified by a point and a direction.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageUID*	The uid of the fuselage
in	*axisPntX*	X-coordinate of the point that specifies the axis of rotation
in	*axisPntY*	Y-coordinate of the point that specifies the axis of rotation
in	*axisPntZ*	Z-coordinate of the point that specifies the axis of rotation
in	*axisDirX*	X-coordinate of the direction that specifies the axis of rotation
in	*axisDirY*	Y-coordinate of the direction that specifies the axis of rotation
in	*axisDirZ*	Z-coordinate of the direction that specifies the axis of rotation
in	*angle*	The angle (in Degree) by which the fuselage should be turned on the axis of rotation
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageUID is not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.4.2.8 tiglFuselageGetNumPointsOnXPlane()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetNumPointsOnXPlane (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int segmentIndex,
             double eta,
             double xpos,
             int * numPointsPtr )
```

Returns the number of points on a fuselage surface for a given fuselage and a give x-position.

Returns the number of points on a fuselage segment of a given fuselage in dependence of parameters eta and at all x-positions with 0.0 <= eta <= 1.0. For eta = 0.0 the point lies on the start profile of the segment, for eta = 1.0 it lies on the end profile of the segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of the fuselage, starting at 1
in	*segmentIndex*	The index of the segment of the fuselage, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*xpos*	X position
out	*numPointsPtr*	Pointer to a integer for the number of intersection points

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.4.2.9 tiglFuselageGetNumPointsOnYPlane()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetNumPointsOnYPlane (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int segmentIndex,
             double eta,
             double ypos,
             int * numPointsPtr )
```

Returns the number of points on a fuselage surface for a given fuselage and a give y-position.

Returns the number of points on a fuselage segment of a given fuselage in dependence of parameters eta and at all y-positions with 0.0 <= eta <= 1.0. For eta = 0.0 the point lies on the start profile of the segment, for eta = 1.0 it lies on the end profile of the segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of the fuselage, starting at 1
in	*segmentIndex*	The index of the segment of the fuselage, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*ypos*	Y position
out	*numPointsPtr*	Pointer to a interger for the number of intersection points

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.4.2.10 tiglFuselageGetPoint()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPoint (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            double eta,
            double zeta,
            double * pointXPtr,
            double * pointYPtr,
            double * pointZPtr )
```

Returns a point on a fuselage surface for a given fuselage and segment index.

Returns a point on a fuselage segment of a given fuselage in dependence of parameters eta and zeta with 0.0 <= eta <= 1.0 and 0.0 <= zeta <= 1.0. For eta = 0.0 the point lies on the start profile of the segment, for eta = 1.0 it lies on the end profile of the segment. For zeta = 0.0 the point is the identical to the start point of the profile wire, for zeta = 1.0 it is identical to the last profile point. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of the fuselage, starting at 1
in	*segmentIndex*	The index of the segment of the fuselage, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*zeta*	zeta in the range 0.0 <= zeta <= 1.0
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.4.2.11 tiglFuselageGetPointAngle()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointAngle (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int segmentIndex,
             double eta,
             double alpha,
             double * pointXPtr,
             double * pointYPtr,
             double * pointZPtr )
```

Returns a point on a fuselage surface for a given fuselage and segment index and an angle alpha (degree).

Returns a point on a fuselage segment of a given fuselage in dependence of parameters eta and at all y-positions with 0.0 <= eta <= 1.0. For eta = 0.0 the point lies on the start profile of the segment, for eta = 1.0 it lies on the end profile of the segment. The angle alpha is calculated in degrees. Alpha=0 degree is meant to be "up" in the direction of the positive z-axis like specifies in cpacs. It's orientation is the mathematical negative rotation direction around the X-axis, i.e. looking in flight direction, an angle of 45 degrees resembles a point on the top-left fuselage. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of the fuselage, starting at 1
in	*segmentIndex*	The index of the segment of the fuselage, starting at 1
in	*eta*	Eta in the range 0.0 <= eta <= 1.0
in	*alpha*	Angle alpha in degrees. No range restrictions.
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred, for example if there is no point at the given eta.

9.4.2.12 tiglFuselageGetPointAngleTranslated()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointAngleTranslated (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int segmentIndex,
             double eta,
             double alpha,
             double y_cs,
             double z_cs,
             double * pointXPtr,
             double * pointYPtr,
             double * pointZPtr )
```

Returns a point on a fuselage surface for a given fuselage and segment index and an angle alpha (degree). 0 degree of the angle alpha is meant to be "up" in the direction of the positive z-axis like specifies in cpacs. The origin of the line that will be rotated with the angle alpha could be translated via the parameters y_cs and z_cs.

Returns a point on a fuselage segment of a given fuselage in dependence of parameters eta and at all y-positions with 0.0 <= eta <= 1.0. For eta = 0.0 the point lies on the start profile of the segment, for eta = 1.0 it lies on the end profile of the segment. The angle alpha is calculated in degrees. It's orientation is the mathematical negative rotation direction around the X-axis, i.e. looking in flight direction, an angle of 45 degrees resembles a point on the top-left fuselage. The parameters y_cs and z_cs must be in absolute world coordinates. The point is returned in absolute world coordinates.

Parameters

`in`	*cpacsHandle*	Handle for the CPACS configuration
`in`	*fuselageIndex*	The index of the fuselage, starting at 1
`in`	*segmentIndex*	The index of the segment of the fuselage, starting at 1
`in`	*eta*	eta in the range 0.0 <= eta <= 1.0
`in`	*alpha*	Angle alpha in degrees. No range restrictions.
`in`	*y_cs*	Shifts the origin of the angle alpha in y-direction.
`in`	*z_cs*	Shifts the origin of the angle alpha in z-direction.
`out`	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
`out`	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
`out`	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred, for example if there is no point at the given eta and the given shifting parameters y_cs and z_cs.

9.4.2.13 tiglFuselageGetPointOnXPlane()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointOnXPlane (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            double eta,
            double xpos,
            int pointIndex,
            double * pointXPtr,
            double * pointYPtr,
            double * pointZPtr )
```

Returns a point on a fuselage surface for a given fuselage and segment index.

Returns a point on a fuselage segment of a given fuselage in dependence of parameters eta and at all y-positions with 0.0 <= eta <= 1.0. For eta = 0.0 the point lies on the start profile of the segment, for eta = 1.0 it lies on the end profile of the segment. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of the fuselage, starting at 1
in	*segmentIndex*	The index of the segment of the fuselage, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*xpos*	x position of a cutting plane
in	*pointIndex*	Defines witch point if more than one.
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.4.2.14 tiglFuselageGetPointOnYPlane()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointOnYPlane (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            double eta,
            double ypos,
            int pointIndex,
            double * pointXPtr,
            double * pointYPtr,
            double * pointZPtr )
```

Returns a point on a fuselage surface for a given fuselage and segment index.

Returns a point on a fuselage segment of a given fuselage in dependence of parameters eta and at all y-positions with 0.0 <= eta <= 1.0. For eta = 0.0 the point lies on the start profile of the segment, for eta = 1.0 it lies on the end profile of the segment. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of the fuselage, starting at 1

Parameters

in	*segmentIndex*	The index of the segment of the fuselage, starting at 1
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*ypos*	Y position
in	*pointIndex*	Defines witch point if more than one.
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if a point was found
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.4.2.15 tiglFuselageGetProfileName()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetProfileName (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int sectionIndex,
             int elementIndex,
             char ** profileNamePtr )
```

Returns the name of a fuselage profile. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*sectionIndex*	The index of a section, starting at 1
in	*elementIndex*	The index of an element on the section
out	*profileNamePtr*	The name of the wing profile

Usage example:

```
TiglReturnCode returnCode;
char* namePtr = 0;
returnCode = tiglFuselageGetProfileName(cpacsHandle, fuselage, section, element, &namePtr);
printf("Profile name is %s\n", namePtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if profileNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.16 tiglFuselageGetSectionCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSectionCount (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int * sectionCount )
```

Returns the number of sections of a fuselage.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
out	*sectionCount*	The number of sections of the fuselage

Usage example:

```
   TiglReturnCode returnCode;
   int sectionCount = 0;
   returnCode = tiglFuselageGetSectionUID(cpacsHandle, fuselageIndex, &sectionCount);
   printf("The Number of sections of fuselage %d is %d\n", fuselageIndex, sectionCount);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex is not valid
- TIGL_NULL_POINTER if sectionCount is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.17 tiglFuselageGetSectionUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSectionUID (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int sectionIndex,
             char ** uidNamePtr )
```

Returns the UID of a section of a fuselage. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*sectionIndex*	The index of a section, starting at 1
out	*uidNamePtr*	The uid of the fuselage

Usage example:

```
TiglReturnCode returnCode;
char* uidPtr = 0;
returnCode = tiglFuselageGetSectionUID(cpacsHandle, fuselage, sectionUID, &uidPtr);
printf("The UID of the section of fuselage %d is %s\n", fuselage, uidPtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if profileNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.18 tiglFuselageGetSegmentCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int * segmentCountPtr )
```

Returns the number of segments for a fuselage in a CPACS configuration.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of the fuselage, starting at 1
out	*segmentCountPtr*	Pointer to the number of segments

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex is not valid
- TIGL_NULL_POINTER if segmentCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.19 tiglFuselageGetSegmentIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * segmentUID,
            int * segmentIndexPtr,
            int * fuselageIndexPtr )
```

Returns the Index of a segment of a fuselage.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*segmentUID*	The uid of the fuselage
out	*segmentIndexPtr*	The index of a segment, starting at 1
out	*fuselageIndexPtr*	The index of a fuselage, starting at 1

Usage example:

```
TiglReturnCode returnCode;
int segmentIndex, fuselageIndex;
returnCode = tiglFuselageGetSegmentIndex(cpacsHandle, segmentUID, &segmentIndex, &fuselageIndex);
printf("The Index of the segment of fuselage %d is %d\n", fuselageIndex, segmentIndex);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex is not valid
- TIGL_UID_ERROR if the segmentUID does not exist
- TIGL_NULL_POINTER if segmentUID is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.20 tiglFuselageGetSegmentUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            char ** uidNamePtr )
```

Returns the UID of a segment of a fuselage. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*uidNamePtr*	The uid of the fuselage

Usage example:

```
TiglReturnCode returnCode;
char* uidPtr = 0;
returnCode = tiglFuselageGetSegmentUID(cpacsHandle, fuselage, segmentID, &uidPtr);
printf("The UID of the segment of fuselage %d is %s\n", fuselage, uidPtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if profileNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.21 tiglFuselageGetStartConnectedSegmentCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartConnectedSegmentCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            int * segmentCountPtr )
```

Returns the count of segments connected to the start section of a given fuselage segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*segmentCountPtr*	Pointer to the count of connected segments

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if segmentCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.22 tiglFuselageGetStartConnectedSegmentIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartConnectedSegmentIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            int n,
            int * connectedIndexPtr )
```

Returns the index (number) of the n-th segment connected to the start section of a given fuselage segment. n starts at 1.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
in	*n*	n-th segment searched, 1 <= n <= tiglFuselageGetStartConnectedSegmentCount(...)
out	*connectedIndexPtr*	Pointer to the segment index of the n-th connected segment

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle or no n-th connected segment was found

- TIGL_INDEX_ERROR if fuselageIndex, segmentIndex or n are not valid
- TIGL_NULL_POINTER if segmentIndexPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.23 tiglFuselageGetStartSectionAndElementIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartSectionAndElementIndex (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int segmentIndex,
             int * sectionIndexPtr,
             int * elementIndexPtr )
```

Returns the section index and section element index of the start side of a given fuselage segment.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*sectionIndexPtr*	The section UID of the start side
out	*elementIndexPtr*	The section element UID of the start side

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_NULL_POINTER if sectionIndexPtr or elementIndexPtr are a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.24 tiglFuselageGetStartSectionAndElementUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartSectionAndElementUID (
             TiglCPACSConfigurationHandle cpacsHandle,
             int fuselageIndex,
             int segmentIndex,
             char ** sectionUIDPtr,
             char ** elementUIDPtr )
```

Returns the section UID and section element UID of the start side of a given fuselage segment.

Important change: The memory necessary for the two UIDs must not to be freed by the user anymore.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*sectionUIDPtr*	The section UID of the start side
out	*elementUIDPtr*	The section element UID of the start side

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex or segmentIndex are not valid
- TIGL_ERROR if some other error occurred

9.4.2.25 tiglFuselageGetSymmetry()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSymmetry (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            TiglSymmetryAxis * symmetryAxisPtr )
```

Returns the Symmetry Enum if the fuselage has symmetry-axis.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	Index of the fuselage in the cpacs file
out	*symmetryAxisPtr*	Returning TiglSymmetryAxis enum pointer

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.4.2.26 tiglFuselageGetUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            char ** uidNamePtr )
```

Returns the UID of a fuselage. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
out	*uidNamePtr*	The uid of the fuselage

Usage example:

```
TiglReturnCode returnCode;
char* uidPtr = 0;
returnCode = tiglFuselageGetUID(cpacsHandle, fuselage, &uidPtr);
printf("The UID of the fuselage is %s\n", uidPtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if profileNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.4.2.27 tiglGetFuselageCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglGetFuselageCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            int * fuselageCountPtr )
```

Returns the number of fuselages in a CPACS configuration.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
out	*fuselageCountPtr*	Pointer to the number of fuselages

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if fuselageCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.5 Functions for rotor calculations

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglGetRotorCount (TiglCPACSConfigurationHandle cpacs↩ Handle, int ∗rotorCountPtr)

  Returns the number of rotors in a CPACS configuration.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetIndex (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗rotorUID, int ∗rotorIndexPtr)

  Returns the Index of a rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetRadius (TiglCPACSConfigurationHandle cpacs↩ Handle, int rotorIndex, double ∗radiusPtr)

  Returns the radius of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetReferenceArea (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, double ∗referenceAreaPtr)

  Returns the reference area of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetSolidity (TiglCPACSConfigurationHandle cpacs↩ Handle, int rotorIndex, double ∗solidityPtr)

  Returns the solidity of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetSurfaceArea (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, double ∗surfaceAreaPtr)

  Returns the surface area of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetTipSpeed (TiglCPACSConfigurationHandle cpacs↩ Handle, int rotorIndex, double ∗tipSpeedPtr)

  Returns the tip speed of the rotor in [m/s].
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetTotalBladePlanformArea (TiglCPACSConfiguration↩ Handle cpacsHandle, int rotorIndex, double ∗totalBladePlanformAreaPtr)

  Returns the total blade planform area of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetUID (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, char ∗∗uidNamePtr)

  Returns the UID of a rotor. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetVolume (TiglCPACSConfigurationHandle cpacs↩ Handle, int rotorIndex, double ∗volumePtr)

  Returns the volume of the rotor.

9.5.1 Detailed Description

Functions to handle rotor geometries with TIGL.

9.5.2 Function Documentation

9.5.2.1 tiglGetRotorCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglGetRotorCount (
             TiglCPACSConfigurationHandle cpacsHandle,
             int ∗ rotorCountPtr )
```

Returns the number of rotors in a CPACS configuration.

Fortran syntax:

tigl_get_rotor_count(integer cpacsHandle, integer rotorCountPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
out	*rotorCountPtr*	Pointer to the number of rotors

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if rotorCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.5.2.2 tiglRotorGetIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * rotorUID,
            int * rotorIndexPtr )
```

Returns the Index of a rotor.

Fortran syntax:

tigl_rotor_get_index(integer cpacsHandle, character*n uIDNamePtr, integer rotorIndex, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorUID*	The uid of the rotor
out	*rotorIndexPtr*	The index of a rotor, starting at 1

Usage example:

```
TiglReturnCode returnCode;
int rotorIndex;
returnCode = tiglRotorGetUID(cpacsHandle, rotorUID, &rotorIndex);
printf("The Index of the rotor is %d\n", rotorIndex);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if rotorUID does not exist
- TIGL_NULL_POINTER if rotorUID is a null pointer
- TIGL_ERROR if some other error occurred

9.5.2.3 tiglRotorGetRadius()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetRadius (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            double * radiusPtr )
```

Returns the radius of the rotor.

This function returns the radius of the largest blade attached to the rotor hub.

Fortran syntax:

tigl_rotor_get_radius(integer cpacsHandle, int rotorIndex, real radiusPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor to calculate the radius, starting at 1
out	*radiusPtr*	The radius of the rotor

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex is less or equal zero or greater than the rotor count
- TIGL_NULL_POINTER if radiusPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.5.2.4 tiglRotorGetReferenceArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetReferenceArea (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            double * referenceAreaPtr )
```

Returns the reference area of the rotor.

The area of the rotor disk is taken as reference area of the rotor. It is calculated using the formula pi∗r^2, where r denotes the radius of the largest attached blade.

Fortran syntax:

tigl_rotor_get_reference_area(integer cpacsHandle, int rotorIndex, real referenceAreaPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor to calculate the area, starting at 1
out	*referenceAreaPtr*	The reference area of the rotor

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex is less or equal zero or greater than the rotor count
- TIGL_NULL_POINTER if referenceAreaPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.5.2.5 tiglRotorGetSolidity()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetSolidity (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            double * solidityPtr )
```

Returns the solidity of the rotor.

The rotor solidity ratio is calculated by dividing the total blade planform area by the rotor disk area.

Fortran syntax:

tigl_rotor_get_solidity(integer cpacsHandle, int rotorIndex, real solidityPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor to calculate the area, starting at 1
out	*solidityPtr*	The reference area of the rotor

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex is less or equal zero or greater than the rotor count
- TIGL_NULL_POINTER if solidityPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.5.2.6 tiglRotorGetSurfaceArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetSurfaceArea (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            double * surfaceAreaPtr )
```

Returns the surface area of the rotor.

The returned surface area is the sum of the surface areas of all attached rotor blades.

Fortran syntax:

tigl_rotor_get_surface_area(integer cpacsHandle, int rotorIndex, real surfaceAreaPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the Rotor to calculate the area, starting at 1
out	*surfaceAreaPtr*	The surface area of the rotor

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex is less or equal zero or greater than the rotor count
- TIGL_NULL_POINTER if surfaceAreaPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.5.2.7 tiglRotorGetTipSpeed()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetTipSpeed (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            double * tipSpeedPtr )
```

Returns the tip speed of the rotor in [m/s].

The rotor tip speed is calculated using the nominal rotation speed of the rotor and the rotor radius.

Fortran syntax:

tigl_rotor_get_tip_speed(integer cpacsHandle, int rotorIndex, real tipSpeedPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor to calculate the area, starting at 1
out	*tipSpeedPtr*	The tip speed of the rotor

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex is less or equal zero or greater than the rotor count
- TIGL_NULL_POINTER if tipSpeedPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.5.2.8 tiglRotorGetTotalBladePlanformArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetTotalBladePlanformArea (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            double * totalBladePlanformAreaPtr )
```

Returns the total blade planform area of the rotor.

This function calculates the sum of the planform areas of all blades attached to the rotor hub.

Fortran syntax:

tigl_rotor_get_total_blade_planform_area(integer cpacsHandle, int rotorIndex, real totalBladePlanformAreaPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor to calculate the area, starting at 1
out	*totalBladePlanformAreaPtr*	The total blade planform area of the rotor

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex is less or equal zero or greater than the rotor count
- TIGL_NULL_POINTER if totalBladePlanformAreaPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.5.2.9 tiglRotorGetUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetUID (
             TiglCPACSConfigurationHandle cpacsHandle,
             int rotorIndex,
             char ** uidNamePtr )
```

Returns the UID of a rotor. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

Fortran syntax:

tigl_rotor_get_uid(integer cpacsHandle, integer rotorIndex, character*n uIDNamePtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	The index of a rotor, starting at 1
out	*uidNamePtr*	The uid of the rotor

Usage example:

```
TiglReturnCode returnCode;
char* uidPtr = 0;
returnCode = tiglRotorGetUID(cpacsHandle, rotor, &uidPtr);
printf("The UID of the rotor is %s\n", uidPtr);
```

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex is not valid
- TIGL_NULL_POINTER if profileNamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.5.2.10 tiglRotorGetVolume()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetVolume (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            double * volumePtr )
```

Returns the volume of the rotor.

The returned volume is the sum of the volumes of all attached rotor blades.

Fortran syntax:

tigl_rotor_get_volume(integer cpacsHandle, int rotorIndex, real volumePtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor to calculate the volume, starting at 1
out	*volumePtr*	The volume of the rotor

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex is less or equal zero or greater than the rotor count
- TIGL_NULL_POINTER if volumePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6 Functions for rotor blade calculations

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetAzimuthAngle (TiglCPACSConfiguration↩
Handle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗azimuthAnglePtr)

  Returns the azimuth angle of a rotor blade in degrees.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetLocalChord (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, int segmentIndex, double eta, double ∗chordPtr)

  Returns the local chord length of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetLocalRadius (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, int segmentIndex, double eta, double ∗radiusPtr)

  Returns the local radius of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetLocalTwistAngle (TiglCPACSConfiguration↩
Handle cpacsHandle, int rotorIndex, int rotorBladeIndex, int segmentIndex, double eta, double ∗twistAngle↩
Ptr)

  Returns the local twist angle [deg] of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetPlanformArea (TiglCPACSConfiguration↩
Handle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗planformAreaPtr)

  Returns the planform area of the rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetRadius (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗radiusPtr)

  Returns the radius of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetSurfaceArea (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗surfaceAreaPtr)

  Returns the surface area of the rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetTipSpeed (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗tipSpeedPtr)

  Returns the tip speed of a rotor blade [m/s].
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetVolume (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗volumePtr)

  Returns the volume of the rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetWingIndex (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, int ∗wingIndexPtr)

  Returns the index of the parent wing definition of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetWingUID (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, char ∗∗wingUIDPtr)

  Returns the UID of the parent wing definition of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetRotorBladeCount (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int ∗rotorBladeCountPtr)

  Returns the total number of rotor blades attached to a rotor.

9.6.1 Detailed Description

Functions to handle rotor blade geometries with TIGL.

9.6.2 Function Documentation

9.6.2.1 tiglRotorBladeGetAzimuthAngle()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetAzimuthAngle (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            double * azimuthAnglePtr )
```

Returns the azimuth angle of a rotor blade in degrees.

Returns the azimuth angle in degrees of the rotor blade with the index rotorBladeIndex attached to the rotor with the index rotorIndex.

Fortran syntax:

tigl_get_rotor_blade_azimuth(integer cpacsHandle, integer rotorIndex, integer rotorBladeIndex, real azimuthAngle, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade
out	*azimuthAnglePtr*	Pointer to the azimuth angle

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex or rotorBladeIndex are invalid
- TIGL_NULL_POINTER if azimuthAnglePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.2 tiglRotorBladeGetLocalChord()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetLocalChord (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            int segmentIndex,
            double eta,
            double * chordPtr )
```

Returns the local chord length of a rotor blade.

Returns the local chord length of the section at the relative spanwise coordinate eta of the segment segmentIndex of the rotor blade with the index rotorBladeIndex attached to the rotor with the index rotorIndex.

Fortran syntax:

tigl_get_rotor_blade_local_chord(integer cpacsHandle, integer rotorIndex, integer rotorBladeIndex, integer segmentIndex, real eta, real chordPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade
in	*segmentIndex*	Index of the segment of the referenced wing definition
in	*eta*	Relative spanwise segment coordinate: eta in the range 0.0 <= eta <= 1.0; eta = 0 for inner section , eta = 1 for outer section
out	*chordPtr*	Pointer to the local chord length

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle or no point was found
- TIGL_INDEX_ERROR if rotorIndex, rotorBladeIndex or segmentIndex are invalid
- TIGL_NULL_POINTER if chordPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.3 tiglRotorBladeGetLocalRadius()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetLocalRadius (
             TiglCPACSConfigurationHandle cpacsHandle,
             int rotorIndex,
             int rotorBladeIndex,
             int segmentIndex,
             double eta,
             double * radiusPtr )
```

Returns the local radius of a rotor blade.

Returns the local radius of the rotor blade with the index rotorBladeIndex attached to the rotor with the index rotor↩Index. The distance of the point with relative spanwise coordinate eta on the quarter chord line of the rotor blade segment segmentIndex from the z axis of the rotor coordinate system is taken as the local radius. It is calculated for the rotor blade at azimuth=0 and with no hinge transformations applied.

Fortran syntax:

tigl_get_rotor_blade_local_radius(integer cpacsHandle, integer rotorIndex, integer rotorBladeIndex, integer segmentIndex, real eta, real radiusPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade
in	*segmentIndex*	Index of the segment of the referenced wing definition
in	*eta*	Relative spanwise segment coordinate: eta in the range 0.0 <= eta <= 1.0; eta = 0 for inner section , eta = 1 for outer section
out	*radiusPtr*	Pointer to the local radius

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle or no point was found
- TIGL_INDEX_ERROR if rotorIndex, rotorBladeIndex or segmentIndex are invalid
- TIGL_NULL_POINTER if radiusPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.4 tiglRotorBladeGetLocalTwistAngle()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetLocalTwistAngle (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            int segmentIndex,
            double eta,
            double * twistAnglePtr )
```

Returns the local twist angle [deg] of a rotor blade.

Returns the twist angle in degrees of the section at the relative spanwise coordinate eta of the segment segment↵
Index of the rotor blade with the index rotorBladeIndex attached to the rotor with the index rotorIndex.

Fortran syntax:

tigl_get_rotor_blade_local_twist_angle(integer cpacsHandle, integer rotorIndex, integer rotorBladeIndex, integer segmentIndex, real eta, real twistAnglePtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade
in	*segmentIndex*	Index of the segment of the referenced wing definition
in	*eta*	Relative spanwise segment coordinate: eta in the range 0.0 <= eta <= 1.0; eta = 0 for inner section , eta = 1 for outer section
out	*twistAnglePtr*	Pointer to the local twist angle

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle or no point was found
- TIGL_INDEX_ERROR if rotorIndex, rotorBladeIndex or segmentIndex are invalid
- TIGL_NULL_POINTER if twistAnglePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.5 tiglRotorBladeGetPlanformArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetPlanformArea (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            double * planformAreaPtr )
```

Returns the planform area of the rotor blade.

Fortran syntax:

tigl_rotor_blade_get_planform_area(integer cpacsHandle, int rotorIndex, int rotorBladeIndex, real planformAreaPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade to calculate the area, starting at 1
out	*planformAreaPtr*	The planform area of the rotor blade

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex or rotorBladeIndex are invalid
- TIGL_NULL_POINTER if planformAreaPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.6 tiglRotorBladeGetRadius()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetRadius (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            double * radiusPtr )
```

Returns the radius of a rotor blade.

Returns the radius of the rotor blade with the index rotorBladeIndex attached to the rotor with the index rotor↩ Index. The maximum distance of a point on the quarter chord line of the rotor blade from the z axis of the rotor coordinate system is taken as the rotor blade radius. It is calculated for the rotor blade at azimuth=0 and with no hinge transformations applied.

Fortran syntax:

tigl_get_rotor_blade_radius(integer cpacsHandle, integer rotorIndex, integer rotorBladeIndex, real radiusPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade
out	*radiusPtr*	Pointer to the radius

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex or rotorBladeIndex are invalid
- TIGL_NULL_POINTER if radiusPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.7 tiglRotorBladeGetSurfaceArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetSurfaceArea (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            double * surfaceAreaPtr )
```

Returns the surface area of the rotor blade.

Fortran syntax:

tigl_rotor_blade_get_surface_area(integer cpacsHandle, int rotorIndex, int rotorBladeIndex, real surfaceAreaPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade to calculate the area, starting at 1
out	*surfaceAreaPtr*	The surface area of the rotor blade

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex or rotorBladeIndex are invalid
- TIGL_NULL_POINTER if surfaceAreaPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.8 tiglRotorBladeGetTipSpeed()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetTipSpeed (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            double * tipSpeedPtr )
```

Returns the tip speed of a rotor blade [m/s].

The rotor blade tip speed is calculated using the nominal rotation speed of the rotor and the rotor blade radius.

Fortran syntax:

tigl_get_rotor_blade_tip_speed(integer cpacsHandle, integer rotorIndex, integer rotorBladeIndex, real tipSpeedPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade
out	*tipSpeedPtr*	Pointer to the radius

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex or rotorBladeIndex are invalid
- TIGL_NULL_POINTER if tipSpeedPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.9 tiglRotorBladeGetVolume()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetVolume (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            double * volumePtr )
```

Returns the volume of the rotor blade.

Fortran syntax:

tigl_rotor_blade_get_volume(integer cpacsHandle, int rotorIndex, int rotorBladeIndex, real volumePtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade to calculate the volume, starting at 1
out	*volumePtr*	The volume of the rotor blade

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex or rotorBladeIndex are invalid
- TIGL_NULL_POINTER if volumePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.10 tiglRotorBladeGetWingIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetWingIndex (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            int * wingIndexPtr )
```

Returns the index of the parent wing definition of a rotor blade.

Returns the index of the wing definition referenced by the parent rotor blade attachment of the rotor blade with the index rotorBladeIndex.

Fortran syntax:

tigl_get_rotor_blade_wing_index(integer cpacsHandle, integer rotorIndex, integer rotorBladeIndex, integer wing↩ IndexPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade
out	*wingIndexPtr*	Pointer to the wing index

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex or rotorBladeIndex are invalid
- TIGL_NULL_POINTER if wingIndexPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.11 tiglRotorBladeGetWingUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetWingUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int rotorBladeIndex,
            char ** wingUIDPtr )
```

Returns the UID of the parent wing definition of a rotor blade.

Returns the UID of the wing definition referenced by the parent rotor blade attachment of the rotor blade with the index rotorBladeIndex.

Fortran syntax:

tigl_get_rotor_blade_wing_uid(integer cpacsHandle, integer rotorIndex, integer rotorBladeIndex, character*n wingUIDPtr, integer returnCode)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor
in	*rotorBladeIndex*	Index of the rotor blade
out	*wingUIDPtr*	Pointer to the wing index

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if rotorIndex or rotorBladeIndex are invalid
- TIGL_NULL_POINTER if wingUIDPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.6.2.12 tiglRotorGetRotorBladeCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetRotorBladeCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            int rotorIndex,
            int * rotorBladeCountPtr )
```

Returns the total number of rotor blades attached to a rotor.

Fortran syntax:

tigl_get_rotor_blade_count(integer cpacsHandle, integer rotorIndex, integer rotorBladeCountPtr, integer return↩
Code)

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*rotorIndex*	Index of the rotor to count the attached rotor blades
out	*rotorBladeCountPtr*	Pointer to the number of rotor blades

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if rotorBladeCountPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.7 Functions for boolean calculations

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglComponentIntersectionLineCount (TiglCPACSConfiguration↩Handle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, int ∗numWires)

  The function returns the number of intersection lines of two geometric components.
- TIGL_COMMON_EXPORT TiglReturnCode tiglComponentIntersectionPoint (TiglCPACSConfiguration↩Handle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, int lineID, double eta, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  The function returns a point on the intersection line of two geometric components. Often there are more that one intersection line, therefore you need to specify the line.
- TIGL_COMMON_EXPORT TiglReturnCode tiglComponentIntersectionPoints (TiglCPACSConfiguration↩Handle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, int lineID, const double ∗etaArray, int numberOfPoints, double ∗pointXArray, double ∗pointYArray, double ∗pointZArray)

  Convienience function to returns a list of points on the intersection line of two geometric components. Often there are more that one intersection line, therefore you need to specify the line.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectComponents (TiglCPACSConfigurationHandle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, char ∗∗intersectionID)

  tiglIntersectComponents computes the intersection line(s) between two shapes specified by their CPACS uid. It returns an intersection ID for further computations on the result. To query points on the intersection line, tigl↩IntersectGetPoint has to be called.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectGetLineCount (TiglCPACSConfigurationHandle cpacsHandle, const char ∗intersectionID, int ∗lineCount)

  tiglIntersectGetLineCount return the number of intersection lines computed by tiglIntersectComponents or tigl↩IntersectWithPlane for the given intersectionID.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectGetPoint (TiglCPACSConfigurationHandle cpacs↩Handle, const char ∗intersectionID, int lineIdx, double eta, double ∗pointX, double ∗pointY, double ∗pointZ)

  tiglIntersectGetPoint samples a point on an intersection line calculated by tiglIntersectComponents or tiglIntersect↩WithPlane.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectWithPlane (TiglCPACSConfigurationHandle cpacs↩Handle, const char ∗componentUid, double px, double py, double pz, double nx, double ny, double nz, char ∗∗intersectionID)

  tiglIntersectWithPlane computes the intersection line(s) between a shape and a plane. It returns an intersection ID for further computations on the result. To query points on the intersection line, tiglIntersectGetPoint has to be called.

9.7.1 Detailed Description

Function for boolean calculations on wings/fuselages.

These function currently only implement intersection algorithms between two shapes defined in cpacs ot a shape and a plane. Shapes or geometries are identified with their cpacs uid.

Currently only wings, wing segments, fuselages, and fuselage segments can be used in the intersection routines.

9.7.2 Function Documentation

9.7.2.1 tiglComponentIntersectionLineCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglComponentIntersectionLineCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentUidOne,
            const char * componentUidTwo,
            int * numWires )
```

The function returns the number of intersection lines of two geometric components.

Deprecated This is a deprecated function and will be removed in future releases. Use tiglIntersectGetLineCount in combination with tiglIntersectGetPoint instead.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentUidOne*	The UID of the first component
in	*componentUidTwo*	The UID of the second component
out	*numWires*	The number of intersection lines

Returns

- TIGL_SUCCESS if there are not intersection lines
- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_ERROR if some other error occurred

9.7.2.2 tiglComponentIntersectionPoint()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglComponentIntersectionPoint (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentUidOne,
            const char * componentUidTwo,
            int lineID,
            double eta,
            double * pointXPtr,
            double * pointYPtr,
            double * pointZPtr )
```

The function returns a point on the intersection line of two geometric components. Often there are more that one intersection line, therefore you need to specify the line.

Deprecated This is a deprecated function and will be removed in future releases. Use tiglIntersectComponents and tiglIntersectGetPoint instead.

Returns a point on the intersection line between a surface and a wing in dependence of parameter eta which range from 0.0 to 1.0. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentUidOne*	The UID of the first component
in	*componentUidTwo*	The UID of the second component
in	*lineID*	The index of the intersection wire, get wire number with "tiglComponentIntersectionLineCount"
in	*eta*	eta in the range 0.0 <= eta <= 1.0
out	*pointXPtr*	Pointer to the x-coordinate of the point in absolute world coordinates
out	*pointYPtr*	Pointer to the y-coordinate of the point in absolute world coordinates
out	*pointZPtr*	Pointer to the z-coordinate of the point in absolute world coordinates

Returns

- TIGL_SUCCESS if a point was found

- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.7.2.3 tiglComponentIntersectionPoints()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglComponentIntersectionPoints (
             TiglCPACSConfigurationHandle cpacsHandle,
             const char * componentUidOne,
             const char * componentUidTwo,
             int lineID,
             const double * etaArray,
             int numberOfPoints,
             double * pointXArray,
             double * pointYArray,
             double * pointZArray )
```

Convienience function to returns a list of points on the intersection line of two geometric components. Often there are more that one intersection line, therefore you need to specify the line.

Deprecated This is a deprecated function and will be removed in future releases. Use tiglIntersectComponents and tiglIntersectGetPoint instead.

Returns a point on the intersection line between a surface and a wing in dependence of parameter eta which range from 0.0 to 1.0. The point is returned in absolute world coordinates.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentUidOne*	The UID of the first component
in	*componentUidTwo*	The UID of the second component
in	*lineID*	The index of the intersection wire, get wire number with "tiglComponentIntersectionLineCount"
in	*etaArray*	Array of eta values in the range 0.0 <= eta <= 1.0.
in	*numberOfPoints*	The number of points to calculate i.e. the size of the etaArray.
out	*pointXArray*	Array of x-coordinates of the points in absolute world coordinates. The Array must be preallocated with the size numberOfPoints.
out	*pointYArray*	Array of y-coordinates of the points in absolute world coordinates. The Array must be preallocated with the size numberOfPoints.
out	*pointZArray*	Array of z-coordinates of the points in absolute world coordinates. The Array must be preallocated with the size numberOfPoints.

Returns

- TIGL_NOT_FOUND if no point was found or the cpacs handle is not valid
- TIGL_NULL_POINTER if pointXPtr, pointYPtr or pointZPtr are null pointers
- TIGL_ERROR if some other error occurred

9.7.2.4 tiglIntersectComponents()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectComponents (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentUidOne,
            const char * componentUidTwo,
            char ** intersectionID )
```

tiglIntersectComponents computes the intersection line(s) between two shapes specified by their CPACS uid. It returns an intersection ID for further computations on the result. To query points on the intersection line, tigl↩ IntersectGetPoint has to be called.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentUidOne*	The UID of the first component
in	*componentUidTwo*	The UID of the second component
out	*intersectionID*	A unique identifier that is associated with the computed intersection.

Returns

- TIGL_SUCCESS if an intersection could be computed
- TIGL_NOT_FOUND if the cpacs handle is not valid
- TIGL_NULL_POINTER if either componentUidOne, componentUidTwo, or intersectionID are NULL pointers
- TIGL_UID_ERROR if componentUidOne or componentUidTwo can not be found in the CPACS file

9.7.2.5 tiglIntersectGetLineCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectGetLineCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * intersectionID,
            int * lineCount )
```

tiglIntersectGetLineCount return the number of intersection lines computed by tiglIntersectComponents or tigl↩ IntersectWithPlane for the given intersectionID.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*intersectionID*	The intersection identifier returned by tiglIntersectComponents or tiglIntersectWithPlane
out	*lineCount*	Number of intersection lines computed by tiglIntersectComponents or tiglIntersectWithPlane. If no intersection could be computed, line count is 0.

Returns

- TIGL_SUCCESS if no error occured
- TIGL_NOT_FOUND if the cpacs handle or the intersectionID is not valid
- TIGL_NULL_POINTER if lineCount is a NULL pointer

9.7.2.6 tiglIntersectGetPoint()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectGetPoint (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * intersectionID,
            int lineIdx,
            double eta,
            double * pointX,
            double * pointY,
            double * pointZ )
```

tiglIntersectGetPoint samples a point on an intersection line calculated by tiglIntersectComponents or tiglIntersect↩WithPlane.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*intersectionID*	The intersection identifier returned by tiglIntersectComponents or tiglIntersectWithPlane
in	*lineIdx*	Line index to sample from. To get the number of lines, call tiglIntersectGetLineCount. 1 <= lineIdx <= lineCount.
in	*eta*	Parameter on the curve that determines the point position, with 0 <= eta <= 1.
out	*pointX*	X coordinate of the resulting point.
out	*pointY*	Y coordinate of the resulting point.
out	*pointZ*	Z coordinate of the resulting point.

Returns

- TIGL_SUCCESS if no error occured
- TIGL_NOT_FOUND if the cpacs handle or the intersectionID is not valid
- TIGL_NULL_POINTER if pointX, pointY, or pointZ are NULL pointers
- TIGL_INDEX_ERROR if lineIdx is not in valid range
- TIGL_MATH_ERROR if eta is not in range 0 <= eta <= 1

9.7.2.7 tiglIntersectWithPlane()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectWithPlane (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * componentUid,
            double px,
            double py,
            double pz,
            double nx,
            double ny,
            double nz,
            char ** intersectionID )
```

tiglIntersectWithPlane computes the intersection line(s) between a shape and a plane. It returns an intersection ID for further computations on the result. To query points on the intersection line, tiglIntersectGetPoint has to be called.

The shape has to be specified by its CPACS UID. The plane is specified by a central point p on the plane and a normal vector n, which is perpendicular to the plane. The normal vector must not be zero!

Parameters

`in`	*cpacsHandle*	Handle for the CPACS configuration
`in`	*componentUid*	The UID of the CPACS shape
`in`	*px*	X Coordinate of the plane center point
`in`	*py*	Y Coordinate of the plane center point
`in`	*pz*	Z Coordinate of the plane center point
`in`	*nx*	X value of the plane normal vector
`in`	*ny*	Y value of the plane normal vector
`in`	*nz*	Z value of the plane normal vector
`out`	*intersectionID*	A unique identifier that is associated with the computed intersection.

Returns

- TIGL_SUCCESS if an intersection could be computed
- TIGL_NOT_FOUND if the cpacs handle is not valid
- TIGL_NULL_POINTER if either componentUid or intersectionID are NULL pointers
- TIGL_UID_ERROR if componentUid can not be found in the CPACS file
- TIGL_MATH_ERROR if the normal vector is zero

9.8 Export Functions

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglExportFusedBREP (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗filename)

  Exports the fused/trimmed geometry of a CPACS configuration to BREP format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportFusedSTEP (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗filenamePtr)

  Exports the fused/trimmed geometry of a CPACS configuration to STEP format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportFusedWingFuselageIGES (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗filenamePtr)

  Exports the boolean fused geometry of a CPACS configuration to IGES format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportFuselageColladaByUID (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗fuselageUID, const char ∗filename, double deflection)

  Exports the boolean fused geometry of a fuselage (selected by uid) meshed to Collada (∗.dae) format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportIGES (TiglCPACSConfigurationHandle cpacsHandle, const char ∗filenamePtr)

  Exports the geometry of a CPACS configuration to IGES format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageSTL (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a fuselage meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageSTLByUID (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗fuselageUID, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a fuselage meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageVTKByIndex (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const int fuselageIndex, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a fuselage (selected by index) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageVTKByUID (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗fuselageUID, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a fuselage (selected by uid) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageVTKSimpleByUID (const TiglCPA↩ CSConfigurationHandle cpacsHandle, const char ∗fuselageUID, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a fuselage (selected by uid) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedGeometrySTL (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of the whole configuration meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedGeometryVTK (const TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of the whole configuration meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedGeometryVTKSimple (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of the whole configuration meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingSTL (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a wing meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingSTLByUID (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗wingUID, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a wing meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingVTKByIndex (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const int wingIndex, const char ∗filenamePtr, const double deflection)

  Exports the boolean fused geometry of a wing (selected by id) meshed to VTK format.

- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingVTKByUID (const TiglCPACS←↩ ConfigurationHandle cpacsHandle, const char ∗wingUID, const char ∗filenamePtr, double deflection)

    Exports the boolean fused geometry of a wing (selected by UID) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingVTKSimpleByUID (const TiglCPACS←↩ ConfigurationHandle cpacsHandle, const char ∗wingUID, const char ∗filenamePtr, double deflection)

    Exports the boolean fused geometry of a wing (selected by UID) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportSTEP (TiglCPACSConfigurationHandle cpacsHandle, const char ∗filenamePtr)

    Exports the geometry of a CPACS configuration to STEP format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportVTKSetOptions (const char ∗key, const char ∗value)

    Sets options for the VTK Export.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportWingColladaByUID (const TiglCPACSConfiguration←↩ Handle cpacsHandle, const char ∗wingUID, const char ∗filename, double deflection)

    Exports the boolean fused geometry of a wing (selected by uid) meshed to Collada (∗.dae) format.

9.8.1 Detailed Description

Functions for export of wings/fuselages.

VTK-Export

There a various different VTK exports functions in TIGL. All functions starting with 'tiglExportVTK[Fuselage|Wing]...' are exporting a special triangulation with no duplicated points into a VTK file formated in XML (file extension .vtp) with some custom informations added to the file.

In addition to the triangulated geometry, additional data are written to the VTK file. These data currently include:

- uID: The UID of the fuselage or wing component segment on which the triangle exists.
- segmentIndex: The segmentIndex of the fuselage or wing component segment on which the triangle exists. Kind of redundant to the UID.
- eta/xsi: The parameters in the surface parametrical space of the triangle.
- isOnTop: Flag that indicates whether the triangle is on the top of the wing or not. Please see the cpacs documentation how "up" is defined for wings.

Please note that at this time these information are only valid for wings!

There are two ways, how these additional data are attached to the VTK file. The first is the official VTK way to declare additional data for each polygon/triangle. For each data entry, a <DataArray> tag is added under the xpath

```
/VTKFile/PolyData/Piece/CellData
```

Each CellData contains a vector(list) of values, each of them corresponding the data of one triangle. For example the data entry for the wing segment eta coordinates for 4 triangles looks like.

```
<DataArray type="Float64" Name="eta" NumberOfComponents="1"
    format="ascii" RangeMin="0.000000" RangeMax="1.000000">
        0.25 0.5 0.75 1.0
</DataArray>
```

The second way these data are stored is by using the "MetaData" mechanism of VTK. Here, a <MetaData> tag is added under the xpath

```
/VTKFile/PolyData/Piece/Polys
```

A typical exported MetaData tag looks like the following:

```
<MetaData elements="uID segmentIndex eta xsi isOnTop">
  "rootToInnerkink" 1 3.18702 0.551342 0
  "rootToInnerkink" 1 2.93939 0.581634 0
  "rootToInnerkink" 1 4.15239 0.520915 0
  ...
</MetaData>
```

The 'elements' attribute indicates the number and the names of the additional information tags as a whitespace separated list. In this example you could see 5 information fields with the name.

9.8.2 Function Documentation

9.8.2.1 tiglExportFusedBREP()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportFusedBREP (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * filename )
```

Exports the fused/trimmed geometry of a CPACS configuration to BREP format.

In order to fuse the geometry, boolean operations are performed. Depending on the complexity of the configuration, the fusing can take several minutes.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*filename*	BREP export file name

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.8.2.2 tiglExportFusedSTEP()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportFusedSTEP (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * filenamePtr )
```

Exports the fused/trimmed geometry of a CPACS configuration to STEP format.

In order to fuse the geometry, boolean operations are performed. Depending on the complexity of the configuration, the fusing can take several minutes.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*filenamePtr*	Pointer to an STEP export file name

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.8.2.3 tiglExportFusedWingFuselageIGES()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportFusedWingFuselageIGES (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * filenamePtr )
```

Exports the boolean fused geometry of a CPACS configuration to IGES format.

To maintain compatibility with CATIA, the file suffix should be ".igs".

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*filenamePtr*	Pointer to an IGES export file name

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.8.2.4 tiglExportFuselageColladaByUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportFuselageColladaByUID (
            const TiglCPACSConfigurationHandle cpacsHandle,
            const char * fuselageUID,
            const char * filename,
            double deflection )
```

Exports the boolean fused geometry of a fuselage (selected by uid) meshed to Collada (∗.dae) format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageUID*	UID of the Fuselage to export
in	*filename*	Filename of the resulting collada file. It should contain the .dae file name ending.
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filename is a null pointer
- TIGL_INDEX_ERROR if fuselageUID does not exists
- TIGL_ERROR if some other error occurred

9.8.2.5 tiglExportIGES()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportIGES (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * filenamePtr )
```

Exports the geometry of a CPACS configuration to IGES format.

To maintain compatibility with CATIA, the file suffix should be ".igs".

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*filenamePtr*	Pointer to an IGES export file name

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.8.2.6 tiglExportMeshedFuselageSTL()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageSTL (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            const char * filenamePtr,
            double deflection )
```

Exports the boolean fused geometry of a fuselage meshed to STL format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	Index of the Fuselage to export
in	*filenamePtr*	Pointer to an STL export file name
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred

- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if fuselageIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.8.2.7 tiglExportMeshedFuselageSTLByUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageSTLByUID (
             TiglCPACSConfigurationHandle cpacsHandle,
             const char * fuselageUID,
             const char * filenamePtr,
             double deflection )
```

Exports the boolean fused geometry of a fuselage meshed to STL format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageUID*	UID of the Fuselage to export
in	*filenamePtr*	Pointer to an STL export file name
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if fuselageIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.8.2.8 tiglExportMeshedFuselageVTKByIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageVTKByIndex (
             const TiglCPACSConfigurationHandle cpacsHandle,
             const int fuselageIndex,
             const char * filenamePtr,
             double deflection )
```

Exports the boolean fused geometry of a fuselage (selected by index) meshed to VTK format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	Index of the Fuselage to export
in	*filenamePtr*	Pointer to an VTK export file name ($*$.vtp = polygonal XML_VTK)
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if fuselageIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.8.2.9 tiglExportMeshedFuselageVTKByUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageVTKByUID (
            const TiglCPACSConfigurationHandle cpacsHandle,
            const char * fuselageUID,
            const char * filenamePtr,
            double deflection )
```

Exports the boolean fused geometry of a fuselage (selected by uid) meshed to VTK format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageUID*	UID of the Fuselage to export
in	*filenamePtr*	Pointer to an VTK export file name (∗.vtp = polygonal XML_VTK)
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if fuselageUID does not exists
- TIGL_ERROR if some other error occurred

9.8.2.10 tiglExportMeshedFuselageVTKSimpleByUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageVTKSimpleByUID (
            const TiglCPACSConfigurationHandle cpacsHandle,
            const char * fuselageUID,
            const char * filenamePtr,
            double deflection )
```

Exports the boolean fused geometry of a fuselage (selected by uid) meshed to VTK format.

This function does only a very simple, but also very fast meshing on the fuselage and exports them to a VTK file. No additional CPACS relevant information are computed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageUID*	UID of the Fuselage to export
in	*filenamePtr*	Pointer to an VTK export file name (∗.vtp = polygonal XML_VTK)
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if fuselageUID does not exists
- TIGL_ERROR if some other error occurred

9.8.2.11 tiglExportMeshedGeometrySTL()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedGeometrySTL (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * filenamePtr,
            double deflection )
```

Exports the boolean fused geometry of the whole configuration meshed to STL format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*filenamePtr*	Pointer to an STL export file name
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.8.2.12 tiglExportMeshedGeometryVTK()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedGeometryVTK (
            const TiglCPACSConfigurationHandle cpacsHandle,
            const char * filenamePtr,
            double deflection )
```

Exports the boolean fused geometry of the whole configuration meshed to VTK format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*filenamePtr*	Pointer to an VTK export file name (∗.vtp = polygonal XML_VTK)
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle

- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.8.2.13 tiglExportMeshedGeometryVTKSimple()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedGeometryVTKSimple (
             const TiglCPACSConfigurationHandle cpacsHandle,
             const char * filenamePtr,
             double deflection )
```

Exports the boolean fused geometry of the whole configuration meshed to VTK format.

This function does only a very simple, but also very fast meshing on the geometry and exports them to a VTK file. No additional CPACS relevant information are computed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*filenamePtr*	Pointer to an VTK export file name (∗.vtp = polygonal XML_VTK)
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.8.2.14 tiglExportMeshedWingSTL()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingSTL (
             TiglCPACSConfigurationHandle cpacsHandle,
             int wingIndex,
             const char * filenamePtr,
             double deflection )
```

Exports the boolean fused geometry of a wing meshed to STL format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	Index of the Wing to export
in	*filenamePtr*	Pointer to an STL export file name
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle

- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if wingIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.8.2.15 tiglExportMeshedWingSTLByUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingSTLByUID (
             TiglCPACSConfigurationHandle cpacsHandle,
             const char * wingUID,
             const char * filenamePtr,
             double deflection )
```

Exports the boolean fused geometry of a wing meshed to STL format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingUID*	UID of the Wing to export
in	*filenamePtr*	Pointer to an STL export file name
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if wingIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.8.2.16 tiglExportMeshedWingVTKByIndex()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingVTKByIndex (
             const TiglCPACSConfigurationHandle cpacsHandle,
             const int wingIndex,
             const char * filenamePtr,
             const double deflection )
```

Exports the boolean fused geometry of a wing (selected by id) meshed to VTK format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	Index of the Wing to export
in	*filenamePtr*	Pointer to an VTK export file name (∗.vtp = polygonal XML_VTK)
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred

- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if wingIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.8.2.17 tiglExportMeshedWingVTKByUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingVTKByUID (
             const TiglCPACSConfigurationHandle cpacsHandle,
             const char * wingUID,
             const char * filenamePtr,
             double deflection )
```

Exports the boolean fused geometry of a wing (selected by UID) meshed to VTK format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingUID*	UID of the Wing to export
in	*filenamePtr*	Pointer to an VTK export file name (∗.vtp = polygonal XML_VTK)
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if the wing UID does not exists
- TIGL_ERROR if some other error occurred

9.8.2.18 tiglExportMeshedWingVTKSimpleByUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingVTKSimpleByUID (
             const TiglCPACSConfigurationHandle cpacsHandle,
             const char * wingUID,
             const char * filenamePtr,
             double deflection )
```

Exports the boolean fused geometry of a wing (selected by UID) meshed to VTK format.

This function does only a very simple, but also very fast meshing on the wing segments and exports them to a VTK file. No additional CPACS relevant information are computed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingUID*	UID of the Wing to export
in	*filenamePtr*	Pointer to an VTK export file name (∗.vtp = polygonal XML_VTK)
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_INDEX_ERROR if the wing UID does not exists
- TIGL_ERROR if some other error occurred

9.8.2.19 tiglExportSTEP()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportSTEP (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * filenamePtr )
```

Exports the geometry of a CPACS configuration to STEP format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*filenamePtr*	Pointer to an STEP export file name

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filenamePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.8.2.20 tiglExportVTKSetOptions()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportVTKSetOptions (
            const char * key,
            const char * value )
```

Sets options for the VTK Export.

Available Settings:

- *key*: "normals_enabled" *valid values*: "0 or 1" *default*: "1".

  Enables or disables the output of normal vectors. A Normal vector and a vertex belong to the same logical structure. If there are two identical vertices but with different normal vectors, the VTK export stores them as two entries. Thus, a VTK file may have "duplicate" vertices. To disable this behavior, set "normal_enabled" to "0".

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NULL_POINTER if key or value are a null pointer
- TIGL_ERROR if the specified key/value pair is invalid

9.8.2.21 tiglExportWingColladaByUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglExportWingColladaByUID (
            const TiglCPACSConfigurationHandle cpacsHandle,
            const char * wingUID,
            const char * filename,
            double deflection )
```

Exports the boolean fused geometry of a wing (selected by uid) meshed to Collada (∗.dae) format.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingUID*	UID of the Wing to export
in	*filename*	Filename of the resulting collada file. It should contain the .dae file name ending.
in	*deflection*	Maximum deflection of the triangulation from the real surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if filename is a null pointer
- TIGL_INDEX_ERROR if fuselageUID does not exists
- TIGL_ERROR if some other error occurred

9.9 Material functions

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetMaterialCount (TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗compSegmentUID, TiglStructureType structureType, double eta, double xsi, int ∗materialCount)

  Returns the number of materials defined at a point on the wing component segment surface.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetMaterialThickness (TiglCPA↩ CSConfigurationHandle cpacsHandle, const char ∗compSegmentUID, TiglStructureType structureType, double eta, double xsi, int materialIndex, double ∗thickness)

  Returns one of the material thicknesses of a given point on the wing component segment surface. The number of materials on that point has to be first queried using tiglWingComponentSegmentGetMaterialCount.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetMaterialUID (TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗compSegmentUID, TiglStructureType structureType, double eta, double xsi, int materialIndex, char ∗∗uid)

  Returns one of the material UIDs of a given point on the wing component segment surface. The number of materials on that point has to be first queried using tiglWingComponentSegmentGetMaterialCount.

9.9.1 Detailed Description

Functions to query material information of wings/fuselages. Materials are currently ony implemented for the wing component segment. Here, materials for the lower and upper wing surface can be queried, which can be a material for the whole skin/surface or a material defined inside a wing cell. A wing cell material overwrites the global skin material, i.e. if the whole wing skin material is aluminum and the trailing edge is made of radar absorbing material, only the absorbing material is returned by the querying functions.

9.9.2 Function Documentation

9.9.2.1 tiglWingComponentSegmentGetMaterialCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetMaterialCount (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * compSegmentUID,
            TiglStructureType structureType,
            double eta,
            double xsi,
            int * materialCount )
```

Returns the number of materials defined at a point on the wing component segment surface.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*compSegmentUID*	UID of the component segment
in	*structureType*	Type of structure, where the materials are queried
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*xsi*	xsi in the range 0.0 <= xsi <= 1.0
out	*materialCount*	Number of materials defined at the given coordinate

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if compSegmentUID or materialCount is a null pointer
- TIGL_INDEX_ERROR if compSegmentUID or materialIndex is invalid
- TIGL_ERROR if some other error occurred

9.9.2.2 tiglWingComponentSegmentGetMaterialThickness()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetMaterialThickness (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * compSegmentUID,
            TiglStructureType structureType,
            double eta,
            double xsi,
            int materialIndex,
            double * thickness )
```

Returns one of the material thicknesses of a given point on the wing component segment surface. The number of materials on that point has to be first queried using tiglWingComponentSegmentGetMaterialCount.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*compSegmentUID*	UID of the component segment
in	*structureType*	Type of structure, where the materials are queried
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*xsi*	xsi in the range 0.0 <= xsi <= 1.0
in	*materialIndex*	Index of the material to query (1 <= index <= materialCount)
out	*thickness*	Material thickness at the given coordinate. If no thickness is defined, thickness gets a negative value and TIGL_UNINITIALIZED is returned.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if compSegmentUID or thickness is a null pointer
- TIGL_INDEX_ERROR if compSegmentUID or materialIndex is invalid
- TIGL_UNINITIALIZED if no thickness is defined for the material
- TIGL_ERROR if some other error occurred

9.9.2.3 tiglWingComponentSegmentGetMaterialUID()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetMaterialUID (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * compSegmentUID,
            TiglStructureType structureType,
            double eta,
            double xsi,
            int materialIndex,
            char ** uid )
```

Returns one of the material UIDs of a given point on the wing component segment surface. The number of materials on that point has to be first queried using tiglWingComponentSegmentGetMaterialCount.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*compSegmentUID*	UID of the component segment
in	*structureType*	Type of structure, where the materials are queried
in	*eta*	eta in the range 0.0 <= eta <= 1.0
in	*xsi*	xsi in the range 0.0 <= xsi <= 1.0
in	*materialIndex*	Index of the material to query (1 <= index <= materialCount)
out	*uid*	Material uid at the given coordinate

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if compSegmentUID is a null pointer
- TIGL_INDEX_ERROR if compSegmentUID or materialIndex is invalid
- TIGL_ERROR if some other error occurred

9.10 Functions for volume calculations

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentVolume (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double ∗volumePtr)

  Returns the volume of a segment of a fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetVolume (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, double ∗volumePtr)

  Returns the volume of the fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentVolume (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double ∗volumePtr)

  Returns the volume of a segment of a wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetVolume (TiglCPACSConfigurationHandle cpacs↩ Handle, int wingIndex, double ∗volumePtr)

  Returns the volume of the wing.

9.10.1 Detailed Description

Function for volume calculations on wings/fuselages.

9.10.2 Function Documentation

9.10.2.1 tiglFuselageGetSegmentVolume()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentVolume (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            double ∗ volumePtr )
```

Returns the volume of a segment of a fuselage.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*volumePtr*	The pointer to a variable for the volume of the fuselage

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if volumePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.10.2.2 tiglFuselageGetVolume()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetVolume (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            double * volumePtr )
```

Returns the volume of the fuselage.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	Index of the fuselage to calculate the volume, starting at 1
out	*volumePtr*	The volume of the fuselage

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.10.2.3 tiglWingGetSegmentVolume()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentVolume (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            double * volumePtr )
```

Returns the volume of a segment of a wing.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*volumePtr*	The pointer to a variable for the volume of the wing

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if volumePtr is a null pointer
- TIGL_ERROR if some other error occurred

9.10.2.4 tiglWingGetVolume()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetVolume (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            double * volumePtr )
```

Returns the volume of the wing.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	Index of the Wing to calculate the volume, starting at 1
out	*volumePtr*	The volume of the wing

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.11 Functions for surface area calculations

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentSurfaceArea (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double ∗surfaceAreaPtr)

  Returns the surface area of a segment of a fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSurfaceArea (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, double ∗surfaceAreaPtr)

  Returns the surface area of the fuselage. Currently, the area includes also the faces on the fuselage symmetry plane (in case of a symmetric wing). This is in particular a problem for fuselages, where only one half side is defined in CPACS. In future releases, these faces will not belong anymore to the surface area calculation.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetReferenceArea (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, TiglSymmetryAxis symPlane, double ∗referenceAreaPtr)

  Returns the reference area of the wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentLowerSurfaceAreaTrimmed (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta1, double xsi1, double eta2, double xsi2, double eta3, double xsi3, double eta4, double xsi4, double ∗surfaceArea)

  Computes the area of the trimmed lower wing segment surface. This function can be e.g. used to determine the area of the wing flaps.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentSurfaceArea (TiglCPACSConfiguration↩ Handle cpacsHandle, int wingIndex, int segmentIndex, double ∗surfaceAreaPtr)

  Returns the surface area of a segment of a wing. This includes only the area of the upper and lower wing segment surface and does not include the trailing egde or any closing faces.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentUpperSurfaceAreaTrimmed (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta1, double xsi1, double eta2, double xsi2, double eta3, double xsi3, double eta4, double xsi4, double ∗surfaceArea)

  Computes the area of the trimmed upper wing segment surface. This function can be e.g. used to determine the area of the wing flaps.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSurfaceArea (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, double ∗surfaceAreaPtr)

  Returns the surface area of the wing. Currently, the area includes also the faces on the wing symmetry plane (in case of a symmetric wing). In coming releases, these faces will not belong anymore to the surface area calculation.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetWettedArea (TiglCPACSConfigurationHandle cpacsHandle, char ∗wingUID, double ∗wettedAreaPtr)

  Returns the wetted area of the wing.

9.11.1 Detailed Description

Function for surface area calculations of wings/fuselages.

9.11.2 Function Documentation

9.11.2.1 tiglFuselageGetSegmentSurfaceArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentSurfaceArea (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            int segmentIndex,
            double * surfaceAreaPtr )
```

Returns the surface area of a segment of a fuselage.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	The index of a fuselage, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*surfaceAreaPtr*	The pointer to a variable for the surface area of the fuselage-segment

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if surfaceAreaPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.11.2.2 tiglFuselageGetSurfaceArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSurfaceArea (
            TiglCPACSConfigurationHandle cpacsHandle,
            int fuselageIndex,
            double * surfaceAreaPtr )
```

Returns the surface area of the fuselage. Currently, the area includes also the faces on the fuselage symmetry plane (in case of a symmetric wing). This is in particular a problem for fuselages, where only one half side is defined in CPACS. In future releases, these faces will not belong anymore to the surface area calculation.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*fuselageIndex*	Index of the Fuselage to calculate the area, starting at 1
out	*surfaceAreaPtr*	The surface area of the wing

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if fuselageIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.11.2.3 tiglWingGetReferenceArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetReferenceArea (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            TiglSymmetryAxis symPlane,
            double * referenceAreaPtr )
```

Returns the reference area of the wing.

The reference area of the wing is calculated by taking account the quadrilateral portions of each wing segment by projecting the wing segments into the plane defined by the user. If projection should be avoided, use TIGL_NO_↩ SYMMETRY as symPlane argument.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	Index of the Wing to calculate the area, starting at 1
in	*symPlane*	Plane on which the wing is projected for calculating the refarea. Values can be: • TIGL_NO_SYMMETRY, the wing is not projected but its true 3D area is calculated • TIGL_X_Y_PLANE, the wing is projected onto the x-y plane (use for e.g. main wings and HTPs) • TIGL_X_Z_PLANE, the wing is projected onto the x-z plane (use for e.g. VTPs) • TIGL_Y_Z_PLANE, the wing is projected onto the y-z plane
out	*referenceAreaPtr*	The reference area of the wing

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.11.2.4 tiglWingGetSegmentLowerSurfaceAreaTrimmed()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentLowerSurfaceAreaTrimmed (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            double eta1,
            double xsi1,
            double eta2,
            double xsi2,
            double eta3,
            double xsi3,
            double eta4,
            double xsi4,
            double * surfaceArea )
```

Computes the area of the trimmed lower wing segment surface. This function can be e.g. used to determine the area of the wing flaps.

The use of this function is analog to tiglWingGetSegmentUpperSurfaceAreaTrimmed.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
in	*eta1*	Eta value of P1 in range [0,1]
in	*xsi1*	Xsi value of P1 in range [0,1]
in	*eta2*	Eta value of P2 in range [0,1]

Parameters

in	*xsi2*	Xsi value of P2 in range [0,1]
in	*eta3*	Eta value of P3 in range [0,1]
in	*xsi3*	Xsi value of P3 in range [0,1]
in	*eta4*	Eta value of P4 in range [0,1]
in	*xsi4*	Xsi value of P4 in range [0,1]
out	*surfaceArea*	Area of the trimmed lower wing surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex ot segmentIndex are not valid
- TIGL_NULL_POINTER if surfaceArea is a null pointer
- TIGL_ERROR if the eta/xsi coordinates are not in the valid range [0,1] or another error occured

9.11.2.5 tiglWingGetSegmentSurfaceArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentSurfaceArea (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            double * surfaceAreaPtr )
```

Returns the surface area of a segment of a wing. This includes only the area of the upper and lower wing segment surface and does not include the trailing egde or any closing faces.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
out	*surfaceAreaPtr*	The pointer to a variable for the surface area of the wing

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex, sectionIndex or elementIndex are not valid
- TIGL_NULL_POINTER if surfaceAreaPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.11.2.6 tiglWingGetSegmentUpperSurfaceAreaTrimmed()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentUpperSurfaceAreaTrimmed (
            TiglCPACSConfigurationHandle cpacsHandle,
            int wingIndex,
            int segmentIndex,
            double eta1,
            double xsi1,
            double eta2,
            double xsi2,
            double eta3,
            double xsi3,
            double eta4,
            double xsi4,
            double * surfaceArea )
```

Computes the area of the trimmed upper wing segment surface. This function can be e.g. used to determine the area of the wing flaps.

The computed area does not include the trailing edge or any closing side faces.

All eta and xsi values must be in the range [0,1]. The trimmed area is defined with the four corner point P1, P2, P3, and P4. The order of the points should be right handed, as shown the the image below.

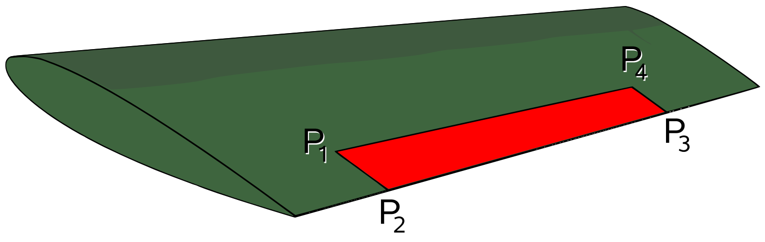


Figure 11 Location of the four corner points

Each of the points is defined with an eta/xsi coordinate pair in the wing segment system.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	The index of a wing, starting at 1
in	*segmentIndex*	The index of a segment, starting at 1
in	*eta1*	Eta value of P1 in range [0,1]
in	*xsi1*	Xsi value of P1 in range [0,1]
in	*eta2*	Eta value of P2 in range [0,1]
in	*xsi2*	Xsi value of P2 in range [0,1]
in	*eta3*	Eta value of P3 in range [0,1]
in	*xsi3*	Xsi value of P3 in range [0,1]
in	*eta4*	Eta value of P4 in range [0,1]
in	*xsi4*	Xsi value of P4 in range [0,1]
out	*surfaceArea*	Area of the trimmed upper wing surface

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex ot segmentIndex are not valid
- TIGL_NULL_POINTER if surfaceArea is a null pointer

- TIGL_ERROR if the eta/xsi coordinates are not in the valid range [0,1] or another error occured

9.11.2.7 tiglWingGetSurfaceArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSurfaceArea (
             TiglCPACSConfigurationHandle cpacsHandle,
             int wingIndex,
             double * surfaceAreaPtr )
```

Returns the surface area of the wing. Currently, the area includes also the faces on the wing symmetry plane (in case of a symmetric wing). In coming releases, these faces will not belong anymore to the surface area calculation.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingIndex*	Index of the Wing to calculate the area, starting at 1
out	*surfaceAreaPtr*	The surface area of the wing

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_INDEX_ERROR if wingIndex is less or equal zero
- TIGL_ERROR if some other error occurred

9.11.2.8 tiglWingGetWettedArea()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetWettedArea (
             TiglCPACSConfigurationHandle cpacsHandle,
             char * wingUID,
             double * wettedAreaPtr )
```

Returns the wetted area of the wing.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingUID*	UID of the Wing to calculate the wetted area
out	*wettedAreaPtr*	The wetted area of the wing

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if wingUID is wrong
- TIGL_NULL_POINTER if wingIUD is NULL
- TIGL_ERROR if some other error occurred

9.12 General geometry functions.

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglProfileGetBSplineCount (TiglCPACSConfigurationHandle cpacsHandle, const char ∗profileUID, int ∗curveCount)

  Returns the number of curves the given profile is made of.
- TIGL_COMMON_EXPORT TiglReturnCode tiglProfileGetBSplineData (TiglCPACSConfigurationHandle cpacsHandle, const char ∗profileUID, int curveid, int nControlPoints, double ∗cpx, double ∗cpy, double ∗cpz, int nKnots, double ∗knots)

  Returns the B-Spline data of the given profile curve. This includes the knot vector and the control points of the B-Spline.
- TIGL_COMMON_EXPORT TiglReturnCode tiglProfileGetBSplineDataSizes (TiglCPACSConfiguration←Handle cpacsHandle, const char ∗profileUID, int curveid, int ∗degree, int ∗nControlPoints, int ∗nKnots)

  Returns the B-Spline data sizes for a given curve on a profile. This includes size of the knot vector, size of the control point vector and degree of the spline.

9.12.1 Detailed Description

9.12.2 Function Documentation

9.12.2.1 tiglProfileGetBSplineCount()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglProfileGetBSplineCount (
             TiglCPACSConfigurationHandle cpacsHandle,
             const char ∗ profileUID,
             int ∗ curveCount )
```

Returns the number of curves the given profile is made of.

The given profile may be a fuselage profile or a wing profile. Typically, wing profiles consist of two curves (lower and upper curve), fuselage profiles consist of only one curve.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*profileUID*	UID of the profile
out	*curveCount*	Number of curves

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if profileUID is wrong
- TIGL_NULL_POINTER if profileUID or curveCount NULL
- TIGL_ERROR if some other error occurred

9.12.2.2 tiglProfileGetBSplineData()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglProfileGetBSplineData (
             TiglCPACSConfigurationHandle cpacsHandle,
             const char * profileUID,
             int curveid,
             int nControlPoints,
             double * cpx,
             double * cpy,
             double * cpz,
             int nKnots,
             double * knots )
```

Returns the B-Spline data of the given profile curve. This includes the knot vector and the control points of the B-Spline.

The output arrays cpx, cpy, cpz, and knots have to be allocated by the user first. The control point vector arrays must have the size nControlPoints. This value must be queried with tiglProfileGetBSplineDataSizes first. The knot vector array must have the size kKnots. This value has to be queried also using the function tiglProfileGetBSpline↩ DataSizes.

The given profile may be a fuselage profile or a wing profile. Typically, wing profiles consist of two curves (lower and upper curve), fuselage profiles consist of only one curve.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*profileUID*	UID of the profile
in	*curveid*	Index of the curve. Number of curves must be queried with tiglProfileGetBSplineCount. 1 <= index <= count
in	*nControlPoints*	Size of the control point vector. To be queried with tiglProfileGetBSplineDataSizes first.
out	*cpx*	X-values of the control point vector.
out	*cpy*	Y-values of the control point vector.
out	*cpz*	Z-values of the control point vector.
in	*nKnots*	Size of the knot vector. To be queried with tiglProfileGetBSplineDataSizes first.
out	*knots*	Knot vector values.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if profileUID is wrong
- TIGL_INDEX_ERROR if curveid knot in range [1, curveCount]
- TIGL_NULL_POINTER if the argument profileUID, cpx, cpy, cpz, or knots are NULL
- TIGL_ERROR if the values nControlPoints, or nKnots are wrong

9.12.2.3 tiglProfileGetBSplineDataSizes()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglProfileGetBSplineDataSizes (
              TiglCPACSConfigurationHandle cpacsHandle,
              const char * profileUID,
              int curveid,
              int * degree,
              int * nControlPoints,
              int * nKnots )
```

Returns the B-Spline data sizes for a given curve on a profile. This includes size of the knot vector, size of the control point vector and degree of the spline.

The given profile may be a fuselage profile or a wing profile. Typically, wing profiles consist of two curves (lower and upper curve), fuselage profiles consist of only one curve.

Parameters

`in`	*cpacsHandle*	Handle for the CPACS configuration
`in`	*profileUID*	UID of the profile
`in`	*curveid*	Index of the curve. Number of curves must be queried with tiglProfileGetBSplineCount. 1 <= index <= count
`out`	*degree*	Degree of the B-Spline
`out`	*nControlPoints*	Size of the control point vector
`out`	*nKnots*	Size of the knot vector

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_UID_ERROR if profileUID is wrong
- TIGL_INDEX_ERROR if curveid knot in range [1, curveCount]
- TIGL_NULL_POINTER if the argument profileUID, degree, ncontrolPoints, or nKnots are NULL
- TIGL_ERROR if some other error occurred

9.13 Logging functions.

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglLogSetFileEnding (const char ∗ending)

  Sets the file ending for logging files. Default is "log".
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogSetTimeInFilenameEnabled (TiglBoolean enabled)

  Enables or disables appending a unique date/time identifier inside the log file name (behind the file prefix). By default, the time identifier is enabled.
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogSetVerbosity (TiglLogLevel level)

  Set the console verbosity level.
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogToFileDisabled ()

  Disabled file logging. If a log file is currently opened by TiGL it will be closed. The log messages are printed to console. This is the default logging mechanism of TIGL.
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogToFileEnabled (const char ∗filePrefix)

  Sets up the tigl logging mechanism to send all log messages into a file.
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogToFileStreamEnabled (FILE ∗fp)

  Sets up the tigl logging mechanism to send all log messages into an already opened file.

9.13.1 Detailed Description

The following functions are used to customize the behaviour how messages are handled by TiGL. By default, only error messages and warnings are printed to the console.

In addition, TiGL can be told to write message into a log file using the function tiglLogToFileEnabled and tiglLogTo←
FileStreamEnabled. By enabling file logging, other log messages than errors and warnings can be inspected. File logging is disabled by default.

In order to change the verbosity of the TiGL messages printed on console, use tiglLogSetVerbosity.

9.13.2 Function Documentation

9.13.2.1 tiglLogSetFileEnding()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglLogSetFileEnding (
            const char * ending )
```

Sets the file ending for logging files. Default is "log".

This function has to be called before tiglLogToFileEnabled to have the desired effect.

Parameters

in	*ending*	File ending of the logging file. Default is "log".

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NULL_POINTER if ending is NULL
- TIGL_ERROR if some error occurred

9.13.2.2 tiglLogSetTimeInFilenameEnabled()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglLogSetTimeInFilenameEnabled (
             TiglBoolean enabled )
```

Enables or disables appending a unique date/time identifier inside the log file name (behind the file prefix). By default, the time identifier is enabled.

This function has to be called before tiglLogToFileEnabled to have the desired effect.

Parameters

in	*enabled*	Set to true, if time identifier should be enabled.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_ERROR if some error occurred

9.13.2.3 tiglLogSetVerbosity()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglLogSetVerbosity (
             TiglLogLevel level )
```

Set the console verbosity level.

This function shall be used change, what kind of logging information is displayed on the console. By default, only errors and warnings are printed on console.

Parameters

in	*level*	Verbosity level for console messages.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_ERROR if some error occurred

9.13.2.4 tiglLogToFileDisabled()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglLogToFileDisabled ( )
```

Disabled file logging. If a log file is currently opened by TiGL it will be closed. The log messages are printed to console. This is the default logging mechanism of TIGL.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_ERROR if some other error occurred

9.13.2.5 tiglLogToFileEnabled()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglLogToFileEnabled (
            const char * filePrefix )
```

Sets up the tigl logging mechanism to send all log messages into a file.

Typically this function has to be called before opening any cpacs configuration. The console logging mechanism remains untouched.

Parameters

in	*filePrefix*	Prefix of the filename to be created. The filename consists of the prefix, a date and time string and the ending ".log".

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NULL_POINTER if filePrefix is NULL
- TIGL_OPEN_FAILED if file can not be opened for writing
- TIGL_ERROR if some other error occurred

9.13.2.6 tiglLogToFileStreamEnabled()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglLogToFileStreamEnabled (
            FILE * fp )
```

Sets up the tigl logging mechanism to send all log messages into an already opened file.

In contrast to tiglLogToFileEnabled, the messages are appended to an already opened file. This file might be an already opened logging file set up by another library or program. The console logging mechanism remains untouched.

Typically this function has to be called before opening any cpacs configuration.

Parameters

in	*fp*	File pointer to an already opened file.

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NULL_POINTER if fp is NULL , i.e. not opened for writing.
- TIGL_ERROR if some other error occurred

9.14 Generic utility functions.

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglComponentGetHashCode (TiglCPACSConfigurationHandle cpacsHandle, const char ∗componentUID, int ∗hashCodePtr)

  Returns a unique HashCode for a geometric component.
- TIGL_COMMON_EXPORT TiglReturnCode tiglConfigurationGetLength (TiglCPACSConfigurationHandle cpacsHandle, double ∗pLength)

  Returns the length of the plane.
- TIGL_COMMON_EXPORT const char ∗ tiglGetErrorString (TiglReturnCode errorCode)

  Translates an error code into a string.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetMAC (TiglCPACSConfigurationHandle cpacs↩Handle, const char ∗wingUID, double ∗mac_chord, double ∗mac_x, double ∗mac_y, double ∗mac_z)

  This function calculates location of the quarter of mean aerodynamic chord, and gives the chord lenght as well.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSpan (TiglCPACSConfigurationHandle cpacs↩Handle, const char ∗wingUID, double ∗pSpan)

  Returns the span of a wing.

9.14.1 Detailed Description

Generic utility functions for geometric components that fits not only to wings *or* fuselages.

9.14.2 Function Documentation

9.14.2.1 tiglComponentGetHashCode()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglComponentGetHashCode (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char ∗ componentUID,
            int ∗ hashCodePtr )
```

Returns a unique HashCode for a geometric component.

The component, for example a wing or a fuselage, could be specified via its UID. The HashCode is the same as long as the geometry of this component has not changed. The HashCode is valid through the current session only! The hash value is computed from the value of the underlying shape reference.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*componentUID*	The uid of the component for which the hash should be computed
out	*hashCodePtr*	The pointer to a hash value to represent this shape

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NOT_FOUND if no configuration was found for the given handle
- TIGL_NULL_POINTER if surfaceAreaPtr is a null pointer
- TIGL_ERROR if some other error occurred

9.14.2.2 tiglConfigurationGetLength()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglConfigurationGetLength (
             TiglCPACSConfigurationHandle cpacsHandle,
             double * pLength )
```

Returns the length of the plane.

The calculation of the airplane lenght is realized as follows:

All part of the configuration (currently all wing and fuselage segments) are put into a bounding box. The length of the plane is returned as the length of the box in x-direction.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
out	*pLength*	Length of plane

Returns

Error code

9.14.2.3 tiglGetErrorString()

```
TIGL_COMMON_EXPORT const char* tiglGetErrorString (
             TiglReturnCode errorCode )
```

Translates an error code into a string.

Parameters

in	*errorCode*	Return value of a tigl function

Returns

Error code as a string.

9.14.2.4 tiglWingGetMAC()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetMAC (
             TiglCPACSConfigurationHandle cpacsHandle,
             const char * wingUID,
             double * mac_chord,
             double * mac_x,
             double * mac_y,
             double * mac_z )
```

This function calculates location of the quarter of mean aerodynamic chord, and gives the chord lenght as well.

It uses the classical method that can be applied to trapozaidal wings. This method is used for each segment. The values are found by taking into account of sweep and dihedral. But the effect of insidance angle is neglected. These values should coinside with the values found with tornado tool.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingUID*	UID of the Wing
out	*mac_chord*	Mean areadynamic chord length
out	*mac_x,mac_y,mac↩_z*	- Position of the MAC

Returns

- TIGL_SUCCESS if no error occurred
- TIGL_NULL_POINTER if wingUID, mac_chord, mac_x, mac_y or mac_z are null pointers
- TIGL_ERROR In case of an unknown error

9.14.2.5 tiglWingGetSpan()

```
TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSpan (
            TiglCPACSConfigurationHandle cpacsHandle,
            const char * wingUID,
            double * pSpan )
```

Returns the span of a wing.

The calculation of the wing span is realized as follows:

- If the wing is mirrored at a symmetry plane (like the main wing), the wing body and its mirrored counterpart are computed and are put into a bounding box. The length of the box in a specific space dimension is returned as the wing span depending on the symmetry plane (y direction for x-z planes, z direction for x-y planes, x direction for y-z symmetry planes).
- If no symmetry plane is defined (e.g. for the fins), the largest dimension of the bounding box around the wing is returned.

Parameters

in	*cpacsHandle*	Handle for the CPACS configuration
in	*wingUID*	UID of the Wing
out	*pSpan*	Wing span

Returns

Error code

10 File Documentation

10.1 build/doc/ChangeLog.md File Reference

10.2 build/src/api/tigl_version.h File Reference

Definition of the tigl version numbers.

Macros

Version Number

Printable format: "%d.%d.%d", MAJOR, MINOR, PATCHLEVEL

- #define TIGL_MAJOR_VERSION 2
- #define TIGL_MINOR_VERSION 2
- #define TIGL_PATCHLEVEL 1
- #define TIGL_VERSION_STRING "2.2.1"
- #define TIGL_REVISION ""
- #define TIGL_VERSION_HEX (TIGL_MAJOR_VERSION << 16 | TIGL_MINOR_VERSION << 8 | TI↩ GL_PATCHLEVEL)

10.2.1 Detailed Description

Definition of the tigl version numbers.

10.2.2 Macro Definition Documentation

10.2.2.1 TIGL_MAJOR_VERSION

```
#define TIGL_MAJOR_VERSION 2
```

10.2.2.2 TIGL_MINOR_VERSION

```
#define TIGL_MINOR_VERSION 2
```

10.2.2.3 TIGL_PATCHLEVEL

```
#define TIGL_PATCHLEVEL 1
```

10.2.2.4 TIGL_REVISION

```
#define TIGL_REVISION ""
```

10.2.2.5 TIGL_VERSION_HEX

```
#define TIGL_VERSION_HEX (TIGL_MAJOR_VERSION << 16 | TIGL_MINOR_VERSION << 8 | TIGL_PATCHLEV↩
EL)
```

TIGL_VERSION_HEX returns a unique integer which is increasing with each release This can be e.g. used in ifdefs

10.2.2.6 TIGL_VERSION_STRING

```
#define TIGL_VERSION_STRING "2.2.1"
```

10.3 doc/mainpage.md File Reference

10.4 doc/tiglviewer.md File Reference

10.5 doc/tiglviewer_console.md File Reference

10.6 doc/usage.md File Reference

10.7 examples/README.md File Reference

10.8 src/api/tigl.h File Reference

Declaration of the TIGL C interface.

```
#include "tixi.h"
#include <stdio.h>
```

Macros

- #define TIGL_COMMON_EXPORT
- #define TIGL_COMPONENT_FUSELAGE 8
- #define TIGL_COMPONENT_FUSELSEGMENT 128
- #define TIGL_COMPONENT_GENERICSYSTEM 1024
- #define TIGL_COMPONENT_LOGICAL 2
- #define TIGL_COMPONENT_PHYSICAL 1
- #define TIGL_COMPONENT_PLANE 4
- #define TIGL_COMPONENT_ROTOR 2048
- #define TIGL_COMPONENT_ROTORBLADE 4096
- #define TIGL_COMPONENT_SEGMENT 32
- #define TIGL_COMPONENT_WING 16
- #define TIGL_COMPONENT_WINGCOMPSEGMENT 256
- #define TIGL_COMPONENT_WINGSEGMENT 64
- #define TIGL_COMPONENT_WINGSHELL 512
- #define TIGL_H

Typedefs

- typedef enum TiglAlgorithmCode TiglAlgorithmCode

  Typedef for possible algorithm types used in calculations.
- typedef enum TiglBoolean TiglBoolean

  Definition of boolean type used in TIGL.
- typedef enum TiglContinuity TiglContinuity
- typedef enum TiglCoordinateSystem TiglCoordinateSystem
- typedef int TiglCPACSConfigurationHandle

  Datatype for a CPACS configuration handle.
- typedef unsigned int TiglGeometricComponentType
- typedef enum TiglImportExportFormat TiglImportExportFormat

  Definition of the different file formats used for import/export used in TIGL.
- typedef enum TiglLoftSide TiglLoftSide
- typedef enum TiglLogLevel TiglLogLevel

  Definition of logging levels.
- typedef enum TiglReturnCode TiglReturnCode

  Definition of error return type.
- typedef const char ∗∗ TiglStringList
- typedef enum TiglStructureType TiglStructureType
- typedef enum TiglSymmetryAxis TiglSymmetryAxis

  Definition of Symmetry Axis used in TIGL.

Enumerations

- enum TiglAlgorithmCode { TIGL_INTERPOLATE_LINEAR_WIRE = 0, TIGL_INTERPOLATE_BSPLINE_↩
  WIRE = 1, TIGL_APPROXIMATE_BSPLINE_WIRE = 2 }
- enum TiglBoolean { TIGL_FALSE = 0, TIGL_TRUE = 1 }
- enum TiglContinuity { C0 = 0, C1 = 1, C2 = 2 }
- enum TiglCoordinateSystem { GLOBAL_COORDINATE_SYSTEM = 0, WING_COORDINATE_SYSTEM =
  1, FUSELAGE_COORDINATE_SYSTEM = 2 }
- enum TiglImportExportFormat { TIGL_IMPORTEXPORT_IGES = 0, TIGL_IMPORTEXPORT_STEP = 1, T↩
  IGL_IMPORTEXPORT_STL = 2, TIGL_IMPORTEXPORT_VTK = 3 }
- enum TiglLoftSide { UPPER_SIDE = 0, LOWER_SIDE = 1 }
- enum TiglLogLevel {
  TILOG_SILENT =0, TILOG_ERROR =1, TILOG_WARNING =2, TILOG_INFO =3,
  TILOG_DEBUG =4, TILOG_DEBUG1 =5, TILOG_DEBUG2 =6, TILOG_DEBUG3 =7,
  TILOG_DEBUG4 =8 }
- enum TiglReturnCode {
  TIGL_SUCCESS = 0, TIGL_ERROR = 1, TIGL_NULL_POINTER = 2, TIGL_NOT_FOUND = 3,
  TIGL_XML_ERROR = 4, TIGL_OPEN_FAILED = 5, TIGL_CLOSE_FAILED = 6, TIGL_INDEX_ERROR = 7,
  TIGL_STRING_TRUNCATED = 8, TIGL_WRONG_TIXI_VERSION = 9, TIGL_UID_ERROR = 10, TIGL_↩
  WRONG_CPACS_VERSION = 11,
  TIGL_UNINITIALIZED = 12, TIGL_MATH_ERROR = 13, TIGL_WRITE_FAILED = 14 }
- enum TiglStructureType { UPPER_SHELL = 0, LOWER_SHELL = 1, INNER_STRUCTURE = 2 }
- enum TiglSymmetryAxis { TIGL_NO_SYMMETRY = 0, TIGL_X_Y_PLANE = 1, TIGL_X_Z_PLANE = 2, TI↩
  GL_Y_Z_PLANE = 3 }

Functions

- TIGL_COMMON_EXPORT TiglReturnCode tiglCloseCPACSConfiguration (TiglCPACSConfigurationHandle cpacsHandle)

    Closes a CPACS configuration and cleans up all memory used by the configuration. After closing a configuration the associated configuration handle is no longer valid. When the CPACS configuration has been closed, the companion tixi document can also be closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglComponentGetHashCode (TiglCPACSConfigurationHandle cpacsHandle, const char ∗componentUID, int ∗hashCodePtr)

    Returns a unique HashCode for a geometric component.
- TIGL_COMMON_EXPORT TiglReturnCode tiglComponentIntersectionLineCount (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, int ∗numWires)

    The function returns the number of intersection lines of two geometric components.
- TIGL_COMMON_EXPORT TiglReturnCode tiglComponentIntersectionPoint (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, int lineID, double eta, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

    The function returns a point on the intersection line of two geometric components. Often there are more that one intersection line, therefore you need to specify the line.
- TIGL_COMMON_EXPORT TiglReturnCode tiglComponentIntersectionPoints (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, int lineID, const double ∗etaArray, int numberOfPoints, double ∗pointXArray, double ∗pointYArray, double ∗pointZArray)

    Convienience function to returns a list of points on the intersection line of two geometric components. Often there are more that one intersection line, therefore you need to specify the line.
- TIGL_COMMON_EXPORT TiglReturnCode tiglConfigurationGetLength (TiglCPACSConfigurationHandle cpacsHandle, double ∗pLength)

    Returns the length of the plane.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportFusedBREP (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗filename)

    Exports the fused/trimmed geometry of a CPACS configuration to BREP format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportFusedSTEP (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗filenamePtr)

    Exports the fused/trimmed geometry of a CPACS configuration to STEP format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportFusedWingFuselageIGES (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗filenamePtr)

    Exports the boolean fused geometry of a CPACS configuration to IGES format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportFuselageColladaByUID (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗fuselageUID, const char ∗filename, double deflection)

    Exports the boolean fused geometry of a fuselage (selected by uid) meshed to Collada (∗.dae) format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportIGES (TiglCPACSConfigurationHandle cpacsHandle, const char ∗filenamePtr)

    Exports the geometry of a CPACS configuration to IGES format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageSTL (TiglCPACSConfiguration↩ Handle cpacsHandle, int fuselageIndex, const char ∗filenamePtr, double deflection)

    Exports the boolean fused geometry of a fuselage meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageSTLByUID (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗fuselageUID, const char ∗filenamePtr, double deflection)

    Exports the boolean fused geometry of a fuselage meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageVTKByIndex (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const int fuselageIndex, const char ∗filenamePtr, double deflection)

    Exports the boolean fused geometry of a fuselage (selected by index) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageVTKByUID (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗fuselageUID, const char ∗filenamePtr, double deflection)

    Exports the boolean fused geometry of a fuselage (selected by uid) meshed to VTK format.

- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedFuselageVTKSimpleByUID (const TiglCPA↩ CSConfigurationHandle cpacsHandle, const char ∗fuselageUID, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a fuselage (selected by uid) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedGeometrySTL (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of the whole configuration meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedGeometryVTK (const TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of the whole configuration meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedGeometryVTKSimple (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of the whole configuration meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingSTL (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a wing meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingSTLByUID (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗wingUID, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a wing meshed to STL format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingVTKByIndex (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const int wingIndex, const char ∗filenamePtr, const double deflection)

  Exports the boolean fused geometry of a wing (selected by id) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingVTKByUID (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗wingUID, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a wing (selected by UID) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportMeshedWingVTKSimpleByUID (const TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗wingUID, const char ∗filenamePtr, double deflection)

  Exports the boolean fused geometry of a wing (selected by UID) meshed to VTK format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportSTEP (TiglCPACSConfigurationHandle cpacsHandle, const char ∗filenamePtr)

  Exports the geometry of a CPACS configuration to STEP format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportVTKSetOptions (const char ∗key, const char ∗value)

  Sets options for the VTK Export.
- TIGL_COMMON_EXPORT TiglReturnCode tiglExportWingColladaByUID (const TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗wingUID, const char ∗filename, double deflection)

  Exports the boolean fused geometry of a wing (selected by uid) meshed to Collada (∗.dae) format.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetCircumference (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double ∗circumferencePtr)

  Returns the circumference of a fuselage surface for a given fuselage and segment index and an eta.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndConnectedSegmentCount (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int ∗segmentCountPtr)

  Returns the count of segments connected to the end section of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndConnectedSegmentIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int n, int ∗connectedIndexPtr)

  Returns the index (number) of the n-th segment connected to the end section of a given fuselage segment. n starts at 1.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndSectionAndElementIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int ∗sectionIndexPtr, int ∗element↩ IndexPtr)

  Returns the section index and section element index of the end side of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetEndSectionAndElementUID (TiglCPACS↩ ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, char ∗∗sectionUIDPtr, char ∗∗elementUIDPtr)

  Returns the section UID and section element UID of the end side of a given fuselage segment.

- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetIndex (TiglCPACSConfigurationHandle cpacs←↩ Handle, const char ∗fuselageUID, int ∗fuselageIndexPtr)

  Returns the index of a fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetMinumumDistanceToGround (TiglCPACS←↩ ConfigurationHandle cpacsHandle, char ∗fuselageUID, double axisPntX, double axisPntY, double axisPntZ, double axisDirX, double axisDirY, double axisDirZ, double angle, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns the point where the distance between the selected fuselage and the ground is at minimum. The Fuselage could be turned with a given angle at at given axis, specified by a point and a direction.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetNumPointsOnXPlane (TiglCPACSConfiguration←↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double xpos, int ∗numPointsPtr)

  Returns the number of points on a fuselage surface for a given fuselage and a give x-position.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetNumPointsOnYPlane (TiglCPACSConfiguration←↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double ypos, int ∗numPointsPtr)

  Returns the number of points on a fuselage surface for a given fuselage and a give y-position.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPoint (TiglCPACSConfigurationHandle cpacs←↩ Handle, int fuselageIndex, int segmentIndex, double eta, double zeta, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on a fuselage surface for a given fuselage and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointAngle (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double alpha, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on a fuselage surface for a given fuselage and segment index and an angle alpha (degree).
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointAngleTranslated (TiglCPACSConfiguration←↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double alpha, double y_cs, double z_cs, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on a fuselage surface for a given fuselage and segment index and an angle alpha (degree). 0 degree of the angle alpha is meant to be "up" in the direction of the positive z-axis like specifies in cpacs. The origin of the line that will be rotated with the angle alpha could be translated via the parameters y_cs and z_cs.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointOnXPlane (TiglCPACSConfiguration←↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double xpos, int pointIndex, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on a fuselage surface for a given fuselage and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetPointOnYPlane (TiglCPACSConfiguration←↩ Handle cpacsHandle, int fuselageIndex, int segmentIndex, double eta, double ypos, int pointIndex, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on a fuselage surface for a given fuselage and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetProfileName (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int sectionIndex, int elementIndex, char ∗∗profileNamePtr)

  Returns the name of a fuselage profile. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSectionCount (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int ∗sectionCount)

  Returns the number of sections of a fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSectionUID (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int sectionIndex, char ∗∗uidNamePtr)

  Returns the UID of a section of a fuselage. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentCount (TiglCPACSConfiguration←↩ Handle cpacsHandle, int fuselageIndex, int ∗segmentCountPtr)

  Returns the number of segments for a fuselage in a CPACS configuration.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentIndex (TiglCPACSConfigurationHandle cpacsHandle, const char ∗segmentUID, int ∗segmentIndexPtr, int ∗fuselageIndexPtr)

  Returns the Index of a segment of a fuselage.

- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentSurfaceArea (TiglCPACSConfiguration↩
Handle cpacsHandle, int fuselageIndex, int segmentIndex, double ∗surfaceAreaPtr)

    Returns the surface area of a segment of a fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentUID (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, char ∗∗uidNamePtr)

    Returns the UID of a segment of a fuselage. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSegmentVolume (TiglCPACSConfiguration↩
Handle cpacsHandle, int fuselageIndex, int segmentIndex, double ∗volumePtr)

    Returns the volume of a segment of a fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartConnectedSegmentCount (TiglCPACS↩
ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int ∗segmentCountPtr)

    Returns the count of segments connected to the start section of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartConnectedSegmentIndex (TiglCPACS↩
ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int n, int ∗connectedIndexPtr)

    Returns the index (number) of the n-th segment connected to the start section of a given fuselage segment. n starts at 1.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartSectionAndElementIndex (TiglCPACS↩
ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, int ∗sectionIndexPtr, int ∗element↩
IndexPtr)

    Returns the section index and section element index of the start side of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetStartSectionAndElementUID (TiglCPACS↩
ConfigurationHandle cpacsHandle, int fuselageIndex, int segmentIndex, char ∗∗sectionUIDPtr, char ∗∗elementUIDPtr)

    Returns the section UID and section element UID of the start side of a given fuselage segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSurfaceArea (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, double ∗surfaceAreaPtr)

    Returns the surface area of the fuselage. Currently, the area includes also the faces on the fuselage symmetry plane (in case of a symmetric wing). This is in particular a problem for fuselages, where only one half side is defined in CPACS. In future releases, these faces will not belong anymore to the surface area calculation.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetSymmetry (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, TiglSymmetryAxis ∗symmetryAxisPtr)

    Returns the Symmetry Enum if the fuselage has symmetry-axis.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetUID (TiglCPACSConfigurationHandle cpacs↩
Handle, int fuselageIndex, char ∗∗uidNamePtr)

    Returns the UID of a fuselage. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglFuselageGetVolume (TiglCPACSConfigurationHandle cpacsHandle, int fuselageIndex, double ∗volumePtr)

    Returns the volume of the fuselage.
- TIGL_COMMON_EXPORT TiglReturnCode tiglGetCPACSTixiHandle (TiglCPACSConfigurationHandle cpacsHandle, TixiDocumentHandle ∗tixiHandlePtr)

    Returns the underlying TixiDocumentHandle for a given CPACS configuration handle.
- TIGL_COMMON_EXPORT const char ∗ tiglGetErrorString (TiglReturnCode errorCode)

    Translates an error code into a string.
- TIGL_COMMON_EXPORT TiglReturnCode tiglGetFuselageCount (TiglCPACSConfigurationHandle cpacs↩
Handle, int ∗fuselageCountPtr)

    Returns the number of fuselages in a CPACS configuration.
- TIGL_COMMON_EXPORT TiglReturnCode tiglGetRotorCount (TiglCPACSConfigurationHandle cpacs↩
Handle, int ∗rotorCountPtr)

    Returns the number of rotors in a CPACS configuration.
- TIGL_COMMON_EXPORT const char ∗ tiglGetVersion ()

    Returns the version number of this TIGL version.

- TIGL_COMMON_EXPORT TiglReturnCode tiglGetWingCount (TiglCPACSConfigurationHandle cpacs↩ Handle, int ∗wingCountPtr)

  Returns the number of wings in a CPACS configuration.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectComponents (TiglCPACSConfigurationHandle cpacsHandle, const char ∗componentUidOne, const char ∗componentUidTwo, char ∗∗intersectionID)

  tiglIntersectComponents computes the intersection line(s) between two shapes specified by their CPACS uid. It returns an intersection ID for further computations on the result. To query points on the intersection line, tigl↩ IntersectGetPoint has to be called.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectGetLineCount (TiglCPACSConfigurationHandle cpacsHandle, const char ∗intersectionID, int ∗lineCount)

  tiglIntersectGetLineCount return the number of intersection lines computed by tiglIntersectComponents or tigl↩ IntersectWithPlane for the given intersectionID.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectGetPoint (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗intersectionID, int lineIdx, double eta, double ∗pointX, double ∗pointY, double ∗pointZ)

  tiglIntersectGetPoint samples a point on an intersection line calculated by tiglIntersectComponents or tiglIntersect↩ WithPlane.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIntersectWithPlane (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗componentUid, double px, double py, double pz, double nx, double ny, double nz, char ∗∗intersectionID)

  tiglIntersectWithPlane computes the intersection line(s) between a shape and a plane. It returns an intersection ID for further computations on the result. To query points on the intersection line, tiglIntersectGetPoint has to be called.
- TIGL_COMMON_EXPORT TiglReturnCode tiglIsCPACSConfigurationHandleValid (TiglCPACSConfiguration↩ Handle cpacsHandle, TiglBoolean ∗isValidPtr)

  Checks if a given CPACS configuration handle is a valid.
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogSetFileEnding (const char ∗ending)

  Sets the file ending for logging files. Default is "log".
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogSetTimeInFilenameEnabled (TiglBoolean enabled)

  Enables or disables appending a unique date/time identifier inside the log file name (behind the file prefix). By default, the time identifier is enabled.
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogSetVerbosity (TiglLogLevel level)

  Set the console verbosity level.
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogToFileDisabled ()

  Disabled file logging. If a log file is currently opened by TiGL it will be closed. The log messages are printed to console. This is the default logging mechanism of TIGL.
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogToFileEnabled (const char ∗filePrefix)

  Sets up the tigl logging mechanism to send all log messages into a file.
- TIGL_COMMON_EXPORT TiglReturnCode tiglLogToFileStreamEnabled (FILE ∗fp)

  Sets up the tigl logging mechanism to send all log messages into an already opened file.
- TIGL_COMMON_EXPORT TiglReturnCode tiglOpenCPACSConfiguration (TixiDocumentHandle tixiHandle, const char ∗configurationUID, TiglCPACSConfigurationHandle ∗cpacsHandlePtr)

  Opens a CPACS configuration and builds up the data and geometry structure in memory.
- TIGL_COMMON_EXPORT TiglReturnCode tiglProfileGetBSplineCount (TiglCPACSConfigurationHandle cpacsHandle, const char ∗profileUID, int ∗curveCount)

  Returns the number of curves the given profile is made of.
- TIGL_COMMON_EXPORT TiglReturnCode tiglProfileGetBSplineData (TiglCPACSConfigurationHandle cpacsHandle, const char ∗profileUID, int curveid, int nControlPoints, double ∗cpx, double ∗cpy, double ∗cpz, int nKnots, double ∗knots)

  Returns the B-Spline data of the given profile curve. This includes the knot vector and the control points of the B-Spline.
- TIGL_COMMON_EXPORT TiglReturnCode tiglProfileGetBSplineDataSizes (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗profileUID, int curveid, int ∗degree, int ∗nControlPoints, int ∗nKnots)

  Returns the B-Spline data sizes for a given curve on a profile. This includes size of the knot vector, size of the control point vector and degree of the spline.

- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetAzimuthAngle (TiglCPACSConfiguration↩ Handle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗azimuthAnglePtr)

  Returns the azimuth angle of a rotor blade in degrees.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetLocalChord (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, int segmentIndex, double eta, double ∗chordPtr)

  Returns the local chord length of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetLocalRadius (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, int segmentIndex, double eta, double ∗radiusPtr)

  Returns the local radius of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetLocalTwistAngle (TiglCPACSConfiguration↩ Handle cpacsHandle, int rotorIndex, int rotorBladeIndex, int segmentIndex, double eta, double ∗twistAngle↩ Ptr)

  Returns the local twist angle [deg] of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetPlanformArea (TiglCPACSConfiguration↩ Handle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗planformAreaPtr)

  Returns the planform area of the rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetRadius (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗radiusPtr)

  Returns the radius of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetSurfaceArea (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗surfaceAreaPtr)

  Returns the surface area of the rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetTipSpeed (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗tipSpeedPtr)

  Returns the tip speed of a rotor blade [m/s].
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetVolume (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, double ∗volumePtr)

  Returns the volume of the rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetWingIndex (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, int ∗wingIndexPtr)

  Returns the index of the parent wing definition of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorBladeGetWingUID (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int rotorBladeIndex, char ∗∗wingUIDPtr)

  Returns the UID of the parent wing definition of a rotor blade.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetIndex (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗rotorUID, int ∗rotorIndexPtr)

  Returns the Index of a rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetRadius (TiglCPACSConfigurationHandle cpacs↩ Handle, int rotorIndex, double ∗radiusPtr)

  Returns the radius of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetReferenceArea (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, double ∗referenceAreaPtr)

  Returns the reference area of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetRotorBladeCount (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, int ∗rotorBladeCountPtr)

  Returns the total number of rotor blades attached to a rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetSolidity (TiglCPACSConfigurationHandle cpacs↩ Handle, int rotorIndex, double ∗solidityPtr)

  Returns the solidity of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetSurfaceArea (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, double ∗surfaceAreaPtr)

  Returns the surface area of the rotor.

- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetTipSpeed (TiglCPACSConfigurationHandle cpacs↩ Handle, int rotorIndex, double ∗tipSpeedPtr)

  Returns the tip speed of the rotor in [m/s].
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetTotalBladePlanformArea (TiglCPACSConfiguration↩ Handle cpacsHandle, int rotorIndex, double ∗totalBladePlanformAreaPtr)

  Returns the total blade planform area of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetUID (TiglCPACSConfigurationHandle cpacsHandle, int rotorIndex, char ∗∗uidNamePtr)

  Returns the UID of a rotor. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglRotorGetVolume (TiglCPACSConfigurationHandle cpacs↩ Handle, int rotorIndex, double ∗volumePtr)

  Returns the volume of the rotor.
- TIGL_COMMON_EXPORT TiglReturnCode tiglSaveCPACSConfiguration (const char ∗configurationUID, TiglCPACSConfigurationHandle cpacsHandle)

  Writes a CPACS configuration based on the data and geometry structure in memory.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentComputeEtaIntersection (TiglCP↩ ACSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, double csEta1, double csXsi1, double csEta2, double csXsi2, double eta, double ∗xsi, TiglBoolean ∗hasWarning)

  Given a straight line in space defined by a pair of component segment (eta,xsi) coordinates, the function computes the intersection of the line with a component segment iso-eta line.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentFindSegment (TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, double x, double y, double z, char ∗∗segmentUID, char ∗∗wingUID)

  Returns the segmentUID and wingUID for a given point on a componentSegment. The returned strings must not be freed by the user anymore.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetMaterialCount (TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗compSegmentUID, TiglStructureType structureType, double eta, double xsi, int ∗materialCount)

  Returns the number of materials defined at a point on the wing component segment surface.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetMaterialThickness (TiglCPA↩ CSConfigurationHandle cpacsHandle, const char ∗compSegmentUID, TiglStructureType structureType, double eta, double xsi, int materialIndex, double ∗thickness)

  Returns one of the material thicknesses of a given point on the wing component segment surface. The number of materials on that point has to be first queried using tiglWingComponentSegmentGetMaterialCount.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetMaterialUID (TiglCPACS↩ ConfigurationHandle cpacsHandle, const char ∗compSegmentUID, TiglStructureType structureType, double eta, double xsi, int materialIndex, char ∗∗uid)

  Returns one of the material UIDs of a given point on the wing component segment surface. The number of materials on that point has to be first queried using tiglWingComponentSegmentGetMaterialCount.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetNumberOfSegments (TiglCP↩ ACSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, int ∗nsegments)

  Returns the number of segments belonging to a component segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetPoint (TiglCPACSConfiguration↩ Handle cpacsHandle, const char ∗componentSegmentUID, double eta, double xsi, double ∗x, double ∗y, double ∗z)

  Returns x,y,z koordinates for a given eta and xsi on a componentSegment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetSegmentIntersection (TiglCP↩ ACSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, const char ∗segmentUID, double csEta1, double csXsi1, double csEta2, double csXsi2, double segmentEta, double ∗segmentXsi, Tigl↩ Boolean ∗hasWarning)

  Computes the intersection of a line (defined by component segment coordinates) with an iso-eta line on a specified wing segment.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentGetSegmentUID (TiglCPA↩ CSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, int segmentIndex, char ∗∗segmentUID)

  Returns the segment UID of a segment belonging to a component segment. The segment is specified with its index, which is in the 1...nsegments. The number of segments nsegments can be queried with tiglWingComponent↩ SegmentGetNumberOfSegments.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingComponentSegmentPointGetSegmentEtaXsi (TiglCP↩ ACSConfigurationHandle cpacsHandle, const char ∗componentSegmentUID, double eta, double xsi, char ∗∗wingUID, char ∗∗segmentUID, double ∗segmentEta, double ∗segmentXsi, double ∗errorDistance)

  Returns eta, xsi, segmentUID and wingUID for a given eta and xsi on a componentSegment.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetChordNormal (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double ∗normalXPtr, double ∗normal↩ YPtr, double ∗normalZPtr)

  Returns a normal vector on the wing chord surface for a a given wing and segment index.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetChordPoint (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on the wing chord surface for a a given wing and segment index.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetComponentSegmentCount (TiglCPACSConfiguration↩ Handle cpacsHandle, int wingIndex, int ∗compSegmentCountPtr)

  Returns the number of component segments for a wing in a CPACS configuration.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetComponentSegmentIndex (TiglCPACSConfiguration↩ Handle cpacsHandle, int wingIndex, const char ∗compSegmentUID, int ∗segmentIndexPtr)

  Returns the Index of a component segment of a wing.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetComponentSegmentUID (TiglCPACSConfiguration↩ Handle cpacsHandle, int wingIndex, int compSegmentIndex, char ∗∗uidNamePtr)

  Returns the UID of a component segment of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetIndex (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗wingUID, int ∗wingIndexPtr)

  Returns the Index of a wing.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerConnectedSegmentCount (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int ∗segmentCountPtr)

  Returns the count of wing segments connected to the inner section of a given segment.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerConnectedSegmentIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int n, int ∗connectedIndexPtr)

  Returns the index (number) of the n-th wing segment connected to the inner section of a given segment. n starts at 1.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerSectionAndElementIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int ∗sectionIndexPtr, int ∗element↩ IndexPtr)

  Returns the section index and section element index of the inner side of a given wing segment.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetInnerSectionAndElementUID (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, char ∗∗sectionUIDPtr, char ∗∗element↩ UIDPtr)

  Returns the section UID and section element UID of the inner side of a given wing segment.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetLowerPoint (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on the lower wing surface for a a given wing and segment index.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetLowerPointAtDirection (TiglCPACSConfiguration↩ Handle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double dirx, double diry, double dirz, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr, double ∗errorDistance)

Returns a point on the lower wing surface for a a given wing and segment index. This function is different from tigl↩ WingGetLowerPoint: First, a point on the wing chord surface is computed (defined by segment index and eta,xsi). Then, a line is constructed, which is defined by this point and a direction given by the user. The intersection of this line with the lower wing surface is finally returned. The point is returned in absolute world coordinates.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetMAC (TiglCPACSConfigurationHandle cpacs↩ Handle, const char ∗wingUID, double ∗mac_chord, double ∗mac_x, double ∗mac_y, double ∗mac_z)

  This function calculates location of the quarter of mean aerodynamic chord, and gives the chord lenght as well.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterConnectedSegmentCount (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int ∗segmentCountPtr)

  Returns the count of wing segments connected to the outer section of a given segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterConnectedSegmentIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int n, int ∗connectedIndexPtr)

  Returns the index (number) of the n-th wing segment connected to the outer section of a given segment. n starts at 1.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterSectionAndElementIndex (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, int ∗sectionIndexPtr, int ∗element↩ IndexPtr)

  Returns the section index and section element index of the outer side of a given wing segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetOuterSectionAndElementUID (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, char ∗∗sectionUIDPtr, char ∗∗element↩ UIDPtr)

  Returns the section UID and section element UID of the outer side of a given wing segment.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetProfileName (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int sectionIndex, int elementIndex, char ∗∗profileNamePtr)

  Returns the name of a wing profile. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetReferenceArea (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, TiglSymmetryAxis symPlane, double ∗referenceAreaPtr)

  Returns the reference area of the wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSectionCount (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int ∗sectionCount)

  Returns the number of sections of a wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSectionUID (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int sectionIndex, char ∗∗uidNamePtr)

  Returns the UID of a section of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentCount (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int ∗segmentCountPtr)

  Returns the number of segments for a wing in a CPACS configuration.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentEtaXsi (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, double pointX, double pointY, double pointZ, int ∗segmentIndex, double ∗eta, double ∗xsi, int ∗isOnTop)

  Inverse function to tiglWingGetLowerPoint and tiglWingGetLowerPoint. Calculates to a point (x,y,z) in global coordinates the wing segment coordinates and the wing segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentIndex (TiglCPACSConfigurationHandle cpacsHandle, const char ∗segmentUID, int ∗segmentIndexPtr, int ∗wingIndexPtr)

  Returns the Index of a segment of a wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentLowerSurfaceAreaTrimmed (TiglCPACS↩ ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta1, double xsi1, double eta2, double xsi2, double eta3, double xsi3, double eta4, double xsi4, double ∗surfaceArea)

  Computes the area of the trimmed lower wing segment surface. This function can be e.g. used to determine the area of the wing flaps.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentSurfaceArea (TiglCPACSConfiguration↩ Handle cpacsHandle, int wingIndex, int segmentIndex, double ∗surfaceAreaPtr)

Returns the surface area of a segment of a wing. This includes only the area of the upper and lower wing segment surface and does not include the trailing egde or any closing faces.

- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentUID (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, char ∗∗uidNamePtr)

  Returns the UID of a segment of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentUpperSurfaceAreaTrimmed (TiglCPACS←ConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta1, double xsi1, double eta2, double xsi2, double eta3, double xsi3, double eta4, double xsi4, double ∗surfaceArea)

  Computes the area of the trimmed upper wing segment surface. This function can be e.g. used to determine the area of the wing flaps.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSegmentVolume (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double ∗volumePtr)

  Returns the volume of a segment of a wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSpan (TiglCPACSConfigurationHandle cpacs←Handle, const char ∗wingUID, double ∗pSpan)

  Returns the span of a wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSurfaceArea (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, double ∗surfaceAreaPtr)

  Returns the surface area of the wing. Currently, the area includes also the faces on the wing symmetry plane (in case of a symmetric wing). In coming releases, these faces will not belong anymore to the surface area calculation.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetSymmetry (TiglCPACSConfigurationHandle cpacs←Handle, int wingIndex, TiglSymmetryAxis ∗symmetryAxisPtr)

  Returns the Symmetry Enum if the wing has symmetry-axis.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetUID (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, char ∗∗uidNamePtr)

  Returns the UID of a wing. The string returned must not be deleted by the caller via free(). It will be deleted when the CPACS configuration is closed.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetUpperPoint (TiglCPACSConfigurationHandle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr)

  Returns a point on the upper wing surface for a a given wing and segment index.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetUpperPointAtDirection (TiglCPACSConfiguration←Handle cpacsHandle, int wingIndex, int segmentIndex, double eta, double xsi, double dirx, double diry, double dirz, double ∗pointXPtr, double ∗pointYPtr, double ∗pointZPtr, double ∗errorDistance)

  Returns a point on the upper wing surface for a a given wing and segment index. This function is different from tiglWingGetUpperPoint: First, a point on the wing chord surface is computed (defined by segment index and eta,xsi). Then, a line is constructed, which is defined by this point and a direction given by the user. The intersection of this line with the upper wing surface is finally returned. The point is returned in absolute world coordinates.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetVolume (TiglCPACSConfigurationHandle cpacs←Handle, int wingIndex, double ∗volumePtr)

  Returns the volume of the wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingGetWettedArea (TiglCPACSConfigurationHandle cpacsHandle, char ∗wingUID, double ∗wettedAreaPtr)

  Returns the wetted area of the wing.
- TIGL_COMMON_EXPORT TiglReturnCode tiglWingSegmentPointGetComponentSegmentEtaXsi (TiglCP←ACSConfigurationHandle cpacsHandle, const char ∗segmentUID, const char ∗componentSegmentUID, double segmentEta, double segmentXsi, double ∗eta, double ∗xsi)

  Returns eta, xsi coordinates of a componentSegment given segmentEta and segmentXsi on a wing segment.

10.8.1 Detailed Description

Declaration of the TIGL C interface.

10.8.2 Macro Definition Documentation

10.8.2.1 TIGL_COMMON_EXPORT

```
#define TIGL_COMMON_EXPORT
```

10.8.2.2 TIGL_COMPONENT_FUSELAGE

```
#define TIGL_COMPONENT_FUSELAGE 8
```

The Component is a fuselage

10.8.2.3 TIGL_COMPONENT_FUSELSEGMENT

```
#define TIGL_COMPONENT_FUSELSEGMENT 128
```

The Component is a fuselage segment

10.8.2.4 TIGL_COMPONENT_GENERICSYSTEM

```
#define TIGL_COMPONENT_GENERICSYSTEM 1024
```

The Component is a generic system

10.8.2.5 TIGL_COMPONENT_LOGICAL

```
#define TIGL_COMPONENT_LOGICAL 2
```

A logical component, like a wing segment

10.8.2.6 TIGL_COMPONENT_PHYSICAL

```
#define TIGL_COMPONENT_PHYSICAL 1
```

A phyisical component like a fuselage, wing, nacelle, something you could touch

10.8.2.7 TIGL_COMPONENT_PLANE

```
#define TIGL_COMPONENT_PLANE 4
```

The whole aircraft

10.8.2.8 TIGL_COMPONENT_ROTOR

```
#define TIGL_COMPONENT_ROTOR 2048
```

The Component is a rotor

10.8.2.9 TIGL_COMPONENT_ROTORBLADE

```
#define TIGL_COMPONENT_ROTORBLADE 4096
```

The Component is a rotor blade

10.8.2.10 TIGL_COMPONENT_SEGMENT

```
#define TIGL_COMPONENT_SEGMENT 32
```

The Component is a general segment

10.8.2.11 TIGL_COMPONENT_WING

```
#define TIGL_COMPONENT_WING 16
```

The Component is a wing

10.8.2.12 TIGL_COMPONENT_WINGCOMPSEGMENT

```
#define TIGL_COMPONENT_WINGCOMPSEGMENT 256
```

The Component is a wing component segment

10.8.2.13 TIGL_COMPONENT_WINGSEGMENT

```
#define TIGL_COMPONENT_WINGSEGMENT 64
```

The Component is a wing segment

10.8.2.14 TIGL_COMPONENT_WINGSHELL

```
#define TIGL_COMPONENT_WINGSHELL 512
```

The Component is a face of the wing (e.g. upper wing surface)

10.8.2.15 TIGL_H

```
#define TIGL_H
```

10.8.3 Typedef Documentation

10.8.3.1 TiglAlgorithmCode

```
typedef enum TiglAlgorithmCode TiglAlgorithmCode
```

Typedef for possible algorithm types used in calculations.

Possible values for variables of type TiglAlgorithmCode are:

- TIGL_INTERPOLATE_LINEAR_WIRE: Use a linear interpolation between the points of a wire.
- TIGL_INTERPOLATE_BSPLINE_WIRE: Use a BSpline interpolation between the points of a wire.
- TIGL_APPROXIMATE_BSPLINE_WIRE: Use a BSpline approximation for the points of a wire.

10.8.3.2 TiglBoolean

```
typedef enum TiglBoolean TiglBoolean
```

Definition of boolean type used in TIGL.

Possible values are:

- TIGL_FALSE
- TIGL_TRUE

10.8.3.3 TiglContinuity

```
typedef enum TiglContinuity TiglContinuity
```

10.8.3.4 TiglCoordinateSystem

```
typedef enum TiglCoordinateSystem TiglCoordinateSystem
```

10.8.3.5 TiglCPACSConfigurationHandle

```
typedef int TiglCPACSConfigurationHandle
```

Datatype for a CPACS configuration handle.

10.8.3.6 TiglImportExportFormat

```
typedef enum TiglImportExportFormat TiglImportExportFormat
```

Definition of the different file formats used for import/export used in TIGL.

Possible values are:

- TIGL_IMPORTEXPORT_IGES: Use IGES format for geometry import/export.
- TIGL_IMPORTEXPORT_STEP: Use STEP format for geometry import/export.
- TIGL_IMPORTEXPORT_STL: Use STL format for geometry import/export.
- TIGL_IMPORTEXPORT_VTK: Use VTK format for geometry import/export.

10.8.3.7 TiglLoftSide

```
typedef enum TiglLoftSide TiglLoftSide
```

10.8.3.8 TiglLogLevel

```
typedef enum TiglLogLevel TiglLogLevel
```

Definition of logging levels.

Possible values are:

- TILOG_SILENT
- TILOG_ERROR
- TILOG_WARNING
- TILOG_INFO
- TILOG_DEBUG
- TILOG_DEBUG1
- TILOG_DEBUG2
- TILOG_DEBUG3
- TILOG_DEBUG4

10.8.3.9 TiglReturnCode

```
typedef enum TiglReturnCode TiglReturnCode
```

Definition of error return type.

Possible values are:

- TIGL_SUCCESS
- TIGL_ERROR
- TIGL_NULL_POINTER
- TIGL_NOT_FOUND
- TIGL_XML_ERROR
- TIGL_OPEN_FAILED
- TIGL_CLOSE_FAILED
- TIGL_INDEX_ERROR
- TIGL_STRING_TRUNCATED
- TIGL_WRONG_TIXI_VERSION
- TIGL_UID_ERROR

10.8.3.10 TiglStringList

```
typedef const char** TiglStringList
```

10.8.3.11 TiglStructureType

```
typedef enum TiglStructureType TiglStructureType
```

10.8.3.12 TiglSymmetryAxis

```
typedef enum TiglSymmetryAxis TiglSymmetryAxis
```

Definition of Symmetry Axis used in TIGL.

Possible values are:

- TIGL_NO_SYMMETRY
- TIGL_X_Y_PLANE
- TIGL_X_Z_PLANE
- TIGL_Y_Z_PLANE

10.8.4 Enumeration Type Documentation

10.8.4.1 TiglContinuity

```
enum TiglContinuity
```

Enumerator

C0	
C1	
C2	

10.8.4.2 TiglCoordinateSystem

```
enum TiglCoordinateSystem
```

Enumerator

GLOBAL_COORDINATE_SYSTEM	
WING_COORDINATE_SYSTEM	
FUSELAGE_COORDINATE_SYSTEM	

10.8.4.3 TiglLoftSide

```
enum TiglLoftSide
```

Enumerator

UPPER_SIDE	
LOWER_SIDE	

10.8.4.4 TiglStructureType

```
enum TiglStructureType
```

Enumerator

UPPER_SHELL	
LOWER_SHELL	
INNER_STRUCTURE	